ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
Τμήμα Ηλεκτρολόγων & Ηλεκτρονικών Μηχανικών

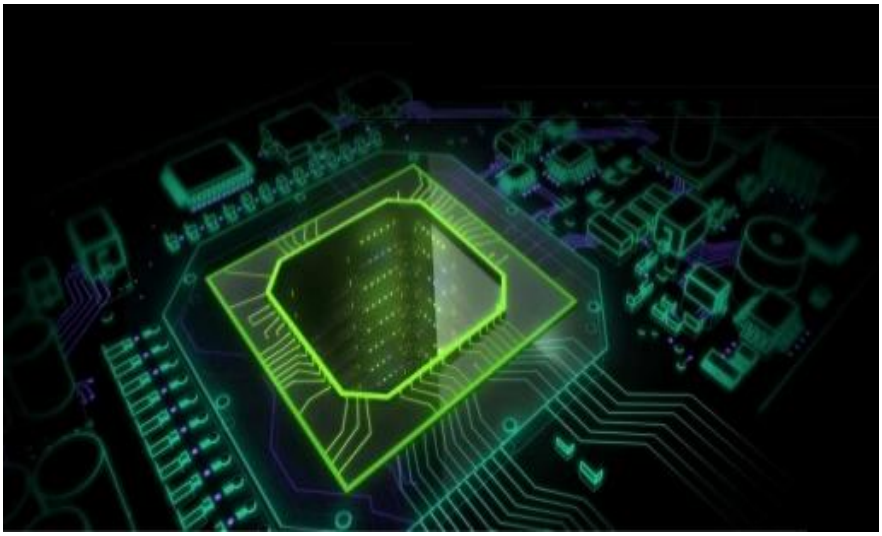
UNIVERSITY of WEST ATTICA
FACULTY OF ENGINEERING
Department of Electrical & Electronics Engineering

Πρόγραμμα Μεταπτυχιακών Σπουδών
Διαδικτυωμένα Ηλεκτρονικά Συστήματα

Master of Science in
Internetworked Electronic Systems

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«ΤΕΧΝΙΚΕΣ ΓΙΑ ΧΑΜΗΛΗ ΚΑΤΑΝΑΛΩΣΗ ΙΣΧΥΟΣ ΣΕ ΣΥΓΧΡΟΝΟΥΣ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΕΣ ΚΑΙ ΜΙΚΡΟΕΛΕΓΚΤΕΣ»



Μεταπτυχιακός Φοιτητής: ΚΟΤΑΜΠΙΤΣΗΣ ΧΡΗΣΤΟΣ, ΑΜ: IES-0044
Επιβλέπουσα: ΡΑΓΚΟΥΣΗ ΜΑΡΙΑ, Καθηγήτρια

ΑΙΓΑΛΕΩ, ΦΕΒΡΟΥΑΡΙΟΣ 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
Τμήμα Ηλεκτρολόγων & Ηλεκτρονικών Μηχανικών

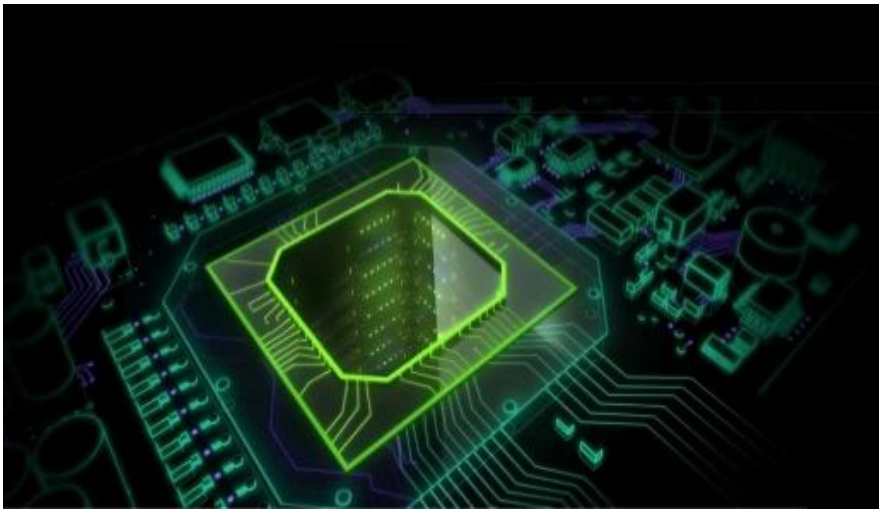
UNIVERSITY of WEST ATTICA
FACULTY OF ENGINEERING
Department of Electrical & Electronics Engineering

Πρόγραμμα Μεταπτυχιακών Σπουδών
Διαδικτυωμένα Ηλεκτρονικά Συστήματα

Master of Science in
Internetworked Electronic Systems

MSc Thesis

"TECHNIQUES FOR LOW POWER CONSUMPTION IN MODERN MICROPROCESSORS AND MICROCONTROLLERS"



Student: KOTAMPITSIS CHRISTOS, Reg. Nr. IES-0044
MSc Thesis Supervisor: Prof. RANGOUSSI MARIA

ATHENS-EGALEO, FEBRUARY 2021

ΠΕΡΙΛΗΨΗ

Ένας από τους πιο σημαντικούς περιορισμούς αλλά και ένα από τα βασικά κριτήρια βελτιστοποίησης και αξιολόγησης ενός κυκλώματος ή συστήματος, είναι η κατανάλωση ισχύος. Η ανάγκη για χαμηλή κατανάλωση ισχύος, τέθηκε για πρώτη φορά σαν ένα επείγον ζήτημα την δεκαετία του 1970 όπου ήταν φανερή η σημασία της επίτευξής της, αφού θα επέτρεπε μεγαλύτερη αυτονομία και μικρότερο μέγεθος στις φορητές συσκευές. Τις επόμενες δεκαετίες, η στροφή της βιομηχανίας προς την τεχνολογία CMOS ήταν επιβεβλημένη λόγω της αυξημένης κατανάλωσης ισχύος των μικροεπεξεργαστών, των μνημών καθώς και των custom σχεδιασμών και αυτό γιατί σε αντίθεση με τις υπόλοιπες τεχνολογίες, η τεχνολογία CMOS εμφανίζει μηδενική στατική κατανάλωση ισχύος, γεγονός που τη καθιέρωσε μέχρι και σήμερα για την υλοποίηση των ψηφιακών κυκλωμάτων.

Παραδοσιακά, η έρευνα για την ενεργειακή απόδοση είχε επικεντρωθεί στη μείωση της κατανάλωσης ενέργειας σε επίπεδο υλικού. Όπως όμως έχει αποδειχτεί, ένας σημαντικός παράγοντας κατανάλωσης ισχύος είναι και το λογισμικό, που κατευθύνει μεγάλο μέρος της δραστηριότητας του υλικού και παρέχει πολλές ευκαιρίες για βελτίωση της ενεργειακής απόδοσης.

Συμπερασματικά, το θέμα της μείωσης της κατανάλωσης ισχύος, αντιμετωπίζεται με διαφορετικούς τρόπους και μεθοδολογίες στα διαφορετικά επίπεδα ανάπτυξης, όπως το κύκλωμα, η λογική, η αρχιτεκτονική, ο αλγόριθμος μέχρι και το λειτουργικό σύστημα.

Στη παρούσα διπλωματική εργασία, παρουσιάζονται και αναλύονται οι τεχνικές σε επίπεδο hardware και software, που βοηθούν στη μείωση της κατανάλωσης ισχύος σε σύγχρονους επεξεργαστές και μικροελεγκτές.

ΛΕΞΕΙΣ – ΚΛΕΙΔΙΑ:

cmos, cpu, επεξεργαστές, ψηφιακά κυκλώματα, hardware, λογισμικό, μείωση κατανάλωσης ισχύος, μικροελεγκτές, software, τεχνικές σχεδίασης

ABSTRACT

One of the most important limitations but also one of the basic criteria for optimizing and evaluating a circuit or system, is power consumption. The need for low power consumption was first raised as a critical issue in the 1970's where the importance of achieving it, was obvious, as it would allow more autonomy and smaller size in mobile devices. In the following decades, the industry shifted to CMOS technology due to the increased power consumption of microprocessors, memory and custom applications, because unlike other technologies, CMOS technology shows zero static power consumption, which established it to this day, for the implementation of digital circuits.

Traditionally, energy efficiency research has focused on reducing energy consumption at the material level. But as it turns out, an important factor in power consumption is the software, which directs much of the hardware activity and provides many opportunities to improve energy efficiency.

The conclusion is that the issue of reducing the power consumption of an integrated circuit or system is addressed in different ways and methodologies, at each of the different levels of development, such as circuit, logic, architecture, algorithm and even operating system.

In this thesis, are presented and analyzed the techniques of reducing power consumption, in terms of hardware and software in modern microprocessors and microcontrollers.

KEYWORDS:

cmos, cpu, digital circuits, hardware, mcu, microcontrollers, microprocessors, power consumption reduction, software, digital design methods

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες και την εκτίμησή μου στην επιβλέπουσα καθηγήτρια, κ. Μαρία Ραγκούση για την στήριξη, καθοδήγηση και εμπιστοσύνη της κατά την διάρκεια εκπόνησης της Μεταπτυχιακής Διπλωματικής μου εργασίας.

Θα ήθελα επίσης να ευχαριστήσω το σύνολο των Καθηγητών και του προσωπικού του Μεταπτυχιακού προγράμματος «Διαδικτυωμένα Ηλεκτρονικά Συστήματα» του τμήματος Ηλεκτρολόγων & Ηλεκτρονικών Μηχανικών, της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, για τις πολύτιμες γνώσεις και πληροφορίες που μου μετέδωσαν κατά την διάρκεια της φοίτησης μου.

Ένα μεγάλο ευχαριστώ στην αδερφή μου Μαρία, για την στήριξη και βοήθεια που μου προσέφερε καθ' όλη τη διάρκεια του μεταπτυχιακού προγράμματος.

Τέλος, θα ήθελα να αφιερώσω τη Διπλωματική μου εργασία στη μνήμη των γονιών μου Αναστασίου και Άννας.

ΠΙΝΑΚΑΣ ΣΥΜΒΟΛΩΝ-ΑΚΡΩΝΥΜΙΩΝ-ΣΥΝΤΟΜΟΓΡΑΦΙΩΝ

AC	Approximate Computing
ACC	Apache Commons Collections
ADPCM	Adaptive Differential Pulse-Code Modulation
ALU	Arithmetic Logic Unit
AMBA	Advanced Microcontroller Bus Architecture
API	Application Programming Interface
ARM	Advanced RISC Machines
ATD	Address Transition Detection
BIPS	Billion Instructions Per Second
CCC	Crossbar Connected Cache
CISC	Complex Instruction Set Computer
CMOS	Complementary MOSFET
CPI	Clock Cycles Per Instruction
CPL	CMOS Pass Logic
CPSR	Current Program Status Register
CPU	Central Processing Unit
CVSL	Current Voltage Switch Logic
DFS	Dynamic Frequency Scaling
DMA	Direct Memory Access
DRAM	Dynamic Random-Access Memory
DVFS	Dynamic Voltage and Frequency Scaling
DVS	Dynamic Voltage Scaling
EEPROM	Electrically Erasable Programmable Read-Only Memory
EPROM	Erasable Programmable Read-Only Memory
ESP	Early design Stage Power and performance simulator
ESR	Equivalent Series Resistance
FET	Field-Effect Transistor
FPU	Floating-Point Unit
GPS	Global Positioning System
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HDMI	High-Definition Multimedia Interface
I/O	Input/Output

ILP	Instruction-Level Parallelism
ILPA	Instruction-Level Power Analysis
ISA	Instruction Set Architecture
JCF	Java Collection Framework
JCL	Java Collection Libraries
JVM	Java Virtual Machines
Lx	Level x cache
LC	Level Conversion
LCFF	Level Conversion Flip-Flop
LDO	Low Dropout Regulator
MCU	MicroController Unit
MIPS	Million Instructions Per Second
MOSFET	Metal-Oxide-Semiconductor Field Effect Transistor
MPEG	Moving Picture Experts Group
MSV	Multi-Scale Voltage
NMOS	N-Type Metal Oxide Semiconductor
OS	Operating System
PC	Program Counter
PCVSL	Precharged CVSL
PEEK	Proactive Energy-awarE development Kit
PLL	Phase Locked Loop
PM	Power Manager
PMOS	P-Type Metal Oxide Semiconductor
PMU	Power Management Unit
POR	Power-On Reset
PROM	Programmable ROM
QoS	Quality of Service
RAM	Random Access Memory
RC6	Rivest Cipher 6
RISC	Reduced Instruction Set Computer
ROM	Read Only Memory
RTOS	Real Time Operating System
SEEDS	Software Engineer's Energy-optimization Decision Support
SoC	System On a Chip
SPEC	Standard Performance Evaluation Corporation

SRAM	Static RAM
USB	Universal Serial Bus
VDC	Voltage-Down Converter
VLIW	Very Long Instruction Word
x86	Intel Instruction Set Architectures
XTEA	eXtended Tiny Encryption Algorithm

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΕΡΙΛΗΨΗ **3**
ABSTRACT **4**
ΕΥΧΑΡΙΣΤΙΕΣ **5**
ΠΙΝΑΚΑΣ ΣΥΜΒΟΛΩΝ-ΑΚΡΩΝΥΜΙΩΝ-ΣΥΝΤΟΜΟΓΡΑΦΙΩΝ **6**
ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ **9**
ΕΙΣΑΓΩΓΗ: **13**
ΚΕΦΑΛΑΙΟ 1: Ο CMOS Inverter και η κατανάλωση ισχύος **15**
1.1 Εισαγωγή 15
1.2 Ο αναστροφέας CMOS 15
1.3 Η λειτουργία του κυκλώματος 17
1.3.1 Τα περιθώρια θορύβου 19
1.4 Η κατανάλωση ισχύος στα ολοκληρωμένα κυκλώματα CMOS 21
1.4.1 Δυναμική κατανάλωση ισχύος 21
1.4.2 Στατική κατανάλωση ισχύος 22
1.4.3 Κατανάλωση ισχύος λόγω του ρεύματος βραχυκύκλωσης 23
1.5 Άλλες πηγές κατανάλωσης ισχύος 24
1.5.1 Ανεπιθύμητες μεταβάσεις 24
1.6 Οι παράγοντες που επηρεάζουν την κατανάλωση ενέργειας 25
1.6.1 Η σχέση της συχνότητας λειτουργίας και κατανάλωσης ενέργειας 25
1.6.2 Η σχέση της τάσης λειτουργίας και κατανάλωσης ενέργειας 27
1.6.3 Η σχέση της θερμοκρασίας λειτουργίας και κατανάλωσης ενέργειας 28
ΒΙΒΛΙΟΓΡΑΦΙΑ 1ου ΚΕΦΑΛΑΙΟΥ 30
ΚΕΦΑΛΑΙΟ 2: System On a Chip (SoC) και αρχιτεκτονικές επεξεργαστών **32**
2.1 Εισαγωγή 32
2.2 System on a Chip (SoC) 32
2.2.1 Γιατί SoC; 32
2.2.2 Πλεονεκτήματα SoC 34
2.2.3 SoC vs CPU 34
2.3 Αρχιτεκτονική RISC vs CISC 34
2.3.1 Σύγκριση ISA (Instruction Set Architecture) - η αρχιτεκτονική CISC 35
2.3.2 Η αρχιτεκτονική RISC 36
2.3.3 Σύγκριση απόδοσης 37
2.3.4 Ισχύς και κατανάλωση ενέργειας 37
2.3.5 Συμπεράσματα 38

2.4 Η αρχιτεκτονική ARM 38
2.4.1 The Acorn RISC Machine 39
2.4.2. Τα χαρακτηριστικά της αρχιτεκτονικής ARM 40
2.4.3. Οι καταχωρητές 40
2.4.4. Οι τρόποι διευθυνσιοδότησης 42
2.4.5. Το σετ των εντολών 43
ΒΙΒΛΙΟΓΡΑΦΙΑ 2ου ΚΕΦΑΛΑΙΟΥ 45
ΚΕΦΑΛΑΙΟ 3: Τεχνικές χαμηλής κατανάλωσης ισχύος σε επίπεδο hardware 47
3.1 Εισαγωγή 47
3.2 Βασικές αρχές σχεδίασης των τεχνικών μείωσης της κατανάλωσης ισχύος και παράγοντες που την επηρεάζουν 47
3.3 Μετρικές ισχύος και απόδοσης 52
3.4 Τεχνικές χαμηλής κατανάλωσης ισχύος σε κυκλωματικό και λογικό επίπεδο 53
3.4.1 Βελτιστοποίηση της τοπογραφίας (Layout optimization) 53
3.4.2 Ελαχιστοποίηση λογικής (Logic minimization) 54
3.4.3 Διάσπαση λογικού κυκλώματος (Logic decomposition) 55
3.4.4 Technology mapping 56
3.4.5 Η επίδραση της λογικής οικογένειας στην κατανάλωση ισχύος 56
3.4.6 Ελαχιστοποίηση των ανεπιθύμητων μεταβάσεων (Signal glitching) 58
3.4.7 Βελτιστοποίηση διαστάσεων των τρανζίστορ 58
3.4.8 Κυκλώματα CMOS πολλαπλής τάσης κατωφλίου(Multiple-Threshold) 59
3.4.9 Λογική με Πολλαπλές Τάσεις Τροφοδοσίας 64
3.4.10 Δυναμική Κλιμάκωση της Τάσης (Dynamic Voltage Scaling - DVS) 66
3.4.11 Δυναμική Προσαρμογή της Τάσης και της Συχνότητας (DynamicVoltage and Frequency Scaling - DVFS) 68
3.5 Τεχνικές χαμηλής κατανάλωσης ισχύος σε επίπεδο αρχιτεκτονικής 72
3.5.1 Τεχνική χαμηλής κατανάλωσης με απομόνωση της ισχύος (Power Gating) 72
3.5.2 Τεχνική χαμηλής κατανάλωσης με απενεργοποίηση του ρολογιού κατά τμήματα (Clock gating) 74
3.5.3 Η τεχνική των τμημάτων τάσης (Voltage islands) 78
3.5.4 Διασωλήνωση – Pipelining 81
3.5.5 Παραλληλισμός (Parallelism) 87
3.6 Η μονάδα διαχείρισης ενέργειας (Power Management Unit -PMU) 90
3.6.1 Εισαγωγή 90
3.6.2 Τα μπλοκ της μονάδας διαχείρισης ισχύος 90
3.6.3 Μονάδα διαχείρισης ενέργειας μικροεπεξεργαστή 92
3.6.4 Η μονάδα διαχείρισης ισχύος σε υλοποίηση low power SoC 93
3.7 Γραμμικοί ρυθμιστές τάσης 97

3.7.1 Ο Low Dropout Γραμμικός Ρυθμιστής 100
3.7.2 Έννοιες των Low-Dropout Regulators (LDO) 103
ΒΙΒΛΙΟΓΡΑΦΙΑ 3ου ΚΕΦΑΛΑΙΟΥ 113
ΚΕΦΑΛΑΙΟ 4: Μνήμες και μείωση της κατανάλωσης ισχύος 118
4.1 Εισαγωγή 118
4.2 Βασικές αρχές σχεδίασης των τεχνικών μείωσης της κατανάλωσης ισχύος 118
4.3 Το κελί μνήμης (memory cell) 119
4.4 Ταξινόμηση μνημών 120
4.5 Κριτήρια απόδοσης μνημών 122
4.6 Οι μνήμες DRAM και SRAM 123
4.6.1 Η κατανάλωση ισχύος στις μνήμες DRAM και SRAM 123
4.6.2 SRAM vs DRAM 124
4.6.3 Δυναμική RAM - DRAM 124
4.6.4 Κυκλώματα κελιών δυναμικής μνήμης ανάγνωσης-εγγραφής (DRAM) 125
4.6.5 Τεχνικές χαμηλής ισχύος DRAM 127
4.7 Στατική RAM - SRAM 129
4.7.1 Κυκλώματα κελιών Στατικής μνήμης ανάγνωσης-εγγραφής (SRAM) 129
4.7.2 Τεχνικές χαμηλής κατανάλωσης ισχύος SRAM 130
4.7.3 Μείωση χωρητικότητας 131
4.7.4 Η τεχνική Memory Bank Partitioning 132
4.7.5 Μείωση της ισχύος αναμονής (standby) 132
4.8 Οι μνήμες ROM 133
4.8.1 Κυκλώματα μνήμης μόνο για ανάγνωση (ROM) 134
4.8.2 FLASH memory 136
4.8.3 Flash Memory Architecture 136
4.8.4 Κατανάλωση ισχύος στις μνήμες flash 137
4.8.5 Τεχνικές χαμηλής κατανάλωσης ισχύος για τις μνήμες flash 138
4.9 Η CACHE memory 138
4.9.1 Αριθμός caches 139
4.9.2 Multilevel caches 140
4.9.3 Unified vs Split caches 141
4.9.4 Μείωση της κατανάλωσης ισχύος στις μνήμες cache 141
4.10 Μείωση της κατανάλωσης ισχύος με τη χρήση Direct Memory Access (DMA) 145
4.10.1 Εισαγωγή 145
4.10.2 DMA και χαμηλή κατανάλωση ισχύος 149
4.10.3 Πλεονεκτήματα της χρήσης DMA 149
ΒΙΒΛΙΟΓΡΑΦΙΑ 4ου ΚΕΦΑΛΑΙΟΥ 151
ΚΕΦΑΛΑΙΟ 5: Τεχνικές χαμηλής κατανάλωσης ισχύος σε επίπεδο software 154

5.1 Εισαγωγή 154
5.2 Power vs Energy 154
5.3 Power/Energy vs Performance 156
5.4 Οι παράγοντες κατανάλωσης της ισχύος 156
5.5 Η εκτίμηση της κατανάλωσης ισχύος του λογισμικού και η CPU 158
5.5.1 Εκτίμηση ισχύος σε επίπεδο πύλης 158
5.5.2 Εκτίμηση ισχύος σε επίπεδο αρχιτεκτονικής 158
5.5.3 Ανάλυση της ισχύος σε επίπεδο εντολής (Instruction-Level Power Analysis) 159
5.6 Η δραστηριότητα μεταγωγής διαύλου (Bus switching activity) 161
5.7 Low power τεχνικές στο λογισμικό 161
5.7.1 Οι βελτιστοποιήσεις στον κώδικα 161
5.7.2 Η βελτιστοποίηση του datapath bit-width 164
5.7.3 Instruction scheduling-based βελτιστοποίηση για τη μείωση της δραστηριότητας μεταγωγής 166
5.7.4 Compiler-based προσέγγιση για τη μείωση των cache misses 167
5.7.5 Κωδικοποίηση διαύλου (Bus encoding) 168
5.7.6 Επιλεκτική συμπίεση κώδικα, μία εναλλακτική προσέγγιση στην κωδικοποίηση διαύλου 169
5.7.7 Ελαχιστοποίηση του ενεργειακού κόστους των προσβάσεων στη μνήμη 170
5.8 Πρακτικές και μέθοδοι για την υλοποίηση low power λογισμικού 172
5.8.1 Παράλληλος προγραμματισμός (Parallel programming) 172
5.8.2 Approximate computing (AC) 176
5.8.3 Ανάλυση του πηγαίου κώδικα (Source code analysis) 178
5.8.4 Οι γλώσσες προγραμματισμού (Programming languages) 180
5.8.5 Δομές δεδομένων (Data structures) 184
5.8.6 Πρακτικές κωδικοποίησης (Coding practices) 186
5.8.7 Εργαλεία μέτρησης της καταναλισκόμενης ισχύος και ενέργειας 188
5.8.8 Η συντήρηση του λογισμικού (maintenance) 191
5.9 Το λειτουργικό σύστημα (OS), οι εφαρμογές (applications) και η κατανάλωση ενέργειας 194
5.9.1 Χρονοπρογραμματισμός (Scheduling) 194
5.9.2 Ο διαχειριστής ενέργειας (Power Manager – PM) 195
5.9.3 Εφαρμογές (Applications) 196
5.10 Real Time Operating System (RTOS) - Το λειτουργικό σύστημα πραγματικού χρόνου 197
5.10.1 Τα χαρακτηριστικά του RTOS 197
5.10.2 Οι καταστάσεις ισχύος (Power states) 198
5.10.3 RTOS low power χρονοπρογραμματισμός 199
ΒΙΒΛΙΟΓΡΑΦΙΑ 5ου ΚΕΦΑΛΑΙΟΥ 203
ΚΕΦΑΛΑΙΟ 6: Συμπεράσματα 206

ΕΙΣΑΓΩΓΗ

Στη παρούσα διατριβή, παρουσιάζονται εκείνες οι προσεγγίσεις και τεχνικές σε επίπεδο hardware και software, που έχουν σαν σκοπό την μείωση της κατανάλωσης ισχύος των κυκλωμάτων. Η κατανόηση των μηχανισμών κατανάλωσης ισχύος είναι κρίσιμη για την ανάπτυξη τέτοιων τεχνικών τόσο στο επίπεδο του κυκλώματος, της λογικής, της αρχιτεκτονικής όσο και στο επίπεδο του λογισμικού.

Στόχος της εργασίας είναι η παρουσίαση και παροχή πληροφοριών για τους περιορισμούς, τις δυνατότητες και τα tradeoffs που σχετίζονται με την ενεργειακή απόδοση των προσεγγίσεων και τεχνικών, καθώς και να καταδειχθεί η αλληλένδετη και αλληλεπιδραστική σχέση που υπάρχει μεταξύ τους στο πρόβλημα της μείωσης της κατανάλωσης ενέργειας. Το ενεργειακά αποδοτικό hardware, δεν σημαίνει κατ' ανάγκη ότι αποδίδει το αναμενόμενο ενεργειακό όφελος εάν το λογισμικό είναι κακογραμμένο και το αντίθετο.

Με δεδομένο ότι η δυναμική κατανάλωση αποτελεί τη σημαντικότερη παράμετρο στην κατανάλωση ενέργειας των κυκλωμάτων CMOS, χωρίς ταυτόχρονα να αγνοείται η στατική κατανάλωση που τα τελευταία χρόνια με την ολοένα αυξανόμενη ολοκλήρωση έχει γίνει και αυτή σημαντικός παράγοντας, παρουσιάζονται τεχνικές μείωσης της κατανάλωσης ισχύος στο κυκλωματικό και λογικό επίπεδο, καθώς και στο επίπεδο της αρχιτεκτονικής. Ξεχωριστό κεφάλαιο αφιερώνεται στις τεχνικές μείωσης της κατανάλωσης ισχύος των μνημών, οι οποίες αποτελούν έναν από τους μεγαλύτερους καταναλωτές, με μερίδιο έως και 25% του συνόλου της.

Όσον αφορά την κατανάλωση ισχύος του λογισμικού, η σύνταξη βιώσιμου, αποδοτικού και πράσινου λογισμικού (green software), απαιτεί την κατανόηση της συμπεριφοράς κατανάλωσης ισχύος ενός προγράμματος. Οι τεχνικές βελτιστοποίησης, αφορούν στις διαδρομές των δεδομένων, στους διαύλους του συστήματος, στις προσβάσεις στη μνήμη, σε τεχνικές μεταγλωττιστών καθώς και σε πρακτικές όπως ο παράλληλος προγραμματισμός και το approximate computing. Αναφορά γίνεται στο λειτουργικό σύστημα πραγματικού χρόνου RTOS - Real Time Operating System και στις τεχνικές διαχείρισης της ενέργειας σε επίπεδο λειτουργικού συστήματος με τον χρονοπρογραμματισμό των διαδικασιών.

Η εργασία αυτή είναι χωρισμένη σε έξι κεφάλαια. Αρχικά παρουσιάζονται οι στόχοι της εργασίας καθώς και η δομή της διατριβής.

Στο 1ο κεφάλαιο, παρουσιάζεται ο αναστροφέας CMOS, ο οποίος αποτελεί τον θεμέλιο λίθο της σχεδίασης των ψηφιακών ολοκληρωμένων κυκλωμάτων CMOS και αναλύεται η λειτουργία του. Ακολουθούν οι περιγραφές των καταναλώσεων ισχύος του αναστροφέα CMOS, καθώς και της σχέσης των παραγόντων της συχνότητας, της τάσης και της θερμοκρασίας με τη κατανάλωση ισχύος.

Στο 2ο κεφάλαιο, αναλύεται η αρχιτεκτονική SoC (System On a Chip) και παρουσιάζονται οι ISA αρχιτεκτονικές CISC και RISC. Ακολούθως, γίνεται η παρουσίαση του RISC επεξεργαστή ARM και των κύριων χαρακτηριστικών του.

Στο 3ο κεφάλαιο, παρουσιάζονται οι τεχνικές χαμηλής κατανάλωσης ισχύος σε επίπεδο hardware, κατηγοριοποιημένες στις τεχνικές σε κυκλωματικό και λογικό επίπεδο όπως οι τεχνικές DVS και DVFS και στις τεχνικές σε επίπεδο αρχιτεκτονικής όπως το Power gating, το Clock gating και η τεχνική των Voltage islands. Εκτενής αναφορά γίνεται στους LDO ρυθμιστές τάσης και τα χαρακτηριστικά τους.

Στο 4ο κεφάλαιο, αναπτύσσονται οι τεχνικές μείωσης της κατανάλωσης ισχύος σε διάφορους τύπους μνημών, οι οποίες με τη διαρκώς αυξανόμενη ζήτηση για μεγαλύτερη χωρητικότητα αποθήκευσης, απαιτούν πιο συμπαγείς υλοποιήσεις και κατά συνέπεια έχουν σημαντική συμβολή στην αύξηση της κατανάλωσης ισχύος.

Στο 5ο κεφάλαιο, περιγράφονται οι τεχνικές χαμηλής κατανάλωσης ισχύος σε επίπεδο software. Γίνεται αναφορά στους παράγοντες της κατανάλωσης της ισχύος, όπως οι δίαυλοι συστήματος, οι διαδρομές των δεδομένων και οι προσβάσεις στη μνήμη. Αναφέρονται επίσης τρόποι εκτίμησης της κατανάλωσης ισχύος του λογισμικού σε επίπεδο πύλης, αρχιτεκτονικής και εντολής, παρουσιάζονται βελτιστοποιήσεις στον κώδικα, καθώς και πρακτικές – μέθοδοι για τη μείωση της κατανάλωσης ισχύος όπως το approximate computing και ο παράλληλος προγραμματισμός. Αναφορά γίνεται στο λειτουργικό σύστημα πραγματικού χρόνου RTOS – Real Time Operating System και στα χαρακτηριστικά του.

Τέλος, η διατριβή ολοκληρώνεται με το κεφάλαιο των συμπερασμάτων.

ΚΕΦΑΛΑΙΟ 1:
Ο CMOS Inverter και η κατανάλωση ισχύος

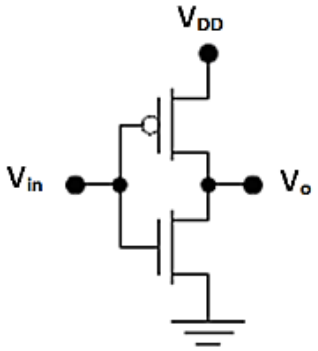
1.1 Εισαγωγή

Η τεχνολογία CMOS (Complementary MOSFET), χρησιμοποιείται ευρέως σήμερα για τη δημιουργία κυκλωμάτων σε πολλές και ποικίλες εφαρμογές. Οι σημερινοί επεξεργαστές των υπολογιστών και τα κινητά τηλέφωνα, χρησιμοποιούν την CMOS υλοποίηση λόγω πολλών βασικών πλεονεκτημάτων που προσφέρει, όπως χαμηλή κατανάλωση ισχύος, σχετικά υψηλή ταχύτητα, υψηλά περιθώρια θορύβου και στις δύο καταστάσεις και λειτουργεί σε ένα ευρύ φάσμα τάσεων πηγής και εισόδου. Στο Κεφάλαιο 1 παρουσιάζεται ο θεμέλιος λίθος της σχεδίασης των ψηφιακών ολοκληρωμένων κυκλωμάτων CMOS, ο αναστροφέας CMOS και αναλύεται η λειτουργία του. Ακολουθεί η περιγραφή των καταναλώσεων ισχύος του αναστροφέα CMOS, όπως είναι η δυναμική, η στατική, καθώς και η κατανάλωση ισχύος που οφείλεται στο ρεύμα βραχυκύκλωσης και στη συνέχεια περιγράφεται η σχέση των παραγόντων της συχνότητας, της τάσης και της θερμοκρασίας με τη κατανάλωση ισχύος.

1.2 Ο αναστροφέας CMOS

Ο αναστροφέας CMOS, αποτελεί το βασικό τμήμα των ψηφιακών ολοκληρωμένων κυκλωμάτων CMOS. Το χαρακτηριστικό της μετάβασης (switching) του αναστροφέα CMOS, καθορίζει και τη ταχύτητα διάδοσης του σήματος των CMOS ψηφιακών κυκλωμάτων.

Η δομή του αναστροφέα CMOS, αποτελείται από ένα ζεύγος συμπληρωματικών MOSFET (ενός τρανζίστορ εμπλουτισμού NMOS και ενός PMOS τρανζίστορ εμπλουτισμού, επειδή αυτός ο τύπος MOSFET έχει καλύτερες επιδόσεις σε σύγκριση με τον τύπο αραίωσης), τα οποία λειτουργούν σε συμπληρωματική λειτουργία. Το κύκλωμα του αναστροφέα CMOS εμφανίζεται στο σχήμα 1.1. [1]

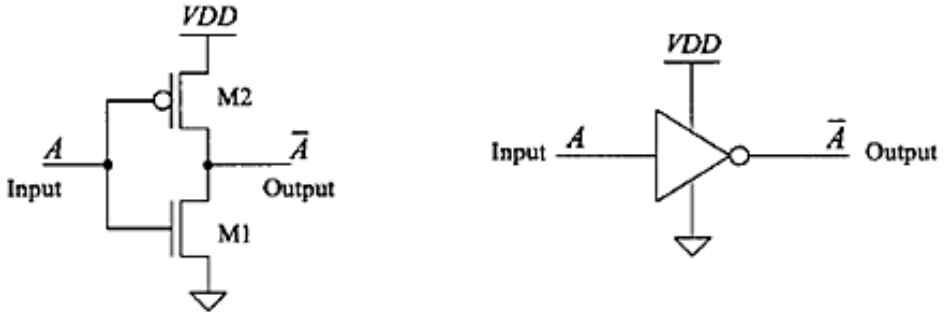


Σχήμα 1.1. Ο αναστροφέας CMOS.

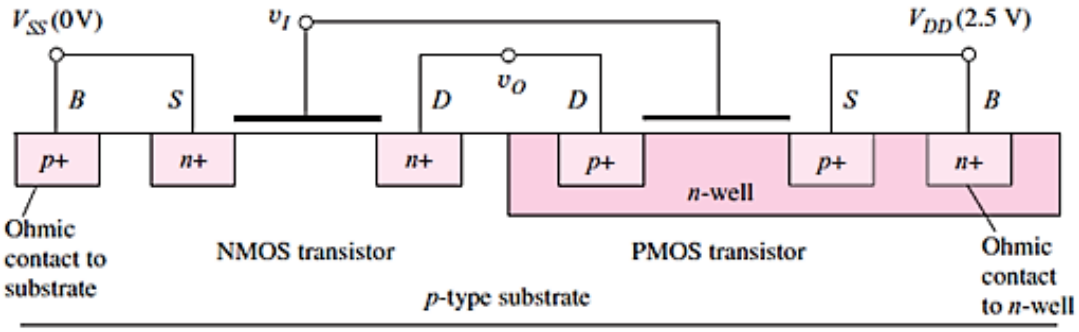
Από το σχήμα 1.2, φαίνεται ότι ο μετατροπέας εκτελεί τη λογική λειτουργία της αναστροφής του σήματος εισόδου A. Όταν η είσοδος του μετατροπέα συνδέεται στη γείωση, η έξοδος γίνεται V_{DD} μέσω του PMOS M2 το οποίο έρχεται σε κατάσταση on (το M1 είναι σε κατάσταση off). Όταν η είσοδος συνδεθεί στη V_{DD}, το δυναμικό της εξόδου γίνεται ίσο με αυτό της γης, μέσω του τρανζίστορ NMOS M1 το οποίο βρίσκεται σε κατάσταση on (το M2 βρίσκεται σε κατάσταση off).

Ο μετατροπέας CMOS έχει πολλά θετικά χαρακτηριστικά όπως για παράδειγμα σε αντίθεση με άλλες λογικές οικογένειες που δεν φτάνουν σε τέτοια επίπεδα τροφοδοσίας, η τάση εξόδου του που μπορεί να κυμαίνεται από τη γείωση έως τη V_{DD}. Επίσης, η στατική κατανάλωση ισχύος του μετατροπέα CMOS που είναι πρακτικά μηδέν, η ελάχιστη ευαισθησία στο θόρυβο, όπως και το switching κατώφλι που μπορεί να ρυθμιστεί αλλάζοντας το μέγεθος του τρανζίστορ. [2]

Δεδομένου ότι τα τρανζίστορ nMOS και pMOS πρέπει να κατασκευάζονται στο ίδιο ic, η διαδικασία κατασκευής CMOS είναι πιο περίπλοκη από την τυπική διαδικασία μόνο για nMOS. Συγκεκριμένα, η διαδικασία CMOS πρέπει να παρέχει ένα υπόστρωμα τύπου n για τα τρανζίστορ pMOS και έναν τύπο p υποστρώματος για τα τρανζίστορ nMOS. Αυτό μπορεί να επιτευχθεί με την κατασκευή είτε φρεατίων(wells) τύπου n σε wafer τύπου p, ή κατασκευάζοντας φρεάτια τύπου p σε wafer τύπου n (Σχήμα 1.3). [3], [4]



Σχήμα 1.2. Η λογική λειτουργία του CMOS αναστροφέα.



Σχήμα 1.3. Η κατασκευαστική δομή του CMOS inverter.

1.3 Η λειτουργία του κυκλώματος

Στο σχήμα 4, η τάση εισόδου συνδέεται στη πύλη και του nMOS και του pMOS τρανζίστορ. Έτσι, και τα δύο τρανζίστορ οδηγούνται απευθείας από τη τάση εισόδου Vin. Το υπόστρωμα του τρανζίστορ nMOS είναι συνδεδεμένο στο γη, ενώ το υπόστρωμα του τρανζίστορ του pMOS συνδέεται με την τάση V_{DD}, προκειμένου να πολωθούν ανάστροφα οι επαφές πηγής (source) και υποδοχής (drain).

Από το διάγραμμα του κυκλώματος στο σχήμα 1.4 έχουμε:

$V_{GS},n=Vin$, $V_{DS},n=Vout$ και,

$V_{GS},p=-(V_{DD}-Vin)$, $V_{DS},p=-(V_{DD}-Vout)$

Θα ξεκινήσουμε την ανάλυση εξετάζοντας δύο περιπτώσεις:

Όταν η τάση εισόδου είναι μικρότερη από την τάση κατωφλίου του nMOS, δηλαδή όταν $Vin<V_{T0},n$ το τρανζίστορ nMOS είναι σε αποκοπή. Ταυτόχρονα, το τρανζίστορ pMOS είναι ενεργοποιημένο και λειτουργεί στη γραμμική περιοχή.

Δεδομένου ότι τα drain ρεύματα και των δύο τρανζίστορ είναι περίπου ίσα με το μηδέν (εκτός από το μικρά ρεύματα διαρροής), δηλαδή, $I_D,n = I_D,p = 0$ η τάση V_{DS} του τρανζίστορ pMOS είναι επίσης ίση με το μηδέν και η τάση εξόδου V_{OH}, είναι ίση με την τάση τροφοδοσίας.

$Vout = V_{OH} = V_{DD}$

Στη δεύτερη περίπτωση, όταν η τάση εισόδου υπερβαίνει ($V_{DD} + V_{T0},p$), το τρανζίστορ pMOS είναι απενεργοποιημένο. Σε αυτήν την περίπτωση, το τρανζίστορ nMOS λειτουργεί στη γραμμική περιοχή, αλλά η τάση V_{DS} είναι ίση με το μηδέν. Κατά συνέπεια, η τάση εξόδου του κυκλώματος είναι:

$Vout= V_{OL} =0$

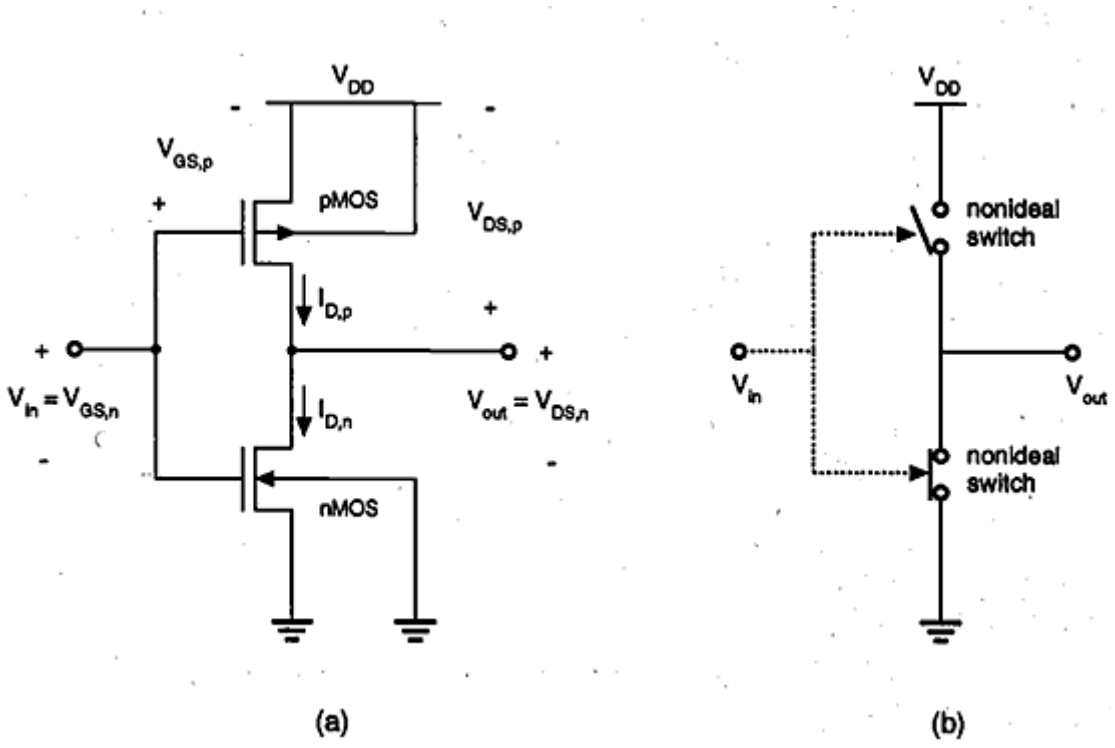
Στη συνέχεια, εξετάζεται ο τρόπος λειτουργίας των τρανζίστορ nMOS και pMOS σε σχέση με τις τάσεις εισόδου και εξόδου. Το τρανζίστορ nMOS λειτουργεί στον κορεσμό εάν Vin> $V_{T0,n}$ και εάν ισχύουν τα ακόλουθα:

$V_{DS,n} >= V_{GS,n} - V_{T0,n} \Leftrightarrow Vout >= Vin - V_{T0,n}$

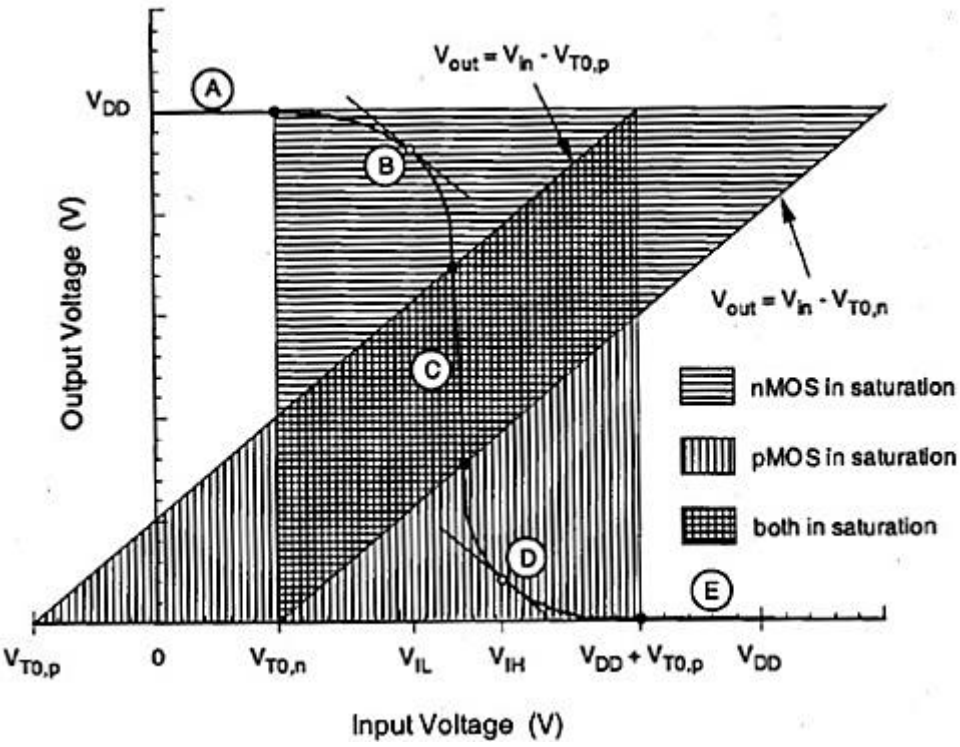
Το pMOS τρανζίστορ λειτουργεί στον κορεσμό εάν Vin<(V_{DD}+$V_{T0,p}$) και εάν ισχύουν τα ακόλουθα:

$V_{DS,p} <= V_{GS,p} - V_{T0,p} \Leftrightarrow Vout <= Vin - V_{T0,p}$

Όλα τα πιό πάνω φαίνονται στο σχήμα 1.5 και στον πίνακα 1.1, με τις σκιαγραφημένες περιοχές της χαρακτηριστικής καμπύλης μετάβασης τάσης Vout–Vin. [3]



Σχήμα 1.4. (a) Το κύκλωμα του CMOS inverter (b) απλοποιημένο κύκλωμα.



Σχήμα 1.5. Περιοχές λειτουργίας του CMOS

Πίνακας 1.1. Περιοχές λειτουργίας και αντίστοιχα επίπεδα τάσεων Vin – Vout.

Region	V_{in}	V_{out}	nMOS	pMOS
A	$< V_{T0,n}$	V_{OH}	cut-off	linear
B	V_{IL}	high $\approx V_{OH}$	saturation	linear
C	V_{th}	V_{th}	saturation	saturation
D	V_{IH}	low $\approx V_{OL}$	linear	saturation
E	$>\left(V_{DD} + V_{T0,p}\right)$	V_{OL}	linear	cut-off

1.3.1 Τα περιθώρια θορύβου

Το περιθώριο θορύβου είναι μια σημαντική παράμετρος στο λογική σχεδίαση. Ορίζεται ως η επιτρεπόμενη τάση θορύβου στην είσοδο έτσι ώστε να μην επηρεάζεται η

έξοδος. Τα επίπεδα λογικής μπορούν να εξαχθούν από τα DC χαρακτηριστικά. Όπως απεικονίζεται στο σχήμα 1.6 ορίζουμε τα επίπεδα λογικής σην είσοδο από:

- Λογική 0: για $0 <= V_{IN} <= V_{IL}$
- Λογική 1: για $V_{IH} <= V_{IN} <= V_{DD}$

και στην έξοδο από:

- Λογική 0: για $0 <= V_O <= V_{OL}$
- Λογική 1: για $V_{OH} <= V_O <= V_{DD}$

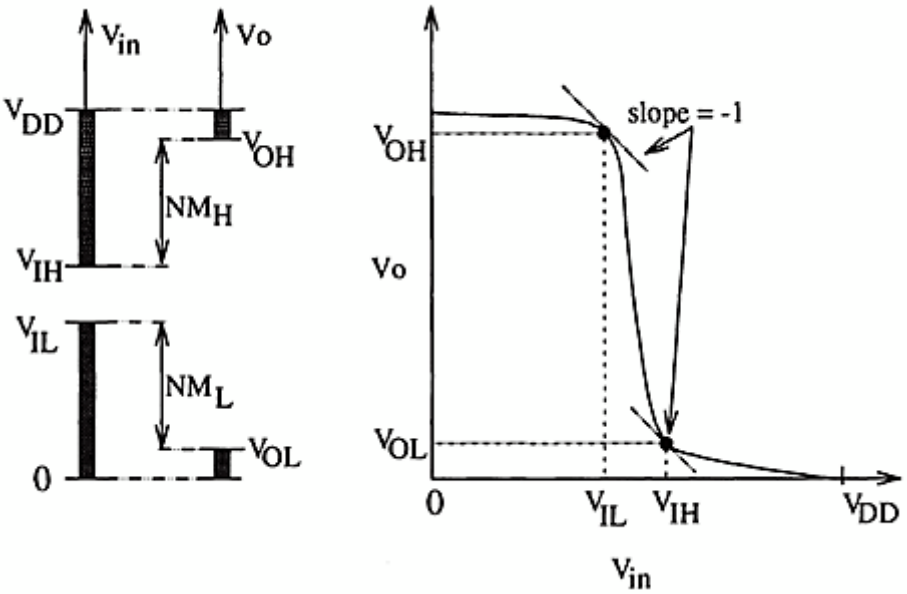
Το χαμηλό περιθώριο θορύβου ορίζεται από:

$NM_L = |V_{IL}-V_{OL}|$

και το υψηλό περιθώριο θορύβου ορίζεται από:

$N_{MH}=|V_{OH}-V_{IH}|$

Για την ύπαρξη καλών περιθωρίων θορύβου, είναι επιθυμητό τα V_{IL} και V_{IH} να είναι κοντά το ένα στο άλλο και γύρω από το σημείο $V_{DD} / 2$. [12]



Σχήμα 1.6. Ορισμός περιθωρίων θορύβου.

1.4 Η κατανάλωση ισχύος στα ολοκληρωμένα κυκλώματα CMOS

Η κατανάλωση ισχύος είναι ένα από τα κυριότερα προβλήματα προς αντιμετώπιση. Ο εντοπισμός των πηγών κατανάλωσης ισχύος είναι κρίσιμος για την ανάπτυξη τεχνικών χαμηλής κατανάλωσης κατά την εργοστασιακή κατασκευή τους τόσο στο επίπεδο του κυκλώματος όσο και στο επίπεδο της αρχιτεκτονικής. Υπάρχουν τρεις πηγές κατανάλωσης ισχύος στα κυκλώματα CMOS, η δυναμική κατανάλωση, η στατική κατανάλωση ή κατανάλωση ρευμάτων διαρροής(leakage) και η κατανάλωση ρεύματος βραχυκυκλώματος. Η συνολική κατανάλωση ισχύος ενός κυκλώματος CMOS είναι:

$$P_{total} = P_{dynamic} + P_{leakage} + P_{short\text{-}circuit}$$

όπου $P_{dynamic}$ είναι η δυναμική κατανάλωση που οφείλεται στη δραστηριότητα του κυκλώματος κατά την οποία φορτίζονται και εκφορτίζονται οι παρασιτικές χωρητικότητες κατά τη διάρκεια των μεταβάσεων του σήματος. Η $P_{leakage}$ οφείλεται σε διαρροές ρεύματος και εξακολουθεί να υφίσταται ακόμη και όταν δεν υπάρχει δραστηριότητα στο κύκλωμα (π.χ. είναι σε κατάσταση αναμονής). Η $P_{short\text{-}circuit}$ είναι η ισχύς που καταναλώνεται κατά τη μετάβαση ενός σήματος εισόδου όπου και τα PMOS και NMOS τρανζίστορς μιας πύλης CMOS άγουν ταυτόχρονα.

1.4.1 Δυναμική κατανάλωση ισχύος

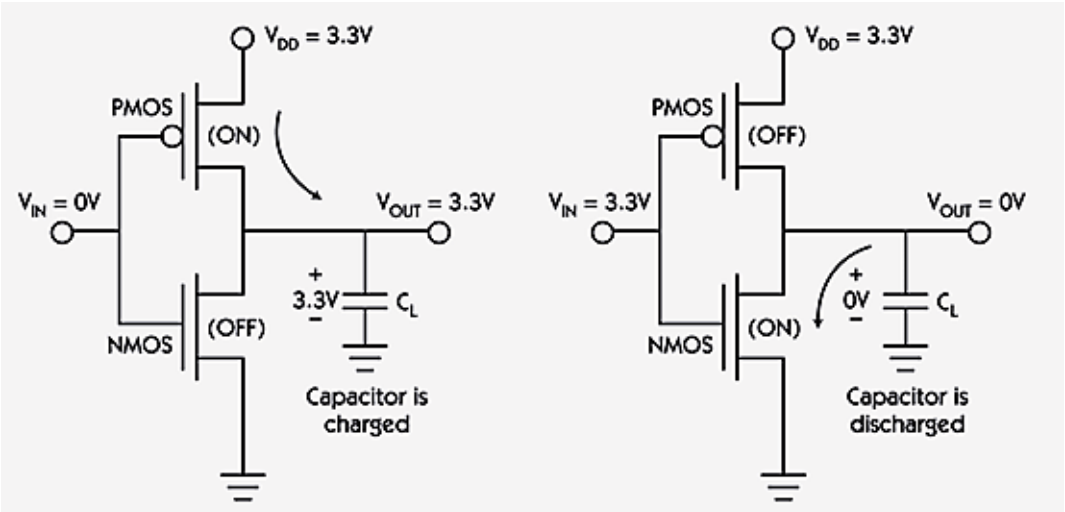
Μιά σημαντική πηγή κατανάλωσης ισχύος σε ένα τυπικό κύκλωμα CMOS είναι η δυναμική κατανάλωση. Η δυναμική κατανάλωση οφείλεται στην φόρτιση και αποφόρτιση των παρασιτικών χωρητικοτήτων κατά τη διάρκεια μεταβάσεων της τάσης στους κόμβους ενός κυκλώματος CMOS.

Η δυναμική κατανάλωση ισχύος είναι ανεξάρτητη από τον τύπο της πύλης όπου γίνεται η μετάβαση και του σχήματος της κυματομορφής εισόδου (χρόνοι ανόδου και καθόδου του σήματος εισόδου). Εξαρτάται όμως από την τάση τροφοδοσίας, τη συχνότητα μετάβασης και την ισοδύναμη χωρητικότητα στον κόμβο μετάβασης (Σχήμα 1.7). [5]

Κατά την φόρτιση της χωρητικότητας εξόδου, η μισή από την ενέργεια που αντλείται από την τροφοδοσία αποθηκεύεται στον πυκνωτή, ενώ η υπόλοιπη καταναλώνεται σαν θερμότητα στο PMOS. Στην συνέχεια κατά την εκφόρτιση του πυκνωτή, η αποθηκευμένη σε αυτόν ενέργεια καταναλώνεται σαν θερμότητα στο NMOS. Να σημειωθεί ότι η ποσότητα ενέργειας που καταναλώνεται στο PMOS και NMOS είναι πάντοτε η ίδια, ανεξάρτητα από τις διαστάσεις τους (στην περίπτωση αυτή αγνοείται η επίδραση των παρασιτικών χωρητικοτήτων, οι οποίες αυξάνουν με την αύξηση των διαστάσεων των τρανζίστορ).

Η πιο κάτω σχέση δίνει τη δυναμική κατανάλωση ισχύος στην απλή περίπτωση ενός αντιστροφέα CMOS, ο οποίος εκτελεί μεταβάσεις με τη συχνότητα του ρολογιού. Οι μεταβάσεις ενός κυκλώματος ποτέ δεν γίνονται σύμφωνα με τη συχνότητα του ρολογιού, αλλά με ρυθμό σημαντικά μικρότερο. Στην περίπτωση αυτή η δυναμική κατανάλωση ισχύος θα είναι: [6]

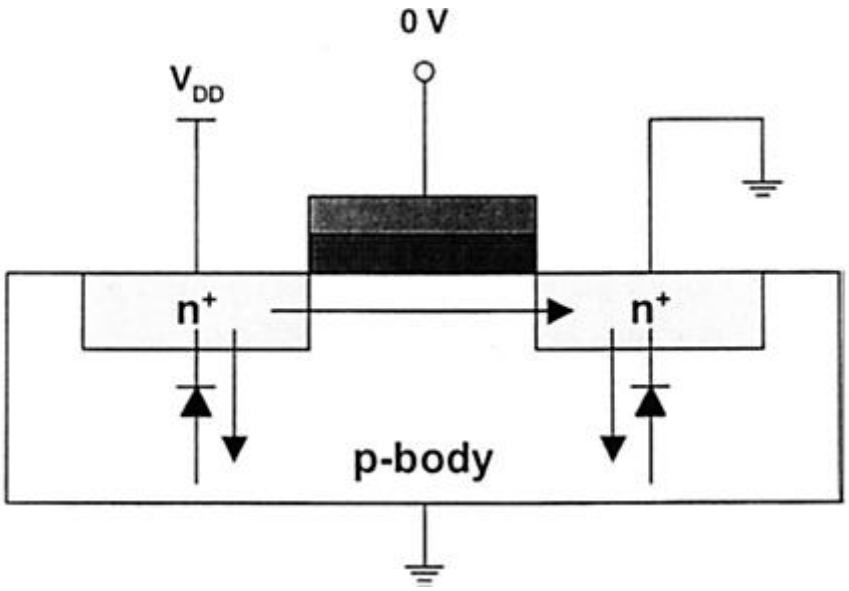
$$P_{dynamic} = C_L * V^2_{DD} * f$$



Σχήμα 1.7. Φόρτιση και εκφόρτιση των παρασιτικών χωρητικοτήτων.

1.4.2 Στατική κατανάλωση ισχύος

Στην ιδανική περίπτωση, η κατανάλωση ισχύος σε μια συμπληρωματική πύλη CMOS είναι μηδενική σε συνθήκες ηρεμίας καθώς πάντοτε ένα από τα δύο τρανζίστορ βρίσκεται σε αποκοπή. Στην πραγματικότητα, όμως, υπάρχει μια μικρή στατική κατανάλωση η οποία προέρχεται τόσο από τα ρεύματα διαρροής των τρανζίστορ που βρίσκονται στην περιοχή της αποκοπής, ή την περιοχή υποκατωφλίου (subthreshold) όπως είναι αλλιώς γνωστή, όσο και από τα αντίστοιχα ρεύματα των ανάστροφα πολωμένων παρασιτικών διόδων που σχηματίζονται ανάμεσα στις περιοχές πηγής-υποδοχής και στο υπόστρωμα κάθε τρανζίστορ, όπως φαίνεται στο ακόλουθο σχήμα 1.8.



Σχήμα 1.8. Μηχανισμοί δημιουργίας ρευμάτων διαρροής.

Το άθροισμα των ρευμάτων ανάστροφης πόλωσης και υποκατωφλίου δίνει το συνολικό ρεύμα διαρροής της πύλης, οπότε η στατική κατανάλωση διαρροής σε συνθήκες ηρεμίας μπορεί να θεωρηθεί σταθερή ως προς το χρόνο.

Τα ρεύματα και η στατική ισχύς διαρροής εξαρτώνται κυρίως από τις παραμέτρους της τεχνολογίας ενώ το μέγεθος τους είναι αρκετά μικρό, πράγμα που σημαίνει ότι ο ρόλος τους θα περιορίζεται αποκλειστικά και μόνο στην περίπτωση όπου το κύκλωμα βρίσκεται σε ηρεμία και δεν καταναλώνει δυναμική ισχύ, ενώ θα μπορούν να αγνοούνται με ασφάλεια κατά τις μεταβάσεις μεταξύ λογικών καταστάσεων. Πάντως, το σχετικό μέγεθος των στατικών ποσοτήτων αυξάνεται με τη μείωση των τάσεων τροφοδοσίας και κατωφλίου.

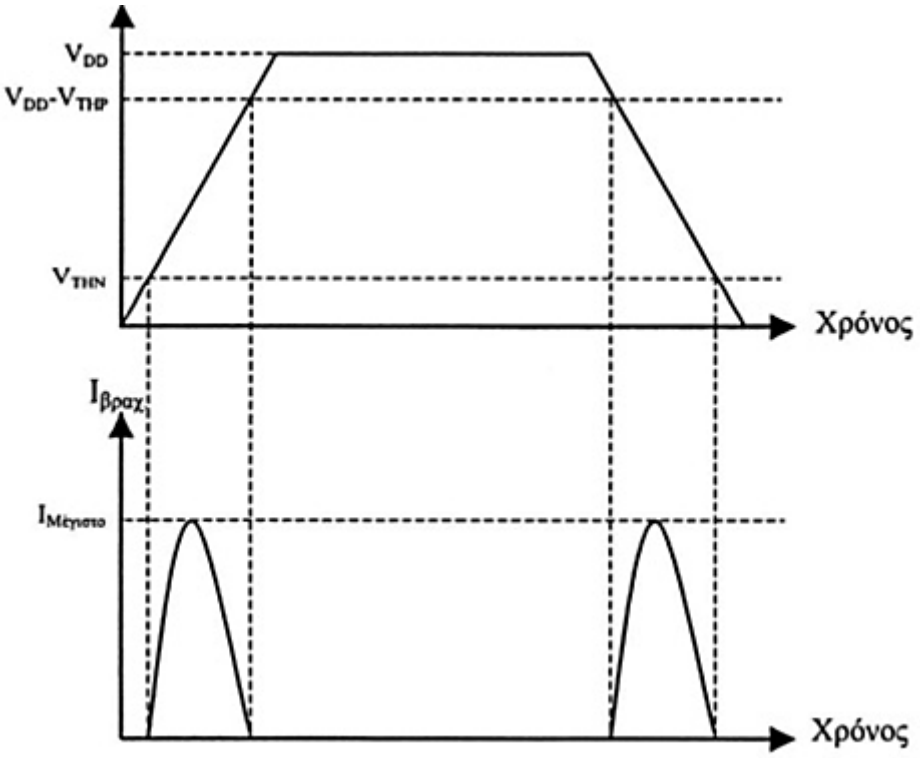
1.4.3 Κατανάλωση ισχύος λόγω του ρεύματος βραχυκύκλωσης

Τη χρονική διάρκεια κατά την οποία η είσοδος ενός κυκλώματος CMOS βρίσκεται στη μεταβατική κατάσταση, τα NMOS και τα PMOS τρανζίστορ άγουν ταυτόχρονα, προκαλώντας ένα ρεύμα βραχυκυκλώματος από την τροφοδοσία προς τη γη. Το συγκεκριμένο ρεύμα παρουσιάζεται όσο η τάση εισόδου βρίσκεται μεταξύ των τιμών Vtn, Vdd-Vtp, όπως φαίνεται στο σχήμα 1.9.

Το ρεύμα βραχυκυκλώματος γίνεται ιδιαίτερα σημαντικό όταν ο χρόνος ανόδου/καθόδου του παλμού εισόδου είναι μεγάλος. Στην περίπτωση αυτή ο αγώγιμος δρόμος από την τροφοδοσία προς τη γη υφίσταται για περισσότερο χρόνο. Το ρεύμα βραχυκυκλώματος μπορεί να ελαχιστοποιηθεί σημαντικά με τη κατάλληλη κυκλωματική σχεδίαση, ώστε οι χρόνοι ανόδου/καθόδου του παλμού εισόδου να είναι

μικρότεροι ή τουλάχιστον ίσοι με τους χρόνους ανόδου/καθόδου του παλμού εξόδου. Υπό αυτή την συνθήκη, η κατανάλωση ισχύος λόγω

ρεύματος βραχυκυκλώματος μπορεί είναι μικρότερη από το 10% της συνολικής δυναμικής κατανάλωσης ισχύος. [6]



Σχήμα 1.9. Ρεύμα βραχυκυκλώματος.

1.5 Άλλες πηγές κατανάλωσης ισχύος

1.5.1 Ανεπιθύμητες μεταβάσεις

Κατά την ανάλυση της πιθανότητας μετάβασης ενός κόμβου, αγνοήθηκε το γεγονός ότι μια λογική πύλη εμφανίζει μη μηδενική καθυστέρηση διάδοσης του σήματος. Κατά συνέπεια, είναι δυνατόν σε μια λογική πύλη οδηγούμενη από άλλες, να εμφανίζεται διαφορά στους χρόνους άφιξης των σημάτων στις εισόδους της. Η χρονική απόκλιση των σημάτων εισόδου είναι δυνατόν να προκαλέσει επιπλέον μεταβάσεις σε έναν κόμβο, προτού η τάση σε αυτόν αποκατασταθεί στην τελική λογική τιμή της. Οι μεταβάσεις αυτές καλούνται ανεπιθύμητες ή ψευδείς μεταβάσεις (στα αγγλικά

δηλώνονται ως: spurious transitions, glitches, dynamic hazards) και συνεισφέρουν στην αύξηση της ολικής κατανάλωσης της ισχύος.

Αν η χρονική διάρκεια μιας ανεπιθύμητης μετάβασης είναι μεγάλη, είναι δυνατόν να διαδοθεί στις επόμενες βαθμίδες. Αυτό έχει σαν συνέπεια σε κυκλώματα μεγάλου λογικού βάθους (logical depth) να αυξάνεται η εμφάνιση ανεπιθύμητων μεταβάσεων για δύο λόγους: επειδή η ύπαρξη μεγάλου λογικού βάθους συνεπάγεται μεγάλες χρονικές αποκλίσεις μεταξύ των σημάτων και γιατί οι ανεπιθύμητες μεταβάσεις διαδίδονται στους γειτονικούς κόμβους (μέχρι να συναντήσουν κύκλωμα ελεγχόμενο από ρολόι. [6]

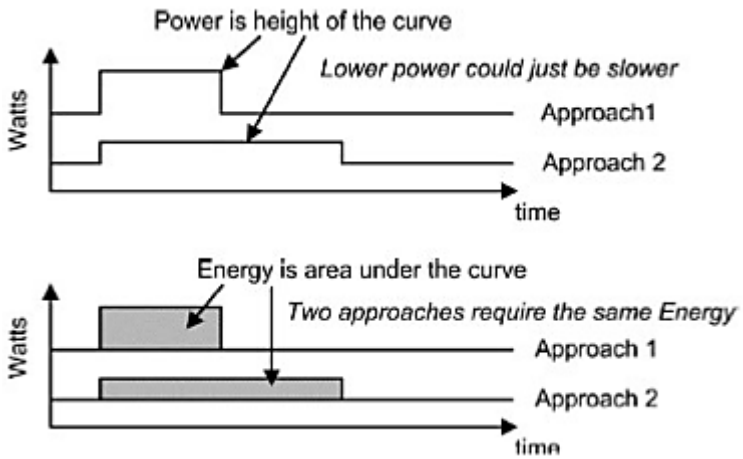
1.6 Οι παράγοντες που επηρεάζουν την κατανάλωση ενέργειας

Στη CMOS τεχνολογία η δυναμική κατανάλωση ισχύος αποτελεί μια σημαντική παράμετρο και οφείλεται κυρίως στο ρεύμα που απαιτείται για να φορτιστούν οι εσωτερικές χωρητικότητες και σε μικρότερο βαθμό από μικρά ρεύματα που εμφανίζονται κατά τη διάρκεια ενός γεγονότος μετάβασης (switching event). Σύμφωνα με την εξίσωση $P_{active}=C*V^2*F$(1.1), η μείωση της δυναμικής κατανάλωσης μπορεί να προκύψει από μείωση της συχνότητας του ρολογιού, της χωρητικότητας ή της τάσης τροφοδοσίας. Δυστυχώς όλες αυτές οι μέθοδοι υποβαθμίζουν την ταχύτητα διάδοσης του σήματος και την υπολογιστική απόδοση. Η θερμοκρασία λειτουργίας είναι επίσης ένας παράγοντας που πρέπει να λαμβάνεται υπόψη. Οι προδιαγραφές των ολοκληρωμένων κυκλωμάτων που δίνουν οι κατασκευαστές, αναφέρουν ένα συγκεκριμένο εύρος λειτουργίας θερμοκρασιών. Είναι γνωστό ότι με την αύξηση της θερμοκρασίας, η κατανάλωση ισχύος μπορεί να αυξηθεί δραματικά και καθιστά την αποτελεσματικότητα της μονάδας ψύξης καθοριστικό παράγοντα της ενέργειας που μπορεί να διαχειριστεί το κύκλωμα. [5]

1.6.1 Η σχέση της συχνότητας λειτουργίας και κατανάλωσης ενέργειας

Όπως αναφέρθηκε, η συχνότητα ρολογιού είναι ένας παράγοντας που επηρεάζει την κατανάλωση ενέργειας. Σε πολλές περιπτώσεις, με μικρότερη συχνότητα ρολογιού, η κατανάλωση ενέργειας μειώνεται και αυξάνεται ο χρόνος ανταπόκρισης του κυκλώματος και αντίστροφα για υψηλότερες συχνότητες ρολογιού. Τόσο η ταχύτητα διάδοσης του σήματος, όσο και η δυναμική κατανάλωση ενέργειας διπλασιάζεται περίπου όταν διπλασιάζεται η συχνότητα ρολογιού. Εάν η στατική κατανάλωση ενέργειας ήταν μηδέν, δεν θα είχε σημασία η συχνότητα που χρησιμοποιούμε, καθώς ένα βραδύτερο ρολόι θα οδηγούσε σε αναλογική μείωση της κατανάλωσης ενέργειας, αφήνοντας την καταναλισκόμενη ενέργεια ανά υπολογισμό ίδια. Ωστόσο, η στατική κατανάλωση ενέργειας καθιστά τη συσκευή λιγότερο αποτελεσματική στις χαμηλότερες συχνότητες. Ως εκ τούτου, γενικά είναι αποδοτικότερο ενεργειακά οι εργασίες να γίνονται σε μεγαλύτερες συχνότητες, να εκτελούνται όσο το δυνατόν γρηγορότερα για να αυξηθεί το ποσοστό του χρόνου που δαπανάται σε στατική κατανάλωση ενέργειας.

Αυτό όμως δεν είναι πάντοτε αληθές, όπως συμβαίνει σε περιπτώσεις όπου η κατάσταση αναμονής βρίσκεται σε υψηλότερες συχνότητες ρολογιού, με αποτέλεσμα το όφελος για την ταχύτητα διάδοσης του σήματος να μην είναι γραμμικό με τη συχνότητα. Σε τέτοιες περιπτώσεις μπορεί να είναι πιο αποδοτικό οι εργασίες να εκτελούνται με συχνότητα λίγο κάτω από το όριο αναμονής (sleep mode). Το σχήμα 1.10 απεικονίζει τη διαφορά μεταξύ της καταναλισκόμενης ισχύος, όταν οι εργασίες εκτελούνται με υψηλότερη συχνότητα ρολογιού (προσέγγιση 1) και για χαμηλότερες συχνότητες ρολογιού (προσέγγιση 2). Η ισχύς που απεικονίζεται αναφέρεται στη στιγμιαία ισχύ της συσκευής και η ενέργεια αναπαρίσταται με το γραμμοσκιασμένο τμήμα.



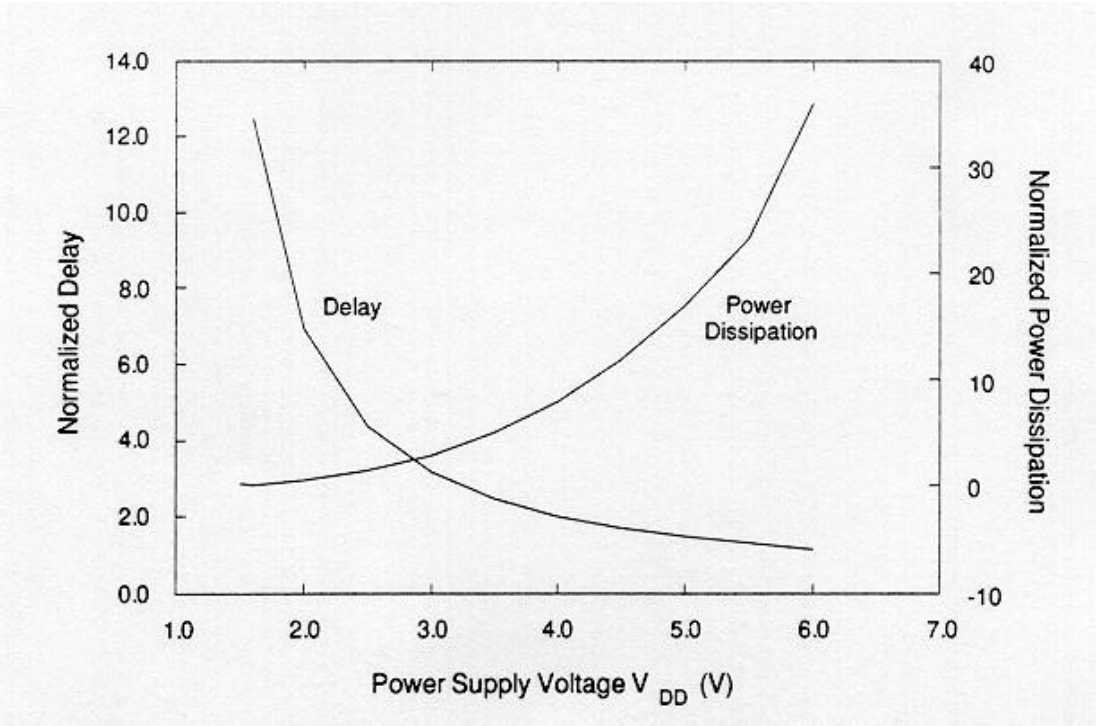
Σχήμα 1.10. Σχέση καταναλισκόμενης ισχύος και ενέργειας για διαφορετικές συχνότητες λειτουργίας.

Μία άλλη για παράδειγμα σημαντική κατανάλωση ενέργειας, που δεν επηρεάζει τη ταχύτητα διάδοσης του σήματος, είναι η περίπτωση κατά την οποία αυτή οφείλεται σε λογικά ενεργά κυκλώματα που δεν συνεισφέρουν στη λειτουργικότητα του συστήματος. Το ρολόι αυτών των κυκλωματικών μπλοκ μπορεί να απενεργοποιηθεί για να αποφευχθούν οι μεταβάσεις σήματος (switching activity) και η κατανάλωση ενέργειας σε αυτά τα μπλοκ. Αυτή η τεχνική χαμηλής κατανάλωσης με απενεργοποίηση του ρολογιού κατά τμήματα (clock gating technique) μπορεί να μειώσει σημαντικά τη δυναμική κατανάλωση μέσα σε ένα ολοκληρωμένο κύκλωμα. Σε μη ενεργό κυκλωματικό μπλοκ οι είσοδοι, υπό κανονικές συνθήκες, δεν αλλάζουν. Τότε οι δυναμική κατανάλωση του μπλοκ καθορίζεται από το δέντρο διανομής του σήματος ρολογιού (clock tree). Επειδή αυτά τα στοιχεία του κυκλώματος έχουν δραστηριότητα μετάβασης 100%, η δυναμική κατανάλωση μπορεί να φθάσει και το 30% της συνολικής δυναμικής κατανάλωσης. Συνεπώς, η τεχνική απενεργοποίησης του ρολογιού κατά τμήματα είναι μια αποτελεσματική τεχνική ακόμα και όταν οι είσοδοι δεν αλλάζουν.

Το κύριο μειονέκτημα της τεχνικής απενεργοποίησης του ρολογιού κατά τμήματα είναι ότι τα ρεύματα τροφοδοσίας παρουσιάζουν απότομες μεταβάσεις λόγω της ενεργοποίησης και απενεργοποίησης μεγάλων υποσυστημάτων. Εάν ο ρυθμιστής τάσης (voltage regulator) δεν έχει σχεδιαστεί να λειτουργεί γρήγορα και χωρίς μεγάλες αιχμές η τάση στο ολοκληρωμένο μπορεί να διαταραχθεί και να οδηγήσει σε αποτυχία λειτουργίας στα ενεργά κυκλωματικά μπλοκ. [5]

1.6.2 Η σχέση της τάσης λειτουργίας και κατανάλωσης ενέργειας

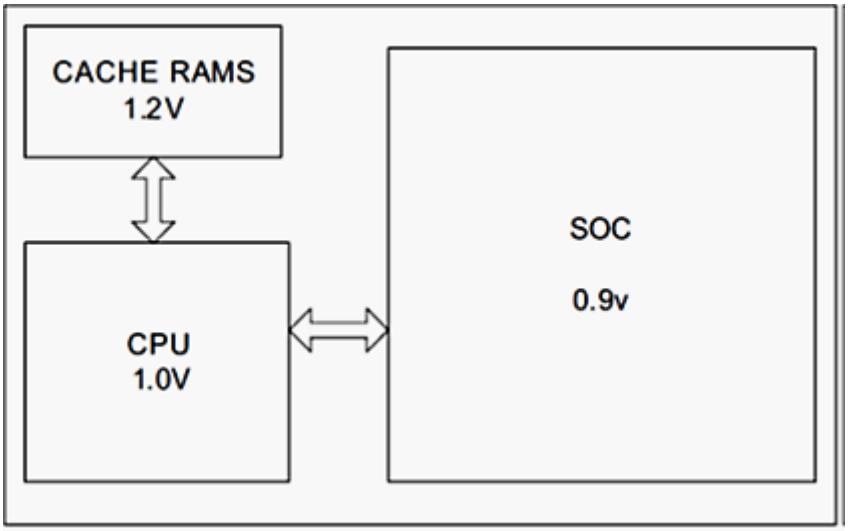
Η μείωση της τάσης σε επίπεδο σχεδίασης του ολοκληρωμένου κυκλώματος είναι ένας βασικός παράγοντας για τη μείωση της κατανάλωσης ενέργειας σύμφωνα με την εξίσωση (1.1). Λόγω του ότι η τάση υψώνεται στο τετράγωνο, οποιαδήποτε μεταβολή της μπορεί να φέρει σημαντικές αλλαγές στη κατανάλωση. Η μείωση της τάσης ενώ μπορεί να επιφέρει σημαντική μείωση της κατανάλωσης(π.χ. 30% μείωση της τάσης λειτουργίας μειώνει τη κατανάλωση ισχύος κατά 49%), έχει ως αποτέλεσμα την πτώση της απόδοσης, λόγω του ότι τα στοιχεία CMOS γίνονται πιο αργά(Σχήμα 1.11) και πιο ευαίσθητα στον θόρυβο.



Σχήμα 11. Σχέση τάσης τροφοδοσίας – καθυστέρησης και τάσης τροφοδοσίας - κατανάλωσης ισχύος.

Η μείωση αυτή μπορεί να πραγματοποιηθεί με μεθόδους μεταβολής της τάσης σε επίπεδο ολοκληρωμένου κυκλώματος. Όλες οι μονάδες σε ένα ic δεν είναι απαραίτητο να λειτουργούν με την ίδια τάση. Για παράδειγμα η αρχιτεκτονική με πολλαπλές τάσεις

τροφοδοσίας(Multi - Voltage Architecture) μπορεί να μειώσει σημαντικά την κατανάλωση της ισχύος. Στο σχήμα 1.12 εμφανίζεται μια διάταξη με πολλαπλές τάσεις τροφοδοσίας. [8]



Σχήμα 1.12. Αρχιτεκτονική με πολλαπλές τάσεις τροφοδοσίας.

Αυτή η τεχνική απομακρύνεται από την παραδοσιακή προσέγγιση της χρήσης μιας ενιαίας σταθερής τροφοδοσίας για όλες τις (εσωτερικές) πύλες σε ένα ολοκληρωμένο κύκλωμα.

Η πιο βασική μορφή αυτής της νέας προσέγγισης είναι να χωρίσει την εσωτερική λογική του ολοκληρωμένου κυκλώματος σε περιοχές πολλαπλών τάσεων ή τομείς ισχύος, το καθένα με τη δική του τροφοδοσία. Αυτή η προσέγγιση ονομάζεται σχεδιασμός πολλαπλών τάσεων. Βασίζεται στην αντίληψη ότι σε ένα σύγχρονο σχεδιασμό ic, διαφορετικά μπλοκ έχουν διαφορετικούς στόχους επιδόσεων και διαφορετικούς περιορισμούς. Ένας επεξεργαστής για παράδειγμα, θα εκτελέσει τους υπολογισμούς με τη μέγιστη ταχύτητα που του επιτρέπει η τεχνολογία ημιαγωγών με την οποία έχει υλοποιηθεί. Σε αυτή την περίπτωση απαιτείται μια σχετικά υψηλή τάση τροφοδοσίας. Ένα μπλοκ USB, από την άλλη πλευρά, μπορεί να εργαστεί σε μια σταθερά χαμηλή ταχύτητα, η οποία καθορίζεται από το πρωτόκολλο τις εκάστοτε εφαρμογής. Στην περίπτωση αυτή, μια μικρότερη τάση τροφοδοσίας μπορεί να είναι αρκετή για να καλύψει τις απαιτήσεις των μπλοκ. [8], [10]

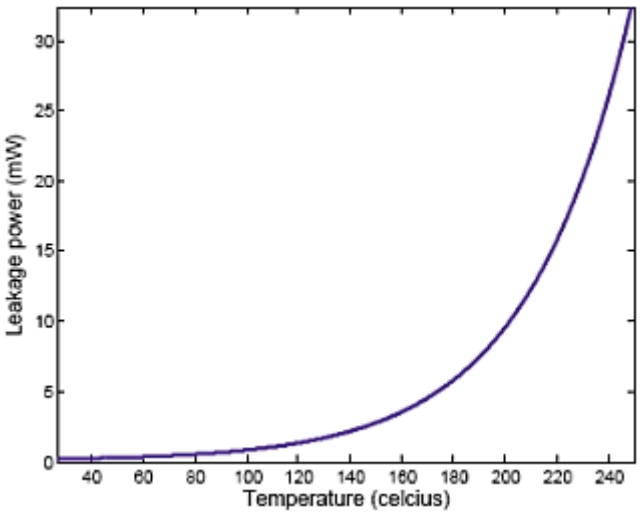
1.6.3 Η σχέση της θερμοκρασίας λειτουργίας και κατανάλωσης ενέργειας

Η θερμική διαχείριση είναι ένα κρίσιμο θέμα στον σχεδιασμό ολοκληρωμένων κυκλωμάτων. Σε κάθε νέα γενιά τεχνολογίας, το μέγεθος της συσκευής μειώνεται ενώ οι ταχύτητες της διάδοσης του σήματος αυξάνονται με ταυτόχρονη αύξηση της πυκνότητας του κυκλώματος. Όλοι αυτοί οι παράγοντες συμβάλλουν στις αυξημένες

θερμοκρασίες που βλάπτουν την απόδοση και την αξιοπιστία του κυκλώματος. Επιπλέον, hot spots λόγω της χωρικά μη ομοιόμορφης ροής θερμότητας στα ic μπορούν να προκαλέσουν περαιτέρω μείωση στην αξιοπιστία λειτουργίας. Τα ic φτάνουν σε τέτοια επίπεδα θερμοκρασίας που θα καταστήσουν τις τρέχουσες τεχνικές ψύξης ανεπαρκείς. Αυτή η υπερθέρμανση είναι η κύρια αιτία για μεγάλο μέρος των αστοχιών των υλικών. Η τάση στη βιομηχανία ημιαγωγών είναι προς μικρότερα, ταχύτερα και υψηλότερης ισχύος ic. Έτσι, η θερμική διαχείριση γίνεται όλο και πιο σημαντική. Όσο η απόδοση του ολοκληρωμένου αυξάνεται, η υψηλότερη θερμοκρασία και η χαμηλότερη αξιοπιστία γίνονται σημαντικοί παράγοντες στο σχεδιασμό. Η αύξηση της πυκνότητας στο packaging μειώνει το συνολικό μέγεθος της συσκευής και είναι οικονομικά αποδοτικότερη, όπως επίσης βελτιώνει τη συνολική ηλεκτρική απόδοση, αλλά από θερμική άποψη, η υψηλή ισχύς σε μικρότερες συσκευές είναι ένας κακός συνδυασμός. Αυτή η αύξηση της υψηλής πυκνότητας ισχύος πρέπει να οδηγήσει σε καλύτερη θερμική διαχείριση. [11]

Ο πρώτος νόμος της θερμοδυναμικής αναφέρει ότι σε σταθερή λειτουργία η εισερχόμενη ενέργεια ενός συστήματος, είναι ίση στην αποδιδόμενη ενέργεια του συστήματος. Έτσι, απόντων άλλων ενεργειακών αλληλεπιδράσεων, η περισσότερη ενέργεια που αποδίδει ένα ic είναι με τη μορφή θερμότητας, που προκύπτει από τα ρεύματα που ρέουν στα ηλεκτρονικά στοιχεία. Επομένως, η παραγωγή θερμότητας του ic είναι ανάλογη με την κατανάλωση ενέργειας. [7], [9]

Και καθώς οι διαστάσεις κατασκευής των CMOS έχουν πέσει κάτω από τα 100nm, η ισχύς διαρροής (leakage power) έχει γίνει ένα σημαντικό κομμάτι της συνολικής κατανάλωσης ισχύος του ic. Τα ρεύματα διαρροής (leakage currents) καθόσον είναι εξαρτώμενα από τη θερμοκρασία και επειδή αυτή η ισχύς διαρροής παρουσιάζει εκθετική σχέση με τη θερμοκρασία λειτουργίας του ic, σημαντική μείωση ισχύος μπορεί να επιτευχθεί μειώνοντας τη θερμοκρασία. (Σχήμα 1.13.) [11]



Σχήμα 1.13. Σχέση μεταξύ ισχύος διαρροής και θερμοκρασίας.

ΒΙΒΛΙΟΓΡΑΦΙΑ 1ου ΚΕΦΑΛΑΙΟΥ

[1] THE MOST SIGNIFICANT MOSFET PARAMETERS IMPACT IN CMOS INVERTER SWITCHING CHARACTERISTICS, Milaim Zabeli, Nebi Caka, Myzafere Limani, Qamil Kabashi*University of Prishtina, May 2018. International Journal Of Circuits, Systems And Signal Processing, Volume 12, 2018

[2] CMOS CIRCUIT DESIGN, LAYOUT, AND SIMULATION, Third Edition, R. Jacob Baker, IEEE Press Series on Microelectronic Systems, Stuart K. Tewksbury and Joe E. Brewer, Series Editors, Copyright © 2010 by the Institute of Electrical and Electronics Engineers, Inc.

[3] CMOS DIGITAL INTERGRATED CIRCUITS, ANALYSIS AND DESIGN, Sung-Mo (Steve) Kang, University of Illinois at Urbana- Champaign, Yusuf Leblebigi, Worcester Polytechnic Institute, Swiss Federal Institute of Technology-Lausanne, McGraw-Hill Education, 3rd Edition, 2002.

[4] MICROELECTRONIC CIRCUIT DESIGN, Richard C. Jaeger, Auburn University, Travis N. Blalock, University of Virginia, Published by McGraw-Hill, Copyright c 2011 by The McGraw-Hill Companies, Inc.

[5] "CMOS ΛΟΓΙΚΗ ΟΙΚΟΓΕΝΕΙΑ ΜΕ ΧΡΗΣΗ ΔΙΟΔΙΚΟΥ ΤΡΑΝΖΙΣΤΟΡ ΓΙΑ ΧΑΜΗΛΗ ΔΥΝΑΜΙΚΗ ΚΑΤΑΝΑΛΩΣΗ ΕΝΕΡΓΕΙΑΣ", Αικατερίνη Κατσαρού, Μεταπτυχιακή Εργασία, Πανεπιστήμιο Ιωαννίνων, Σχολή Θετικών Επιστημών, Τμήμα Μηχανικών Η/Υ & Πληροφορικής, Ιούνιος 2012.

[6] "ΚΥΚΛΩΜΑΤΑ CMOS ΧΑΜΗΛΗΣ ΚΑΤΑΝΑΛΩΣΗΣ ΙΣΧΥΟΣ ΚΑΙ ΧΑΜΗΛΗΣ ΤΑΣΗΣ ΤΡΟΦΟΔΟΣΙΑΣ", Διδακτορική Διατριβή, Ιωάννης Χ. Μωυσιάδης, ΕΘΝΙΚΟ & ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ, Σχολή Θετικών Επιστημών Τμήμα Πληροφορικής, Αθήνα 2000.

[7] MODELING THE TEMPERATURE BIAS OF POWER CONSUMPTION FOR NANOMETER-SCALE CPUs IN APPLICATION PROCESSORS - Karel DeVogeleer,

Gerard Memmi, Pierre Jouvelot† and Fabien Coelho† TELECOM ParisTech – INFRES – CNRS LTCI - UMR 5141 – Paris, France, April 2014.

[8] LOW POWER METHODOLOGY MANUAL FOR SYSTEM-ON-CHIP DESIGN, Michael Keating, David Flynn, Robert Aitken, Alan Gibons, Kaijian Shi, Springer, 2007.

[9] AT11489: LOW POWER TECHNIQUES FOR ATMEL SMART ARM MCUs, Application Note, Atmel.

[10] DESIGNING LOW-ENERGY EMBEDDED SYSTEMS FROM SILICON TO SOFTWARE Part 2 – Software Decisions, Rev 1.0 Silicon Laboratories.

[11] THERMAL EFFECTS WITH LEAKAGE POWER CONSIDERED IN 2D/3D FLOORPLANNING,Pingqiang Zhou, Yuchun Ma, Qiang Zhou, Xianlong Hong, Tsinghua University, Beijing, P.R.China

[12] LOW-POWER DIGITAL VLSI DESIGN CIRCUITS AND SYSTEMS, by Abdellatif Bellaouar University of Waterloo and Mohamed I. Elmasry University of Waterloo, SPRINGER SCIENCE+BUSINESS MEDIA, LLC, 1995 Springer Science+Business Media New York Originally published by Kluwer Academic Publishers in 1995.

ΚΕΦΑΛΑΙΟ 2:
System On a Chip (SoC) και αρχιτεκτονικές επεξεργαστών

2.1 Εισαγωγή

Το System On a Chip (SoC) ορίζεται ως εκείνο το λειτουργικό μπλοκ, που εμπεριέχει τα περισσότερα από τα βασικά λειτουργικά τμήματα ενός συστήματος. Σε αυτό το κεφάλαιο, αναλύεται η αρχιτεκτονική SoC (System On a Chip) που μπορεί να υλοποιηθεί είτε με MCU είτε με CPU.

Ακολουθεί η σύγκριση των ISA(Instruction Set Architectures) αρχιτεκτονικών CISC και RISC, με την RISC αρχιτεκτονική να αποτελεί μία από τις πιο σημαντικές εξελίξεις στην τεχνολογία των επεξεργαστών, που υπό ορισμένες συνθήκες προσφέρει σημαντικά πλεονεκτήματα σε σχέση με την αντίστοιχη αρχιτεκτονική CISC.

Στο τέλος, γίνεται η παρουσίαση του επεξεργαστή ARM που βασίζεται στην αρχιτεκτονική RISC, καθώς και η περιγραφή των κύριων χαρακτηριστικών του.

2.2 System on a Chip (SoC)

Ένα System on a Chip, επίσης γνωστό ως SoC, είναι ουσιαστικά ένα ολοκληρωμένο κύκλωμα όπου σε μια ενιαία πλατφόρμα ενσωματώνει ένα ολόκληρο ηλεκτρονικό ή υπολογιστικό σύστημα σε αυτό. Είναι, ακριβώς όπως υποδηλώνει το όνομά του, ένα ολόκληρο σύστημα σε ένα ic. Τα στοιχεία που ένα SoC γενικά φαίνεται να ενσωματώνει, περιλαμβάνουν μια κεντρική μονάδα επεξεργασίας, θύρες εισόδου και εξόδου, εσωτερική μνήμη, καθώς και αναλογικά μπλοκ εισόδου και εξόδου.

2.2.1 Γιατί SoC;

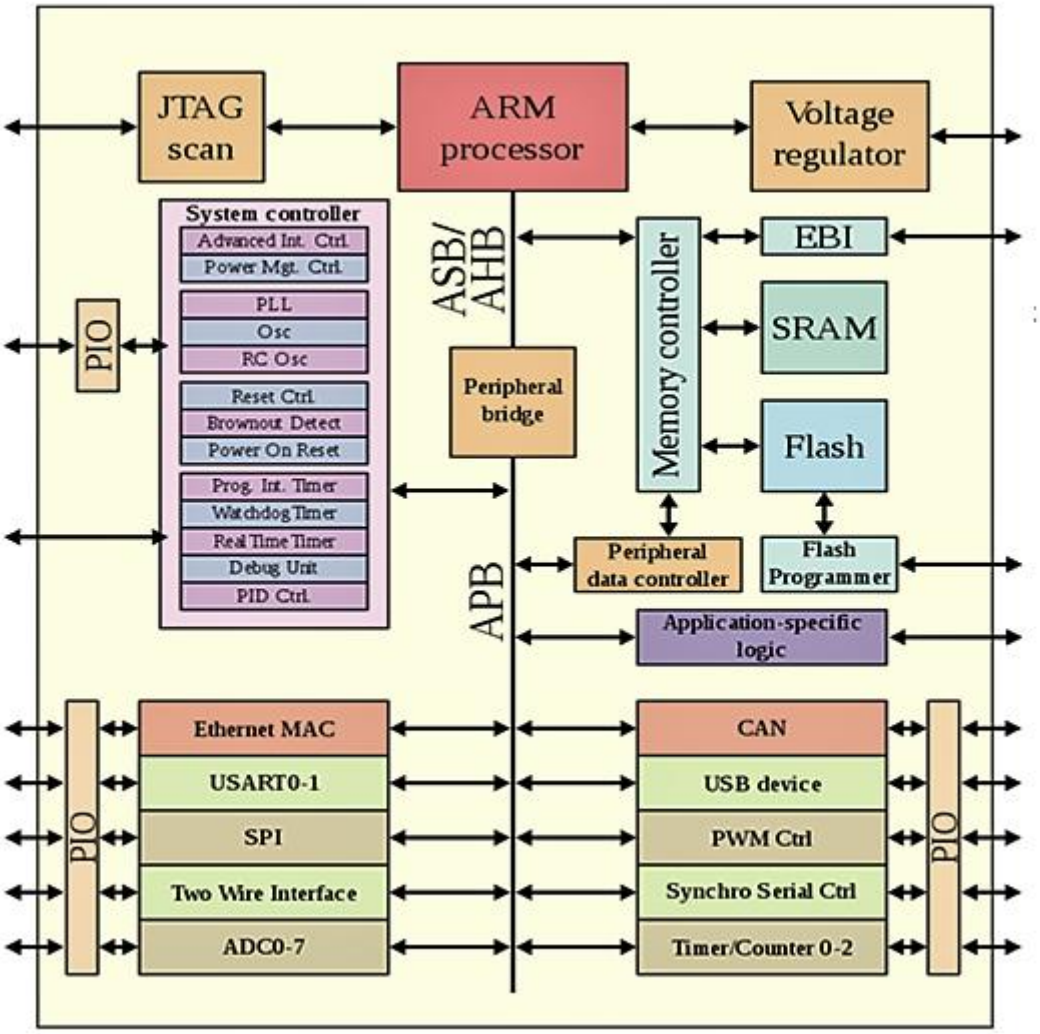
Το κίνητρο πίσω από τη δημιουργία SoC είναι το γεγονός ότι πρωταρχικοί στόχοι μας είναι να μειωθεί η κατανάλωση ενέργειας, το κόστος σχεδίασης και υλοποίησης, καθώς και ο χώρος που καταλαμβάνουν τα συστήματα. Με ένα SoC, επιτυγχάνονται όλοι αυτοί οι στόχοι καθώς ουσιαστικά ελαχιστοποιούνται οι υλοποιήσεις των πολλαπλών ολοκληρωμένων, σε έναν επεξεργαστή που χρησιμοποιεί πολύ λιγότερη ισχύ.

Αυτά τα ολοκληρωμένα μας επέτρεψαν να δημιουργήσουμε μια πληθώρα φορητών συσκευών που μπορούμε εύκολα να μεταφέρουμε οπουδήποτε μαζί μας. Ως εκ τούτου, χρησιμοποιούνται συχνά σε συστήματα που σχετίζονται με το Διαδίκτυο των πραγμάτων, τα ενσωματωμένα συστήματα, καθώς και με τα smartphones, τα αυτοκίνητα και άλλα.

Η εφαρμογή των SoCs είναι πρακτικά απεριόριστη και ανεκτίμητη. Χρησιμοποιούνται στις περισσότερες, αν όχι όλες τις φορητές τεχνολογίες όπως smartphone, κάμερες,

tablet και άλλες ασύρματες τεχνολογίες. Το smartphone είναι ένα καλό παράδειγμα για το πώς λειτουργεί ένα SoC. Όταν χρησιμοποιείται το κινητό τηλέφωνο, εκτός από τη λήψη και πραγματοποίηση κλήσεων, χρησιμοποιείται επίσης για περιήγηση στο διαδίκτυο, σαν video-audio player, σαν φωτογραφική μηχανή και άλλα. Κανένα από αυτά δεν θα ήταν δυνατό χωρίς την ύπαρξη στοιχείων όπως μια κάρτα γραφικών, η υποστήριξη διαδικτύου, ασύρματες συνδέσεις, GPS και πολλά άλλα στοιχεία. Το SoC επέτρεψε να υπάρχουν όλα αυτά τα εξαρτήματα σε ένα ολοκληρωμένο και στο μικρότερο δυνατό μέγεθος.

Η τεχνολογία SoC χρησιμοποιείται επίσης σε προσωπικούς και φορητούς υπολογιστές μικρότερου μεγέθους για τη μείωση της κατανάλωσης ενέργειας και τη βελτίωση της απόδοσης του tablet ή του φορητού υπολογιστή χρησιμοποιώντας ένα μοναδικό ολοκληρωμένο για τη διαχείριση όλων των διαφόρων εργασιών του συστήματος. [2], [Wikipedia]



Σχήμα 2.1 : Block διάγραμμα ενός SoC

Αρχικά, ένα system on chip (Σχήμα 2.1) πρέπει να έχει έναν επεξεργαστή στον πυρήνα του, ο οποίος θα καθορίζει τις λειτουργίες του. Κανονικά, ένα SoC έχει πολλούς πυρήνες επεξεργαστή. Μπορεί να είναι ένας μικροελεγκτής, ένας μικροεπεξεργαστής, ένας ψηφιακός επεξεργαστής σήματος ή ένας επεξεργαστής σετ συγκεκριμένων εφαρμογών. Δεύτερον, το ic πρέπει να έχει τις μνήμες RAM, ROM, EEPROM ή ακόμα και μνήμη flash. Το επόμενο πράγμα που πρέπει να διαθέτει ένα SoC είναι εξωτερικά interfaces με τα τυπικά πρωτόκολλα επικοινωνίας όπως USB, Ethernet και HDMI. Μπορεί επίσης να ενσωματώσει ασύρματη τεχνολογία και να περιλαμβάνει πρωτόκολλα που σχετίζονται με WiFi και Bluetooth. Θα χρειαστεί επίσης μια GPU ή μια μονάδα επεξεργασίας γραφικών για να βοηθήσει στην απεικόνιση. Ένα SoC επίσης περιλαμβάνει voltage regulators, phase lock loop control systems και ταλαντωτές, clocks and timers, analog to digital

και digital to analog μετατροπείς, εσωτερικό δίαυλος διασύνδεσης ή ένα δίκτυο για τη σύνδεση όλων των επιμέρους μπλοκ κ.λπ. [2], [Wikipedia]

2.2.2 Πλεονεκτήματα SoC

Ουσιαστικά τα μεγάλα οφέλη από τη χρήση ενός SoC είναι: εξοικονόμηση ενέργειας, εξοικονόμηση χώρου και μείωση κόστους. Τα SoCs είναι επίσης πολύ πιο αποτελεσματικά ως συστήματα καθώς η απόδοσή τους μεγιστοποιείται ανά watt κατανάλωσης.

2.2.3 SoC vs CPU

Οι μέρες όπου η CPU ήταν το πρωταρχικό και σημαντικότερο μέρος ολόκληρου του υπολογιστικού συστήματος έχουν παρέλθει. Τώρα, η CPU είναι μόνο ένα μέρος που προστίθεται στην «εξίσωση» ενός SoC. Ένα SoC συνδυάζει την ισχύ της CPU με πολλά άλλα στοιχεία που χρειάζεται για την εκτέλεση των λειτουργιών της.

Ο λόγος για τον οποίο τα SoCs προτιμώνται όλο και περισσότερο από τα απλά συστήματα CPU είναι επειδή παρόλο που αφορούν περίπου την «ίδια πλευρά του νομίσματος», ένα SoC εμπεριέχει διπλάσια ισχύ και ικανότητες. Η CPU θα εξακολουθεί να βασίζεται σε πολλά άλλα εξωτερικά υλικά, ενώ ένα SoC έχει χώρο για όλα όσα χρειάζονται να προστεθούν στον ελάχιστο χώρο του. Σε σύγκριση με τις CPU, τα SoC χρησιμοποιούν επίσης μικρότερη καλωδίωση και στη καταναλώνουν λιγότερη ισχύ, καθιστώντας τα πολύ πιο αποτελεσματικά και ενεργειακά έξυπνα. Το μόνο πρόβλημα που μπορείτε να υποστηριχθεί κατά του SoC είναι το γεγονός ότι σε σύγκριση με ένα σύστημα CPU, είναι μη αναβαθμίσιμο. [2], [Wikipedia]

2.3 Αρχιτεκτονική RISC vs CISC

Οι αρχιτεκτονικές RISC και CISC (Πίνακας 2.1) έχουν εξελιχθεί στο χρόνο σε διαφορετικές εφαρμογές. Οι περισσότεροι σύγχρονοι επεξεργαστές CISC

χρησιμοποιούνται σε διακομιστές και επιτραπέζιους υπολογιστές, ενώ οι σύγχρονοι επεξεργαστές RISC κυριαρχούν στις embedded εφαρμογές. Λόγω αυτής της διαφοράς, οι επεξεργαστές RISC έχουν βελτιστοποιηθεί για τα χαμηλής ισχύος συστήματα και οι επεξεργαστές CISC για απόδοση. [1]

Πίνακας 2.1. Τα χαρακτηριστικά των RISC και CISC αρχιτεκτονικών

	Format	Operations	Operands
RISC/ ARM	o Fixed length instructions o Relatively simple encoding o ARM: 4B, THUMB(2B, optional)	o Simple, single function operations o Single cycle	o Operands: registers, immediates o Few addressing modes o ARM: 16 general purpose registers
CISC / x86	o Variable length instructions o Common insts shorter/simpler o Special insts longer/complex o x86: from 1B to 16B long	o Complex, multi-cycle instructions o Transcendentals o Encryption o String manipulation	o Operands: memory, registers, immediates o Many addressing modes o x86: 8 32b & 6 16b registers
Historical Contrasts	o CISC decode latency prevents pipelining o CISC decoders slower/more area o Code density: RISC < CISC	o Even w/ μcode, pipelining hard o CISC latency may be longer than compiler's RISC equivalent	o CISC decoder complexity higher o CISC has more per inst work, longer cycles o Static code size: RISC > CISC
Convergence Trends	o μ-op cache minimizes decoding overheads o x86 decode optimized for common insts o I-cache minimizes code density impact	o CISC insts split into RISC-like micro-ops; optimizations eliminated inefficiencies o Modern compilers pick mostly RISC insts; μ-op counts similar for ARM and x86	o x86 decode optimized for common insts o CISC insts split into RISC-like micro-ops; x86 and ARM μ-op latencies similar o Number of data cache accesses similar
Empirical Questions	o How much variance in x86 inst length? Low variance ⇒ common insts optimized o Are ARM and x86 code densities similar? Similar density ⇒ No ISA effect o What are instruction cache miss rates? Low ⇒ caches hide low code densities	o Are macro-op counts similar? Similar ⇒ RISC-like on both o Are complex instructions used by x86 ISA? Few complex ⇒ Compiler picks RISC-like o Are μ-op counts similar? Similar ⇒ CISC split into RISC-like μ-ops	o Number of data accesses similar? Similar ⇒ no data access inefficiencies

2.3.1 Σύγκριση ISA (Instruction Set Architecture) - η αρχιτεκτονική CISC

Οι σχεδιαστές υπολογιστών δημιούργησαν επεξεργαστές CISC για να απλοποιήσουν τη διαδικασία της μεταγλώττισης. Αυτή η αρχιτεκτονική συνήθως ενσωματώνει πάνω από εκατό εντολές με ποικίλο αριθμό bytes και πολυπλοκότητα.

Ορισμένες από αυτές τις εντολές μπορούν να χειριστούν δεδομένα απευθείας από μνήμη και, ως εκ τούτου, περιλαμβάνει μια μεγάλη διεύθυνση μνήμης στην κωδικοποίηση. Αυτό το χαρακτηριστικό έχει, εκτός από το μεταβλητό μήκος των εντολών, τα ακόλουθα μειονεκτήματα:

- Πιο περίπλοκη ILP (Instruction-level parallelism) και pipeline εφαρμογή λόγω μη ομοιόμορφου μεγέθους εντολών και κωδικοποίησης.
- Η αποκωδικοποίηση και ο προγραμματισμός των εντολών παρουσιάζουν δυσκολία στην εκτέλεση.
- Αύξηση της πολυπλοκότητας της ISA, η οποία οδηγεί σε μεγαλύτερο κόστος συντήρησης.

Παρ' όλα αυτά τα μειονεκτήματα, η αρχιτεκτονική CISC παρουσιάζει μερικά ελκυστικά χαρακτηριστικά. Ως κίνητρο για την ανάπτυξη αυτού του σχεδιασμού, υπήρχε η επιθυμία να βελτιωθεί η προγραμματιστική παραγωγικότητα. Η εκτεταμένη χρήση σύνθετων εντολών παρέχει άμεση εφαρμογή των υψηλού επιπέδου γλωσσών προγραμματισμού, διευκολύνοντας τη μετάφραση σε κώδικα μηχανής. Επίσης, με το μεταβλητό μήκος στις εντολές CISC και τη βελτιωμένη κωδικοποίηση, υπάρχει πιθανή μείωση του μεγέθους του προγράμματος και του μεγέθους της προσβάσιμης κύριας μνήμης, επιτρέποντας μια ταχύτερη εκτέλεση επειδή η CPU θα εξαρτάται λιγότερο από αργούς πόρους του συστήματος. [1],[3]

2.3.2 Η αρχιτεκτονική RISC

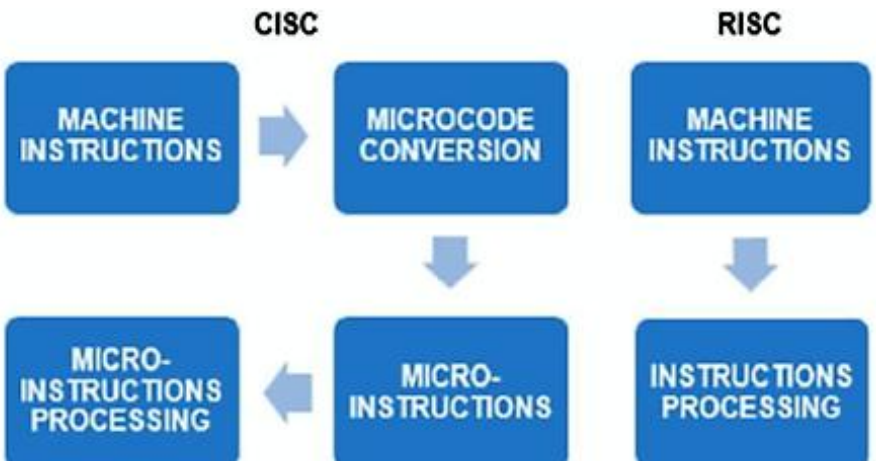
Αυτή η αρχιτεκτονική σχεδιάστηκε για την αντιμετώπιση των πιο πάνω προβλημάτων μειώνοντας την πολυπλοκότητα των εντολών. Οι επεξεργαστές RISC χρησιμοποιούν εντολές σταθερού μήκους, σε αντίθεση με τους CISC, απλοποιώντας έτσι τη διαδικασία αποκωδικοποίησης εντολών. Το σχήμα 2.2 δείχνει την ιδέα πίσω από τις μικρο-λειτουργίες που χρησιμοποιούνται στο CISC σε αντίθεση με την απλοποιημένη διαδικασία στο RISC.

Συνοψίζοντας τα κύρια χαρακτηριστικά αυτής της αρχιτεκτονικής:

- Οι οδηγίες είναι βελτιστοποιημένες και με παρόμοια δομή, επομένως οι πολύπλοκες λειτουργίες είναι μια ακολουθία απλούστερων εντολών που δημιουργούνται από τον μεταγλωττιστή.

- Απαιτείται ένας μεγάλος αριθμός καταχωρητών για τις αριθμητικές πράξεις. Η πρόσβαση στη μνήμη επιτυγχάνεται μέσα από εντολές φόρτωσης και αποθήκευσης, περιορίζοντας την αλληλεπίδραση με τη μνήμη.

- Λόγω των προηγούμενων, το pipelining γίνεται πιο αποτελεσματικό. Η απλότητα που συναντάται στις RISC λειτουργίες επιτρέπει στους επεξεργαστές να έχουν CPI (clock cycles per instruction) ενός κύκλου ρολογιού ανά οδηγία, ενώ η CISC ελαχιστοποιεί τον αριθμό των εντολών ανά πρόγραμμα.



Σχήμα 2.2. Η CISC κάνει χρήση μικροκώδικα για να επιτρέψει την αποκωδικοποίηση του σύνθετων εντολών, ενώ η RISC χρησιμοποιεί ήδη απλές εντολές με συνέπεια, η διαδικασία αποκωδικοποίησης είναι πιο συνοπτική.

Ωστόσο, αυτός ο σχεδιασμός CPU συνοδεύεται από εμπόδια. Επειδή οι εντολές RISC έχουν σταθερό μήκος, απαιτείται περισσότερη μνήμη RAM για την αποθήκευσή τους σε σύγκριση με την αρχιτεκτονική CISC, οπότε ο κώδικας γίνεται μεγαλύτερος. Επιπρόσθετα, η μεταγλώττιση υψηλού επιπέδου γλωσσών σε γλώσσα μηχανής είναι πιο δύσκολη. [3], [4], [7]

2.3.3 Σύγκριση απόδοσης

Σε μια πρόσφατη ανάλυση που μετρά διάφορες πτυχές των ARM και x86 μικροεπεξεργαστών, διαπιστώθηκε ότι η ISA αρχιτεκτονική, δεν έχει σημαντικό αντίκτυπο στην απόδοση. Αυτή η πτυχή καθορίζεται κυρίως από τον τομέα εφαρμογής που στοχεύει ο σχεδιασμός του επεξεργαστή. Ενώ ο σκοπός της αρχιτεκτονικής ARM είναι να επιτευχθεί χαμηλή κατανάλωση ενέργειας και υψηλή απόδοση ακολουθώντας τη φιλοσοφία RISC, η αρχιτεκτονική x86 στοχεύει στην προς τα πίσω συμβατότητα και αποτελεσματικότητα χρησιμοποιώντας μικρο-λειτουργίες, δηλαδή διαχωρίζοντας τις εντολές σε μικρότερες ακολουθίες.

Ανεξάρτητα από τις αντιθέσεις μεταξύ των δύο αρχιτεκτονικών, μελέτες έδειξαν ότι οι επιδράσεις της ISA στην απόδοση είναι πανομοιότυπος στους επεξεργαστές ARM και x86. Οι διαφορές ωστόσο γίνονται ξεκάθαρες κατά τη σύγκριση των μικροαρχιτεκτονικών. [1], [3]

2.3.4 Ισχύς και κατανάλωση ενέργειας

Σύμφωνα με τα αποτελέσματα που συγκεντρώθηκαν από την ίδια ανάλυση, η κατανάλωση ισχύος εξαρτάται περισσότερο από την επιλογή της βελτιστοποίησης του σχεδιασμού του πυρήνα, η οποία μπορεί να δώσει έμφαση είτε στην ισχύ είτε στην

απόδοση. Εξαιτίας αυτού, η αρχιτεκτονική του σετ εντολών δεν επηρεάζει σημαντικά το πόση ενέργεια και ισχύ θα καταναλώσει ο επεξεργαστής. Κατά μέσο όρο, οι εφαρμογές x86 καταναλώνουν περισσότερη ισχύ από τις εφαρμογές ARM. Η x86 ISA απαιτεί περισσότερη ενέργεια για μετάφραση μικρο-λειτουργιών, αλλά αυτό αποδείχθηκε ότι προκαλεί ελάχιστο αντίκτυπο. Και πάλι, η μικροαρχιτεκτονική παίζει σημαντικό ρόλο στον καθορισμό της ενεργειακής απόδοσης.

Συνοψίζοντας αυτά τα αποτελέσματα, η ενέργεια και η αποδοτικότητα ισχύος είναι αποτελεσματικά ανεξάρτητα από την ISA. Η μικροαρχιτεκτονική και η μεθοδολογία του σχεδιασμού είναι οι κύριοι παράγοντες που επηρεάζουν την κατανάλωση. Αν ο σχεδιασμός του πυρήνα εστιάζεται ιδιαίτερα στην ισχύ ή στη βελτιστοποίηση της απόδοσης, μπορεί να έχει επιπλέον απαιτήσεις ενέργειας και, όπως είναι αναμενόμενο, οι επεξεργαστές υψηλότερης απόδοσης χρειάζονται περισσότερη ισχύ από επεξεργαστές χαμηλότερης απόδοσης. Στο τέλος, η μικροαρχιτεκτονική και οι σχεδιαστικές επιλογές είναι αυτές που κάνουν τη διαφορά κατά τη σύγκριση κατανάλωση ενέργειας και ισχύος των επεξεργαστών και όχι της ISA. [1], [3], [4]

2.3.5 Συμπεράσματα

Με βάση πρόσφατες μελέτες φαίνεται λογικό να συμπεράνουμε ότι το χάσμα μεταξύ των αρχιτεκτονικών RISC και CISC γίνεται όλο και πιο μικρό. Οι πρόσφατοι επεξεργαστές x86 χρησιμοποιούν pipelining διατηρώντας επίσης ένα σύνθετο σύνολο εντολών. Εσωτερικά, αυτοί οι επεξεργαστές τμηματοποιούν τις x86 εντολές σε αυτές που μοιάζουν με MIPS. Με αυτόν τον τρόπο, οι τυπικές RISC δυνατότητες, όπως το χαμηλό κόστος και σταθερός χρόνος εκτέλεσης, δεν περιορίζονται πλέον μόνο στους RISC.

Η διαφορά στην απόδοση και την κατανάλωση ισχύος μεταξύ των επεξεργαστών φαίνεται να καθορίζεται από το προοριζόμενη εφαρμογή και όχι από την επιλογή του ISA. Με άλλα λόγια, στους σύγχρονους επεξεργαστές, ο τρόπος εφαρμογής της ISA, δηλαδή, η μικροαρχιτεκτονική, παίζει πιο σημαντικό ρόλο στο προσδιορισμό των επιδόσεων και των χαρακτηριστικών ισχύος από ότι την ίδια την ISA. [3]

2.4 Η αρχιτεκτονική ARM

Ο επεξεργαστής ARM είναι ένας Reduced Instruction Set Computer (RISC). Ο ARM αναπτύχθηκε αρχικά στην Acorn Computers Limited του Cambridge στην Αγγλία και ήταν ο πρώτος μικροεπεξεργαστής RISC που αναπτύχθηκε για εμπορική χρήση.

Το 1990 η ARM Limited ιδρύθηκε ως ξεχωριστή εταιρεία ειδικά για να διευρύνει την εκμετάλλευση της τεχνολογίας ARM, από τότε που ο ARM είχε αδειοδοτηθεί σε πολλούς κατασκευαστές ημιαγωγών σε όλο τον κόσμο και έχει καθιερωθεί ως ηγέτης της αγοράς για ενσωματωμένες εφαρμογές χαμηλής ισχύος και κόστους. [4]

2.4.1 The Acorn RISC Machine

Ο πρώτος επεξεργαστής ARM αναπτύχθηκε από την Acorn Computers Limited του Cambridge στην Αγγλία, μεταξύ Οκτωβρίου 1983 και Απριλίου 1985. Εκείνη την εποχή, και μέχρι το σχηματισμό της Advanced RISC Machines Limited (η οποία αργότερα μετονομάστηκε σε ARM Limited) το 1990, η ARM αντιπροσώπευε το Acorn RISC Machine.

Η Acorn είχε αναπτύξει μια ισχυρή θέση στην αγορά προσωπικών υπολογιστών στο Ηνωμένο Βασίλειο με την επιτυχία του μικροϋπολογιστή BBC micro. Μετά την επιτυχία του BBC micro, οι μηχανικοί της Acorn εξέτασαν διάφορους μικροεπεξεργαστές για την κατασκευή του επόμενου υπολογιστή, αλλά δεν βρήκαν κάποιον που να ικανοποιεί τις απαιτήσεις της εταιρείας. Οι μικροεπεξεργαστές 16-bit CISC που ήταν διαθέσιμοι το 1983 ήταν πιο αργοί από τα τυπικά μέρη μνήμης. Είχαν επίσης εντολές που απαιτούσαν πολλούς κύκλους ρολογιού για την ολοκλήρωσή τους (σε ορισμένες περιπτώσεις, πολλές εκατοντάδες κύκλους ρολογιού), εισάγοντας πολύ μεγάλες καθυστερήσεις. Το BBC micro επωφελήθηκε πολύ από τα γρήγορα interrupts του 6502, έτσι οι σχεδιαστές της Acorn δεν ήταν πρόθυμοι να δεχτούν μια οπισθοδρόμηση σε αυτήν την πτυχή της απόδοσης του επεξεργαστή. Αυτό είχε σαν αποτέλεσμα να εξεταστεί ο σχεδιασμός ενός νέου μικροεπεξεργαστή. Το σημαντικότερο εμπόδιο ήταν ότι η σχεδιαστική ομάδα της Acorn γνώριζε ότι η σχεδίαση και υλοποίηση των μικροεπεξεργαστών απαιτούσε πολυετή προσπάθεια και πολυπληθές ανθρώπινο δυναμικό και η Acorn δεν μπορούσε να επενδύσει σε αυτή την κλίμακα, δεδομένου ότι ήταν μια εταιρεία με μόλις πάνω από 400 υπαλλήλους συνολικά. Έτσι, έπρεπε να παραχθεί ένα καλύτερο σχέδιο μικροεπεξεργαστή σε ένα κλάσμα της σχεδιαστικής προσπάθειας και χωρίς καμία εμπειρία στον custom σχεδιασμό ολοκληρωμένων πέρα από μερικές μικρές συστοιχίες πυλών σχεδιασμένες για το BBC micro. [4]

Σε αυτό το φαινομενικά αδύνατο σενάριο, τα σχέδια για το Berkeley RISC ήρθαν από το πουθενά. Ήταν ένας επεξεργαστής που είχε σχεδιαστεί από μερικούς μεταπτυχιακούς φοιτητές σε λιγότερο από ένα χρόνο, αλλά ήταν ανταγωνιστικός σε σχέση με κορυφαίες εμπορικές προσφορές. Η σχεδίαση ήταν απλή και δεν υπήρχαν πολύπλοκες εντολές που θα εισήγαγαν καθυστέρηση στα interrupts.Ο ARM, γεννήθηκε μέσα από έναν απρόσμενο συνδυασμό παραγόντων και έγινε το βασικό συστατικό της σειράς προϊόντων της Acorn. Αργότερα, μετά από μια λογική τροποποίηση μετονομάστηκε σε Advanced RISC Machine, δανείζοντας το όνομά της στην εταιρεία που δημιουργήθηκε για να διευρύνει την αγορά της πέρα από τη γκάμα προϊόντων της Acorn. Παρά την αλλαγή ονόματος, η αρχιτεκτονική παραμένει κοντά στον πρωτότυπο σχεδιασμό της Acorn.

Τη στιγμή που σχεδιάστηκε το πρώτο ic ARM, τα μόνα παραδείγματα αρχιτεκτονικών RISC ήταν το Berkeley RISC I και II και το Stanford MIPS (που σημαίνει Microprocessor without Interlocking Pipeline Stages), αν και σε κάποιους παλιότερους υπολογιστές όπως το Digital PDP-8, το Cray-1 και το IBM 801, που προηγήθηκαν της

αρχιτεκτονικής RISC, μοιράστηκαν πολλά από τα χαρακτηριστικά που αργότερα συσχετίστηκαν με αυτήν. [4], [5]

2.4.2. Τα χαρακτηριστικά της αρχιτεκτονικής ARM

Η αρχιτεκτονική ARM ενσωμάτωσε μια σειρά χαρακτηριστικών από το Berkeley RISC σχεδιασμό, αλλά ορισμένα άλλα χαρακτηριστικά απορρίφθηκαν. Αυτά που ενσωματώθηκαν ήταν:

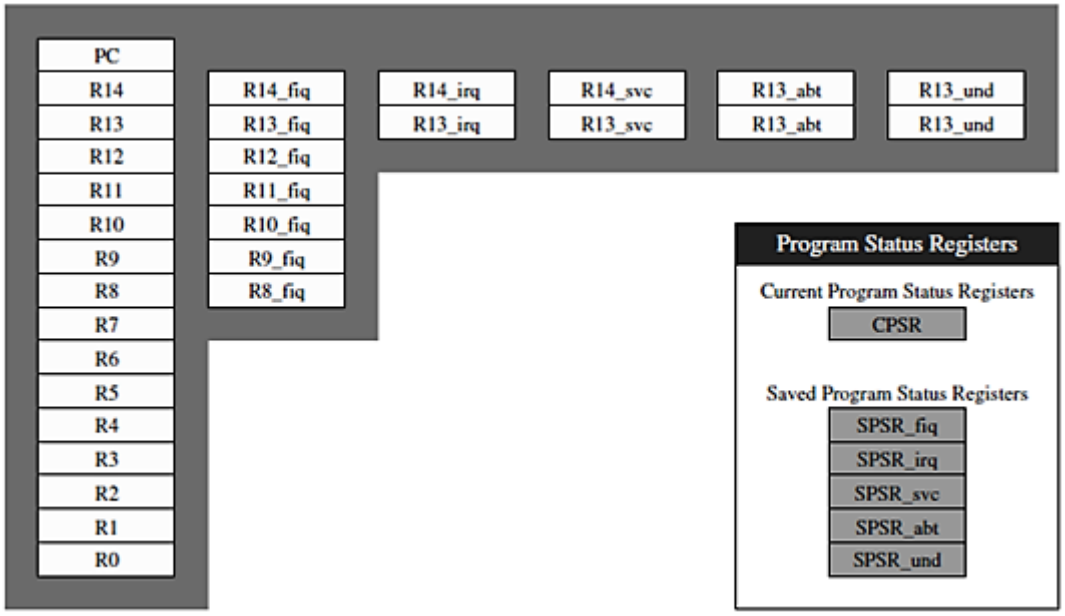
- η αρχιτεκτονική load-store.
- οι σταθερού μήκους 32-bit εντολές.
- Μορφές εντολών 3 διευθύνσεων.

Απορρίφθηκε η single-cycle εκτέλεση όλων των εντολών. Αν και ο ARM εκτελεί τις περισσότερες εντολές επεξεργασίας δεδομένων σε ένα μόνο κύκλο ρολογιού, πολλές άλλες εντολές απαιτούν πολλαπλούς κύκλους ρολογιού για την εκτέλεσή τους.

Έτσι, αντί να εκτελούνται σε ένα κύκλο ρολογιού όλες οι εντολές, ο ARM σχεδιάστηκε για να χρησιμοποιεί τον ελάχιστο αριθμό κύκλων που απαιτούνται για προσβάσεις στη μνήμη. Οπου αυτό ήταν μεγαλύτερο από ένα, οι επιπλέον κύκλοι χρησιμοποιήθηκαν όπου ήταν δυνατόν, για να κάνουν κάτι χρήσιμο. Αυτό μειώνει το συνολικό αριθμό των εντολών ARM που απαιτούνται για την εκτέλεση οποιασδήποτε ακολουθίας λειτουργιών, βελτιώνοντας έτσι την απόδοση και τη πυκνότητα του κώδικα. [6]

2.4.3. Οι καταχωρητές

Η αρχιτεκτονική ARM έχει συνολικά 37 καταχωρητές όπως φαίνεται στο σχήμα 2.3. Αυτοί οι καταχωρητές χωρίζονται σε δύο ομάδες: καταχωρητές γενικού σκοπού και program status καταχωρητές.



Σχήμα 2.3. Το σύνολο των ARM καταχωρητών αποτελείται από 31 καταχωρητές γενικού σκοπού και έξι status καταχωρητές. Όλοι οι καταχωρητές έχουν πλάτος 32 bit.

• **Καταχωρητές γενικού σκοπού:** Υπάρχουν 31 καταχωρητές γενικού σκοπού. Όλοι αυτοί οι καταχωρητές έχουν πλάτος 32 bit και ανά πάσα στιγμή, ο χρήστης μπορεί να έχει πρόσβαση μόνο σε 16 από αυτούς τους καταχωρητές.

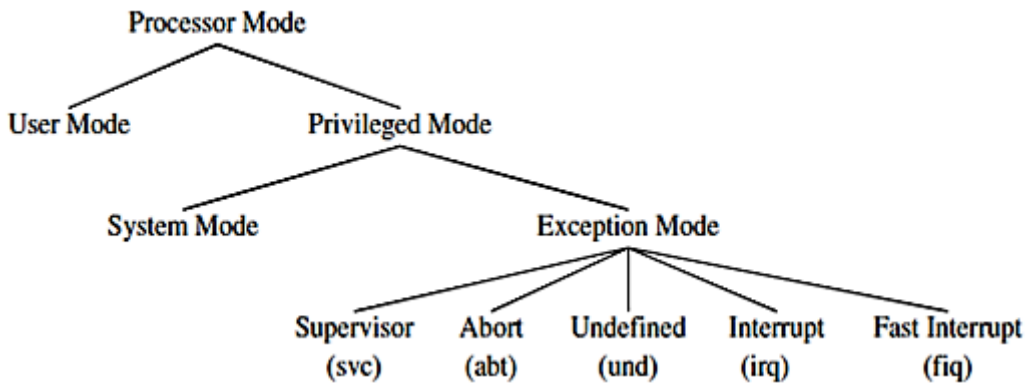
Το πραγματικό σύνολο των ορατών καταχωρητών εξαρτάται από τη κατάσταση(mode) του επεξεργαστή. Για παράδειγμα, στα User και System modes, οι αριστερότεροι 16 καταχωρητές (R0 – R14 και PC) που εμφανίζονται στο σχήμα 2.3 είναι ορατοί.

• **Program status καταχωρητές:** Ο ARM έχει έξι καταχωρητές κατάστασης που διατηρούν την κατάσταση του προγράμματος και έχουν επίσης πλάτος 32 bit. Ο Current Program Status Register (CPSR) είναι διαθέσιμος σε όλες τις καταστάσεις του επεξεργαστή. Η ορατότητα των άλλων καταχωρητών εξαρτάται από τη κατάσταση του επεξεργαστή. [4], [6]

Οι 16 καταχωρητές γενικού σκοπού χωρίζονται σε τρεις ομάδες: unbanked, banked και στον μετρητή προγράμματος(program counter). Οι πρώτοι οκτώ καταχωρητές (R0 – R7) είναι unbanked καταχωρητές. Αυτοί οι καταχωρητές είναι διαθέσιμοι σε όλες τις καταστάσεις του επεξεργαστή. Οι επόμενοι επτά καταχωρητές (R8 – R14) είναι banked καταχωρητές.

Με μερικές εξαιρέσεις, οι περισσότερες εντολές επιτρέπουν τους banked καταχωρητές να χρησιμοποιηθούν όποτε επιτρέπεται η χρήση γενικού σκοπού καταχωρητών. Ο τελευταίος καταχωρητής (R15) χρησιμοποιείται ως καταχωρητής προγράμματος Program Counter (PC).

Ο ARM έχει επτά καταστάσεις επεξεργαστή, οι οποίες χωρίζονται σε Χρήστη(User) και Προνομιακές(Priviliged Modes). Οι Priviliged modes χωρίζονται περαιτέρω σε Συστήματος(System) και Εξαίρεσης(Exception) όπως φαίνεται στο σχήμα 2.4.



Σχήμα 2.4. ARM processor modes.

Τα προγράμματα εφαρμογών εκτελούνται συνήθως στη λειτουργία χρήστη. Αυτή είναι μια μη προνομιακή λειτουργία, που περιορίζει την πρόσβαση των προγραμμάτων σε προστατευμένους πόρους. Επιπλέον, δημιουργείται μία εξαίρεση εάν επιχειρήσετε μία αλλαγή κατάστασης.

Οι υπόλοιπες έξι καταστάσεις είναι προνομιακές καταστάσεις. Αυτές οι καταστάσεις επιτρέπουν την πρόσβαση σε όλους τους πόρους του συστήματος. Επιπλέον, μπορούμε ελεύθερα να αλλάξουμε καταστάσεις συστήματος και εξαίρεσης. Η λειτουργία συστήματος έχει πρόσβαση στο ίδιο σύνολο καταχωρητών όπως και η λειτουργία χρήστη. Ωστόσο, επειδή πρόκειται για προνομιακή κατάσταση, οι περιορισμοί της λειτουργίας χρήστη δεν ισχύουν. Αυτή η κατάσταση χρησιμοποιείται από το λειτουργικό σύστημα. [5]

2.4.4. Οι τρόποι διευθυνσιοδότησης

Οι περισσότεροι επεξεργαστές υποστηρίζουν αρκετούς από αυτούς τους τρόπους διευθυνσιοδότησης (αν και λίγοι τις υποστηρίζουν όλες):

1. **Immediate addressing:** η επιθυμητή τιμή εμφανίζεται ως δυαδική τιμή στην εντολή.

2. **Absolute addressing:** η εντολή περιέχει την πλήρη δυαδική διεύθυνση της επιθυμητής τιμής στη μνήμη.

3. **Indirect addressing:** η εντολή περιέχει τη δυαδική διεύθυνση μιας τοποθεσίας μνήμης που περιέχει τη δυαδική διεύθυνση της επιθυμητής τιμής.

4. **Register addressing:** η επιθυμητή τιμή βρίσκεται σε έναν καταχωρητή και η εντολή περιέχει τον αριθμό καταχωρητή.

5. **Register indirect addressing:** η εντολή περιέχει τον αριθμό ενός καταχωρητή που περιέχει τη διεύθυνση της τιμής στη μνήμη.

6. **Base plus offset addressing:** η εντολή καθορίζει έναν καταχωρητή(τη βάση) και ένα δυαδικό όφσετ που θα προστεθεί στη βάση για να σχηματιστεί η διεύθυνση μνήμης.

7. **Base plus index addressing:** η εντολή καθορίζει έναν καταχωρητή βάσης και έναν άλλο καταχωρητή(το ευρετήριο) που προστίθεται στη βάση για να σχηματίσει τη διεύθυνση μνήμης.

8. **Base plus scaled index addressing**: όπως παραπάνω, αλλά ο δείκτης πολλαπλασιάζεται με ένα σταθερά(συνήθως το μέγεθος του στοιχείου δεδομένων) πριν να προστεθεί στη βάση.

9. **Stack addressing:** ένας έμμεσος ή καθορισμένος καταχωρητής(ο δείκτης στοίβας) δείχνει σε μία περιοχή μνήμης(τη στοίβα) όπου τα στοιχεία δεδομένων γράφονται ή διαβάζονται σε μία βάση last-in-first-out.

Να σημειωθεί ότι οι ονομασίες που χρησιμοποιούνται για αυτούς τους τρόπους διευθυνσιοδότησης από διαφορετικούς κατασκευαστές επεξεργαστών δεν είναι απαραίτητα όπως οι παραπάνω. Η λίστα μπορεί να επεκταθεί σχεδόν απεριόριστα με τη προσθήκη περισσότερων επιπέδων έμμεσης, προσθήκης βάσης, ευρετηρίου, offset και ούτω καθεξής διευθυνσιοδοτήσεων. Ωστόσο, οι περισσότερες από τις κοινές λειτουργίες διευθυνσιοδοτήσεων καλύπτονται από την παραπάνω λίστα. [4], [5], [6]

2.4.5. Το σετ των εντολών

Οι κατηγορίες των εντολών ARM

Οι εντολές ARM γενικά μπορούν να κατηγοριοποιηθούν ως εξής:

- **Data Processing Instructions:** επεξεργασία δεδομένων μέσα στους registers.
- **Branch Instructions:** αλλαγή της ροής των εντολών ή κλήση υπορουτίνας.
- **Load-Store Instructions:** μεταφορά δεδομένων μεταξύ registers και μνήμης.
- **Software Interrupt Instruction:** προκαλεί μία διακοπή interrupt.
- **Program Status Instructions**: διάβασμα/γράψιμο των καταχωρητών κατάστασης του επεξεργαστή. [9]

Το σετ των εντολών:

Ο ARM διαθέτει ένα συμβατικό σετ εντολών ακέραιας επεξεργασίας δεδομένων με παραδοσιακές αριθμητικές, λογικές και ολίσθησης(shift) λειτουργίες (αν και η ολίσθηση εφαρμόζεται με ασυνήθιστο τρόπο). Μία ξεχωριστή εντολή είναι η MLA (πολλαπλασιασμός και πρόσθεση) που δέχεται τέσσερις τελεστές και έχει τη μορφή MLA r1, r2, r3, r4. Του αποτέλεσμά της είναι r1 = r2 * r3 + r4; δηλαδή υπολογίζει ένα γινόμενο και το προσθέτει σε μία προηγούμενη τιμή. Αυτή η φαινομενικά αθώα εντολή βρίσκεται στο επίκεντρο πολλών επεξεργασιών σήματος. [8], [9]

Ο ARM διαθέτει load/store(από καταχωρητή σε καταχωρητή) RISC αρχιτεκτονική. Οι επεξεργαστές RISC ονομάζονται και μηχανές της 1½ διευθύνσης επειδή επιτρέπουν εντολές της μορφής ADD T1, D1 που προσθέτει τα περιεχόμενα της θέσης μνήμης T1 στον καταχωρητή D1 και τοποθετεί το άθροισμα στον D1, αντικαθιστώντας την παλιά τιμή. Ο όρος 1 ½ χρησιμοποιείται για να υποδείξει μια πλήρη διεύθυνση μνήμης και μια σύντομη διεύθυνση καταχωρητή. Οι επεξεργαστές RISC επιτρέπουν λειτουργίες επεξεργασίας δεδομένων μόνο σε καταχωρητές και παρέχουν εντολές της μορφής ADD r1, r2, r3 όπου ο καταχωρητής r2 προστίθεται στον r3 και το άθροισμα τοποθετείται στον r1. Οι μόνες λειτουργίες σχετικές με τη μνήμη είναι η φόρτωση δεδομένων στον

καταχωρητή από τη μνήμη και η αποθήκευση των περιεχομένων ενός καταχωρητή στη μνήμη. [5], [8]

Το σετ των εντολών Thumb-2

Οι ARM επεξεργαστές μπορούν να εκτελέσουν δύο σετ εντολών.Το 32-bit σετ εντολών ARM και το 16-bit σετ εντολών Thumb. Αυτό δίνει τη δυνατότητα στον προγραμματιστή να βελτιστοποιήσει το πρόγραμμά του επιλέγοντας ποιό σετ εντολών θα χρησιμοποιήσει για τις διάφορες διαδικασίες. Το σετ των 32-bit επιλέγεται για μεγαλύτερη ταχύτητα και το σετ των 16-bit για ελαχιστοποίηση του μεγέθους του κώδικα. Η κεντρική μονάδα επεξεργασίας των επεξεργαστών όπως για παράδειγμα οι Cortex, χρησιμοποιεί το σετ εντολών Thumb-2 που είναι μία μίξη των σετ εντολών των 16-bit και των 32-bit. Τα μεγάλα του πλεονεκτήματα είναι ότι προσφέρει βελτίωση στην πυκνότητα του κώδικα κατά 26% σε σχέση με το σετ εντολών 32-bit ARM και βελτίωση στην απόδοση κατά 25% σε σχέση με το σετ εντολών Thumb των 16-bit.Το σετ εντολών Thumb-2 περιέχει επίσης βελτιωμένες εντολές πολλαπλασιασμού που εκτελούνται σε ένα κύκλο μηχανής όπως και διαίρεση του hardware που εκτελείται σε χρονικό διάστημα μεταξύ δύο και εφτά κύκλων μηχανής. Συμπερασματικά,το σετ εντολών Thumb-2 είναι ένας ακόμα λόγος για τον οποίο οι ARM επεξεργαστές είναι ιδανικοί για embedded εφαρμογές που συνδυάζουν την ανάγκη για χαμηλή χρήση της μνήμης και υψηλή απόδοση. [2], [9]

ΒΙΒΛΙΟΓΡΑΦΙΑ 2ου ΚΕΦΑΛΑΙΟΥ

[1] POWER STRUGGLES: REVISITING THE RISC VS. CISC DEBATE ON CONTEMPORARY ARM AND X86 ARCHITECTURES, Ernily Blern, laikrishnan Menon, and Karthikeyan Sankaralingarn, University of Wisconsin – Madison, ©2013 IEEE.

[2] «ΜΕΛΕΤΗ ΚΑΙ ΑΞΙΟΛΟΓΗΣΗ ΚΥΡΙΑΡΧΩΝ ΠΕΡΙΒΑΛΛΟΝΤΩΝ ΑΝΑΠΤΥΞΗΣ ΑΛΓΟΡΙΘΜΩΝ ΓΙΑ ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΜΕ ΕΜΦΑΣΗ ΣΤΗΝ ΠΕΙΡΑΜΑΤΙΚΗ ΑΠΟΤΙΜΗΣΗ ΤΗΣ ΑΠΟΔΟΣΗΣ ΚΑΙ ΚΑΤΑΝΑΛΩΣΗΣ ΜΙΑΣ ARM ΑΝΑΠΤΥΞΙΑΚΗΣ ΠΛΑΤΦΟΡΜΑΣ», Αθανάσιος Θανόπουλος, Αντώνιος Σιτοκωνσταντίνου, Πτυχιακή Εργασία, Τ.Ε.Ι. Δυτικής Ελλάδας, Τμήμα Μηχανικών Πληροφορικής Τ.Ε., Ναύπακτος 2015.

[3] A COMPARISON OF RISC AND CISC ARCHITECTURES Chevtchenko, S. F.1; Vale, R. F.2, Department of Statistics and Informatics, UFRPE Recife, Brazil, 2018.

[4] GUIDE TO RISC PROCESSORS FOR PROGRAMMERS AND ENGINEERS, Sivarama P. Dandamudi, © 2005 Springer Science+Business Media, Inc.

[5] COMPUTER ORGANIZATION AND ARCHITECTURE DESIGNING FOR PERFORMANCE, Tenth Edition, William Stallings, With contribution by Peter Zeno, University of Bridgeport, Copyright © 2016, 2013, 2010 Pearson Education, Inc., Hoboken, NJ 07030.

[6] ARM SYSTEM-ON-CHIP ARCHITECTURE Book by Steve Furber, 2000.

[7] RISC AND CISC COMPUTER ARCHITECTURE, By Farhat Masood BE – Electrical (NUST), College Of Electrical And Mechanical Engineering National University Of Sciences And Technology (Nust), 2011.

[8] ARMs FOR THE POOR: SELECTING A PROCESSOR FOR TEACHING COMPUTER ARCHITECTURE, Alan Clements, ©2010 IEEE, October 27 - 30, 2010, Washington, DC.

[9] «ΚΑΤΑΣΚΕΥΗ ΣΥΣΤΗΜΑΤΟΣ ΑΝΙΧΝΕΥΣΗΣ ΚΑΙ ΚΑΤΑΓΡΑΦΗΣ ΚΙΝΗΣΗΣ ΑΥΤΟΚΙΝΗΤΟΥ ΣΕ ΜΝΗΜΗ SD Ή MMC», Διπλωματική Εργασία, Παπακώστας Ιωάννης, Πανεπιστήμιο Πατρών Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών - Τομέας:Τηλεπικοινωνιών και Τεχνολογίας Πληροφορίας Εργαστήριο Ενσύρματης Τηλεπικοινωνίας, Φεβρουάριος 2013.

ΚΕΦΑΛΑΙΟ 3:
Τεχνικές χαμηλής κατανάλωσης ισχύος σε επίπεδο hardware

3.1 Εισαγωγή

Ένας από τους πιό σημαντικούς περιορισμούς στο σχεδιασμό ψηφιακών συστημάτων είναι η κατανάλωση ισχύος. Η ανάγκη για χαμηλή κατανάλωση ισχύος, ξεκινά από την ολοένα και αυξανόμενη ολοκλήρωση η οποία σύμφωνα με το νόμο του Moore διπλασιάζεται ανά δεκαοκτώ μήνες με αποτέλεσμα να αυξάνει η πυκνότητα ισχύος.

Ένας δεύτερος παράγοντας για την ανάγκη χαμηλής κατανάλωσης ισχύος είναι η αυξανόμενη ζήτηση στην αγορά, φορητών συσκευών τροφοδοτούμενες από μπαταρίες. Η διάρκεια της μπαταρίας γίνεται έτσι ο σημαντικότερος παράγοντας κατά τη σχεδίαση ενός φορητού συστήματος και η χαμηλή κατανάλωση το κρίσιμο στοιχείο της σχεδίασης. Ο περιορισμός αυτός ωστόσο ισχύει και για τον σχεδιασμό σταθερών συστημάτων (π.χ. υπολογιστές), λόγω των απαιτήσεων της θερμοκρασίας, του εμβαδού και του packaging.

Τρίτος παράγοντας είναι η αυξανόμενη απόδοση των υπολογιστικών συστημάτων μέσω της αύξησης της συχνότητας λειτουργίας. Η ισχύς είναι ανάλογη της συχνότητας λειτουργίας και στους σημερινούς επεξεργαστές υψηλής απόδοσης μπορεί να φθάσει τα δεκάδες watts. Ειδικά για τους επεξεργαστές, η κατανάλωση ισχύος έχει φτάσει σε μικρό χρονικό διάστημα από μερικά watts σε δεκάδες, και σε ακόμα περισσότερα.

Οι υπάρχουσες τεχνικές, μπορούν να διαχωριστούν σε εκείνες που εφαρμόζονται κατά τη σχεδίαση και εκείνες που εφαρμόζονται όταν το ολοκληρωμένο βρίσκεται σε χρήση (run-time). Η run-time κατηγορία μπορεί να χωριστεί σε δύο υποκατηγορίες, εκείνη των τεχνικών που εφαρμόζονται όταν το σύστημα είναι σε κατάσταση αναμονής-αδράνειας (idle) και εκείνες που εφαρμόζονται όταν το σύστημα βρίσκεται σε κανονική λειτουργία.

Μία άλλη κατηγοριοποίηση των τεχνικών μπορεί να οριστεί ως τεχνικές χαμηλής κατανάλωσης ισχύος σε επίπεδο κυκλώματος-λογικής και σε επίπεδο αρχιτεκτονικής-συστήματος. [3]

3.2 Βασικές αρχές σχεδίασης των τεχνικών μείωσης της κατανάλωσης ισχύος και παράγοντες που την επηρεάζουν

Η σχεδίαση με τη χρήση τεχνικών χαμηλής κατανάλωσης, εστιάζεται τόσο στη μείωση της δυναμικής (dynamic ή switching) κατανάλωσης ισχύος που εξαρτάται από την χωρητικότητα εξόδου του αναστροφέα, όσο και στη μείωση της κατανάλωσης ισχύος διαρροής (leakage) η οποία είναι πάντα παρούσα και δεν εξαρτάται από την

δραστηριότητα της πύλης. Στη δυναμική κατανάλωση ισχύος, πρέπει να συμπεριληφθεί και η ισχύς βραχυκυκλώματος, που οφείλεται στην αδυναμία των τρανζίστορς να τεθούν στιγμιαία σε κατάσταση on και off.

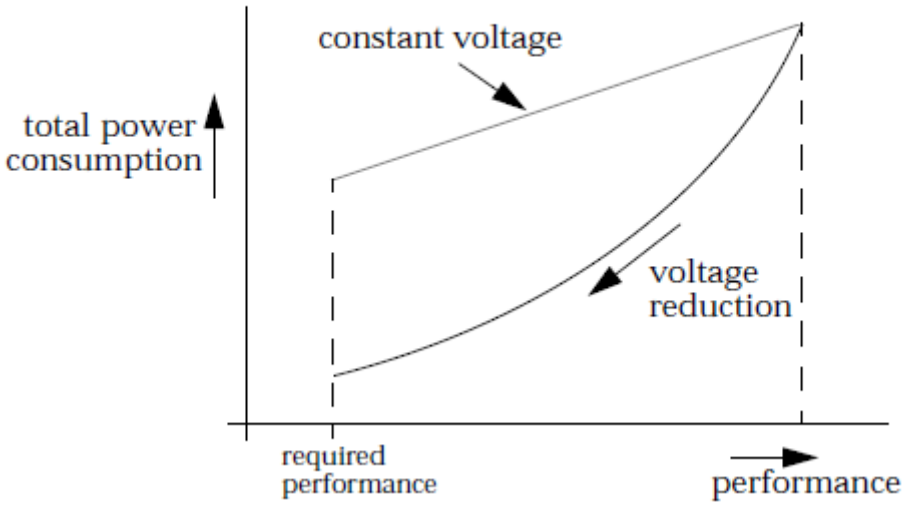
Κύριοι στόχοι είναι η μείωση της τάσης τροφοδοσίας, της συχνότητας λειτουργίας (και κυριότερα της δραστηριότητας μεταγωγής-switching activity) και η μείωση των παρασιτικών χωρητικοτήτων. [15] Γενικά, η μείωση της τάσης τροφοδοσίας V_{DD}, είναι το πιο αποτελεσματικό μέσο για τη μείωση της κατανάλωσης ισχύος της σχεδίασης CMOS. Με χαμηλή V_{DD} ωστόσο, η καθυστέρηση του κυκλώματος αυξάνεται και η απόδοση του chip υποβαθμίζεται. Ωστόσο υπάρχουν δύο τρόποι για να διατηρηθεί η απόδοση του chip και αυτοί είναι α) να μειωθεί η τάση κατωφλίου Vth για να ανακτηθεί η ταχύτητα του κυκλώματος, ή β) να εισαχθούν παράλληλες ή/και αρχιτεκτονικές διασωληνώσεων (pipelining), για την κάλυψη της απώλειας ταχύτητας της συσκευής. Η πρώτη προσέγγιση αυξάνει την κατανάλωση της ισχύος standby λόγω χαμηλού Vth, ενώ η δεύτερη προσέγγιση υποβαθμίζει τη worst case ταχύτητα του κυκλώματος που προκαλείται από τη διακύμανση της Vth λόγω χαμηλής V_{DD}. [9]

Η δυναμική ισχύς και το ρεύμα διαρροής είναι οι κύριες πηγές κατανάλωσης ενέργειας, με τη καταναλισκόμενη ισχύ να επηρεάζεται άμεσα από την συχνότητα με την οποία η έξοδος αλλάζει και πιο συγκεκριμένα εξαρτάται από δυο συνιστώσες, την συχνότητα με την οποία το κύκλωμα λειτουργεί (ταχύτητα ρολογιού, f) και τον συντελεστή μετάβασης (activity factor, α) των πυλών. Ο συντελεστής αυτός είναι μια εκτίμηση του πόσο συχνά μια πύλη μεταβάλει την έξοδό της. Έτσι ο υπολογισμός της δυναμικής ισχύος δίδεται από την παρακάτω εξίσωση:

$$P_{dynamic} = C*V^2*f*a$$

Η κατανάλωση ισχύος που οφείλεται στο ρεύμα διαρροής καθορίζεται από την τάση της πηγής και από την επιφάνεια των πυλών. [8]

Δύο συνιστώσες μπορούν να αλλάξουν κατά την διάρκεια της δυναμικής κατανάλωσης και αυτές είναι η τάση και η συχνότητα λειτουργίας του κυκλώματος. Διακόπτοντας ή μεταβάλλοντας, είτε την τάση είτε τη συχνότητα του ρολογιού του κυκλώματος, η ισχύς θα μειωθεί με συνέπεια να μειωθεί δραστικά η ενέργεια που καταναλώνεται. Απαιτούνται δηλαδή, τεχνικές τέτοιες που θα ενεργοποιούν/απενεργοποιούν ή θα μεταβάλλουν την τάση ή/και το ρολόι ανάλογα με τις απαιτήσεις του συστήματος. Δυστυχώς αυτές οι μέθοδοι ελαττώνουν την ταχύτητα λειτουργίας του τρανζίστορ και την υπολογιστική απόδοση όπως φαίνεται και στο Σχήμα 3.1.



Σχήμα 3.1: Επιπτώσεις της κλιμάκωσης τάσης και της απόδοσης στη συνολική κατανάλωση ισχύος. [6]

Για να εξοικονομηθεί ενέργεια χωρίς απώλεια στην απόδοση, πρέπει να εντοπιστούν τα μέρη του κυκλώματος που υπάρχει σπατάλη ενέργειας. Υπάρχουν μέρη στο κύκλωμα όπου η switching ταχύτητα λειτουργίας είναι μεγαλύτερη από όσο χρειάζεται. Σε ορολογία ανάλυσης χρονισμού, το περιθώριο χρόνου (slack) στο κύκλωμα πρέπει να εντοπιστεί και να μειωθεί με τέτοιο τρόπο ώστε να εξοικονομηθεί ενέργεια. [2]

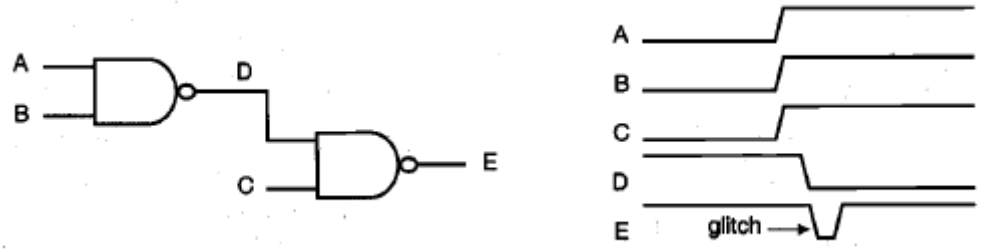
Το μέγεθος των λογικών πυλών καθορίζει την καθυστέρηση διάδοσης σήματος και επίσης την κατανάλωση ισχύος. Έτσι για τις πύλες στην κρίσιμη διαδρομή του σήματος, μπορεί να αλλαχθεί το μέγεθος για να μεγιστοποιηθεί η ταχύτητα. Και στις πύλες σε μη κρίσιμες διαδρομές, δηλαδή σε διαδρομές που έχουν περιθώριο χρόνου μπορούν να αλλάξουν μέγεθος ώστε να γίνει πιο αργή η διάδοση σήματος και για να επιτευχθεί η μείωση της κατανάλωσης ενέργειας.

Εκτός από το μέγεθος του τρανζίστορ και η τοπολογία του κυκλώματος επηρεάζει την ισοστάθμιση (το trade-off) μεταξύ κατανάλωσης ενέργειας και απόδοσης.

Η επιλογή λογικής οικογένειας επίσης παίζει ρόλο ρτην κατανάλωση ισχύος. Για ιδιαίτερα κρίσιμες διαδρομές, ως προς το χρόνο, χρησιμοποιείται δυναμική λογική με αυξημένη κατανάλωση δυναμικής ισχύος. Εάν η καθυστέρηση διάδοσης σήματος δεν είναι κρίσιμη χρησιμοποιείται στατική λογική CMOS.

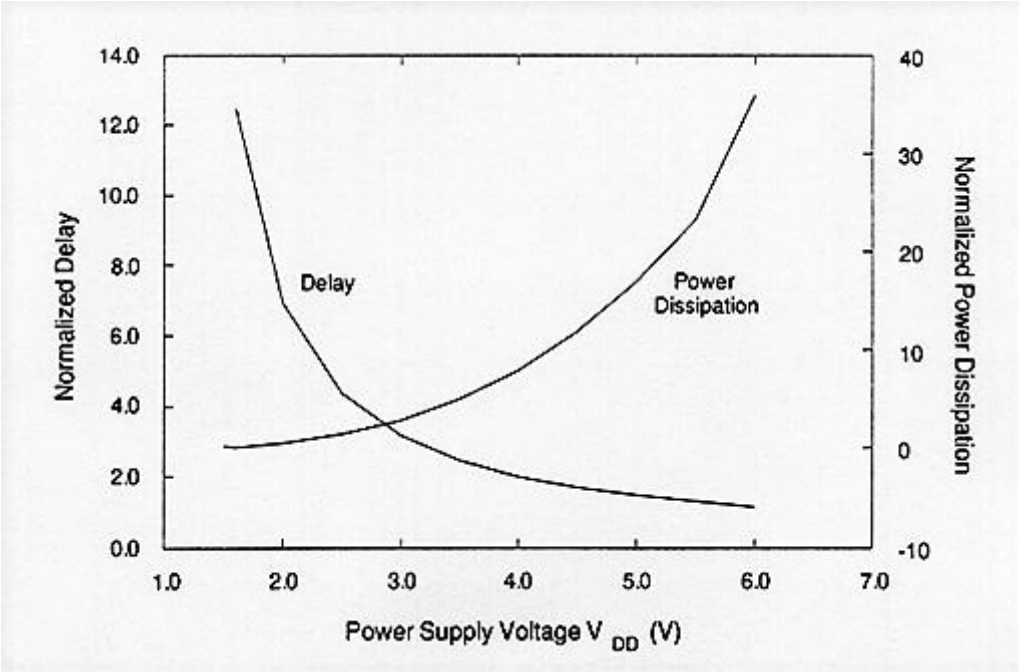
Ένα 20% και περισσότερο της δυναμικής κατανάλωσης προέρχεται από ανεπιθύμητες μεταβάσεις - glitches (Σχήμα 3.2) που προκύπτουν από ανεπιθύμητες καθυστερήσεις των διαφορετικών διαδρομών σήματος και η καθυστέρηση στη χειρότερη περίπτωση, εξαρτάται από την καθυστέρηση στη πιο αργή διαδρομή. Η επιλογή κατάλληλων μεγεθών των τρανζίστορς που βρίσκονται στις γρήγορες

διαδρομές μπορεί να χρησιμοποιηθεί για την εξισορρόπηση των καθυστερήσεων και την αποφυγή άσκοπων μεταβάσεων πριν την τελική μετάβαση της λογικής. [2]



Σχήμα 3.2: Signal glitching.

Μία τεχνική μείωσης της κατανάλωσης ισχύος, είναι η χρήση τοπικά μειωμένης τάσης τροφοδοσίας. Μείωση της τάσης τροφοδοσίας στις πύλες ή στα μπλοκ του κυκλώματος που δεν είναι κρίσιμα οδηγεί σε μείωση του περιθωρίου χρόνου αλλά ταυτόχρονα μειώνει δραστικά την δυναμική κατανάλωση. Αυτή η κλιμάκωση της τάσης τοπικά, είναι γνωστή ως τεχνική πολλαπλών τάσεων (Multi – Scale Voltage – MSV-Multi V_{DD}) τεχνική πολλαπλών τάσεων τροφοδοσίας. Αυτή η τεχνική έχει να κάνει με τη λειτουργία τμημάτων (blocks) της υλοποίησης σε διαφορετικές τάσεις. Η μεταβολή της τάσης θα έχει μεν τετραπλή επίδραση στην συνολικά καταναλισκόμενη ισχύ, μειώνει όμως τη συνολικά απόδοση του συστήματος, μιας και αυξάνονται οι καθυστερήσεις σε επίπεδο πύλης (Σχήμα 3.4). Τέλος, απαιτεί την εισαγωγή level shifters για σήματα μεταξύ blocks που λειτουργούν με διαφορετική τάση, για την ορθή επικοινωνία τους.



Εικόνα 3.4 Η Σχέση τάσης τροφοδοσίας με τη καθυστέρηση και τη κατανάλωσης ισχύος.

Σημαντική επίσης κατανάλωση ενέργειας, που δεν επηρεάζει το περιθώριο χρόνου, οφείλεται σε εκείνα τα λογικά ενεργά κυκλώματα που δεν συνεισφέρουν στη λειτουργικότητα του συστήματος. Το ρολόι αυτών των κυκλωματικών μπλοκ μπορεί να απενεργοποιηθεί με τη χρήση λογικής (παρεμβολή πυλών στο δέντρο ρολογιού) για να αποφευχθούν οι μεταβάσεις σήματος (switching activity) και η κατανάλωση ενέργειας σε αυτά τα μπλοκ. [15] Αυτή η τεχνική χαμηλής κατανάλωσης με απενεργοποίηση του ρολογιού κατά τμήματα (Clock Gating Technique) μπορεί να μειώσει σημαντικά τη δυναμική κατανάλωση μέσα σε ένα ολοκληρωμένο κύκλωμα.

Στα μη ενεργά κυκλωματικά μπλοκ, οι είσοδοι υπό κανονικές συνθήκες, δεν αλλάζουν. Τότε η δυναμική κατανάλωση του μπλοκ καθορίζεται από το δέντρο διανομής του σήματος ρολογιού (clock tree). Επειδή αυτά τα στοιχεία του κυκλώματος έχουν δραστηριότητα μετάβασης 100%, η δυναμική κατανάλωση μπορεί να φθάσει και το 30% της συνολικής δυναμικής κατανάλωσης. Συνεπώς, η τεχνική απενεργοποίησης του ρολογιού κατά τμήματα είναι μια αποτελεσματική τεχνική ακόμα και όταν οι είσοδοι δεν αλλάζουν.

Το κύριο μειονέκτημα της τεχνικής απενεργοποίησης του ρολογιού κατά τμήματα είναι ότι τα ρεύματα τροφοδοσίας παρουσιάζουν απότομες μεταβάσεις λόγω της ενεργοποίησης και απενεργοποίησης μεγάλων υποσυστημάτων. Εάν ο ρυθμιστής τάσης (voltage regulator) δεν έχει σχεδιαστεί να λειτουργεί γρήγορα και χωρίς μεγάλες αιχμές η τάση στο ολοκληρωμένο μπορεί να διαταραχθεί και να οδηγήσει σε δυσλειτουργία τα ενεργά κυκλωματικά μπλοκ. [2]

Οι τεχνικές κλιμάκωσης συχνότητας και τάσης (Dynamic Voltage & Frequency Scaling - DVFS) χρησιμοποιούνται για τη στοχευμένη αυξομείωση της τάσης ή/και συχνότητας, ανάλογα με τις απαιτήσεις της εφαρμογής που χρησιμοποιεί το ολοκληρωμένο. Μιας και επηρεάζουν ταυτόχρονα την τάση και τη συχνότητα, είναι αποδοτικές για τη μείωση και της δυναμικής και της στατικής ισχύος.

Για την υλοποίηση αυτής της τεχνικής απαιτείται η χρήση θερμικών αισθητήρων στο ολοκληρωμένο, η παρακολούθηση του φόρτου του επεξεργαστή από το σύστημα και δύο μονάδες, μία για τη ρύθμιση της τάσης (converter) και μία για το ρολόϊ (Phase locked loop - PLL) αντίστοιχα. Η δυναμική κλιμάκωση τάσης (Dynamic Voltage Scaling - DVS) είναι υποπερίπτωση των DVFS τεχνικών που χρησιμοποιείται για τη μεταβολή μόνο της τάσης λειτουργίας ενός block. Ομοίως, η δυναμική κλιμάκωση συχνότητας (Dynamic Frequency Scaling - DFS) αποτελεί υποπερίπτωση των DVFS τεχνικών που χρησιμοποιείται για τη μεταβολή μόνο της συχνότητας λειτουργίας ενός block. [15]

Μια αποτελεσματική τεχνική για την μείωση των απωλειών στο κύκλωμα όταν αυτό βρεθεί σε αδράνεια, είναι η δημιουργία ενδιάμεσων καταστάσεων ύπνου, όπου τα τρανζίστορς απομόνωσης («ύπνου») δεν οδηγούνται πλήρως στην αποκοπή. Παρόλο

που η διαρροή σε αυτή την περίπτωση είναι αυξημένη σε σχέση με την προσέγγιση της εισαγωγής καταστάσεων «ύπνου» με την χρήση power gating τεχνικής, εξακολουθούμε να έχουμε μείωση της καταναλισκόμενης ενέργειας με ταυτόχρονη όμως μείωση του χρόνου επαναφοράς στην κανονική λειτουργία. Κατά αυτόν τον τρόπο, η ενεργοποίηση του μηχανισμού μπορεί να γίνει συχνότερα (δηλ. για αδρανείς περιόδους που ικανοποιούν λιγότερο αυστηρά χρονικά κριτήρια) αυξάνοντας το συνολικό ενεργειακό κέρδος. Ταυτόχρονα, ανάλογα με το χρόνο αδράνειας του κυκλώματος, αυτό μπορεί να περνά διαδοχικά από ελαφρύτερες καταστάσεις ύπνου, σε βαθύτερες ώστε να επιτευχθούν τα μέγιστα δυνατά ενεργειακά οφέλη. [16]

3.3 Μετρικές ισχύος και απόδοσης

Ποιες μετρικές όμως χρησιμοποιούνται για την μέτρηση της απόδοσης ενός συστήματος ως προς την κατανάλωση, την αποτελεσματικότητα μία τεχνικής μείωσης κατανάλωσης, μίας λογικής συνάρτησης ή ενός τρόπου σχεδιασμού;

Ο πιο άμεσος τρόπος μέτρησης είναι η κατανάλωση ισχύος σε Watt και συνήθως εκφράζει την ολική ισχύ του συστήματος. Για τη μέτρηση της απόδοσης ισχύος ενός ολοκληρωμένου κυκλώματος, χρησιμοποιείται η μονάδα μW/MHz, η οποία εκφράζει την κατανάλωση ενέργειας ανά κύκλο.

Συγκρίνοντας επεξεργαστές, με διαφορετικά σύνολα εντολών ή αρχιτεκτονικές, η μονάδα μW/MHz δεν είναι πλέον επαρκής, διότι η ίδια εντολή μπορεί να διαρκεί διαφορετικό αριθμό κύκλων μηχανής σε κάθε επεξεργαστή. Στις περιπτώσεις αυτές χρησιμοποιείται ένα πιο αντικειμενικό μέτρο, τα μW/MIPS (MIPS: Million Instructions Per Second). Όταν συγκρίνονται επεξεργαστές με διαφορετικές αρχιτεκτονικές, τα μW/MIPS δημιουργούν το ίδιο πρόβλημα με τα μW/MHz. Ένας επεξεργαστής RISC χρησιμοποιεί περισσότερες εντολές για να κάνει κάποια λειτουργία από έναν επεξεργαστή CISC. Επίσης στις αρχιτεκτονικές VLIW (Very Long Instruction Word) μπορούν να εκτελεστούν πολλές λειτουργίες ταυτόχρονα με μία μόνο εντολή. Απαιτείται έτσι η χρήση μετρικής που να αναφέρεται στην αποδοση του επεξεργαστή. Μία τέτοια μετρική είναι το SPEC, δηλαδή η υπολογιστική ταχύτητα που προκύπτει από την εκτέλεση συγκεκριμένου προγράμματος γραμμένου σε γλώσσα υψηλού επιπέδου. Η μονάδα που προκύπτει είναι τα μW/SPEC. Άλλες μετρικές ισχύος-απόδοσης που χρησιμοποιούνται είναι οι $BIPS^3/W$, $BIPS^2/W$ και BIPS/W, έχοντας ως μετρική της απόδοσης τα δισεκατομμύρια οδηγίες ανά δευτερόλεπτο (billions of instructions per second BIPS), όπως και οι ανάστροφες μετρικές W/BIPS και $W/BIPS^2$.

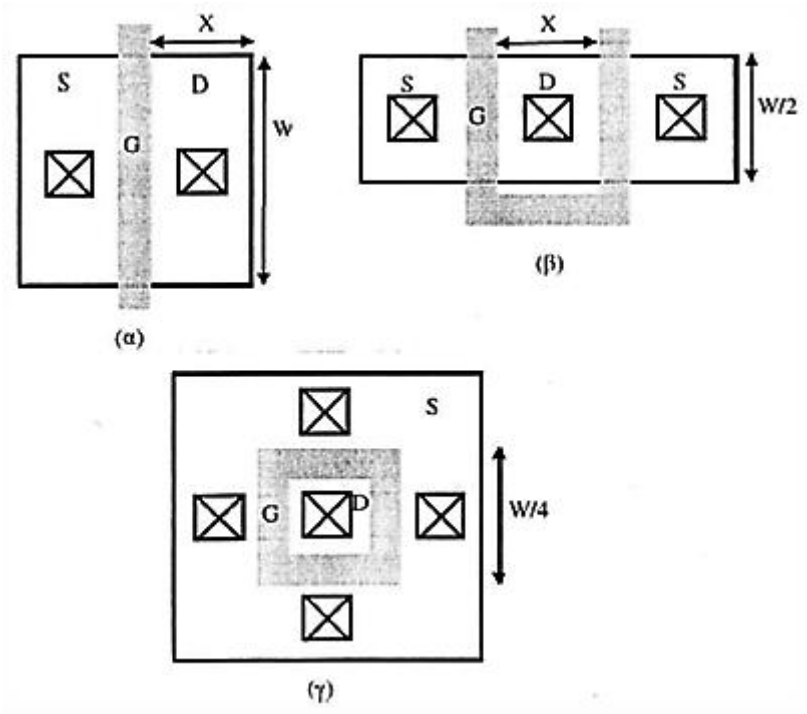
Πολλές άλλες μετρικές χρησιμοποιούνται ανάλογα με το σκοπό. Η επιλογή της μετρικής εξαρτάται από το τύπο της εφαρμογής και τον τρόπο ανάλυσης. Δεν υπάρχει ένα μοναδικό μέτρο το οποίο να αποδίδει καλά σε όλες τις περιπτώσεις, όπως δεν υπάρχει και ένας αποτελεσματικός τρόπος μείωσης της ισχύος για όλες τις εφαρμογές. [33]

3.4 Τεχνικές χαμηλής κατανάλωσης ισχύος σε κυκλωματικό και λογικό επίπεδο

Ακολούθως, παρουσιάζονται οι βασικές τεχνικές ελάττωσης της κατανάλωσης ισχύος σε κυκλωματικό και λογικό επίπεδο. Κάποιες από αυτές εφαρμόζονται σε συμβατική τεχνολογία CMOS, ενώ οι υπόλοιπες απαιτούν τεχνολογία CMOS με τρανζίστορς χαμηλής και υψηλής τάσης κατωφλιού. Μπορούν επίσης να συνδυαστούν πολλές από τις αναφερθείσες μεθοδολογίες, αυξάνοντας το συνολικό κέρδος στη κατανάλωση ισχύος.

3.4.1 Βελτιστοποίηση της τοπογραφίας (Layout optimization)

Με κατάλληλη βελτιστοποίηση της τοπογραφίας, υπάρχει η δυνατότητα να μειωθούν οι παρασιτικές χωρητικότητες των τρανζίστορς, προκαλώντας την μείωση της κατανάλωσης ισχύος. Στο Σχήμα 3.5 παρουσιάζονται τρεις διαφορετικές τοπογραφίες ενός τρανζίστορ μεγάλου μήκους.



ΣΧΗΜΑ 3.5: Διαφορετικές τοπογραφίες τρανζίστορς μεγάλου μήκους.

Η πρώτη αποτελεί την κλασσική τοπογραφία ενός τρανζίστορ μήκους W. Στην περίπτωση αυτή η χωρητικότητα της επαφής του απαγωγού(drain) είναι:

$$C_D = C_{jb}\,(W*X) + C_{jsw}\,(W+2X)$$

όπου C_{jb} είναι η χωρητικότητα οριζόντιας επιφάνειας και C_{jsw} η χωρητικότητα πλευρικής επιφάνειας της επαφής pn μεταξύ απαγωγού και υποστρώματος.

Η δεύτερη τοπογραφία, χρησιμοποιείται προκειμένου να ελαττωθεί η χωρητικότητα της επαφής του απαγωγού η οποία είναι:

$$C_D = C_{jb}((W/2)*X) + C_{jsw}(2X)$$

Για πολύ μεγάλα τρανζίστορς η πύλη είναι δυνατόν να αποτελείται από περισσότερα των δύο τμημάτων.

Τέλος στην τρίτη περίπτωση, ελαχιστοποιείται η συνεισφορά της χωρητικότητας πλευρικής επιφάνειας της επαφής pn μεταξύ απαγωγού και υποστρώματος. Άρα η χωρητικότητα του απαγωγού θα είναι:

$$C_D = C_{jb}*((W^2)/2)$$

Κατά τη φάση της τοποθέτησης (placement) και δρομολόγησης (routing) των τρανζίστορς, πρέπει να ελαχιστοποιείται το μήκος των γραμμών διασύνδεσης των σημάτων υψηλής δραστηριότητας μετάβασης για μια σχεδίαση χαμηλής κατανάλωσης ισχύος. [3]

3.4.2 Ελαχιστοποίηση λογικής (Logic minimization)

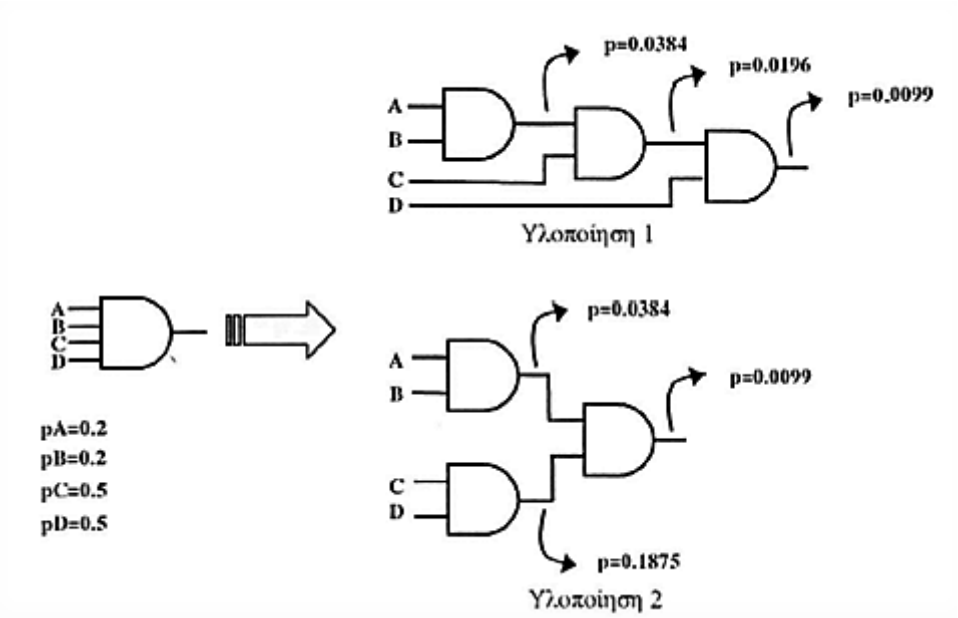
Κατά την ελαχιστοποίηση λογικής, σκοπός είναι η ελαχιστοποίηση της δραστηριότητας μετάβασης (switching activity) του σήματος. Η συγκεκριμένη μεθοδολογία επεμβαίνει στην Boolean έκφραση μιας λογικής συνάρτησης, με σκοπό να μειώσει τη συνολική δραστηριότητα μετάβασης ελαττώνοντας έτσι και την κατανάλωση ισχύος. Ο βασικός περιορισμός που τίθεται κατά τη διάρκεια της ελαχιστοποίησης λογικής είναι η καθυστέρηση, ενώ η επιφάνεια ολοκλήρωσης συνήθως αυξάνεται.

Η συγκεκριμένη μεθοδολογία επιφέρει βελτίωση της κατανάλωσης ισχύος μέχρι και 60%, συνοδευόμενη από αύξηση της επιφάνειας ολοκλήρωσης κατά 8%, Τυπικές τιμές βελτίωσης της κατανάλωσης ισχύος είναι μεταξύ 10-20%.

3.4.3 Διάσπαση λογικού κυκλώματος (Logic decomposition)

Η συγκεκριμένη μεθοδολογία ελαχιστοποιεί την κατανάλωση ισχύος, μετατρέποντας μια ομάδα λογικών συναρτήσεων σε μια άλλη ομάδα που να περιλαμβάνει μόνο λογικές πύλες AND ή OR δύο εισόδων και πύλες NOT. Προφανώς καταλληλότερος συνδυασμός είναι αυτός για τον οποίο η πιθανότητα μετάβασης σε κάθε κόμβο είναι η μικρότερη δυνατή. Με δεδομένη την πιθανότητα μετάβασης των σημάτων στην είσοδο, είναι δυνατόν να βρεθεί η κατάλληλη συνδεσμολογία λογικών πυλών δύο εισόδων που να ελαχιστοποιούν την κατανάλωση ισχύος, Επιπλέον λαμβάνεται υπόψη και η επίδραση των ανεπιθύμητων μεταβάσεων. Ο αλγόριθμος αναζήτησης της κατάλληλης κυκλωματικής δομής βασίζεται στην αρχή ότι προκειμένου να ελαττωθεί η κατανάλωση ισχύος, οι είσοδοι με τις μεγαλύτερες τιμές δραστηριοτήτων μετάβασης πρέπει να οδηγούνται στο κύκλωμα χωρίς να διέρχονται από πολλές λογικές πύλες.

Στο Σχήμα 3.6 παρουσιάζονται δύο διαφορετικές υλοποιήσεις μιας λογικής πύλης AND τεσσάρων εισόδων, με συνδυασμό λογικών πυλών AND δύο εισόδων. [17] Στο ίδιο σχήμα δίνονται και οι πιθανότητες των εισόδων (οι οποίες θεωρούνται ασυσχέτιστες μεταξύ τους) να βρίσκονται στην λογική τιμή 1. Όπως φαίνεται η πρώτη υλοποίηση οδηγεί σε μικρότερη κατανάλωση ισχύος συγκρινόμενη με τη δεύτερη, καθώς η συνολική δραστηριότητα μετάβασης για το πρώτο κύκλωμα είναι 0.888 και για το δεύτερο κύκλωμα 1.056. [3]

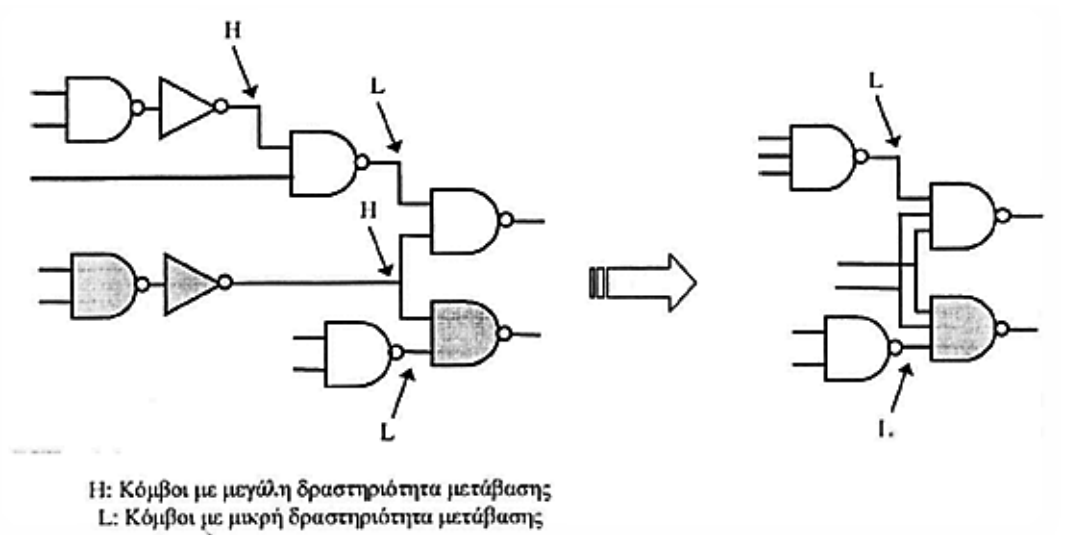


Σχήμα 3.6: Υλοποίηση μια λογικής πύλης AND τεσσάρων εισόδων με συνδυασμό λογικών πυλών AND δύο εισόδων.

3.4.4 Technology mapping

Με τον όρο technology mapping αναφερόμαστε στην υλοποίηση μιας λογικής συνάρτησης, από λογικές πύλες δύο εισόδων μιας συγκεκριμένης βιβλιοθήκης δομικών κυκλωμάτων (cell library). Προκειμένου τα εφαρμοστεί μια μεθοδολογία χαμηλής κατανάλωσης ισχύος κατά τη διαδικασία του technology mapping πρέπει τα δομικά κυκλώματα να είναι χαρακτηρισμένα ως προς την καθυστέρηση, την επιφάνεια ολοκλήρωσης και πρόσθετα ως προς την εσωτερική κατανάλωση ισχύος τους. Οι επιδόσεις και η ποικιλία των δομικών κυκλωμάτων, μπορεί να επηρεάσουν σημαντικά το τελικό κέρδος στην κατανάλωση ισχύος.

Η αρχή που εφαρμόζεται για την ελαχιστοποίηση της κατανάλωσης ισχύος είναι η ενσωμάτωση των κόμβων με μεγάλη δραστηριότητα μετάβασης εντός των λογικών πυλών, έτσι ώστε να οδηγούν μικρότερη χωρητικότητα. Μια πιθανή εφαρμογή της τεχνικής παρουσιάζεται στο Σχήμα 3.7, όπου οι λογικές πύλες με γκρί απόχρωση έχουν ενσωματωθεί σε μία, για την απόκρυψη των κόμβων μεγάλης δραστηριότητας μετάβασης. Στην πράξη αυτό που επιτυγχάνει η συγκεκριμένη μεθοδολογία είναι η μείωση του αριθμού των κόμβων μεγάλης δραστηριότητας μετάβασης, αυξάνοντας όμως τον αριθμό των κόμβων μικρής δραστηριότητας μετάβασης. Σε σχέση με ένα αντίστοιχο κύκλωμα βελτιστοποιημένο ως προς την ταχύτητα λειτουργίας, το κέρδος στην κατανάλωση ισχύος που παρέχει η συγκεκριμένη τεχνική προσεγγίζει το 18% με επιβάρυνση της επιφάνειας ολοκλήρωσης κατά 16%. [3]



Σχήμα 3.7: Η εφαρμογή του technology mapping για ελάττωση της κατανάλωση ισχύος.

3.4.5 Η επίδραση της λογικής οικογένειας στην κατανάλωση ισχύος

Πληθώρα λογικών οικογενειών σε τεχνολογία CMOS προτείνονται για την υλοποίηση λογικών κυκλωμάτων. Κάθε μία έχει συγκεκριμένα πλεονεκτήματα και μειονεκτήματα. [18],[19],[20] Η επιλογή κάποιας συγκεκριμένης λογικής οικογένειας καθορίζεται βάσει συγκεκριμένων κριτηρίων όπως είναι η ταχύτητα λειτουργίας, η απαιτούμενη επιφάνεια ολοκλήρωσης, η κατανάλωση ισχύος, η ευκολία σχεδιασμού που παρέχει, τα περιθώρια θορύβου, ο θόρυβος που επάγει κ.τ.λ.

- **Η λογική οικογένεια CMOS** έχει το πλεονέκτημα ότι είναι εξαιρετικά δοκιμασμένη και για αυτό παρέχει υψηλή δυνατότητα αυτοματοποιημένης σχεδίασης. Επίσης προσφέρει σημαντική ανοχή στον θόρυβο, λόγω της στατικής λειτουργίας της. Παρόλα αυτά είναι ακατάλληλη για την υλοποίηση λογικών κυκλωμάτων μεγάλου βαθμού οδήγησης, διότι οδηγεί σε σημαντική αύξηση της επιφάνειας ολοκλήρωσης και της κατανάλωσης ισχύος. [20]

- **Η λογική οικογένεια CPL** (CMOS pass logic), χρησιμοποιεί NMOS τρανζίστορς διέλευσης, Σε σύγκριση με τα κλασσικά κυκλώματα CMOS, απαιτεί στις περισσότερες περιπτώσεις, λιγότερα και μικρότερου μεγέθους τρανζίστορς, με αποτέλεσμα να παρουσιάζει μικρότερη κατανάλωση ισχύος. Είναι κατάλληλη για την υλοποίηση κυκλωμάτων που βασίζονται στην πύλη XOR (αθροιστές, πολλαπλασιαστές) καθώς χρειάζονται μόνο τέσσερα τρανζίστορς για την υλοποίηση της συγκεκριμένης λογικής πύλης. Το βασικό μειονέκτημα είναι η ανάγκη συμπληρωματικών σημάτων.

- **Η λογική οικογένεια Domino** αποτελεί τον κυριότερο εκπρόσωπο κυκλωμάτων δυναμικής λογικής. Τα δυναμικά κυκλώματα εμφανίζουν ορισμένα σημαντικά πλεονεκτήματα σε σχέση με τα κλασσικά CMOS, όπως είναι η υψηλή ταχύτητα λειτουργίας, μειωμένη κατανάλωση ισχύος λόγω ρεύματος βραχυκύκλωσης, λιγότερες ανεπιθύμητες μεταβάσεις και μικρότερος απαιτούμενος αριθμός τρανζίστορς καθώς έχει απαλειφθεί το PMOS δικτύωμα. Παρόλα αυτά τα κυκλώματα Domino εμφανίζουν μεγαλύτερη δραστηριότητα μετάβασης, καθώς κάθε κόμβος εξόδου φορτίζεται σε δυναμικό Vdd για κάθε κύκλο ρολογιού, αυξάνοντας τη δυναμική κατανάλωση ισχύος. Επίσης απαιτείται κατάλληλη κατανομή του ρολογιού μεταξύ των κυκλωμάτων Domino προκειμένου να μειωθεί η αυξημένη κατανάλωση του ρολογιού.

- **Η λογική οικογένεια CVSL (Current voltage switch logic)**, απαιτεί συμπληρωματικά σήματα εισόδου. Στο τμήμα φόρτισης υπάρχουν δύο PMOS τρανζίστορς σε χιαστή συνδεσμολογία, ώστε να σχηματίζουν μια απλή μονάδα μανδαλωτή (latch). Τα PMOS τρανζίστορς οδηγούνται εναλλάξ από τα δύο

συμπληρωματικά δικτυώματα NMOS εκφόρτισης. Για κάθε διάνυσμα εισόδου, το ένα από τα δύο δικτυώματα εκφόρτισης άγει, εκφορτίζοντας την αντίστοιχη έξοδο, ενώ το άλλο δικτύωμα είναι κλειστό αφήνοντας τη συγκεκριμένη έξοδο να φορτιστεί σε Vdd. Η δυναμική παραλλαγή της λογικής CVSL, είναι η PCVSL (Precharged CVSL).

Οι δυναμικές λογικές οικογένειες (PCVSL και Domino) παρουσιάζουν συγκριτικά την υψηλότερη κατανάλωση ισχύος, ειδικότερα για σήματα εξόδου μικρής δραστηριότητας. Αντίθετα η κλασσική CMOS σχεδίαση παρουσιάζει τη μικρότερη κατανάλωση ισχύος και αποδεικνύεται κατάλληλη για την υλοποίηση απλών λογικών κυκλωμάτων. Οι οικογένειες CVSL και CPL, οι οποίες είναι και οι δύο τεχνικές με συμπληρωματικά σήματα, καταναλώνουν περισσότερη ισχύ από την κλασσική οικονένεια CMOS. Σε σχέση με την CVSL, η λογική CPL απαιτεί περισσότερα τρανζίστορς, ενώ δεν μπορεί να χρησιμοποιηθεί σε εφαρμογές χαμηλής τροφοδοσίας-χαμηλής κατανάλωσης ισχύος, λόγω της πτώσης δυναμικού από την τάση κατωφλιού των NMOS τρανζίστορς. [3]

3.4.6 Ελαχιστοποίηση των ανεπιθύμητων μεταβάσεων (Signal glitching)

Η ελαχιστοποίηση των ανεπιθύμητων μεταβάσεων, βασίζεται κυρίως στην εξισορρόπηση των καθυστερήσεων που εισάγουν διαφορετικοί αγώγιμοι δρόμοι. Η εξισορρόπηση μπορεί να επιτευχθεί υιοθετώντας μια συμμετρική λογική δομή (δέντρου) ή με κατάλληλη επιλογή των διαστάσεων των τρανζίστορς έτσι ώστε τα πιο γρήγορα μονοπάτια να οδηγούνται από τρανζίστορς μικρότερου μεγέθους. Με αυτό τον τρόπο προκαλείται συνήθως αύξηση της καθυστέρησης. Εναλλακτικά μπορεί να τοποθετούνται απομονωτές (buffers), ως στοιχεία καθυστέρησης στα γρήγορα μονοπάτια. Όμως η εισαγωγή πρόσθετων κυκλωματικών στοιχείων πρέπει να γίνεται με ιδιαίτερη προσοχή, καθώς μπορεί να μειώνουν την εμφάνιση ανεπιθύμητων μεταβάσεων αλλά συνολικά να προκαλούν αύξηση της κατανάλωσης ισχύος.

Μείωση των ανεπιθύμητων μεταβάσεων μπορεί επίσης να επιτευχθεί με την εφαρμογή αυτοχρονιζόμενων (self-timing) τεχνικών. Η τεχνική αυτή έχει οδηγήσει σε μείωση της κατανάλωσης ισχύος κατά 25%.

3.4.7 Βελτιστοποίηση διαστάσεων των τρανζίστορ

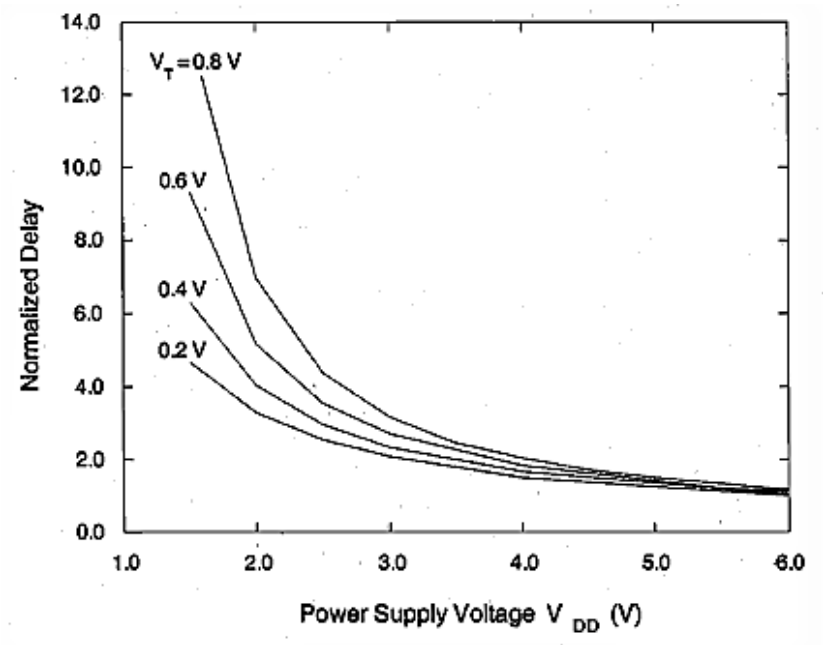
Ανεξάρτητα από την επιλογή της λογικής οικογένειας ή της κυκλωματικής τοπολογίας, η κατάλληλη επιλογή των διαστάσεων των τρανζίστορς μπορεί να επιφέρει σημαντική μείωση της κατανάλωσης ισχύος. Ο ιδανικός λόγος W/L των τρανζίστορ ενός κυκλώματος, προκύπτει διαφορετικός προκειμένου να βελτιστοποιηθεί η ταχύτητα ή η κατανάλωση ισχύος. [21] Προκειμένου το κύκλωμα με τα τρανζίστορς ελάχιστων διαστάσεων και το αντίστοιχο μεγαλύτερο κύκλωμα να παρουσιάζουν την ίδια καθυστέρηση, θα πρέπει η τάση τροφοδοσίας για το μεγαλύτερο κύκλωμα να μειωθεί σύμφωνα με έναν παράγοντα κέρδους της ταχύτητας (εφόσον η καθυστέρηση είναι αντιστρόφως ανάλογη της τάσης τροφοδοσίας).

Συμπερασματικά μπορούμε να πούμε ότι για την οδήγηση φορτίων που αποτελούνται κυρίως από την χωρητικότητα εισόδου των τρανζίστορς, ενδείκνυται η χρησιμοποίηση τρανζίστορς ελάχιστων διαστάσεων για την ελαχιστοποίηση της κατανάλωσης ισχύος. Αντίθετα, στην περίπτωση που η χωρητικότητα φορτίου καθορίζεται κυρίως από την χωρητικότητα της γραμμή διασύνδεσης πρέπει να γίνεται κατάλληλη επιλογή των διαστάσεων των τρανζίστορς. Στην παραπάνω ανάλυση θεωρήθηκε ότι η παρασιτική χωρητικότητα των επαφών είναι ανεξάρτητη από το μέγεθος των τρανζίστορς, το οποίο στην πράξη δεν ισχύει καθώς η επίδραση της συγκεκριμένης χωρητικότητας αυξάνεται με την αύξηση του μεγέθους των τρανζίστορς. [3]

3.4.8 Κυκλώματα CMOS πολλαπλής τάσης κατωφλίου(Multiple-Threshold)

Το κύριο πλεονέκτημα της τεχνικής πολλαπλών τάσεων κατωφλίου σε σχέση με την τεχνική πολλαπλών τάσεων τροφοδοσίας είναι το γεγονός ότι τα λογικά επίπεδα δεν αλλάζουν. Δεν χρειάζονται κυκλώματα ανάκτησης τάσης, που καταναλώνουν χρόνο και ισχύ, στη διεπαφή μεταξύ των διαφορετικών περιοχών τάσεων κατωφλίου. Αφού οι πύλες δε χρειάζεται να ομαδοποιηθούν αλλά μπορούν να τοποθετηθούν αυθαίρετα, δεν χρειάζονται περιορισμοί για τη διαδικασία τοποθέτησης και δρομολόγησής τους. [2]

Η τεχνική πολλαπλών τάσεων κατωφλίου δεν περιορίζεται στο επίπεδο πύλης, για αυτό είναι πιθανό να ανατεθούν διαφορετικές τάσεις κατωφλίου σε μπλοκ με διαφορετικές απαιτήσεις απόδοσης. Μια υψηλότερη τάση κατωφλίου χρησιμοποιείται για ελαχιστοποίηση της διαρροής σε μπλοκ με λιγότερο κρίσιμη απόδοση και η χαμηλή τάση κατωφλίου ανατίθεται σε μπλοκ με την πιο κρίσιμη απόδοση, όπως φαίνεται και στο Σχήμα 3.8. [1]

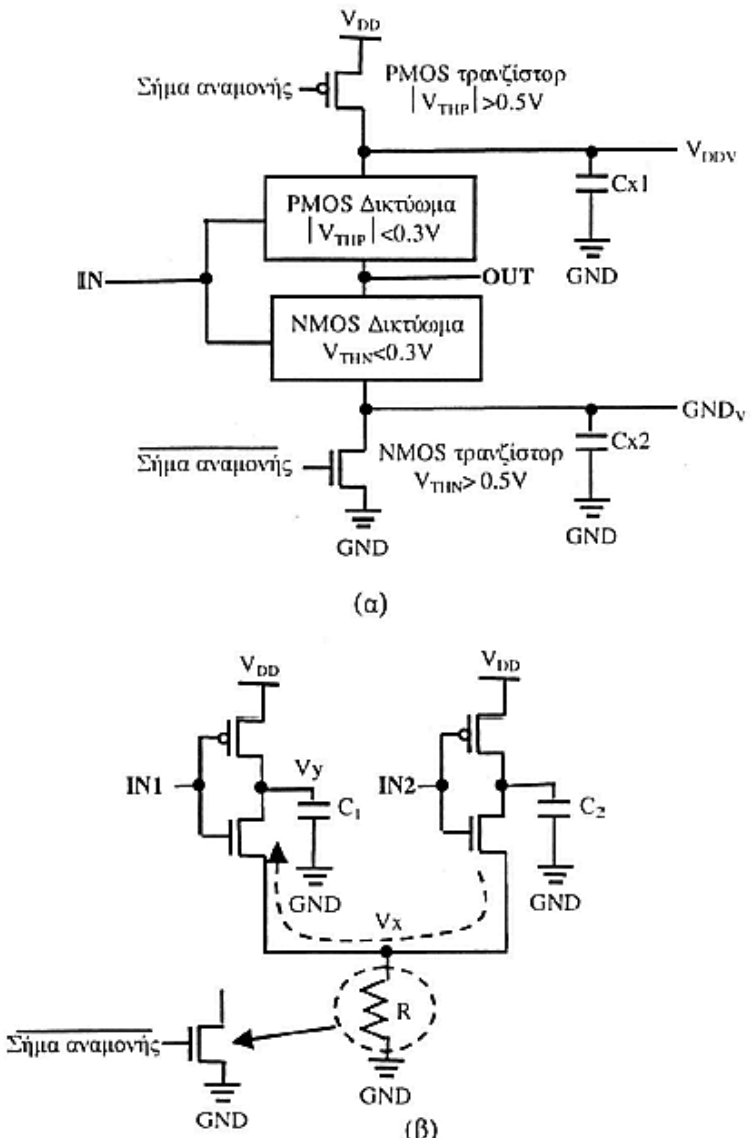


Σχήμα 3.8: Σχέση καθυστέρησης,τάσης τροφοδοσίας και τάσης κατωφλίου V_T.

Στα κυκλώματα CMOS πολλαπλής τάσης κατωφλίου, χρησιμοποιούνται τρανζίστορς με χαμηλή και υψηλή τάση κατωφλίου, με τα τρανζίστορς χαμηλής τάσης κατωφλίου (Vth<0.3V) να υλοποιούνται όλα τα λογικά κυκλώματα που απαιτούν υψηλή ταχύτητα λειτουργίας και με τα τρανζίστορς υψηλότερης τάσης κατωφλίου (Vth>0.5V) να υλοποιούνται σαν διακόπτες, για να απομονώνουν στην κατάσταση αναμονής τα λογικά κυκλώματα από την τροφοδοσία και τη γη εξασφαλίζοντας μικρό υποκατωφλικό ρεύμα. Όπως φαίνεται και από το γενικό διάγραμμα ενός λογικού κυκλώματος πολλαπλής τάσης κατωφλίου (Σχήμα 3.9), το λογικό κύκλωμα δεν συνδέεται απευθείας στις γραμμές τροφοδοσίας (V_{DD}, GND) αλλά στις εικονικές (virtual) γραμμές τροφοδοσίας (V_{DDv}, GND_v).

Στην κατάσταση λειτουργίας (active mode), το σήμα αναμονής είναι στο λογικό 0 και τα τρανζίστορς με υψηλή τάση κατωφλίου άγουν. Λόγω της μικρής αντίστασης που παρουσιάζουν τα MOS τρανζίστορς κατά την κατάσταση αγωγιμότητας, οι κόμβοι V_{DDv} και GND_v θα συμπεριφέρονται σαν πραγματικές γραμμές τροφοδοσίας. Έτσι θα εξασφαλίζεται η κανονική λειτουργία του κυκλώματος, παρέχοντας μεγάλη ταχύτητα λειτουργίας λόγω των τρανζίστορς χαμηλής τάσης κατωφλίου. Στην περίπτωση όπου το κύκλωμα εισέρχεται στην κατάσταση αναμονής το σήμα αναμονής μεταβαίνει στο λογικό 1. Τα τρανζίστορς υψηλής V_{Th} βρίσκονται στην αποκοπή, αποκόπτοντας οποιοδήποτε αγώγιμο μονοπάτι υποκατωφλικού ρεύματος από την τροφοδοσία προς την γη. Έτσι μειώνεται σημαντικά η DC κατανάλωση ισχύος στην κατάσταση αναμονής.

Δύο βασικοί παράγοντες επηρεάζουν τις επιδόσεις ενός κυκλώματος πολλαπλής τάσης κατωφλίου. Ο ένας είναι το μέγεθος των τρανζίστορς υψηλής V_{Th}. Όσο μεγαλύτερο είναι το πλάτος W, τόσο μικρότερη είναι η αντίσταση που παρουσιάζουν στην κατάσταση αγωγιμότητας, προσφέροντας ταχύτερες μεταβάσεις από την κατάσταση αναμονής στην κατάσταση λειτουργίας. Κατάλληλοι αλγόριθμοι έχουν αναπτυχθεί για την εύρεση του βέλτιστου μεγέθους των τρανζίστορς υψηλής V_{TH} ανάλογα με την εφαρμογή. Ο άλλος παράγοντας είναι η χωρητικότητα των κόμβων V_{DDv} και GNDv (Cxi και Cx2 αντίστοιχα). Οι συγκεκριμένες χωρητικότητες, συμβάλλουν στην εξομάλυνση των διακυμάνσεων της τάσης στους κόμβους V_{DDv} και GND_v αντίστοιχα. Οι εν λόγω διακυμάνσεις εμφανίζονται κατά τη διάρκεια των λογικών μεταβάσεων στο εσωτερικό του κυκλώματος, λόγω χωρητικών συζεύξεων. Σαν αποτέλεσμα προκαλείται ανύψωση της τάσης του κόμβου V_{DDV} από την τιμή V_{DD} και αντίστοιχα ελάττωσης της τάσης του κόμβου GND_v από την τιμή GND.



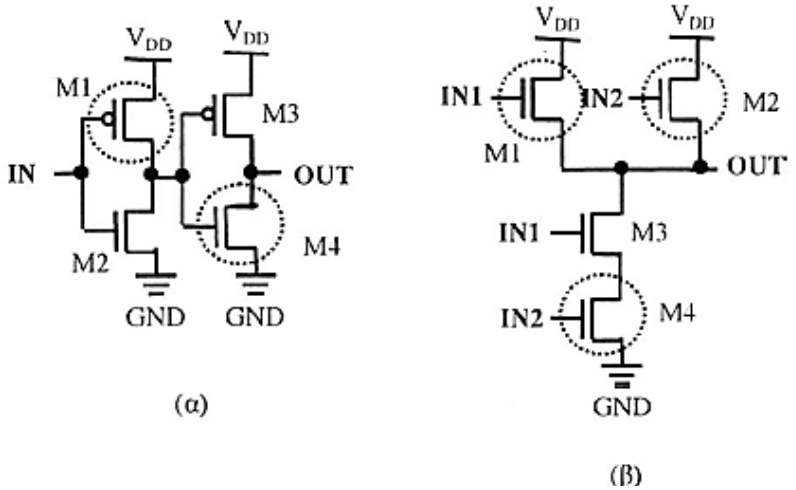
Σχήμα 3.9: α) Κύκλωμα πολλαπλής τάσης κατωφλιού με εικονική τροφοδοσία και γη β) Παρασιτικά φαινόμενα.

Η τεχνική των πολλαπλών τάσεων κατωφλίου, μπορεί να εφαρμοστεί επιτυχώς με την εισαγωγή μόνο ενός PMOS ή NMOS τρανζίστορ υψηλής τάσης κατωφλίου. Ενδείκνυται όμως η χρήση του NMOS τρανζίστορ καθώς μπορεί να εξασφαλίσει τις ίδιες επιδόσεις με ένα PMOS έχοντας μικρότερο μέγεθος. Το βασικότερο μειονέκτημα των κυκλωμάτων πολλαπλών τάσεων κατωφλίου, εκτός από την προφανή αύξηση της επιφάνειας ολοκλήρωσης, είναι η ανάγκη επιπλέον κυκλωμάτωσης, με τρανζίστορς υψηλής τάσης κατωφλίου, για τη διατήρηση των δεδομένων κατά τη διάρκεια της κατάστασης αναμονής.

Μια παραλλαγή της τεχνικής των πολλαπλών τάσεων κατωφλίου υπαγορεύει στα μη κρίσιμα μονοπάτια τα τρανζίστορς να έχουν μεγάλη VTH, ενώ στα κρίσιμα

μονοπάτια (μονοπάτια που καθορίζουν την καθυστέρηση ενός κυκλώματος) τα τρανζίστορς διατηρούν χαμηλή V_{TH}. [22] Με τον τρόπο αυτό εξασφαλίζεται η υψηλή ταχύτητα λειτουργίας ταυτόχρονα με μικρή στατική κατανάλωση ισχύος, ενώ δεν αυξάνεται ο αριθμός των τρανζίστορς. Συγκεκριμένοι αλγόριθμοι σε επίπεδο πυλών [23] και τρανζίστορς έχουν αναπτυχθεί, οι οποίοι επιλέγουν τα κατάλληλα τρανζίστορς για υψηλή V_{TH}, προκειμένου να επιτευχθεί η μεγαλύτερη ελάττωση του υποκατωφλικού ρεύματος με δεδομένους περιορισμούς στην ταχύτητα λειτουργίας.

Στο παρακάτω Σχήμα 3.10, παρουσιάζονται δύο λογικές πύλες με τρανζίστορς χαμηλής τάσης κατωφλίου (εντός κύκλου), στις οποίες επιλεγμένα τρανζίστορς διαθέτουν υψηλή τάση κατωφλίου για την αποκοπή του ρεύματος διαρροής. Στο κύκλωμα των αντιστροφέων, στην κατάσταση αναμονής η είσοδος τίθεται στην λογική τιμή 0. Καθώς τα τρανζίστορς M2 και M3 είναι σε αποκοπή, πρέπει να διαθέτουν υψηλή τάση κατωφλίου, ενώ τα τρανζίστορς MI και M4 τα οποία άγουν διαθέτουν χαμηλή τάση κατωφλίου. Αντίστοιχα στην περίπτωση της πύλης NAND, όταν στην κατάσταση αναμονής οι είσοδοι είναι στην λογική τιμή 0, πρέπει μόνο το τρανζίστορ M3 να έχει υψηλή τάση κατωφλίου προκειμένου να αποκοπεί το ρεύμα διαρροής. Με τον τρόπο αυτό η υποβάθμιση της ταχύτητας λειτουργίας είναι ασήμαντη. [3]



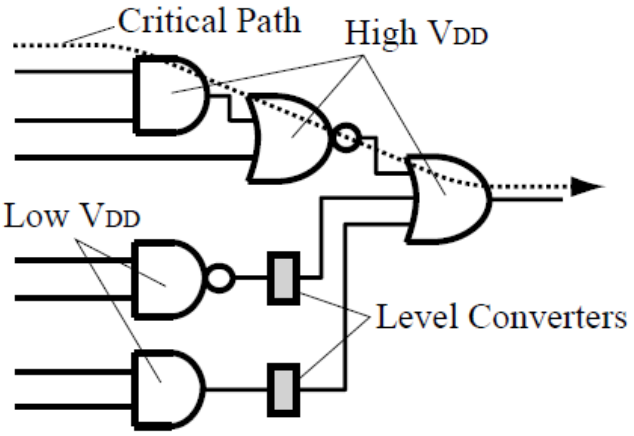
Σχήμα 3.10: Κυκλώματα πολλαπλής τάσης κατωφλιού

Ανακεφαλαιώνοντας, μπορούμε να ισχυριστούμε ότι η χρήση αυξημένης τάσης κατωφλίου για μη-κρίσιμες διαδρομές και μπλοκ του κυκλώματος οδηγεί στη μείωση των διαρροών υποκατωφλικού ρεύματος (subthreshold leakage current). Όμως, το πλήθος των πυλών με χαμηλή τάση κατωφλίου πρέπει να είναι μικρό σε σχέση με το πλήθος των πυλών με υψηλή τάση κατωφλίου για να επιτευχθεί μια σημαντική μείωση της διαρροής ολόκληρου του συστήματος.

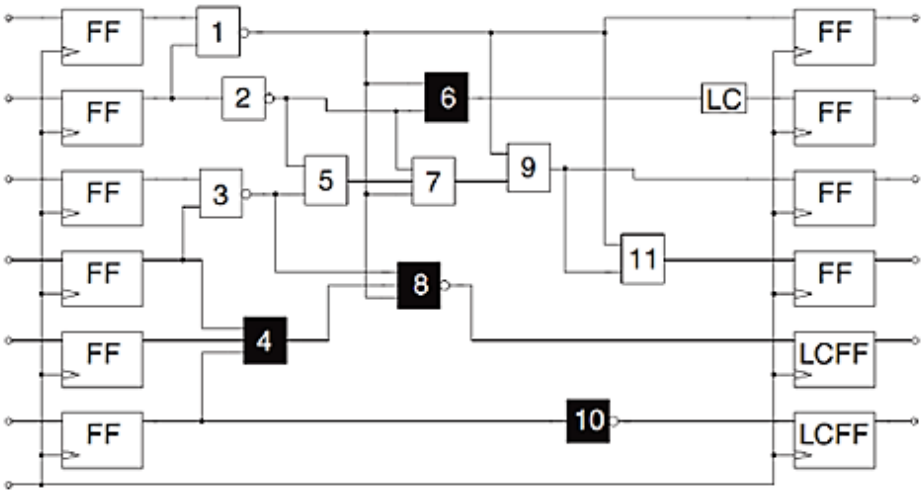
3.4.9 Λογική με Πολλαπλές Τάσεις Τροφοδοσίας

Η τιμή της τάσης τροφοδοσίας των ψηφιακών κυκλωμάτων επιλέγεται με τέτοιο τρόπο ώστε οι κρίσιμες διαδρομές να ικανοποιούν τις χρονικές προδιαγραφές κάτω από οποιεσδήποτε συνθήκες λειτουργίας. Αν και ένα κύκλωμα δεν αποτελείται μόνο από κρίσιμες ως προς το χρόνο διαδρομές, οι συντομότερες διαδρομές έχουν μικρότερη καθυστέρηση σήματος από αυτή που καθορίζεται από τις προδιαγραφές του συστήματος. Ως περιθώριο χρόνου (slack) μιας διαδρομής, ορίζεται η διαφορά μεταξύ της μέγιστης επιτρεπόμενης καθυστέρησης και της πραγματικής καθυστέρησης της αντίστοιχης διαδρομής. Η ύπαρξη του περιθωρίου χρόνου είναι μία ένδειξη ότι σπαταλιέται είτε επιφάνεια πυριτίου, είτε ενέργεια χωρίς κέρδος για την απόδοση του συστήματος. [2]

Για να αποφευχθεί η κατανάλωση ενέργειας ένας τρόπος είναι να τροφοδοτήσουμε τις πύλες που βρίσκονται σε μη κρίσιμες διαδρομές με χαμηλότερη τάση τροφοδοσίας [28], αυξάνοντας την καθυστέρηση διάδοσης σήματος σε αυτές αλλά μειώνοντας ταυτόχρονα τη δυναμική κατανάλωση ως προς το ΔV^2_{DD}. Όσο η καθυστέρηση της συγκεκριμένης διαδρομής είναι μικρότερη από την καθυστέρηση της κρίσιμης διαδρομής, δηλαδή για όσο το περιθώριο χρόνου παραμένει θετικό, η συγκεκριμένη τεχνική χαμηλής κατανάλωσης δεν έχει κανέναν αντίκτυπο στην απόδοση του συστήματος. Η ιδέα των πολλαπλών τάσεων τροφοδοσίας φαίνεται στα Σχήματα 3.11 και 3.12.



Σχήμα 3.11: Παράδειγμα Λειτουργίας της Τεχνικής Πολλαπλών V_{DD}.



Σχήμα 3.12: Αρχή λειτουργίας της τεχνικής πολλαπλών V_{DD} σε επίπεδο πύλης.

Οι πύλες που λειτουργούν με υψηλότερη τάση τροφοδοσίας $V_{DD,high}$ απεικονίζονται με άσπρα τετράγωνα ενώ τα μαύρα τετράγωνα απεικονίζουν πύλες που τροφοδοτούνται με τη μειωμένη τάση $V_{DD,low}$ (σχήμα 3.12). Για να μπορέσει να οδηγήσει μία πύλη με $V_{DD,low}$ μία πύλη με υψηλή τάση τροφοδοσίας χρειάζεται προσαρμογή του επιπέδου τάσης. Για αυτό μετατροπείς επιπέδου τάσης ή ανάκτησης τάσης (level converters - LC) πρέπει να εισαχθούν στο τέλος των διαδρομών με $V_{DD,low}$.

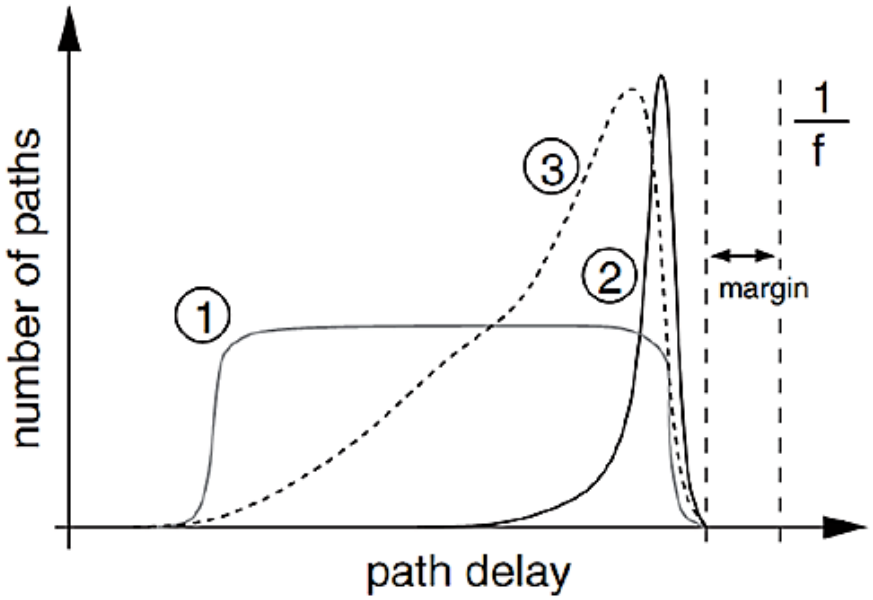
Στο σχήμα 3.11, μία μετάβαση $V_{DD,low}$ σε $V_{DD,high}$ επιτρέπεται μόνο πριν από τα κελιά ανάκτησης τάσης (LC) ή τα ειδικά flip-flops ανάκτησης τάσης (LCFF). Αυτή η τεχνική ονομάζεται ομαδοποίηση κλιμακούμενων τάσεων (clustered voltage scaling –CVS) και κάνει απλή τη σχεδίαση εάν ο αριθμός και η θέση των διεπαφών $V_{DD,low}$ σε $V_{DD,high}$ είναι γνωστά. [29]

Όπως αναφέρθηκε, δυνατότητα εξοικονόμησης ενέργειας της τεχνικής πολλαπλών τάσεων τροφοδοσίας, εξαρτάται σε μεγάλο βαθμό από το διαθέσιμο περιθώριο χρόνου, που δίνεται από την κατανομή καθυστέρησης διάδοσης σήματος της διαδρομής.

Στο Σχήμα 3.13 φαίνονται τρεις περιπτώσεις: Στη πρώτη κατανομή του σχήματος 3.13, η καθυστέρηση διάδοσης σήματος διαδρομής κατανέμεται ομοιόμορφα σε ένα μεγάλο εύρος διαδρομών (περιέχει πολλές σύντομες διαδρομές).

Το κύκλωμα που αντιστοιχεί στη δεύτερη κατανομή του σχήματος 3.3, αποτελείται κυρίως από μεγάλες διαδρομές σήματος (με πολλές κρίσιμες διαδρομές), οπότε δεν υπάρχει καμία περίπτωση να μειώσουμε την κατανάλωση ενέργειας σε βάρος την καθυστέρησης διάδοσης σήματος. Στην πραγματικότητα, μια κατανομή καθυστέρησης διάδοσης σήματος διαδρομής βρίσκεται μεταξύ των δύο προηγούμενων ακραίων περιπτώσεων.

Η τρίτη κατανομή είναι μία τυπική περίπτωση τυχαίας λογικής. Το κλάσμα του αριθμού των μη κρίσιμων διαδρομών ως προς το συνολικό αριθμό διαδρομών, δίνει το ποσοστό εξοικονόμησης ενέργειας της τεχνικής πολλαπλών τάσεων τροφοδοσίας. [2]



Σχήμα 3.13: Κατανομές της Καθυστέρησης Διάδοσης Σήματος Διαδρομής.

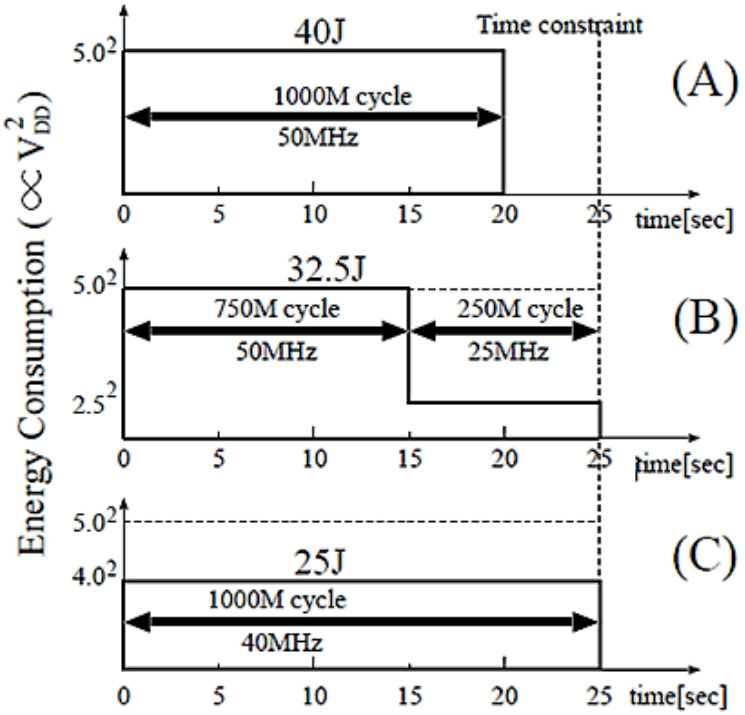
3.4.10 Δυναμική Κλιμάκωση της Τάσης (Dynamic Voltage Scaling - DVS)

Η τάση τροφοδοσίας στα ψηφιακά κυκλώματα CMOS αποτελείται από την τάση που χρειάζεται να λειτουργήσει το κύκλωμα στην τυπική κατάσταση, με επιπρόσθετα τα περιθώρια που λαμβάνουν υπόψη τις αποκλίσεις από την τυπική κατάσταση. Τα περιθώρια στην τροφοδοσία ισχύος εξασφαλίζουν μία υψηλή κατασκευαστική απόδοση. Για αυτό είναι χρήσιμο από ενεργειακής άποψης να μειωθεί η τάση τροφοδοσίας στα γρήγορα ολοκληρωμένα κυκλώματα. Τα γρήγορα ολοκληρωμένα κυκλώματα λειτουργούν με μια μεγαλύτερη συχνότητα, υψηλότερη από αυτή που δίνει η προδιαγραφή αλλά αντιμετωπίζουν πρόβλημα με υπερβολικά υψηλά ρεύματα διαρροής. Μειώνοντας την τάση τροφοδοσίας κατά ένα ποσοστό, η μέγιστη συχνότητα πέφτει στην τιμή της συχνότητας λειτουργίας που δίνεται από την προδιαγραφή αλλά ταυτόχρονα μειώνονται εκθετικά τα ρεύματα διαρροής και η δυναμική κατανάλωση σε αναλογία με το ΔV^2_{DD}. Τα αργά ολοκληρωμένα που θα αποτύχαιναν στον έλεγχο ταχύτητας μπορούν να επιταχυνθούν από μία υψηλότερη τάση τροφοδοσίας. Η κατανάλωση ισχύος αυτών των κυκλωμάτων όταν είναι σε κατάσταση αναμονής δεν είναι πρόβλημα συνήθως γιατί η αργή διάδοση σήματος έχει ισχυρή σχέση με τα χαμηλά ρεύματα διαρροής. Η προσαρμογή της τάσης τροφοδοσίας επιτρέπει παράλληλα την χαμηλή κατανάλωση ισχύος και την υψηλή κατασκευαστική απόδοση. Για προϊόντα όπως οι μικροεπεξεργαστές που πωλούνται σε διαφορετικές κατηγορίες λειτουργίας, η περιγραφόμενη τεχνική μπορεί να χρησιμοποιηθεί για να αυξήσει τον

αριθμό των ολοκληρωμένων κυκλωμάτων. Επιπρόσθετα, η προσαρμογή της απόδοσης μπορεί να επιτευχθεί με συντονισμό των δυναμικών των υποστρωμάτων του ολοκληρωμένου. Παρόλα αυτά, για να επιταχυνθεί ένα κύκλωμα πρέπει να ενισχυθεί η κλιμάκωση της τάσης τροφοδοσίας. [2]

Το Σχήμα 3.14 δείχνει ένα παράδειγμα της δυναμικής κλιμάκωσης τάσης. Ας υποθέσουμε ότι έχουμε έναν επεξεργαστή που χρησιμοποιεί τρεις διαφορετικές τάσεις τροφοδοσίας, 5.0V, 4.0V και 2.5V. Μια εργασία που απαιτεί 1 δισεκατομμύριο κύκλους για να ολοκληρωθεί, εκτελείται στον επεξεργαστή. Οι καταναλώσεις ενέργειας για την εργασία είναι 10nJ/κύκλο, 25nJ/κύκλο και 40nJ/κύκλο στα 2.5V, 4.0V και 5.0V αντίστοιχα. Οι υπολογιστικές ταχύτητες του επεξεργαστή στα 5.0V, 4.0V και 2.5V είναι 50 εκατομμύρια κύκλοι ανά δευτερόλεπτο, 40 εκατομμύρια κύκλοι ανά δευτερόλεπτο και 25 εκατομμύρια κύκλοι ανά δευτερόλεπτο, αντίστοιχα.

Στο Σχήμα 3.14(A), ο επεξεργαστής χρησιμοποιεί τη μέγιστη τάση τροφοδοσίας, 5.0V, για όλη την εκτέλεση της εργασίας. Σε αυτήν την περίπτωση, η συνολική κατανάλωση ενέργειας είναι 40J. Εάν ο επεξεργαστής χρησιμοποιεί 2.5V και 5.0V με τρόπο που ο χρόνος ολοκλήρωσης της εργασίας να ανταποκρίνεται σε δεδομένο περιορισμό χρόνου, η κατανάλωση ενέργειας μπορεί να μειωθεί σε 32.5J όπως φαίνεται στο Σχήμα 3.14(B). Το Σχήμα 3.14(C) δείχνει την καλύτερη περίπτωση αυτού του παραδείγματος. Εάν ο επεξεργαστής χρησιμοποιεί μία μόνο τάση τροφοδοσίας που προσαρμόζει το χρόνο ολοκλήρωσης μόνο στο χρονικό περιορισμό, η συνολική κατανάλωση ενέργειας ελαχιστοποιείται. [5]



Σχήμα 3.14: Παράδειγμα της δυναμικής κλιμάκωσης τάσης.

3.4.11 Δυναμική Προσαρμογή της Τάσης και της Συχνότητας (DynamicVoltage and Frequency Scaling - DVFS)

Η μεταβολή μόνο της τάσης ή του σήματος ρολογιού σε ένα σύστημα μπορεί να επιφέρει εξοικονόμηση σημαντικού ποσοστού ενέργειας, ωστόσο είναι δύσκολο να εφαρμοστεί κάθε τεχνική μεμονωμένα σε εφαρμογές πραγματικού χρόνου. Μειώνοντας ταυτόχρονα και την τάση λειτουργίας και το σήμα ρολογιού κατά συγκεκριμένο παράγοντα, είναι δυνατή η εξοικονόμηση ενέργειας χωρίς το κύκλωμα να παρεκκλίνει από την σωστή λειτουργία του. Η DVFS τεχνική, περιγράφει τη χρήση των δύο τεχνικών εξοικονόμησης ενέργειας, τη δυναμική προσαρμογή της συχνότητας Dynamic Frequency Scaling-DFS και τη δυναμική προσαρμογή της τάσης Dynamic Voltage Scaling-DVS, προσαρμόζοντας δυναμικά την τάση και την συχνότητα λειτουργίας κατά τον χρόνο εκτέλεσης, ανάλογα με τις αλλαγές του φόρτου εργασίας. Οπότε τίθεται το πρόβλημα, σε τι βαθμό μπορούμε να μειώσουμε την κατανάλωση ενέργειας-ισχύος του συστήματος ώστε να μην επηρεαστεί αρνητικά η απόδοσή του. [30]

Δεδομένου ενός απλού συστήματος που λειτουργεί σε τάση 2V και συχνότητα 100Hz, θέτοντας k τον συντελεστή μετάβασης a και την χωρητικότητα C, η εξίσωση διαμορφώνεται ως εξής:

$$P_{base} = k*2^2*100 = 400 * k$$

Υποδιπλασιάζοντας την τάση και την συχνότητα λειτουργίας του κυκλώματος η εξίσωση διαμορφώνεται:

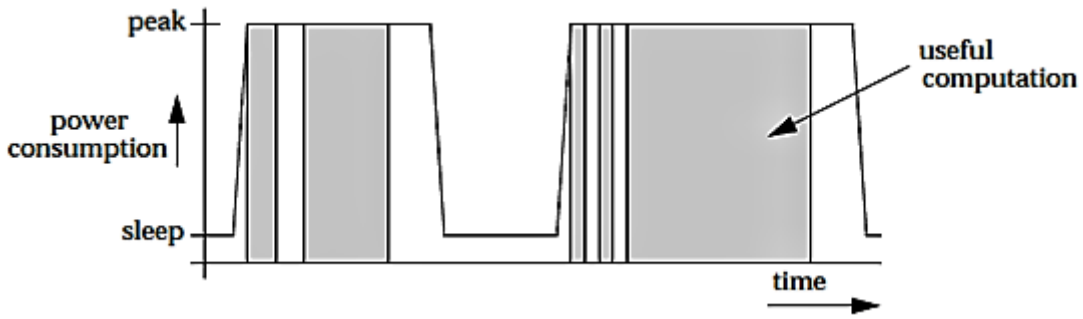
$$P_{Scale} = k*1^2*50 = 50 * k$$

$$P_{Scale} = (1/8) *P_{base}$$

Παρατηρούμε ότι εφαρμόζοντας την μισή τάση τροφοδοσίας και συχνότητα στο κύκλωμα η ισχύς που καταναλώνεται είναι το ένα όγδοο της αρχικής ισχύς. Δεδομένου ότι η συχνότητα του συστήματος μειώθηκε κατά το ήμισυ, ο χρόνος που απαιτείται για τους υπολογισμούς διπλασιάστηκε. Και ο χρόνος είναι σημαντικός συντελεστής για τον υπολογισμό της ενέργειας. [8]

Βασικό χαρακτηριστικό της τεχνικής αυτής είναι να εντοπιστούν τα σημεία που το σύστημα είναι αδρανές(idle) και να γίνει η εκμετάλλευση του system slack ή αλλιώς της «αδράνειας». Όταν ένα σύστημα είναι αδρανές επηρεάζεται ελάχιστα από την μείωση της συχνότητας λειτουργίας ενώ ταυτόχρονα επιτυγχάνεται η κυβική μείωση στη κατανάλωση ισχύος με τη βοήθεια της κλιμάκωσης τάσης. [10]

Στο Σχήμα 3.15. φαίνεται η κατανάλωση ισχύος σε σχέση με το χρόνο ενός τυπικού επεξεργαστή, όπου είναι εμφανή τα διάκενα αδράνειας, όπου δεν εκτελείται χρήσιμο υπολογιστικό έργο από τον επεξεργαστή. [6], [8]

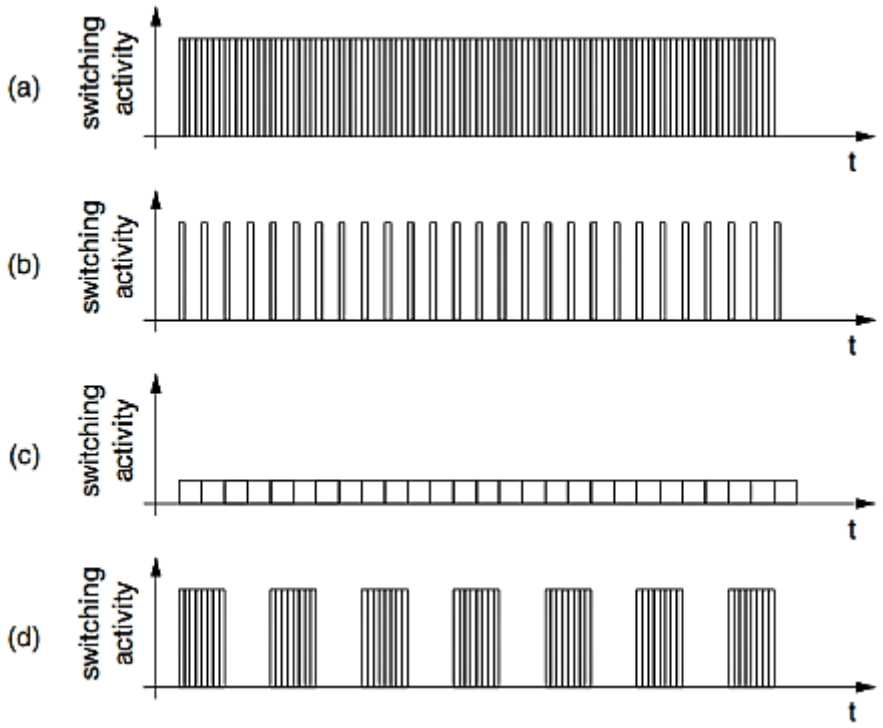


Σχήμα 3.15: Η κατανάλωση ισχύος σε σχέση με το χρόνο ενός τυπικού επεξεργαστή.

Οι περισσότερες εφαρμογές έχουν πολλαπλές καταστάσεις λειτουργίας και άρα και διαφορετικές απαιτήσεις απόδοσης. Εάν η τάση τροφοδοσίας επιλέγεται για τις απαιτήσεις ταχύτητας σε μία κατάσταση υψηλής απόδοσης με μεγάλες χωρητικές συζεύξεις και πτώση τάσης, τότε καταναλώνεται ενέργεια σε όλες τις άλλες καταστάσεις λειτουργίας που οι απαιτήσεις σε ταχύτητα δεν είναι τόσο μεγάλες. Μειώνοντας ταυτόχρονα την τάση τροφοδοσίας και την συχνότητα λειτουργίας σε καταστάσεις χαμηλής απόδοσης επιτυγχάνουμε σημαντική μείωση στη δυναμική κατανάλωση ισχύος. Όπως αναφέρθηκε, η κλιμάκωση της τάσης επηρεάζει τις χωρητικές απώλειες σε αναλογία με το ΔV^2_{DD} και η κλιμάκωση της συχνότητας τις επηρεάζει σε αναλογία με V_{DD}, οπότε οι δυναμικές απώλειες μειώνονται κατά ένα παράγοντα ΔV^3_{DD}. Η κατανάλωση ενέργειας σε κατάσταση λειτουργίας του κυκλώματος οφείλεται κυρίως σε δυναμικές απώλειες για αυτό και αυτό το στοιχείο ισχύος διαπραγματεύεται η κλιμάκωση της τάσης τροφοδοσίας. Φυσικά και όλες οι άλλες απώλειες μειώνονται επίσης με χρήση της κλιμάκωσης τάσης.

Στο Σχήμα 3.16 παρουσιάζονται τέσσερις διαφορετικές καταστάσεις λειτουργίας ενός ψηφιακού συστήματος. [28] Το παράδειγμα στο Σχήμα 3.16(a) μας δείχνει τη μετάβαση σήματος (switching activity) σε λειτουργία υψηλής απόδοσης. Η τάση τροφοδοσίας και η συχνότητα του ρολογιού έχουν και οι δύο υψηλή τιμή. Μειώνοντας μόνο τη συχνότητα του ρολογιού, δηλαδή έχοντας λειτουργία χαμηλής απόδοσης, δεν επηρεάζουμε την ταχύτητα μετάβασης σήματος των πυλών (Σχήμα 3.16(b)). Δηλαδή δεν επηρεάζεται η ταχύτητα διάδοσης σήματος της πύλης αλλά θα υπάρχει ένα διάστημα αδρανοποίησης στο τέλος κάθε κύκλου του ρολογιού (idle period). Στο Σχήμα 3.16(c) μειώνουμε και την τάση τροφοδοσίας για να εκμεταλλευτούμε ολόκληρο τον κύκλο του ρολογιού για τη διάδοση σήματος. Η μείωση της τάσης είναι ανάλογη της μείωσης της συχνότητας. Οπότε έχουμε μείωση δυναμικής κατανάλωσης ενέργειας κατά ΔV^3_{DD} όπως αναφέραμε και προηγουμένως. Άρα στο Σχήμα 3.16(c), το σύστημα

είναι σε λειτουργία χαμηλής απόδοσης και χαμηλής δυναμικής κατανάλωσης. Για λειτουργία χαμηλής στατικής κατανάλωσης (Σχήμα 3.16(d)), υπάρχουν διαστήματα που δε μειώνουμε ούτε την τάση τροφοδοσίας ούτε τη συχνότητα, όποτε έχουμε μία γρήγορη διάδοση σήματος (burst mode) και διαστήματα μεταξύ αυτών όπου το κύκλωμα τίθεται εκτός λειτουργίας (idle mode) με τη χρήση διάφορων τεχνικών ελάττωσης της στατικής ενέργειας, (για παράδειγμα με την τεχνική απομόνωσης ισχύος power gating).



Σχήμα 3.16: (a) Λειτουργία Υψηλής Απόδοσης (b) Μείωση της Συχνότητας του Ρολογιού (c) Λειτουργία Χαμηλής Απόδοσης και Χαμηλής Δυναμικής Κατανάλωσης (d) Λειτουργία Χαμηλής Στατικής Κατανάλωσης.

Οι διάφοροι μηχανισμοί ρύθμισης τάσης–συχνότητας εκμεταλλεύονται περιόδους όπου το κύκλωμα υπολειτουργεί και χωρίζονται σε τρεις κατηγορίες, ανάλογα με το επίπεδο στο οποίο λειτουργούν. Διακρίνονται στα εξής επίπεδα:

1. Επίπεδο συστήματος: σε αυτό το επίπεδο αναγνωρίζονται χρονικές περίοδοι όπου ολόκληρο το σύστημα διακρίνεται από χαμηλή παραγωγή έργου και λαμβάνεται η απόφαση για μείωση της συχνότητας προκειμένου να εξοικονομηθεί ενέργεια.

2. Επίπεδο προγράμματος ή φάσης εντός προγράμματος: οι μηχανισμοί που δρουν σε αυτό το επίπεδο διακρίνουν τις περιόδους όπου το πρόγραμμα χαρακτηρίζεται από χρονοβόρες λειτουργίες μνήμης. Σε αυτή την περίπτωση ο επεξεργαστής μπορεί να

επιβραδυνθεί χαμηλώνοντας τη συχνότητα (και επομένως και την κατανάλωση ισχύος) χωρίς σημαντική επιβάρυνση στην απόδοση.

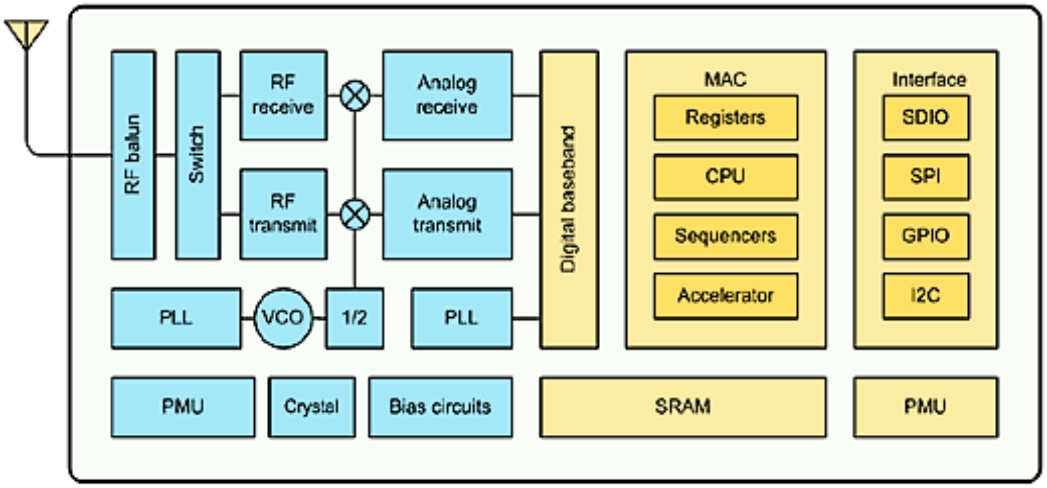
3. Επίπεδο υλικού: το επίπεδο αυτό βρίσκεται ακόμα πιο χαμηλά από το επίπεδο προγράμματος, κατευθείαν στο υλικό, και οι αντίστοιχοι μηχανισμοί προσπαθούν να εκμεταλλευτούν τυχούσα αδράνεια που κρύβεται μεταξύ των λειτουργιών του υλικού. [7]

Η μεγάλη επιτυχία της τεχνικής της δυναμικής κλιμάκωσης τάσης και συχνότητας (Dynamic voltage frequency scaling- DVFS) οφείλεται στην κυβική μείωση της κατανάλωσης ισχύος εξαιτίας της εξάρτησής της από (V, f). Ένας άλλος λόγος επιτυχίας της τεχνικής είναι ότι το γεγονός ότι ο μηχανισμός δυναμικής κλιμάκωσης τάσης και συχνότητας μπορεί να υλοποιηθεί σε πολλούς επεξεργαστές. Η DVFS είναι μια ευρέως διαδεδομένη τεχνική σε εμπορικά συστήματα χαμηλής κατανάλωσης αλλά και σε συστήματα υψηλής απόδοσης. Παραδείγματα επεξεργαστών χαμηλής κατανάλωσης που έχουν DVFS χαρακτηριστικά είναι οι IntelXScale και AMD Mobile K6 Plus, παραδείγματα επεξεργαστών υψηλής απόδοσης αποτελούν οι AMD Opteron Quad-Core και Intel Core i7. [31]

3.5 Τεχνικές χαμηλής κατανάλωσης ισχύος σε επίπεδο αρχιτεκτονικής

3.5.1 Τεχνική χαμηλής κατανάλωσης με απομόνωση της ισχύος (Power Gating)

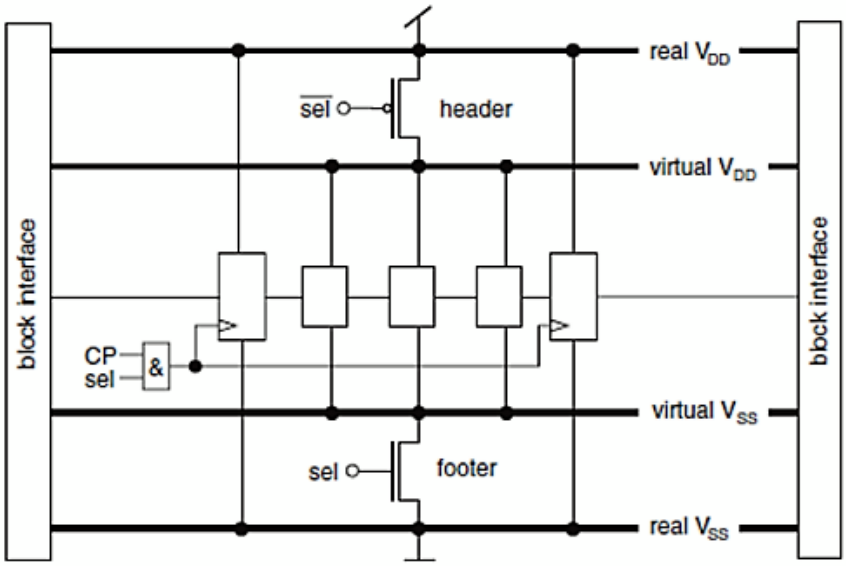
Στο Σχήμα 3.17, παρουσιάζεται ένα αντιπροσωπευτικό παράδειγμα ενός ολοκληρωμένου SoC – System On Chip. Από τα υποσυστήματα του ολοκληρωμένου, πολλά δεν χρειάζεται να είναι συνεχώς σε λειτουργία όπως επίσης είναι δυνατό αυτά τα υποσυστήματα να μην είναι ταυτόχρονα ενεργοποιημένα. Για αυτό το λόγο, η τεχνική χαμηλής κατανάλωσης με απομόνωση του ρολογιού (clock gating) μπορεί να χρησιμοποιηθεί για να αναστείλει τη λειτουργία μεγάλων blocks στο ολοκληρωμένο κύκλωμα για ένα επιλεγμένο χρονικό διάστημα. Αυτή η λύση θα γινόταν αποδεκτή όταν τα ρεύματα διαρροής μπορούσαν να αγνοηθούν στις παλιότερες τεχνολογίες. Στις σύγχρονες deep sub-micron τεχνολογίες, τα αδρανοποιημένα μπλοκς (idle blocks) εμφανίζουν μεγάλα ρεύματα διαρροής που συνεισφέρουν σε μεγάλο ποσοστό στη συνολική κατανάλωση ενέργειας.



Σχήμα 3.17: ESP8266 low cost WiFi SoC.

Μία αποτελεσματική λύση για την μείωση της στατικής κατανάλωσης είναι η τεχνική χαμηλής κατανάλωσης με απομόνωση ισχύος (power gating). Στο Σχήμα 3.18, φαίνεται ότι ένα αδρανοποιημένο κυκλωματικό μπλοκ αποκόπτεται και από το δέντρο διανομής ρολογιού (clock tree) αλλά και από την τροφοδοσία ισχύος (power supply). Για το λόγο αυτό, ένα τρανζίστορ επιβολής κατάστασης ύπνου (sleep transistor) παρεμβάλλεται μεταξύ του κυκλωματικού μπλοκ και της V_{DD} ή της V_{SS}. Το PMOS τρανζίστορ που απομονώνει το κυκλωματικό μπλοκ λέγεται header ενώ το NMOS τρανζίστορ λέγεται footer.

Το τρανζίστορ κατάστασης ύπνου μπορεί να τοποθετηθεί σε μία θέση ή μπορεί να διαιρεθεί σε πολλά μικρά τρανζίστορς που θα τοποθετηθούν διάσπαρτα στη περιοχή του κυκλωματικού μπλοκ. Το τοπικό δίκτυο διανομής της τροφοδοσίας που συνδέεται στη τάση τροφοδοσίας μέσω ενός τρανζίστορ επιβολής κατάστασης ύπνου λέγεται εικονική τάση τροφοδοσίας, έχοντας μία εικονική τάση VDD εάν προστεθεί ένα τρανζίστορ header και μία εικονική τάση VSS εάν προστεθεί ένα τρανζίστορ footer.



Σχήμα 3.18: Η αρχή λειτουργίας της τεχνικής απομόνωσης της ισχύος.

Με την αποκοπή της τροφοδοσίας από ένα κυκλωματικό μπλοκ μειώνονται τα συνολικά ρεύματα διαρροής. Η εναπομείνασα διαρροή καθορίζεται κυρίως από τις ιδιότητες και τις διαστάσεις του διακόπτη. Οπότε για τρανζίστορ σε κατάσταση ύπνου προτιμάται από ένα τρανζίστορ με υψηλή τάση κατωφλίου και παχύ οξείδιο πύλης με το πλάτος του τρανζίστορ (W) να είναι αισθητά μικρότερο από το συνολικό πλάτος των τρανζίστορς των λογικών πυλών στο κύκλωμα. Με αυτό το τρόπο η στατική κατανάλωση μπορεί να μειωθεί δύο και τρεις τάξεις μεγέθους. Μόλις οδηγηθεί στην αποκοπή το τρανζίστορ επιβολής κατάστασης ύπνου, το ρεύμα της συσκευής μειώνεται αυτόματα αλλά όλες οι τάσεις στην συσκευή είναι αρχικά αφόρτιστες.

Οπότε, όλα τα ρεύματα διαρροής στη λογική συνεχίζουν να ρέουν και να αποφορτίζουν τις εσωτερικές χωρητικότητες. Το εικονικό δίκτυο διανομής τάσης προσπαθεί να υπερισχύσει έναντι της τάσης τροφοδοσίας που δεν έχει συνδεθεί με τρανζίστορ σε κατάσταση ύπνου, μέχρι η τάση στη λογική να έχει τόσο μικρή τιμή ώστε το ρεύμα της λογικής να εξισωθεί με το ρεύμα διαρροής στη συσκευή του διακόπτη. Εξαιτίας της κατάρρευσης της τοπικής τάσης τροφοδοσίας, τα εσωτερικά στοιχεία αποθήκευσης (latches και flip-flops) χάνουν την αποθηκευμένη πληροφορία. Για αυτό χρειάζονται ειδικά flip-flops διατήρησης κατάστασης εάν η λογική πληροφορία πρέπει να μη χαθεί κατά τη φάση αδρανοποίησης. [2]

Το μπλοκ μπορεί να τεθεί ξανά σε λειτουργία με το να ενεργοποιήσουμε το τρανζίστορ σε κατάσταση ύπνου, απαιτώντας όμως όλες οι εσωτερικές χωρητικότητες να επαναφορτιστούν, με συνέπεια να εμφανίζεται μία μεγάλη αιχμή ρεύματος (large current spike) κατά την φάση ενεργοποίησης. Για να αποφευχθούν διαταραχές στη λειτουργία των γειτονικών μπλοκ, εξαιτίας αυτής της αιχμής ρεύματος, το μπλοκ πρέπει να ενεργοποιηθεί με ελεγχόμενο τρόπο. Εάν το μπλοκ ξαναλειτουργήσει, το τρανζίστορ σε κατάσταση ύπνου δεν έχει καμία επιρροή στην κατανάλωση ενέργειας, οπότε η τεχνική χαμηλής κατανάλωσης με απομόνωση ισχύος (power gating) είναι αποτελεσματική μόνο όταν υπάρχουν σημαντικά μεγάλες περίοδοι αδρανοποίησης όπου το κύκλωμα δεν απαιτείται να είναι σε λειτουργία.

Παρόλα αυτά υπάρχει η παρασιτική αντίσταση του τρανζίστορ επιβολής κατάστασης ύπνου που προκαλεί υποβάθμιση στην τοπική τάση τροφοδοσίας και αυτή με τη σειρά της μειώνει την ταχύτητα της λογικής. Το μειονέκτημα πρέπει να λαμβάνεται σοβαρά υπόψη κατά τη σχεδίαση του τρανζίστορ σε κατάσταση ύπνου. Το να ενεργοποιήσουμε και να απενεργοποιήσουμε ένα κύκλωμα τρανζίστορ σε κατάσταση ύπνου προκαλεί επιβάρυνση σε ενέργεια (energy overhead) και γι' αυτό η περίοδος αδρανοποίησης πρέπει να είναι αρκετά μεγάλη ώστε να μπορέσουμε να εξοικονομήσουμε την ενέργεια που καταναλώνεται για τη λειτουργία του διακόπτη. Αυτός ο χρόνος αδρανοποίησης λέγεται ελάχιστος χρόνος μη κατανάλωσης ενέργειας (minimum power-down time).

3.5.2 Τεχνική χαμηλής κατανάλωσης με απενεργοποίηση του ρολογιού κατά τμήματα (Clock gating)

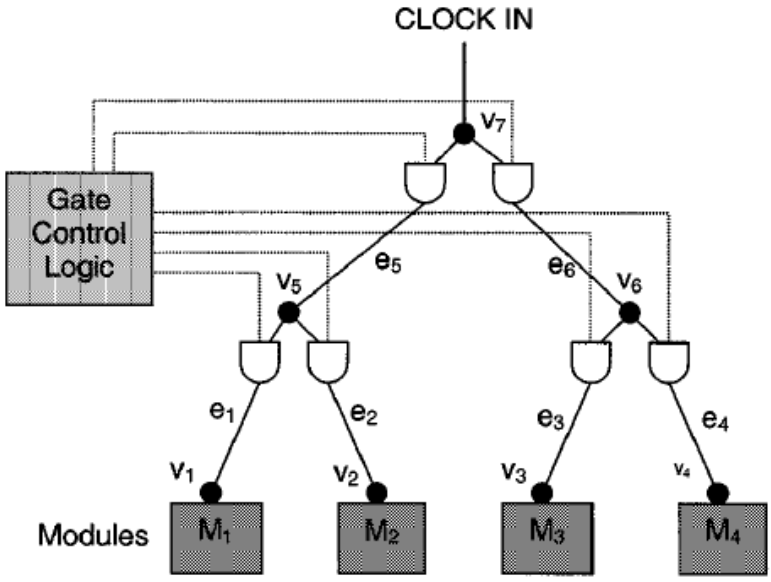
Είναι μια τεχνική που χρησιμοποιείται σε πολλά σύγχρονα κυκλώματα για την μείωση της δυναμικής κατανάλωσης ισχύος. Η τεχνική βασίζεται στο γεγονός ότι σε ένα κύκλωμα ένα μέρος είναι μονίμως ενεργό ενώ κάποια άλλα είναι ενεργά κατά διαστήματα. Η τεχνική αυτή εντοπίζει τα στοιχεία του συστήματος που δεν είναι ενεργά και τα αδρανοποιεί, περιορίζοντας στο ελάχιστο δυνατό τη δραστηριότητα μετάβασης, ελαττώνοντας την κατανάλωση ισχύος. [8], [14]

Η αποτελεσματικότητα του clock gating ωστόσο, απαιτεί μια μεθοδολογία που καθορίζει ποια κυκλώματα απενεργοποιούνται, πότε και για πόσο χρονικό διάστημα. Υλοποιήσεις που είτε έχουν ως αποτέλεσμα τη συχνή εναλλαγή των καταστάσεων ενεργοποίησης και απενεργοποίησης, ή η υλοποίηση σε τόσο μικρά μπλοκ που το clock gating κύκλωμα ελέγχου είναι σχεδόν τόσο μεγάλο όσο τα ίδια τα μπλοκ, μπορεί να οδηγήσουν σε κατανάλωση ισχύος υψηλότερη από όταν χωρίς το clock gating κύκλωμα. [13]

Η εξοικονόμηση ισχύος από τη συγκεκριμένη τεχνική είναι ιδιαίτερα σημαντική, κυρίως για αριθμούς πολλών bits. Υποθέτοντας ότι οι εισερχόμενοι δυαδικοί αριθμοί

έχουν τυχαία κατανομή, η δραστηριότητα μετάβασης άρα και η κατανάλωση ισχύος μπορεί να μειωθεί κατά 50%. [24]

Η αδρανοποίηση αυτή μπορεί να επιτευχθεί στα διάφορα μέρη του δέντρου ρολογιού (clock tree), χρησιμοποιώντας σαν μάσκα μια πύλη AND στο εσωτερικό κόμβο του δέντρου ρολογιού, αδρανοποιώντας τους κόμβους που ανήκουν σε αυτόν τον κλάδο, αποτρέποντας τις περιττές μεταβάσεις σήματος και επιτυγχάνοντας εξοικονόμηση ενέργειας στο σύστημα Σχήμα 3.19. [14], [25].



Σχήμα 3.19: Τοπολογία ενός gated clock tree.

Η τοποθέτηση της μάσκας στα τμήματα του δέντρου που συνδέονται με τους καταχωρητές των αδρανοποιημένων μπλοκ, διατηρεί τις εισόδους της συνδυαστικής λογικής του μπλοκ σταθερές και αποτρέπει κάθε πιθανή μετάβαση κατάστασης. Με αυτόν τον τρόπο μειώνεται η δυναμική κατανάλωση μέσα στο κύκλωμα. Κόβοντας το ρολόι σε αυτά μονοπάτια απενεργοποιούνται επίσης τα flip-flops που περιλαμβάνονται σε αυτά, με αποτέλεσμα να μην εναλλάσσεται η κατάστασή τους. Η μη αλλαγή κατάστασης, μηδενίζει τη κατανάλωση δυναμικής ισχύος, αλλά συνεχίζουν να υπάρχουν τα ρεύματα διαρροής.

Η δυναμική ενέργεια που καταναλώνουν τα δέντρα ρολογιού (clocktrees) είναι πάνω από 50%, και εξαρτάται από:

1. Την ενέργεια που καταναλώνουν τα στοιχεία της συνδυαστικής λογικής των οποίων η κατάσταση αλλάζει σε κάθε κύκλο ρολογιού.

2. Την ενέργεια που καταναλώνουν τα flip-flops ακόμη και όταν η κατάστασή τους δεν μεταβάλλεται.

3. Την ενέργεια που καταναλώνει το δένδρο διάδοσης ρολογιού (clock buffer tree).

Με την τεχνική clock gating μειώνονται οι δυο πρώτοι παραπάνω παράγοντες δυναμικής ενέργειας.

Το clock gating λειτουργεί λαμβάνοντας υπόψη τα enable σήματα των καταχωρητών για να οδηγήσει τα ρολόγια στα επιμέρους τμήματα. Είναι επομένως επιτακτικό ο σχεδιασμός να περιλαμβάνει enable σήματα ώστε το κύκλωμα να επωφεληθεί από το clock gating. Η διαδικασία clock gating εκτός από εξοικονόμηση ενέργειας μπορεί επίσης να συμβάλει στην μείωση χώρου που χρειάζεται ένα κύκλωμα καθώς ένας μεγάλος αριθμός πολυπλεκτών μπορεί να αντικατασταθεί από την clock gating λογική. Αξίζει να σημειωθεί ότι η λογική της clock gating τεχνικής αλλάζει και τη δομή του clock tree.

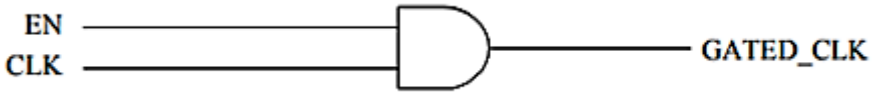
Το clock gating μπορεί να γίνει με δύο βασικούς τρόπους:

i. Clock gating τεχνική χωρίς μανδαλωτές (Latch free clock gating) και

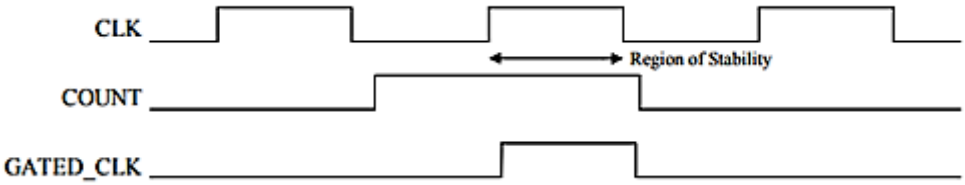
ii. Clock gating τεχνική βασισμένη σε μανδαλωτές (Latch-based clock gating)

Clock gating τεχνική χωρίς μανδαλωτές

Η τεχνική clock gating χωρίς μανδαλωτές (Σχήμα 3.20), χρησιμοποιεί απλές πύλες AND ή OR (ανάλογα με την ακμή που τριγκάρονται τα flip-flops). Αν το σήμα enable απενεργοποιηθεί είτε ενεργοποιηθεί σε έναν παλμό ρολογιού τότε το ρολόι που προκύπτει (gated clock) μπορεί είτε να απενεργοποιηθεί οριστικά είτε να δημιουργήσει πολλούς παλμούς (gated pulses).



Σχήμα 3.20: Gated AND πύλη. [32]

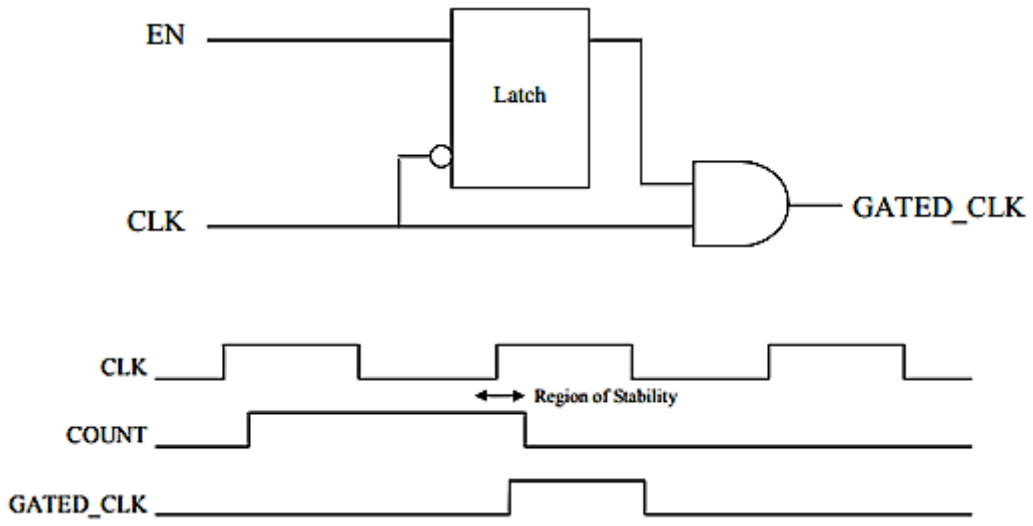


Σχήμα 3.21: Clock Gating χωρίς μανδαλωτές.

Στο παραπάνω Σχήμα 3.21, φαίνεται το παραγόμενο ρολόι Gated_Clk, το οποίο τροφοδοτείται στις εισόδους ρολογιού όλων των flip-flops που έχουν κοινό σήμα επίτρεψης(ENable).

Clock gating τεχνική βασισμένη σε μανδαλωτές

Η τεχνική clock gating που βασίζεται σε μανδαλωτές [26], προσθέτει στη σχεδίαση του συστήματος έναν μανδαλωτή ο οποίος τριγκάρεται σε ακμή του ρολογιού (level - sensitive latch) για να κρατήσει σε σταθερή τιμή το σήμα enable από την ενεργή μέχρι την ανενεργή ακμή του ρολογιού. Το σήμα enable πρέπει να είναι σταθερό κοντά στην ανερχόμενη παρυφή του ρολογιού, αφού ο μαναδαλωτής εντοπίζει και κρατάει την κατάσταση του μέχρι να παραχθεί ένας πλήρης παλμός ρολογιού, όπως σε έναν παραδοσιακό ungated σχεδιασμό.



Σχήμα 3.22: Clock Gating με μανδαλωτές.

Πιο αναλυτικά στο παραπάνω Σχήμα 3.22, έχουμε την πιο απλή μορφή εφαρμογής της τεχνικής, όπου εντοπίζεται το σήμα που καθορίζει εάν το latch θα έχει νέα δεδομένα στο τέλος του κύκλου. Εάν δεν θα έχει, το ρολόι απενεργοποιείται με βάση το σήμα EN που έχει την πληροφορία του εάν το latch θα διατηρήσει την προηγούμενη τιμή του ή εάν θα πάρει νέα είσοδο. Αυτό το σήμα EN μαζί με το σήμα του ρολογιού τροφοδοτούν μία πύλη AND για να επιτύχουν την ενεργοποίηση του ρολογιού κατά τμήματα (gated clock). Αυτή η τεχνική μειώνει τη δυναμική κατανάλωση του σύγχρονου κυκλώματος κατά 5- 10%. [32], [27]

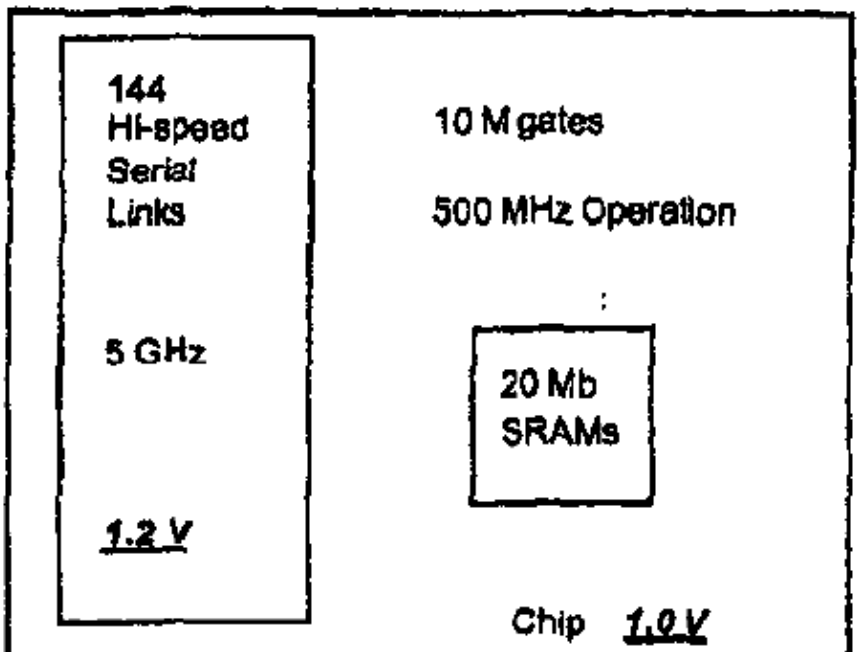
3.5.3 Η τεχνική των τμημάτων τάσης (Voltage islands)

Η ενσωμάτωση σε ένα ολοκληρωμένο όλο και περισσότερων συσκευών, επιτρέπει πολλαπλές εφαρμογές να ενσωματωθούν στις σχεδιάσεις System-on-Chip - SoC. Ένα SoC αποτελείται κυρίως από προγραμματιζόμενους επεξεργαστές και περιφερειακούς πυρήνες που συνδέονται μεταξύ τους με διαμοιραζόμενες αρχιτεκτονικές (bus-based αρχιτεκτονικές). Το να ικανοποιηθούν οι χρονικοί περιορισμοί στις σύγχρονες σχεδιάσεις SoC ήταν μιά δύσκολη υπόθεση και η κατανάλωση ισχύος έχει γίνει μία ακόμα πιό κρίσιμη σχεδιαστική πρόκληση. Υπάρχουν πολλές τεχνικές για μείωση της κατανάλωσης ενέργειας σε κάθε επίπεδο της σχεδιαστικής διαδικασίας ενός SoC.

Η σχεδίαση που βασίζεται σε πυρήνες (core-base design) είναι μια τεχνική που επιτρέπει τη μείωση των στατικών κα δυναμικών χαρακτηριστικών της κατανάλωσης ενέργειας. [11] Πιο συγκεκριμένα, ένα τμήμα τάσης (voltage island) είναι μια ομάδα από πυρήνες πάνω στο chip που τροφοδοτούνται από την ίδια πηγή τάσης, ανεξάρτητα από την πηγή τάσης στο επίπεδο του ολοκληρωμένου. Η χρήση των τμημάτων τάσης (voltage islands) επιτρέπει τη λειτουργία διαφορετικών τμημάτων στη σχεδίαση σε διαφορετικά επίπεδα τάσης ώστε να βελτιστοποιηθεί η συνολική κατανάλωση ενέργειας στο ολοκληρωμένο. Στο πλαίσιο ενός SoC, το τμήμα τάσης επιτρέπει τη βελτιστοποίηση ισχύος σε επίπεδο πυρήνα χρησιμοποιώντας μια πηγή τάσης που είναι διαφορετική από αυτήν της υπόλοιπης σχεδίασης. [2]

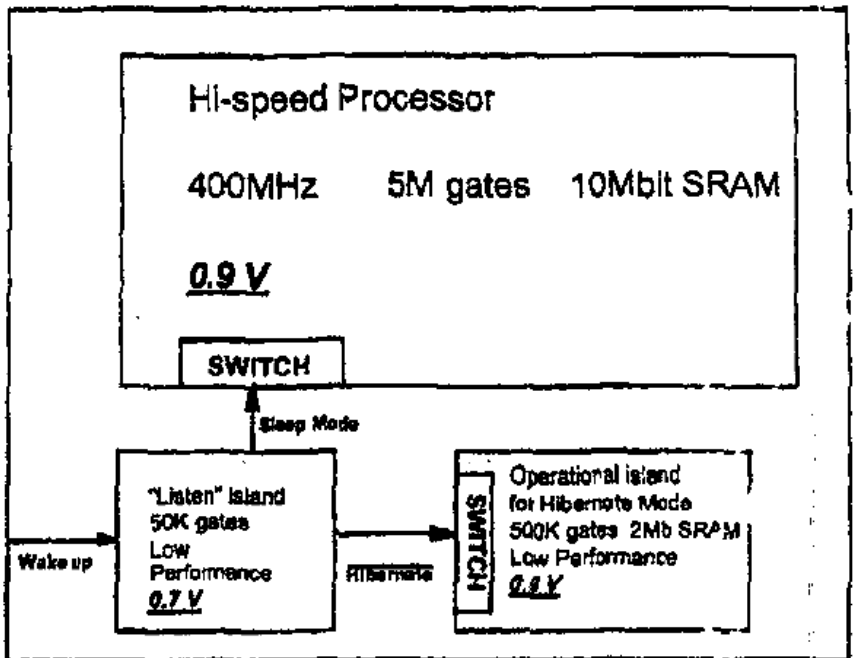
Υπάρχουν διάφορες υλοποιήσεις που δείχνουν το σχεδιαστικό πλεονέκτημα της τεχνικής των “Τμημάτων τάσης”.

Μια τέτοια υλοποίηση, φαίνεται στο Σχήμα 3.23, με τον καθορισμό της ελάχιστης τάσης που απαιτείται για κάθε τμήμα (island) ώστε να επιτυγχάνεται η απαραίτητη απόδοση. Συνήθως το πιο κρίσιμο μπλοκ ως προς την απόδοση σε μία σχεδίαση, είναι ο πυρήνας του επεξεργαστή και απαιτεί το υψηλότερο επίπεδο τάσης που υποστηρίζεται από την τεχνολογία ώστε να μεγιστοποιείται η απόδοση του. Άλλα μπλοκ που συνυπάρχουν στο SoC όπως οι μνήμες και η μονάδα ελέγχου (control unit), μπορεί να μη απαιτούν τόσο υψηλό επίπεδο τάσης, οδηγώντας σε σημαντική μείωσης της δυναμικής κατανάλωσης εάν λειτουργήσουν σε χαμηλότερες τάσεις.



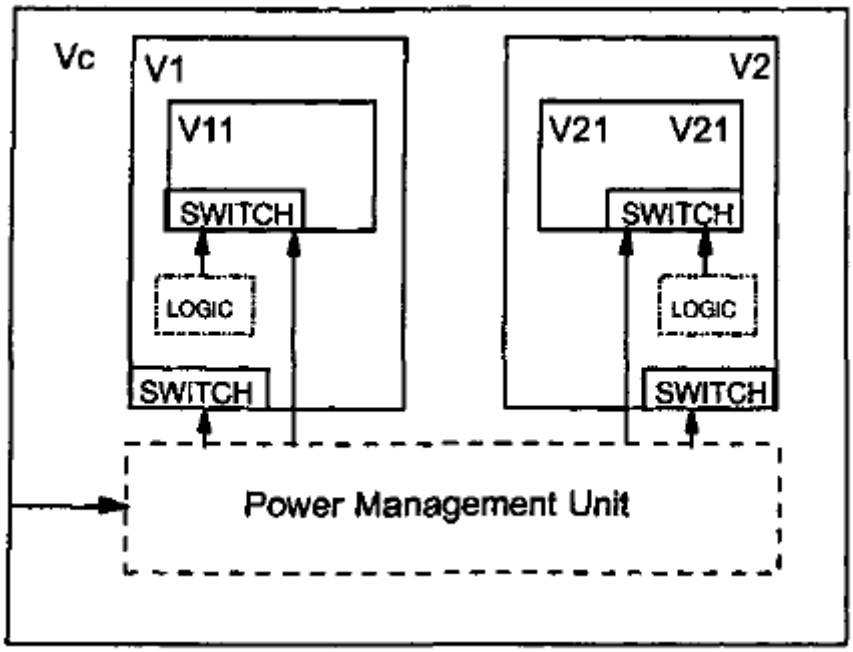
Σχήμα 3.23: Τμήμα τάσης χρονικά-κρίσιμο.

Μία ακόμα υλοποίηση, που φαίνεται στο Σχήμα 3.24, εφαρμόζει εξοικονόμηση ισχύος σε εφαρμογές πιο ευαίσθητες σε στατική κατανάλωση όπως συσκευές που χρησιμοποιούν μπαταρίες. Συνήθως, οι πολύπλοκες σχεδιάσεις SoC αποτελούνται από διαφορετικές μονάδες που λίγες από αυτές λειτουργούν συνεχώς. Τεχνικές όπως η αποσύνδεση του ρολογιού (clock gating) σε ένα τμήμα, μπορούν να χρησιμοποιηθούν για περιορισμό της δυναμικής κατανάλωσης στην κατάσταση αδρανοποίησης, αν και τα ρεύματα διαρροής παραμένουν και μπορεί να είναι σημαντικά σε κυκλώματα υψηλής απόδοσης. Εάν οι τροφοδοσίες τάσης για αυτά τα μπλοκ χωριστούν σε τμήματα (islands) το μπλοκ μπορεί να απενεργοποιηθεί πλήρως, εξαλείφοντας τη δυναμική και τη στατική κατανάλωση.



Σχήμα 3.24: Τμήμα τάσης για διάταξη παροχής ισχύος.

Οι μέθοδοι αρχιτεκτονικής των Voltage Island, μπορούν να βελτιώσουν τη χρησιμότητα των multi-threshold τεχνικών σχεδιασμού. Ένα τμήμα τάσης μπορεί να δημιουργηθεί για να τροφοδοτεί ένα μπλοκ με χαμηλές Vth ευαίσθητο στη δυναμική κατανάλωση ισχύος, σε χαμηλότερη τάση από την υπόλοιπη σχεδίαση. Επιπλέον, αυτό το μπλοκ χαμηλού Vth, μπορεί να απενεργοποιηθεί εντελώς κατά τη διάρκεια των sleep modes έτσι ώστε να ελαχιστοποιηθεί η στατική ισχύς. Ομοίως, τμήματα που είναι πάντα ενεργοποιημένα μπορούν να κρατηθούν σε υψηλότερη τάση έτσι ώστε να παρουσιάσουν μικρότερη διαρροή τα τρανζίστορς υψηλής Vth (Σχήμα 3.25). [11]



Σχήμα 3.25: Multi-level Voltage Islands αρχιτεκτονική.

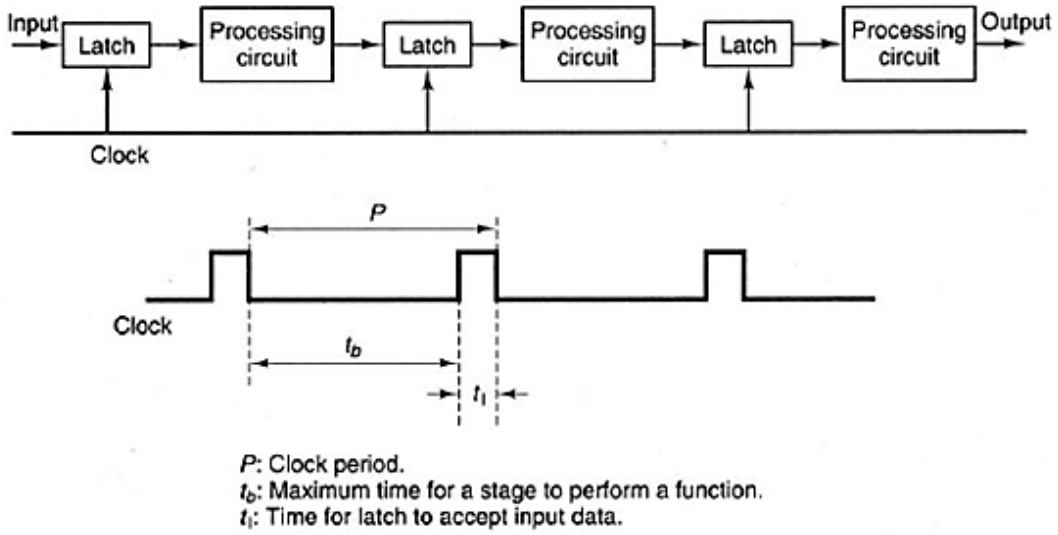
3.5.4 Διασωλήνωση – Pipelining

Το Pipelining είναι ένας τρόπος βελτίωσης της συνολικής απόδοσης επεξεργασίας ενός επεξεργαστή, που κατά συνέπεια επιφέρει μείωση στη κατανάλωση ισχύος. Αυτή η αρχιτεκτονική προσέγγιση επιτρέπει την ταυτόχρονη εκτέλεση πολλών οδηγιών, εκμεταλλευόμενη τον παραλληλισμό σε επίπεδο εντολής επικαλύπτοντας τη διαδικασία εκτέλεσης των εντολών. Είναι ανάλογο με μια γραμμή συναρμολόγησης σε ένα εργοστάσιο, όπου οι εργαζόμενοι εκτελούν μια συγκεκριμένη εργασία και περνούν το μερικώς ολοκληρωμένο προϊόν στον επόμενο εργαζόμενο. [4]

Η δομή του pipelining

Η pipeline τεχνική σχεδιασμού, αποσυνθέτει μια διαδοχική διαδικασία σε διάφορες υποεπεξεργασίες, που ονομάζονται στάδια ή τμήματα. Ένα στάδιο εκτελεί μια συγκεκριμένη λειτουργία και παράγει ένα ενδιάμεσο αποτέλεσμα και αποτελείται από ένα latch εισόδου που ονομάζεται καταχωρητής ή buffer, ακολουθούμενο από το κύκλωμα επεξεργασίας(ένα κύκλωμα επεξεργασίας μπορεί να είναι ένα συνδυαστικό ή ακολουθιακό κύκλωμα). Το κύκλωμα επεξεργασίας ενός δεδομένου σταδίου συνδέεται με το latch εισόδου του επόμενου σταδίου (Σχήμα 3.26). Το σήμα ρολογιού συνδέεται σε κάθε latch εισόδου και σε κάθε παλμό ρολογιού κάθε στάδιο μεταφέρει το ενδιάμεσο αποτέλεσμα στο latch εισόδου του επόμενου σταδίου. Με αυτόν τον τρόπο, το τελικό αποτέλεσμα παράγεται αφού περάσουν τα δεδομένα εισόδου σε ολόκληρη τη

διασωλήνωση, ολοκληρώνοντας ένα στάδιο ανά παλμό ρολογιού. Η περίοδος του παλμού του ρολογιού πρέπει να είναι αρκετά μεγάλη ώστε να παρέχεται επαρκής χρόνος για ένα σήμα να διασχίσει το πιο αργό στάδιο, το οποίο ονομάζεται bottleneck (το στάδιο εκείνο που χρειάζεται τον περισσότερο χρόνο για ολοκλήρωση). Επιπλέον, πρέπει να υπάρχει αρκετός χρόνος έτσι ώστε το latch να αποθηκεύσει τα σήματα εισόδου. Εάν η περίοδος του ρολογιού P, εκφράζεται ως $P=t_b + t_l$, τότε το t_b θα πρέπει να είναι μεγαλύτερο από τη μέγιστη καθυστέρηση του bottleneck σταδίου και το t_l πρέπει να είναι επαρκές για την αποθήκευση των δεδομένων στο latch. [34]



Σχήμα 3.26: Η βασική δομή pipeline.

Η απόδοση του pipelining

Η ικανότητα αλληλεπικάλυψης σταδίων μιας διαδοχικής διαδικασίας για διαφορετικές εργασίες εισόδου (δεδομένα ή λειτουργίες), οδηγεί σε συνολικό θεωρητικό χρόνο ολοκλήρωσης:

$$T_{pipe} = m*P + (n-1)*P$$

όπου n είναι ο αριθμός των εισερχόμενων εργασιών, m είναι ο αριθμός των σταδίων διασωλήνωσης και P είναι η περίοδος ρολογιού.

Ο όρος m * P είναι ο χρόνος που απαιτείται για να περάσει η πρώτη εισερχόμενη εργασία μέσω του «αγωγού» και ο όρος (n-1) * P είναι ο χρόνος που απαιτείται για τις υπόλοιπες εργασίες. Αφού εισαχθούν τα δεδομένα, παράγεται έξοδος σε κάθε κύκλο ρολογιού. Με άλλα λόγια, θα παραχθούν δεδομένα εξόδου τόσο γρήγορα, όσο επιτρέπει το πιο αργό στάδιο. Ακόμα και με αυτόν τον περιορισμό, οι επιδόσεις του

pipelining θα ξεπεράσουν σε μεγάλο βαθμό τις μη pipelined τεχνικές, που απαιτούν την ολοκλήρωση κάθε εργασίας πριν ξεκινήσει η ακολουθία εκτέλεσης μιας άλλης εργασίας. Πιο συγκεκριμένα, όταν το n είναι μεγάλο, ένας επεξεργαστής με διασωληνώσεις μπορεί να παράγει έξοδο περίπου m φορές γρηγορότερα από ό, τι ένας μη-pipelined επεξεργαστής. Από την άλλη πλευρά, σε έναν μη pipelined επεξεργαστή, η παραπάνω διαδοχική διαδικασία απαιτεί ένα χρόνο ολοκλήρωσης:

$$T_{seq} = n^{*}m^{*}P.$$

Η παράμετρος speedup (S) του pipelining, μπορεί να δοθεί ως:

$$S = T_{seq} / T_{pipe} = n^{*}m / (m+n -1)$$

Η τιμή S πλησιάζει το m όταν $n \rightarrow \infty$. Δηλαδή, η μέγιστη ταχύτητα, που ονομάζεται επίσης ιδανική ταχύτητα ενός pipelined επεξεργαστή με στάδια m, έναντι ενός ισοδύναμου επεξεργαστή χωρίς pipeline, είναι m. Με άλλα λόγια, η ιδανική επιτάχυνση ισούται με τον αριθμό των σταδίων του pipelining. Δηλαδή, όταν το n είναι πολύ μεγάλο, μπορεί ένας pipelined επεξεργαστής να παράγει έξοδο περίπου m φορές πιο γρήγορα από έναν επεξεργαστή χωρίς pipelining. Όταν το n είναι μικρό, η επιτάχυνση μειώνεται. Στην πραγματικότητα, για n = 1 ο «αγωγός» έχει την ελάχιστη ταχύτητα 1. [34]

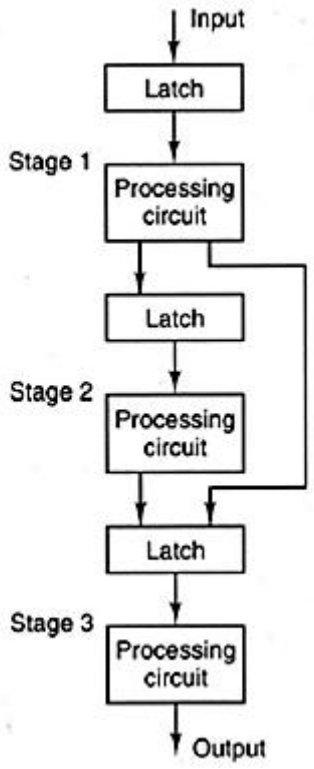
Τύποι pipelining

Οι διασωληνώσεις συνήθως χωρίζονται σε δύο κατηγορίες: στη διασωλήνωση εντολών (instruction) και στην αριθμητική (arithmetic) διασωλήνωση. Η διασωλήνωση σε κάθε μία από αυτές τις κατηγορίες, μπορεί να σχεδιαστεί με δύο τρόπους: στατικό ή δυναμικό.

Μία στατική διασωλήνωση μπορεί να αποδώσει μόνο μία λειτουργία κάθε φορά(όπως η πρόσθεση ή ο πολλαπλασιασμός) και η λειτουργία της μπορεί να αλλάξει μόνο όταν και τα τελευταία δεδομένα εισόδου φτάσουν στην έξοδό της. Για παράδειγμα, σε μία στατική διασωλήνωση που μπορεί να εκτελέσει πρόσθεση και πολλαπλασιασμό, κάθε φορά που αλλάζει η λειτουργία από πολλαπλασιασμό σε λειτουργία πρόσθεσης, πρέπει να έχουν εξαχθεί τα τελευταία δεδομένα πριν ρυθμιστεί για τη νέα λειτουργία, με αποτέλεσμα η απόδοση των στατικών διασωληνώσεων να υποβαθμίζεται σοβαρά όταν οι λειτουργίες αλλάζουν συχνά.

Στη δυναμική διασωλήνωση μπορούν να εκτελεστούν περισσότερες από μία εργασίες τη φορά. Για να εκτελεστεί μια συγκεκριμένη λειτουργία σε δεδομένα εισόδου, τα δεδομένα αυτά πρέπει να περάσουν από μια συγκεκριμένη σειρά σταδίων. Για

παράδειγμα, το Σχήμα 3.27 δείχνει μία δυναμική διασωλήνωση αποτελούμενη από τρία στάδια, που εκτελεί πρόσθεση και πολλαπλασιασμό σε διαφορετικά δεδομένα ταυτόχρονα. Για να εκτελεστεί πολλαπλασιασμός, τα δεδομένα εισαγωγής πρέπει να περάσουν από τα στάδια 1, 2 και 3 και για να πραγματοποιηθεί πρόσθεση, τα δεδομένα χρειάζεται μόνο να περάσουν από τα στάδια 1 και 3. Επομένως, το πρώτο στάδιο της διαδικασίας της πρόσθεσης μπορεί να πραγματοποιηθεί με δεδομένα εισόδου D1 στο στάδιο 1, ενώ ταυτόχρονα το τελευταίο στάδιο της διαδικασίας πολλαπλασιασμού εκτελείται στο στάδιο 3 σε διαφορετικά δεδομένα εισόδου D2. [34]

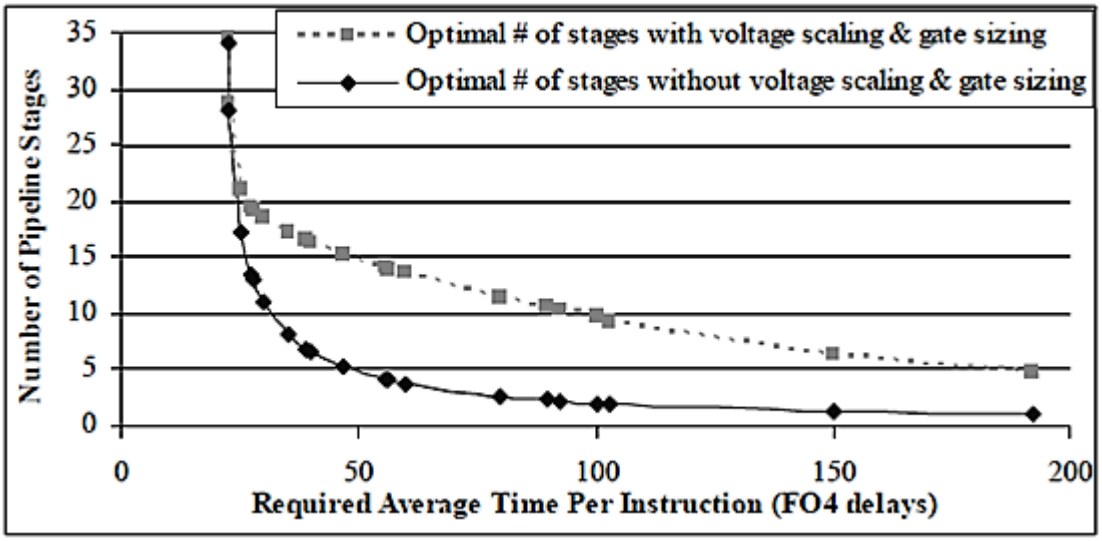


Σχήμα 3.27 Dynamic pipeline τριών σταδίων.

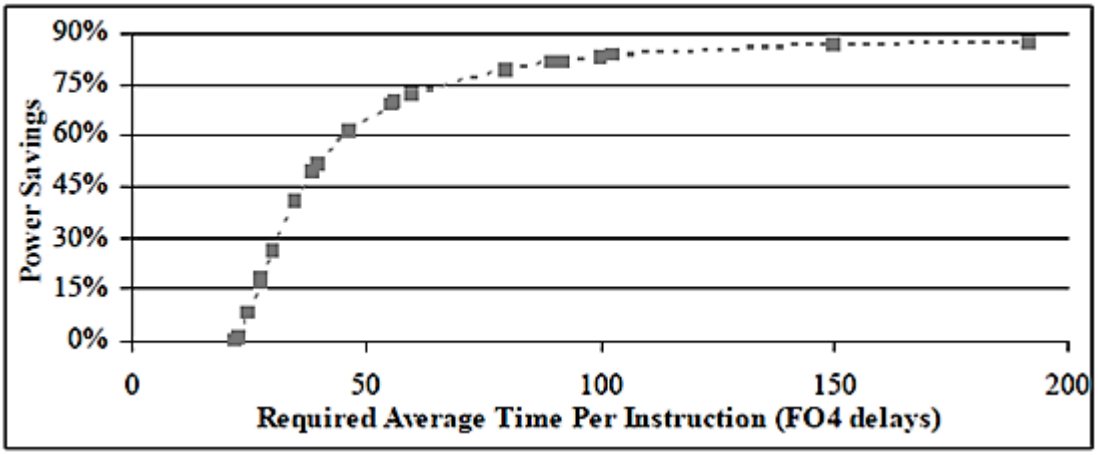
Το pipelining και η κατανάλωση ισχύος

Το pipelining δεν μειώνει την ισχύ από μόνο του. Το pipelining μειώνει τη καθυστέρηση της κρίσιμης διαδρομής, εισάγοντας καταχωρητές μεταξύ της συνδυαστικής λογικής και όσο μικρότερο είναι το μήκος της κρίσιμης διαδρομής τόσο υψηλότερη μπορεί να είναι η συχνότητα ρολογιού, όμως το σήμα ρολογιού των καταχωρητών έχει υψηλή δραστηριότητα η οποία συμβάλλει στη αύξηση της δυναμικής ισχύος. Το pipelining μειώνει τις εντολές ανά κύκλο ρολογιού (IPC – instructions per cycle), μειώνοντας ταυτόχρονα και την ενεργειακή απόδοση. Όμως το περιθώριο χρόνου(timing slack) λόγω pipelining, μπορεί να χρησιμοποιηθεί για τη κλιμάκωση της

τάσης τροφοδοσίας και τη μείωση του μεγέθους της πύλης έτσι ώστε να υπάρξει σημαντική εξοικονόμηση ενέργειας (βλ. Σχήμα 3.28 και Σχήμα 3.29). [33]



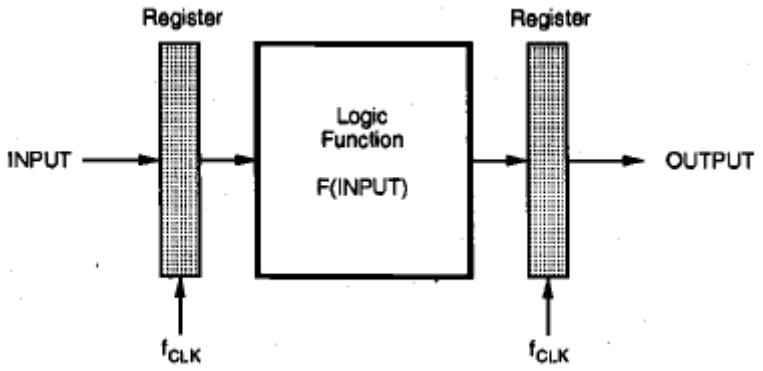
Σχήμα 3.28: Βέλτιστος αριθμός pipeline σταδίων για την ελαχιστοποίηση της ενέργειας/εντολή σε σχέση με τους περιορισμούς της απόδοσης. Σε υψηλό περιορισμό απόδοσης, επιπρόσθετα στάδια επιβαρύνουν περαιτέρω την απόδοση, μειώνουν το περιθώριο χρόνου και υπάρχει μικρότερη δυνατότητα για κλιμάκωση τάσης και μείωση του μεγέθους πύλης.



Σχήμα 3.29: Εξοικονόμηση ενέργειας με επιπρόσθετα pipeline στάδια για εξασφάλιση χρονικού περιθωρίου.

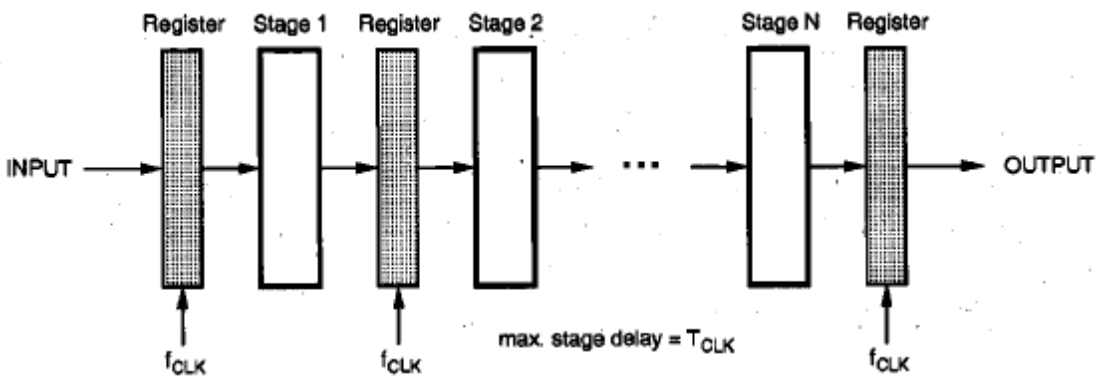
Στο σχήμα 3.30, φαίνεται μία γενική δομή κυκλώματος όπου ένας καταχωρητής χρησιμοποιείται για την προσωρινή αποθήκευση των δεδομένων στην είσοδο και έξοδο του λογικού κυκλώματος. Εάν ορίσουμε ως C_{total} το άθροισμα της χωρητικότητας που αποτελείται από (i) τη χωρητικότητα του καταχωρητή εισόδου, ii) τη χωρητικότητα του λογικού σταδίου και iii) τη χωρητικότητας του καταχωρητή εξόδου, τότε η κατανάλωση ισχύος του συνολικού κυκλώματος θα είναι:

$$P_{reference} = C_{total} \cdot V_{DD}^{2} \cdot f_{CLK}$$



Σχήμα 3.30: Κυκλωματική δομή μίας βαθμιδας.

Ας θεωρήσουμε μια άλλη προσέγγιση στην υλοποίηση μια λογικής συνάρτησης, την τεχνική της διασωλήνωσης (pipeline), τότε στη περίπτωση αυτή το λογικό κύκλωμα διαχωρίζεται σε N λογικές βαθμίδες, μεταξύ των οποίων εισάγονται καταχωρητές (Σχήμα 3.31). [3]



Σχήμα 3.31: Δομή pipeline με N βαθμίδες.

Η κατάτμηση του αρχικού λογικού κυκλώματος σε μικρότερες λογικές μονάδες, έχει σαν αποτέλεσμα τη μείωση του χρόνου καθυστέρησης του κρίσιμου μονοπατιού και ως εκ τούτου, είναι δυνατόν οι λογικές βαθμίδες να λειτουργήσουν N φορές πιο αργά, διατηρώντας τη ρυθμοδότηση εξόδου (throughput) ίδια με αυτή του αρχικού κυκλώματος. Η μείωση της ταχύτητας λειτουργίας κατά N φορές, μπορεί να επιτευχθεί ελαττώνοντας την τάση τροφοδοσίας [3].Έστω $V_{DD,new}$ η νέα τάση τροφοδοσίας του κυκλώματος διασωλήνωσης και η κατανάλωση ισχύος που θα εμφανίζει θα είναι:

$$P_{pipeline} = \left[C_{total} + (N-1)C_{reg}\right] \cdot V_{DD,new}^{2} \cdot f_{CLK}$$

όπου C_{reg} η χωρητικότητα του κάθε καταχωρητή.

Οπότε ο παράγοντας κέρδους της κατανάλωσης ισχύος που εισάγει μια δομή διασωλήνωσης N βαθμιδών είναι:

$$\frac{P_{pipeline}}{P_{reference}} = \left[1 + \frac{C_{reg}}{C_{total}}(N-1)\right] \frac{V_{DD,new}^{2}}{V_{DD}^{2}}$$

3.5.5 Παραλληλισμός (Parallelism)

Μια άλλη τεχνική που επιτρέπει την μείωση της κατανάλωσης της ισχύος, στο επίπεδο της αρχιτεκτονικής, είναι ο παραλληλισμός. [15], [19] Η συγκεκριμένη τεχνική, είναι ιδιαίτερα χρήσιμη όταν η λογική συνάρτηση που πρέπει να υλοποιηθεί δεν επιτρέπει την εφαρμογή της διασωλήνωσης. Μια παράλληλη αρχιτεκτονική αποτελείται από N ίδια κυκλώματα (μονάδες επεξεργασίας), συνδεδεμένα παράλληλα μεταξύ τους (Σχήμα 3.32(a)). Το δεδομένα εισόδου διανέμονται στα N κυκλώματα. Κάθε καταχωρητής εισόδου ενεργοποιείται από ένα τοπικό σήμα ρολογιού με συχνότητα N φορές μικρότερη από την συχνότητα του ρολογιού (Σχήμα 3.32(β)). Υπάρχουν συνολικά N τοπικά σήματα ρολογιού CLK1→CLKN. Προκειμένου κάθε δεδομένο εισόδου να αποθηκεύεται στον αμέσως επόμενο καταχωρητή, κάθε ένα από τα N σήματα ρολογιού είναι καθυστερημένο σε σχέση με το προηγούμενο κατά T_{CLK}. Οι έξοδοι των N κυκλωμάτων αφού πολυπλεχθούν, οδηγούνται στον καταχωρητή εξόδου που λειτουργεί σε συχνότητα f_{CLK}. [1]

Καθώς το ρολόι των καταχωρητών εισόδου λειτουργεί με συχνότητα f_{CLK}/N, ο χρόνος υπολογισμού για κάθε λογικό κύκλωμα αυξάνεται κατά N φορές. Έτσι η τάση τροφοδοσίας μπορεί να ελαττωθεί στην τιμή $V_{DD,new}$, προκειμένου ο χρόνος

καθυστέρησης του κρίσιμου μονοπατιού να ισούται με την περίοδο του τοπικού ρολογιού N*TCLK.

Ο παραλληλισμός και η κατανάλωση ισχύος.

Η συνολική δυναμική κατανάλωση ισχύος της παράλληλης αρχιτεκτονικής με N blocks, δίνεται από τη σχέση:

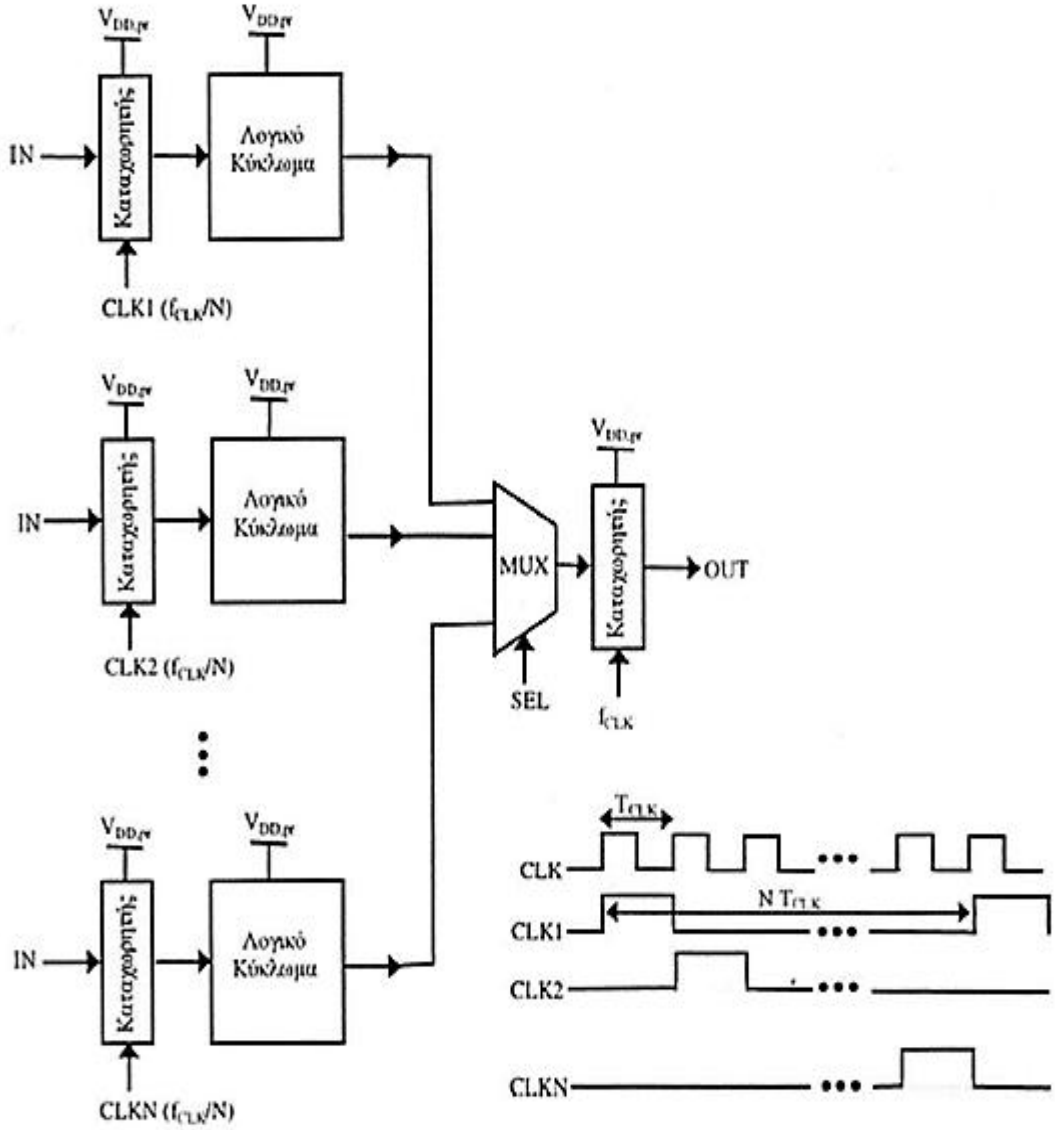
$$P_{parallel} = N \cdot C_{total} \cdot V_{DD,new}{}^2 \cdot \frac{f_{CLK}}{N} + C_{reg} \cdot V_{DD,new}{}^2 \cdot f_{CLK}$$
$$= \left(1 + \frac{C_{reg}}{C_{total}}\right) \cdot C_{total} \cdot V_{DD,new}{}^2 \cdot f_{CLK}$$

Ο πρώτος όρος εκφράζει την κατανάλωση ισχύος από τα N κυκλώματα και καταχωρητές εισόδου, που λειτουργούν με συχνότητα ρολογιού fCLK/N και ο δεύτερος οφείλεται στον καταχωρητή εξόδου που λειτουργεί σε συχνότητα fCLK.

Οπότε το κέρδος σε σχέση με την κλασσική υλοποίηση θα είναι:

$$\frac{P_{parallel}}{P_{reference}} = \frac{V_{DD,new}{}^2}{V_{DD}{}^2} \cdot \left(1 + \frac{C_{reg}}{C_{total}}\right)$$

Καθώς ο αριθμός N αυξάνεται, αυξάνεται και η επιφάνεια ολοκλήρωσης. Αυτή η αύξηση οφείλεται στα επιπλέον κυκλώματα που χρησιμοποιούνται, αλλά και στο πλήθος των γραμμών διασύνδεσης που απαιτούνται για την δρομολόγηση των σημάτων. Κατά συνέπεια αυξάνεται η κατανάλωση ισχύος, επιβάλλοντας έτσι για κάθε εφαρμογή μια βέλτιστη τιμή του N για την οποία ελαχιστοποιείται η κατανάλωση ισχύος. Μεγαλύτερες τιμές του N, αυξάνουν ακόμα περισσότερο την αρχική καθυστέρηση του συστήματος. [3]



ΣΧΗΜΑ 3.32: α) Παράλληλη αρχιτεκτονική, β) Παλμοί ρολογιού.

3.6 Η μονάδα διαχείρισης ενέργειας (Power Management Unit -PMU)

3.6.1 Εισαγωγή

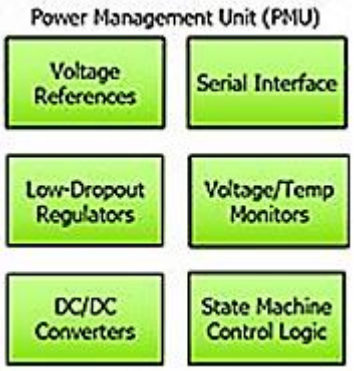
Κυριότερος από πολλούς παράγοντες που πρέπει να ληφθούν υπόψη κατά την ανάπτυξη ενός συστήματος κυρίως σε εφαρμογές System On Chip (SoC), αλλά και κατά την σχεδίαση CPUs και MCUs, είναι τα κυκλώματα διαχείρισης ισχύος. Πολλές υλοποιήσεις συστημάτων, περιέχουν τμήματα κυκλωμάτων που λειτουργούν σε διαφορετικές τάσεις και επομένως, είναι επωφελής η δημιουργία πολλαπλών τάσεων από μία μόνο τροφοδοσία για την αποτελεσματική διαχείριση της ισχύος. Υπάρχουν πολλές τεχνικές για τη σχεδίαση μιας μονάδας διαχείρισης ενέργειας Power Management Unit(PMU), που έχουν σαν κύρια λειτουργία τον έλεγχο του πότε κάθε τμήμα και κάθε περιφερειακό θα τροφοδοτηθεί και με ποια τάση. [41]

Τα τμήματα της μονάδας διαχείρισης ισχύος, είναι κυκλώματα αναλογικού και μικτού σήματος εξειδικευμένα για κάθε εφαρμογή και κάποια από αυτά είναι οι LDO ρυθμιστές, οι DC-DC μετατροπείς, οι αναφορές τάσεων(voltage references) και άλλα μπλοκ κυκλωμάτων όπως ταλαντωτές, παρακολουθητές τάσης, αισθητήρες θερμοκρασίας και power-on reset (POR) κυκλώματα.

Η χρήση αυτών των διαφόρων τμημάτων της PMU, γενικά δεν είναι ανεξάρτητη. Συχνά αυτό που απαιτείται είναι μια πολύ προσεκτικά ελεγχόμενη αλληλουχία τροφοδοσίας και απενεργοποίησης των διαφόρων παροχών τάσης και η δυνατότητα τερματισμού ορισμένων από αυτών κατά τη διάρκεια καθορισμένων καταστάσεων λειτουργίας. Για τον έλεγχο των διαφόρων τμημάτων της PMU, συνήθως προστίθεται και ένα κύκλωμα λογικής κατάστασης(Logic State Machine) για να διασφαλιστεί ο σωστός χρονισμός και αλληλουχία της έναρξης και διακοπής της τροφοδοσίας και για τον καθορισμό των διαφόρων καταστάσεων χαμηλής ισχύος. Αυτή η μηχανή λογικής κατάστασης ελέγχεται συχνά από έναν μικροελεγκτή ή έναν μικροεπεξεργαστή μέσω μιας σειριακής διεπαφής. [38]

3.6.2 Τα μπλοκ της μονάδας διαχείρισης ισχύος

Τα βασικά τμήματα της μονάδας διαχείρισης ισχύος, είναι οι LDO ρυθμιστές, οι DC-DC μετατροπείς, οι αναφορές τάσεων(voltage references) και κάποια άλλα τμήματα κυκλωμάτων όπως ο ταλαντωτής, ο παρακολουθητής τάσης, ο αισθητήρας θερμοκρασίας και το power-on reset (POR) κύκλωμα. (Σχήμα 3.33) [40]



Σχήμα 3.33: Η δομή των blocks της PMU

Οι Low Dropout ρυθμιστές (LDOs)

Οι LDOs χρησιμοποιούνται συχνά για σκοπούς τροφοδοσίας των πολλών διαφορετικών τάσεων που απαιτούνται στο SoC και για την απομόνωση του εσωτερικού κυκλώματος SoC από εξωτερικές πηγές θορύβου. Τα περισσότερα κυκλώματα LDO απαιτούν εξωτερικό πυκνωτή στην έξοδο τους για δύο λόγους. Ο πρώτος λόγος είναι ότι ένας LDO είναι ένα κύκλωμα αρχιτεκτονικής κλειστού βρόχου και απαιτείται πυκνωτής στην έξοδο για την επίτευξη σταθερότητας στη λειτουργία του βρόχου. Η αφαίρεση του πυκνωτή μπορεί να οδηγήσει τον LDO να γίνει ασταθής ή και ακόμη να αρχίσει να ταλαντώνει. Ο δεύτερος λόγος της ύπαρξης του πυκνωτή στην έξοδο, είναι να βοηθήσει το LDO να επιτύχει τη δηλωμένη απόδοση θορύβου, τόσο ως προς την τάση θορύβου στην έξοδο όσο και την απόρριψη θορύβου τροφοδοσίας (PSRR).

Οι μετατροπείς DC-to-DC

Ένα πρόβλημα με τη χρήση LDO για τη δημιουργία χαμηλών τάσεων από μια εισερχόμενη τροφοδοσία, είναι ότι καταναλώνουν ισχύ σε σχέση με το εύρος της πτώσης τάσης από την είσοδο στην έξοδο, ειδικά αν είναι σχετικά υψηλή, οπότε και τα LDO δεν είναι αποδοτικά. Σε αυτήν την περίπτωση, είναι καλύτερη η χρήση dc-to-dc μετατροπέων που μπορούν να επιτύχουν σημαντικά υψηλότερη απόδοση από έναν LDO. Το μειονέκτημα ενός μετατροπέα dc-to-dc είναι ότι πρόκειται για κύκλωμα εναλλαγής λειτουργίας και μπορεί να προκαλέσει μεταγωγή θορύβου στην τάση εξόδου του. Κυκλώματα εξαιρετικά ευαίσθητα στο θόρυβο δεν πρέπει να παρέχονται από dc-to-dc μετατροπείς. Μία λύση για να ξεπεραστεί αυτό είναι να εναλλάσσεται ένας μετατροπέας dc-to-dc για αποτελεσματική πτώση τάσης, ακολουθούμενος από έναν LDO για την απόρριψη του θορύβου τροφοδοσίας.

Υπάρχουν δύο τύποι τεχνολογιών dc-to-dc μετατροπέων. Οι τυπικοί dc-to-dc μετατροπείς βασισμένοι σε επαγωγείς(inductor-based), που απαιτούν εξωτερικό επαγωγέα και πυκνωτή για να δημιουργήσουν το dc επίπεδο εξόδου, και οι switched-

capacitor dc-to-dc μετατροπείς που υλοποιούνται με πυκνωτές on-die, αλλά δεν μπορούν να παρέχουν τόσο ρεύμα όσο οι λύσεις που βασίζονται σε επαγωγείς.

Επιπλέον, οι μετατροπείς dc-to-dc μπορούν να χρησιμοποιηθούν είτε σε λειτουργία buck, για χρήση όπου η τάση εξόδου είναι χαμηλότερη από την τάση εισόδου, ή σε λειτουργία ενίσχυσης(boost mode) όπου η τάση εξόδου είναι υψηλότερη από την τάση εισόδου. Υπάρχουν επίσης συνδυασμοί buck-boost mode dc-to-dc μετατροπέων, που παρέχουν αυτόματα τη σωστή τάση εξόδου ανεξάρτητα από την τάση εισόδου. Αυτό συναντάται ιδιαίτερα σε εφαρμογές με μπαταρία. Όταν η μπαταρία είναι συνδεδεμένη σε φορτιστή ή είναι πλήρως φορτισμένη, η τάση της μπαταρίας μπορεί να είναι υψηλότερη από την απαιτούμενη τάση του κυκλώματος φορτίου και στη συνέχεια καθώς η μπαταρία αποφορτίζεται, η τάση της μπαταρίας είναι χαμηλότερη από την απαιτούμενη τάση του κυκλώματος φορτίου. [40]

Αναφορές τάσεων (voltage references)

Οι αναφορές τάσης(voltage references), είναι απαραίτητο μέρος της μονάδας διαχείρισης ισχύος. Οι αναφορές τάσης χρησιμοποιούνται συνήθως ως αναφορά για διάφορους ενισχυτές op-amp, συγκριτές, μετατροπείς δεδομένων και πολλές άλλες αναλογικές και μικτές λειτουργίες σήματος. Τα επιθυμητά χαρακτηριστικά σε μια αναφορά τάσης περιλαμβάνουν, τη χαμηλή κατανάλωση ισχύος, την υψηλή ακρίβεια και την καλή απόρριψη του θορύβου τροφοδοσίας.

Επιπρόσθετες μονάδες

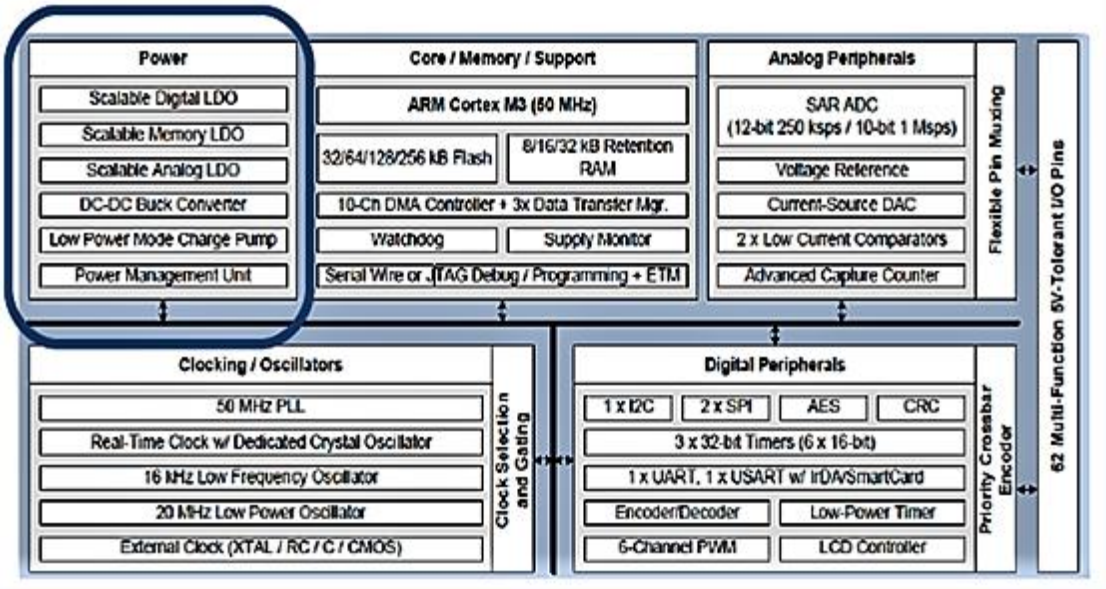
Υπάρχουν πολλά άλλα θεμελιώδη αναλογικά κυκλώματα που χρησιμοποιούνται στη μονάδα διαχείρισης ισχύος. Σε αυτά περιλαμβάνονται τα κυκλώματα ταλαντωτή, παρακολούθησης τάσης, αισθητήρα θερμοκρασίας και επαναφοράς ενεργοποίησης (power-on reset - POR). Υπάρχουν πολλοί τρόποι για την εφαρμογή ταλαντωτών, για λειτουργίες διαχείρισης ισχύος. Τα κυκλώματα παρακολούθησης τάσης και οι αισθητήρες θερμοκρασίας, χρησιμοποιούνται για λόγους ασφαλούς λειτουργίας, ενημερώνοντας τον ελεγκτή για να κλείσει το σύστημα όταν έχει ξεπεραστεί ένα όριο. Τα κυκλώματα Power-on reset (POR), είναι ένα απαραίτητα σε οποιοδήποτε σχεδιασμό, για να αναστείλει την εκκίνηση των λογικών κυκλωμάτων έως ότου οι τάσεις τροφοδοσίας φτάσουν σε επαρκές επίπεδο για να επιτύχουν έγκυρες λογικές καταστάσεις στο σύστημα. [40]

3.6.3 Μονάδα διαχείρισης ενέργειας μικροεπεξεργαστή

Κάποιες από τις βασικές λειτουργίες τις οποίες εκτελεί μία PMU είναι, η ενεργοποίηση ή απενεργοποίηση των αναλογικών συστημάτων τροφοδοσίας συμπεριλαμβανομένων των LDO regulators, η επαναφορά του συστήματος σε ενεργή κατάσταση μετά από ένα I/O ή/και control trigger και ο έλεγχος αν η τάση

τροφοδοσίας στην μνήμη RAM διατηρείται σε επίπεδα τέτοια ώστε να μην χαθούν τα δεδομένα όταν η συσκευή βρίσκεται σε sleep mode.

Στο Σχήμα 3.34, εμφανίζονται σε μπλοκ διάγραμμα οι απαραίτητες μονάδες που πρέπει να περιλαμβάνει ένας μικροεπεξεργαστής χαμηλής κατανάλωσης ισχύος. Οι DC-DC converters χρησιμοποιούνται για την μείωση της τάσης από την πηγή σε μία χαμηλότερη τάση που είναι καταλληλότερη για την τροφοδοσία των διάφορων στοιχείων. Ακόμη, μπορούν να απομονώσουν την τροφοδοσία εισόδου με τα επιμέρους κυκλώματα και να επιτύχουν αποδοτική μεταφορά της ισχύος σε αυτά. Οι low dropout (LDO) γραμμικοί ρυθμιστές τάσης παρέχουν μια συνεχώς ελεγχόμενη, σταθερή και χαμηλού θορύβου τάση εξόδου DC, βελτιώνοντας την αποδοτικότητά του μικροεπεξεργαστή. [38]

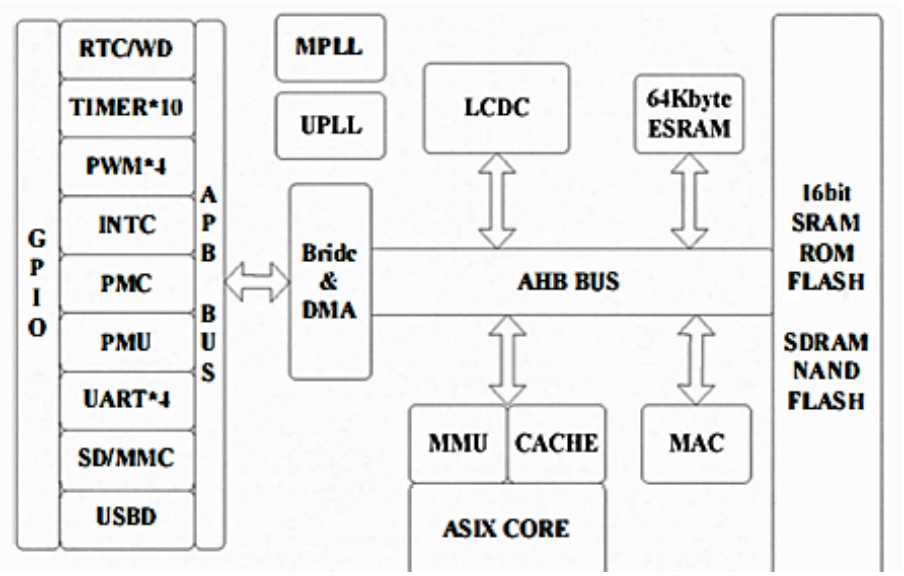


Σχήμα 3.34: Μπλοκ διάγραμμα μικροεπεξεργαστή και οι διάφορες μονάδες τροφοδοσίας.

3.6.4 Η μονάδα διαχείρισης ισχύος σε υλοποίηση low power SoC

Η δομή SoC

Η αρχιτεκτονική SoC με τη μονάδα διαχείρισης ισχύος (PMU) φαίνεται στο Σχήμα 3.35. [39] Το SoC, βασίζεται στη προηγμένη αρχιτεκτονική advanced microcontroller bus architecture (AMBA), με βασικά μέρη την ενσωματωμένη CPU RISC, ενσωματωμένη SRAM και τον ελεγκτή μνήμης.



Σχήμα 3.35: Η αρχιτεκτονική του SoC με μονάδα διαχείρισης ισχύος (PMU).

Πολλαπλοί τρόποι λειτουργίας

Τέσσερις τρόποι λειτουργίας παρέχονται στο SoC: κατάσταση αναστολής λειτουργίας(sleep mode), κατάσταση αδράνειας(idle mode), αργή(slow mode) και κανονική(normal mode). λειτουργία. Σύμφωνα με την ανάλυση της κατανάλωσης ενέργειας ενός συστήματος, η συχνότητα λειτουργίας ρολογιού είναι ένας σημαντικός παράγοντας για την αλλαγή της κατανάλωσης ισχύος.

Διαφορετικοί τρόποι λειτουργίας έχουν διαφορετική πηγή ρολογιού. Το ρολόι λειτουργίας έχει δύο κύριες πηγές: μία εξωτερική που προέρχεται από ταλαντωτή έξω από το σύστημα και μια άλλη εσωτερική που προέρχεται από την έξοδο μιας μονάδας PLL. Μέσω της μονάδας PLL, το ρολόι χαμηλής συχνότητας μπορεί να πολλαπλασιαστεί έως μία υψηλή συχνότητα, το οποίο στη συνέχεια μπορεί να διαιρεθεί για να παρέχει το απαιτούμενο ρολόι για τη καθεμία μονάδα. Όταν το σύστημα ξεκινά ή ολοκληρώνει την επαναφορά, θα λειτουργεί σε αργή συχνότητα και το εξωτερικό ρολόι, χρησιμοποιείται ως ρολόι λειτουργίας.

Εάν η CPU ολοκληρώσει όλες τις εργασίες, το σύστημα θα εισέλθει σε κατάσταση αδράνειας, είτε από αργή κατάσταση λειτουργίας είτε από κανονική. Σε κατάσταση αδράνειας του συστήματος, η CPU θα βρίσκεται σε κατάσταση αδράνειας και θα επιστρέψει στην προηγούμενη λειτουργία κάθε φορά που θα υπάρξει ένα σήμα επαναφοράς(reset) ή διακοπής(interrupt) στη μονάδα της CPU. Εάν το σύστημα βρίσκεται σε κατάσταση αδράνειας για μεγάλο χρονικό διάστημα, το σύστημα θα περιπέσει σε κατάσταση αναστολής λειτουργίας και οι μονάδες και η CPU δεν θα λειτουργούν οπότε δεν απαιτείται και η χρήση ρολογιού. [39]

Δομή της μονάδας διαχείρισης ισχύος

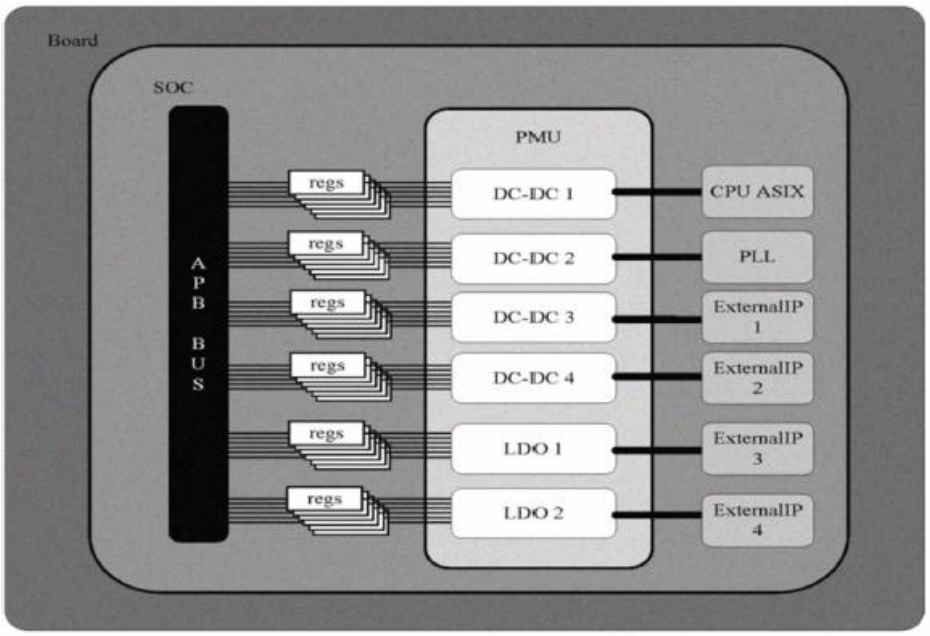
Λαμβάνοντας υπόψη ότι η συχνότητα ρολογιού και η τάση τροφοδοσίας είναι οι σημαντικοί παράγοντες που επηρεάζουν τη δυναμική κατανάλωση ισχύος, η μεταξύ τους σχέση φαίνεται στην παρακάτω εξίσωση.

$$P_{dyn} = a * C * V^2_{dd} * f_c$$

Όπου, a είναι η πιθανότητα μεταγωγής(switching), C είναι η χωρητικότητα φορτίου, V_{dd} είναι η τάση τροφοδοσίας και f_c είναι η συχνότητα ρολογιού. Ανάλογα με τις εφαρμογές, η CPU ή άλλες μονάδες του SoC, ενδέχεται να μην χρειάζεται να λειτουργούν σε υψηλή συχνότητα ρολογιού. Σε τέτοιες συνθήκες, προκειμένου να μειώσουμε την κατανάλωση ενέργειας, μπορούμε να παρέχουμε χαμηλότερη τάση όταν το SoC λειτουργεί σε χαμηλή συχνότητα ρολογιού.

Η διαχείριση ισχύος μπορεί να παρέχει σύνολα διαφορετικών τάσεων που αντιστοιχούν σε διαφορετική συχνότητα ρολογιού. Αποτελείται από έξι τμήματα, που περιλαμβάνουν τις μονάδες DC-DC converters και LDO, τα οποία φαίνονται στο Σχήμα 3.36. Υπάρχουν τέσσερα είδη μονάδων DC-DC σχεδιασμένα για διαφορετικές εφαρμογές. Η πρώτη μονάδα DC-DC είναι η παροχή τροφοδοσίας στην CPU του συστήματος και η παρεχόμενη τάση μπορεί να ρυθμιστεί step by step. Η δεύτερη μονάδα DC-DC είναι ειδικά σχεδιασμένη για παροχή τάσης στη μονάδα PLL. Η τρίτη και η τέταρτη μονάδα DC-DC είναι η παροχή τροφοδοσίας για μονάδες περιφερειακών. Ταυτόχρονα, υπάρχουν δύο LDO ρυθμιστές, οι οποίοι μπορούν να παρέχουν διαμορφωμένες τάσεις για μονάδες περιφερειακών ή για τη μονάδα PLL.

Ανάλογα με την εφαρμογή, το σύστημα μπορεί να λειτουργήσει με κατάλληλη συχνότητα ρολογιού και τάση τροφοδοσίας μέσω της διαχείρισης από την PMU, μέσω ρυθμίσεων από τους καταχωρητές(registers) ή από το λογισμικό του συστήματος. [39]



Σχήμα 3.36: Η αρχιτεκτονική της PMU.

Συμπεράσματα

Από τον Πίνακα 3.1, φαίνεται ότι μέσω του σχεδιασμού πολλαπλών τρόπων λειτουργίας και διαφορετικών συχνοτήτων ρολογιού, η κατανάλωση ενέργειας έχει μειωθεί, ειδικά στη κατάσταση αναστολής λειτουργίας και αργής λειτουργίας. Λόγω της μονάδας διαχείρισης ισχύος και ανάλογα με την εφαρμογή, η CPU και άλλες λειτουργικές μονάδες μπορούν να ολοκληρώσουν εργασίες με διαφορετική τροφοδοσία και ρεύμα με αποτέλεσμα η κατανάλωση ενέργειας έχει μειωθεί περαιτέρω.

Πίνακας 3.1. Κατανάλωση ισχύος της υλοποίησης SoC.

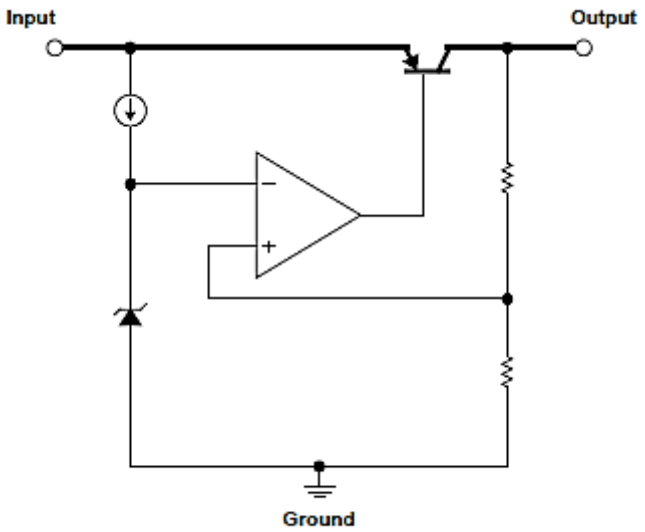
CPU mode	F (MHz)	core voltage (V)	core current (mA)	proposed power (mW)	power [7] (mW)
sleep	X	X	0.18	7.25	19.42
slow	4	1.19	2.27	16.95	45.36
normal	32	1.24	11.12	35.22	75.04
	48	1.26	15.24	42.54	97.26
	88	1.36	35.38	96.24	134.33
idle	32	1.24	6.12	23.42	93.60

3.7 Γραμμικοί ρυθμιστές τάσης

Τι είναι ο γραμμικός ρυθμιστής τάσης

Οι γραμμικοί ρυθμιστές τάσης, παράγουν μια καθαρή, ακριβή τάση εξόδου που χρησιμοποιείται από ευαίσθητα εξαρτήματα. Υπάρχουν δύο τύποι γραμμικών ρυθμιστών: οι standard γραμμικοί ρυθμιστές και οι γραμμικοί ρυθμιστές low dropout (LDOs). Η διαφορά μεταξύ των δύο, είναι στο στοιχείο διέλευσης(pass element) και στο εύρος της dropout τάσης που απαιτείται για τη διατήρηση μιας σταθεροποιημένης τάσης εξόδου. Η dropout τάση είναι η ελάχιστη τάση που απαιτείται στα άκρα του ρυθμιστή για τη διατήρηση της σωστής λειτουργίας του. Οι χαμηλές dropout τάσεις εγγυώνται ότι ένας μικροεπεξεργαστής θα λαμβάνει μια καθαρή, καλά ρυθμιζόμενη τροφοδοσία που αντιδρά γρήγορα στις αλλαγές φορτίου που προκαλούνται από τον επεξεργαστή καθώς και στις αλλαγές της τροφοδοσίας εισόδου. Ο LDO ρυθμιστής είναι ένας εύχρηστος, χαμηλού κόστους και υψηλής απόδοσης μέσο τροφοδοσίας συστημάτων. [36]

Ένα τυπικό διάγραμμα γραμμικού ρυθμιστή φαίνεται στο Σχήμα 3.37, όπου ένα τρανζίστορ διέλευσης ελέγχεται από έναν τελεστικό ενισχυτή ο οποίος συγκρίνει την τάση εξόδου με μια τάση αναφοράς. Καθώς η τάση εξόδου μειώνεται, το op-amp αυξάνει την οδήγηση στο στοιχείο διέλευσης, γεγονός που αυξάνει την τάση εξόδου. Αντίστροφα, εάν η έξοδος αυξηθεί πάνω από το επιθυμητό σημείο ρύθμισης, ο ενισχυτής μειώνει την οδήγηση στο στοιχείο διέλευσης. Αυτές οι διορθώσεις εκτελούνται συνεχώς με τον χρόνο αντίδρασης να περιορίζεται μόνο από την ταχύτητα του βρόχου τελεστικού και τρανζίστορ εξόδου .



Σχήμα 3.37: Τυπικό διάγραμμα γραμμικού σταθεροποιητή.

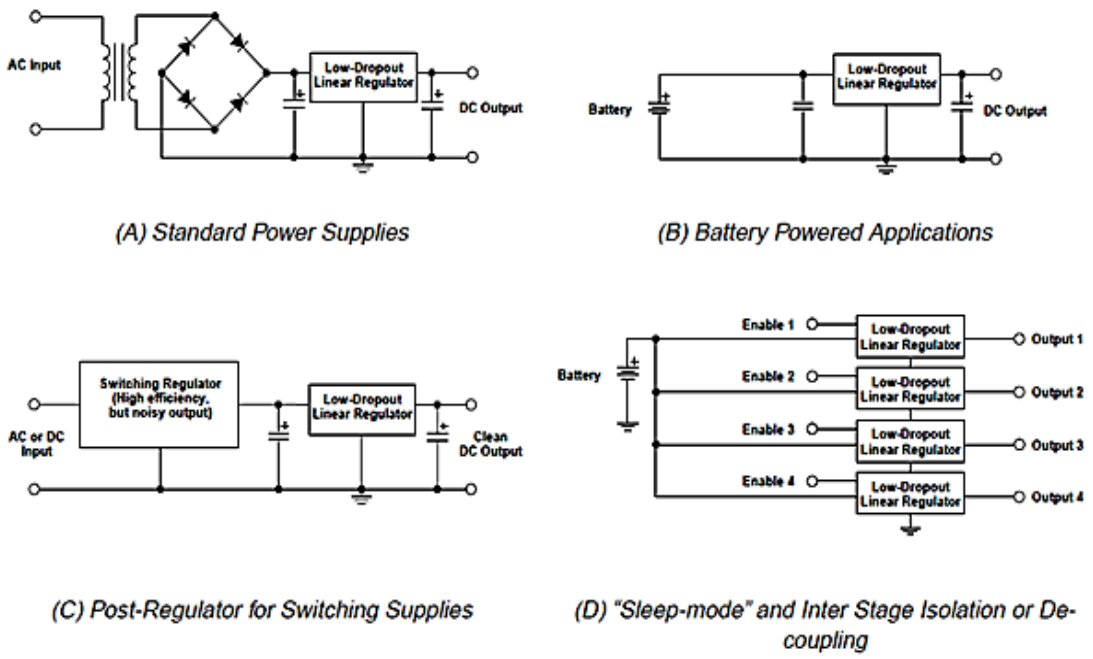
Οι πραγματικοί γραμμικοί ρυθμιστές έχουν μια σειρά από άλλα χαρακτηριστικά, συμπεριλαμβανομένης της προστασίας από βραχυκυκλωμένα φορτία και λόγω υπερθέρμανσης. [37]

Οι λόγοι χρήσης των ρυθμιστών τάσης

Η πιο βασική τους λειτουργία, η ρύθμιση τάσης, παρέχει καθαρή, σταθερή, ακριβή τάση σε ένα κύκλωμα. Οι ρυθμιστές τάσης είναι ένα θεμελιώδες κύκλωμα στα τροφοδοτικά όλων των ηλεκτρονικών συσκευών. Τα βασικά οφέλη και οι εφαρμογές του ρυθμιστή περιλαμβάνουν:

- Ακριβής τάση τροφοδοσίας
- Ενεργό φιλτράρισμα θορύβου
- Προστασία από σφάλματα υπερέντασης
- Απομόνωση μεταξύ σταδίων (decoupling)
- Δημιουργία πολλαπλών τάσεων εξόδου από μία μόνο πηγή
- Χρήση σε σταθερές πηγές ρεύματος

Το Σχήμα 3.38 δείχνει διάφορες τυπικές εφαρμογές για γραμμικούς ρυθμιστές τάσης. Ένα παραδοσιακό τροφοδοτικό AC σε DC εμφανίζεται στο Σχήμα 3.38(A). Εδώ, ο γραμμικός ρυθμιστής εκτελεί απόρριψη κυματισμού, εξάλειψη του AC hum και ρύθμιση τάσης εξόδου. Η τάση εξόδου θα είναι καθαρή και σταθερή, ανεξάρτητα από τις μεταβολές της AC τάσης. Το Σχήμα 3.38(B) χρησιμοποιεί έναν low dropout γραμμικό ρυθμιστή για να παρέχει μια σταθερή τάση εξόδου από μια μπαταρία, καθώς η μπαταρία αποφορτίζεται. Οι ρυθμιστές low dropout είναι εξαιρετικοί για αυτήν την εφαρμογή, δεδομένου ότι επιτρέπουν μεγαλύτερη διάρκεια ζωής για μια δεδομένη μπαταρία. Το Σχήμα 3.38(C) δείχνει έναν γραμμικό ρυθμιστή διαμορφωμένο ως «post regulator» για ένα switching τροφοδοτικό. Τα switching τροφοδοτικά είναι γνωστά για την εξαιρετική τους απόδοση, αλλά η τάση εξόδου παρουσιάζει κυμάτωση η οποία υποβαθμίζει τη ρύθμιση και την απόδοση, ειδικά όταν τροφοδοτούνται αναλογικά κυκλώματα. Ένας γραμμικός ρυθμιστής μετά τον switching ρυθμιστή παρέχει ενεργό φιλτράρισμα και βελτιώνει σημαντικά την ακρίβεια της εξόδου. Όπως δείχνει το Σχήμα 3.38(D), ορισμένοι γραμμικοί ρυθμιστές εξυπηρετούν διπλή λειτουργία τόσο ως ρυθμιστής όσο και ως ελεγκτής ON / OFF της ισχύος. Σε ορισμένες εφαρμογές, διαφορετικά μπλοκ συστήματος τροφοδοτούνται συχνά από διαφορετικούς ρυθμιστές - ακόμη και αν χρησιμοποιούν την ίδια τάση τροφοδοσίας - λόγω της απομόνωσης (decoupling) που παρέχει ο ρυθμιστής. [37]



Σχήμα 3.38: Τυπικές εφαρμογές για γραμμικούς ρυθμιστές τάσης.

Βασικές παραμέτροι των ρυθμιστών τάσης

Ας δούμε τις πιο σημαντικές παραμέτρους των ρυθμιστών τάσης:

· **Η τάση εξόδου(Output voltage)** είναι μια από τις πιο σημαντικές παραμέτρους. Οι γραμμικοί ρυθμιστές διατίθενται τόσο με σταθερή τάση εξόδου όσο και με ρυθμιζόμενη. Οι ρυθμιστές σταθερής τάσης προσφέρουν ευκολία στη χρήση, με τις τάσεις εξόδου τους εργοστασιακά ρυθμισμένες - αλλά μόνο εάν η εφαρμογή χρησιμοποιεί τη διαθέσιμη τάση. Τα ρυθμιζόμενα επιτρέπουν τη χρήση τάσης προσαρμοσμένης στο εκάστοτε κύκλωμα.

· **Το μέγιστο ρεύμα εξόδου(Maximum output current)** είναι η παράμετρος που χρησιμοποιείται γενικά για την κατηγοριοποίηση των ρυθμιστών. Τα μεγαλύτερα μέγιστα ρεύματα εξόδου απαιτούν μεγαλύτερους και ακριβότερους ρυθμιστές.

· **Η Dropout voltage** είναι η επόμενη σημαντική παράμετρος. Αυτή είναι η ελάχιστη πρόσθετη τάση στην είσοδο που απαιτείται για να παράγεται ρυθμιζόμενη έξοδος. Όσο χαμηλότερη είναι η dropout τάση, τόσο το καλύτερο.

· **Το ρεύμα γείωσης(Ground current)** είναι το ρεύμα τροφοδοσίας που χρησιμοποιείται από τον ρυθμιστή και που δεν περνά στο φορτίο. Ένας ιδανικός ρυθμιστής θα ελαχιστοποιήσει το ρεύμα γείωσης. Αυτή η παράμετρος καλείται μερικές φορές και ρεύμα ηρεμίας (quiescent current) , αλλά αυτός ο ορισμός είναι λανθασμένος για ρυθμιστές στοιχείων PNP-pass.

Γραμμικοί ρυθμιστές vs switching ρυθμιστών

Οι γραμμικοί ρυθμιστές είναι λιγότερο ενεργειακά αποδοτικοί από τους switching ρυθμιστές εναλλαγής, αλλά ανάλογα με την εφαρμογή, οι γραμμικοί ρυθμιστές προσφέρουν πολλά θετικά χαρακτηριστικά όπως:

• Ο χαμηλότερος θόρυβος εξόδου είναι σημαντικός για εξοπλισμό επικοινωνιών

• ταχύτερη απόκριση στις μεταβάσεις εισόδου και εξόδου

• ευκολότεροι στη χρήση, επειδή απαιτούν μόνο πυκνωτές φίλτρου για τη λειτουργία τους

• γενικά μικρότεροι σε μέγεθος

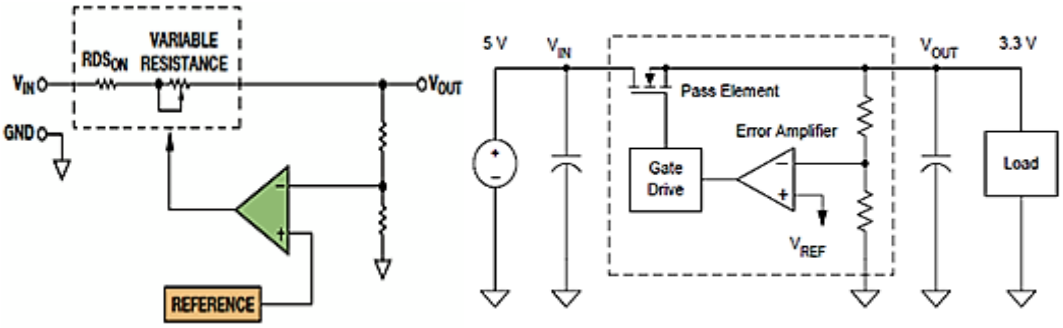
• λιγότερο ακριβοί (απλούστερο εσωτερικό κύκλωμα)

Επιπλέον, σε εφαρμογές που χρησιμοποιούν χαμηλές διαφορές τάσης εισόδου-εξόδου, η απόδοση δεν είναι τόσο κακή. Για παράδειγμα, σε εφαρμογή μικροεπεξεργαστή 5V έως 3.3V, η απόδοση γραμμικού ρυθμιστή πλησιάζει το 66%. [37]

3.7.1 Ο Low Dropout Γραμμικός Ρυθμιστής

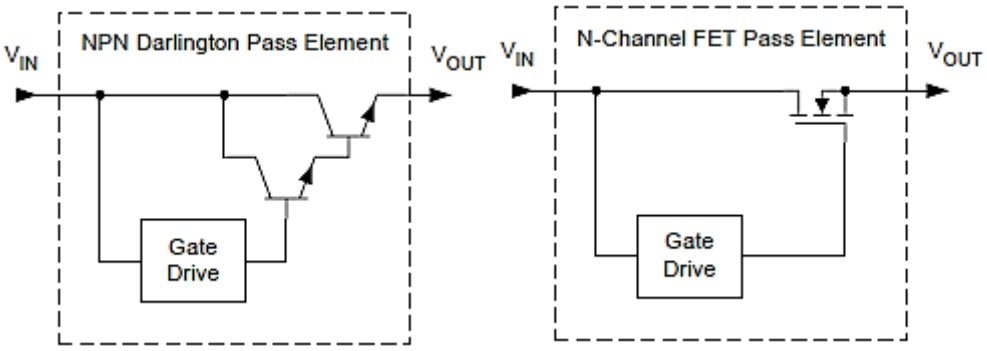
Οι ρυθμιστές low dropout (LDOs) είναι ένας απλός, φθηνός τρόπος ρύθμισης της τάσης εξόδου που τροφοδοτείται από είσοδο υψηλότερης τάσης. Ένας ρυθμιστής low dropout είναι μια κατηγορία γραμμικού ρυθμιστή που έχει σχεδιαστεί για να ελαχιστοποιεί τον κορεσμό του τρανζίστορ διέλευσης εξόδου και τις απαιτήσεις οδήγησής του.

Στο σχήμα Σχήμα 3.39, φαίνεται η βασική μορφή ενός LDO(δεξιά) και το απλοποιημένο διάγραμμα ενός LDO ρυθμιστή(αριστερά). Η τάση εισόδου εφαρμόζεται σε ένα στοιχείο διελεύσεως, το οποίο είναι συνήθως ένα N-channel ή P-channel FET, αλλά μπορεί επίσης να είναι NPN ή PNP τρανζίστορ. Το στοιχείο διέλευσης λειτουργεί στη γραμμική περιοχή για να μειώσει την τάση εισόδου στην επιθυμητή τάση εξόδου. Η προκύπτουσα τάση εξόδου ανιχνεύεται από τον ενισχυτή σφάλματος και συγκρίνεται με μια τάση αναφοράς. Ο ενισχυτής σφάλματος οδηγεί την πύλη του στοιχείου διέλευσης στο κατάλληλο σημείο λειτουργίας για να διασφαλιστεί ότι η τάση εξόδου θα είναι στη σωστή τιμή. Καθώς το ρεύμα λειτουργίας ή η τάση εισόδου μεταβάλλονται, ο ενισχυτής σφάλματος ρυθμίζει το στοιχείο διέλευσης ώστε να διατηρηθεί μια σταθερή τάση εξόδου. Σε σταθερή κατάσταση λειτουργίας, ένα LDO συμπεριφέρεται σαν μία απλή αντίσταση.



Σχήμα 3.39: Η απλοποιημένη(αριστερά) και η βασική μορφή ενός LDO ρυθμιστή(δεξιά).

Για τυπικούς ρυθμιστές, το στοιχείο διέλευσης είναι είτε ένα στάδιο εξόδου Darlington NPN ή PNP. Το Σχήμα 3.40(α) δείχνει ότι ένα τρανζίστορ Darlington έχει μία υψηλή πτώση τάσης μεταξύ συλλέκτη-εκπομπού επειδή το κύκλωμα οδήγησης συναντά δύο πτώσεις τάσης βάσης-εκπομπού πριν φτάσει στην έξοδο. Οι κλασσικοί γραμμικοί ρυθμιστές έχουν πτώση τάσης έως και 2 V που είναι αποδεκτή για εφαρμογές με μεγάλη διαφορά τάσης εισόδου-εξόδου. Οι περισσότεροι LDO χρησιμοποιούν ένα N-channel ή P-channel FET σαν στοιχείο διέλευσης και έτσι μπορούν να έχουν dropout τάσεις μικρότερες από 100 mV. Στο Σχήμα 3.40(β), φαίνεται ότι η dropout τάση ενός N-channel FET LDO, εξαρτάται μόνο από το ελάχιστη πτώση τάσης πάνω στο FET. [36]



(α) (β)

Σχήμα 3.40: Στοιχεία διέλευσης γραμμικών ρυθμιστών.

Το Σχήμα 3.41 δείχνει τους πέντε βασικούς τύπους γραμμικών ρυθμιστών:

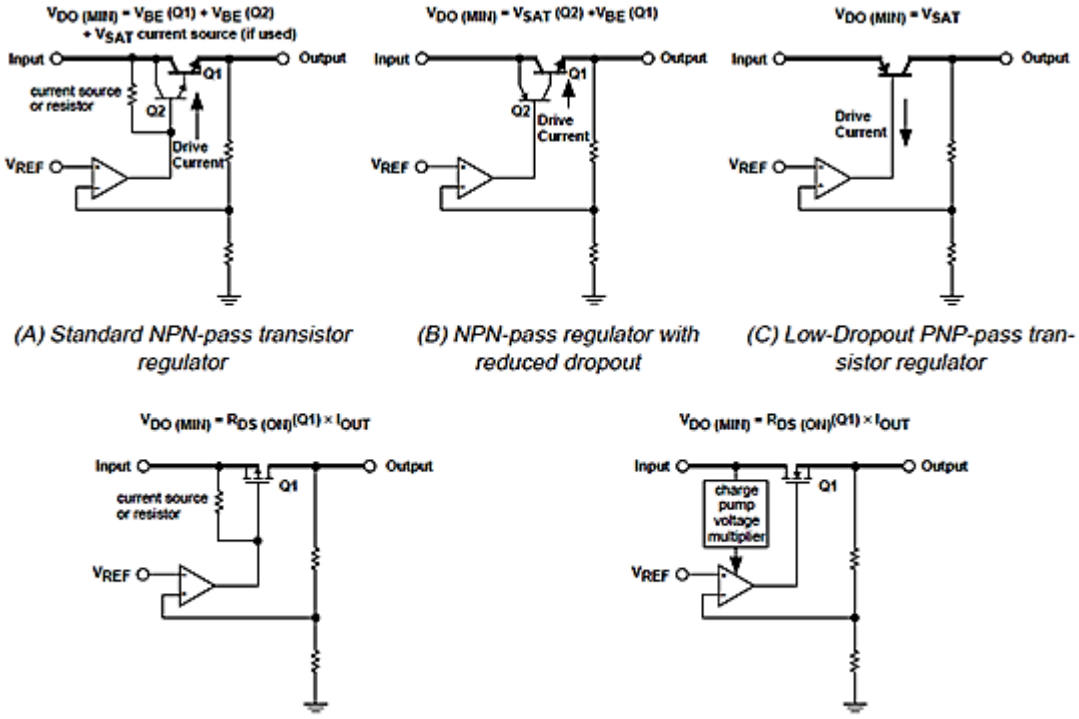
A. Κλασικοί ρυθμιστές που βασίζονται σε NPN και απαιτούν 2.5 έως 3V περιθώριο τάσης εισόδου για να λειτουργήσουν.

B. Ρυθμιστές "Low Dropout NPN", με έξοδο NPN αλλά κύκλωμα οδήγησης βάσης PNP. Αυτές οι συσκευές μειώνουν την απαίτηση για dropout σε 1,2 έως 1,5V.

C. Low dropout PNP-based ρυθμιστές που χρειάζονται επιπλέον 0,3V έως 0,6V για τη λειτουργία τους.

D. P-channel CMOS output ρυθμιστές. Αυτές οι συσκευές έχουν πολύ χαμηλές dropout τάσεις σε χαμηλά ρεύματα, αλλά απαιτούν μεγάλη περιοχή(die area) κατά τον σχεδιασμό τους και έχουν υψηλές απαιτήσεις εσωτερικού ρεύματος οδήγησης όταν εργάζονται με εισόδους με θόρυβο ή με ευρέως μεταβαλλόμενα ρεύματα εξόδου.

E. Regulator controllers. Αυτά είναι ολοκληρωμένα κυκλώματα που παρέχουν τις λειτουργίες αναφοράς και ελέγχου ενός γραμμικού ρυθμιστή, αλλά δεν διαθέτουν το στοιχείο διέλευσης. Παρέχουν το πλεονέκτημα της βελτιστοποίησης του εμβαδού υλοποίησης και του κόστους για εφαρμογές μεγάλων ρευμάτων, αλλά υφίστανται το μειονέκτημα ότι είναι μια multi-package λύση. [37]



(A) Standard NPN-pass transistor regulator

(B) NPN-pass regulator with reduced dropout

(C) Low-Dropout PNP-pass transistor regulator

(D) P-Channel MOSFET-pass transistor regulator

(E) N-Channel MOSFET-pass transistor regulator

Σχήμα 3.41: Βασικοί τύποι γραμμικών ρυθμιστών.

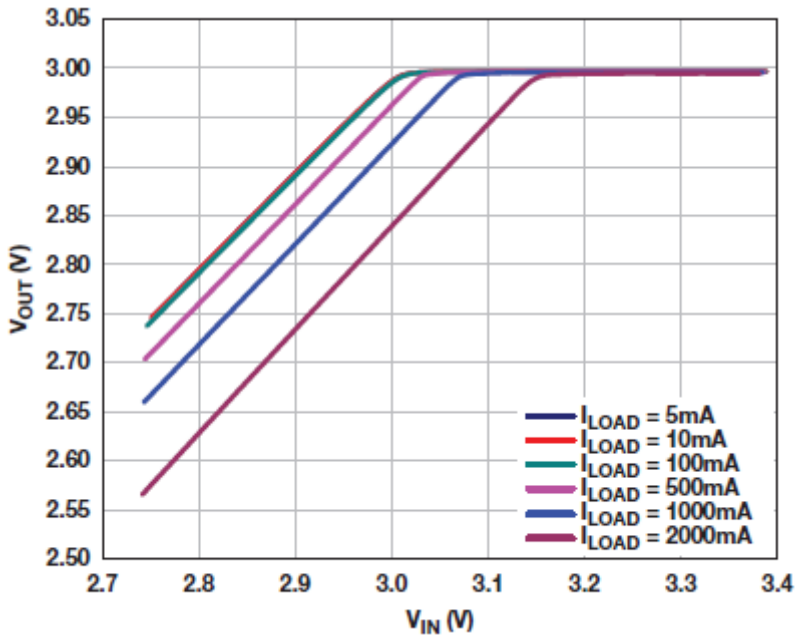
3.7.2 Έννοιες των Low-Dropout Regulators (LDO)

Εδώ παρουσιάζονται ορισμένοι κοινοί όροι που χρησιμοποιούνται με τους LDO, που εξηγούν θεμελιώδεις έννοιες όπως τάση dropout, τάση περιθωρίου(headroom), το ρεύμα ηρεμίας(quiescent current), το ρεύμα γείωσης(ground current), το ρεύμα τερματισμού λειτουργίας(shutdown current), η απόδοση(efficiency), η dc line-and-load ρύθμιση, η transient line-and-load απόκριση, ο λόγος power-supply rejection ratio (PSRR), ο θόρυβος εξόδου(output noise) και η ακρίβεια(accuracy).

Τάση dropout

Η τάση dropout ($V_{DROPOUT}$), είναι η διαφορά τάσης εισόδου-εξόδου στην οποία το LDO δεν είναι πλέον σε θέση να ρυθμίσει, έναντι περαιτέρω μειώσεων της τάσης εισόδου(Σχήμα 3.42). Στην dropout περιοχή, το στοιχείο διέλευσης ενεργεί σαν αντίσταση με τιμή ίση με την αντίσταση drain-to-source (RDSON). Η τάση dropout, εκφρασμένη με τους όρους RDS_{ON} και το ρεύμα φορτίου, είναι:

$$V_{DROPOUT} = I_{LOAD} \times RDS_{ON}.$$



Σχήμα 3.42: Η Dropout περιοχή ενός 3.0V LDO.

Τάση περιθωρίου – Headroom voltage

Η τάση του περιθωρίου είναι η διαφορά τάσης εισόδου προς έξοδο που απαιτείται για ένα LDO να πληροί τις προδιαγραφές του. Το φύλλο δεδομένων(datasheet), δείχνει συνήθως την τάση περιθωρίου ως την κατάσταση στην οποία καθορίζονται οι άλλες παράμετροι. Η τάση του περιθωρίου είναι συνήθως περίπου 400 mV έως 500 mV, αλλά ορισμένα LDO απαιτούν έως και 1,5V. Η τάση του περιθωρίου δεν πρέπει να συγχέεται με την τάση dropout, καθώς είναι η ίδια μόνο όταν το LDO βρίσκεται σε διακοπή. [35]

Ρεύματα ηρεμίας και γείωσης - Quiescent and Ground Current

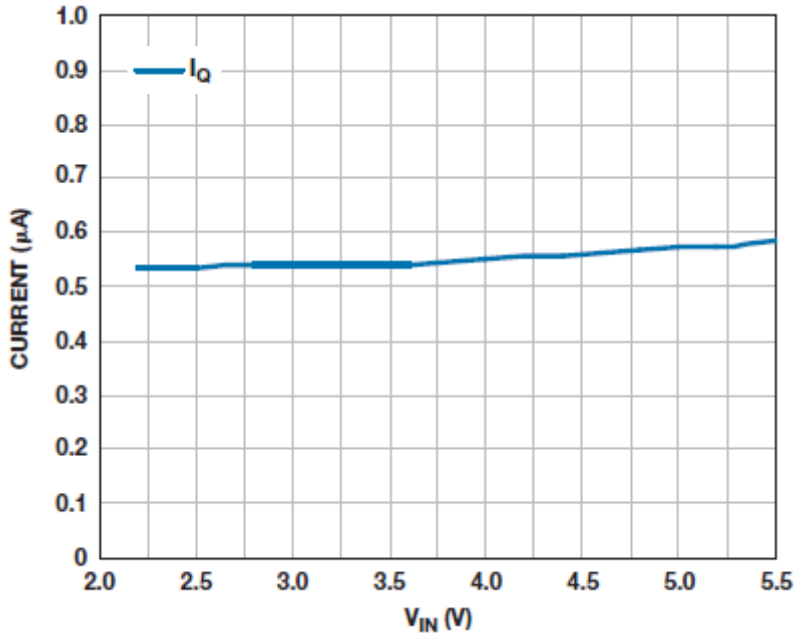
Το ρεύμα ηρεμίας - Quiescent current (I_Q) (Σχήμα 3.43) είναι το απαιτούμενο ρεύμα για την τροφοδοσία του εσωτερικού κυκλώματος του LDO όταν το εξωτερικό ρεύμα φορτίου είναι μηδέν. Περιλαμβάνει τα ρεύματα λειτουργίας του

ενισχυτή σφάλματος, του διαιρέτη της τάσης εξόδου και των κυκλωμάτων ανίχνευσης υπερέντασης και υπερθέρμανσης. Η ποσότητα του ρεύματος ηρεμίας καθορίζεται από την τοπολογία, την τάση εισόδου και τη θερμοκρασία.

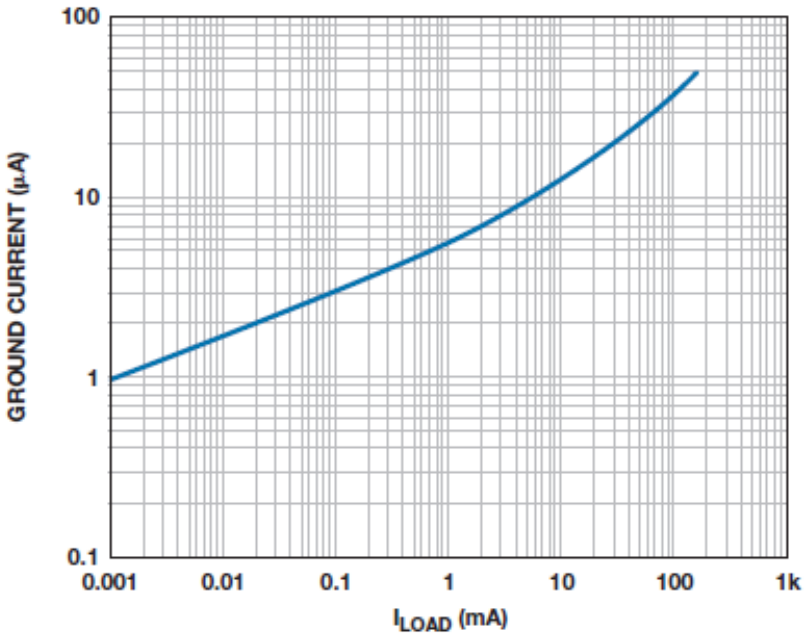
$I_Q = I_{IN}$ @ χωρίς φορτίο

Το ρεύμα γείωσης (I_{GND}) (Σχήμα 3.44) είναι η διαφορά μεταξύ των ρευμάτων εισόδου και εξόδου και περιλαμβάνει απαραιτήτως το ρεύμα ηρεμίας. Ένα χαμηλό ρεύμα γείωσης μεγιστοποιεί την απόδοση του LDO.

$I_{GND} = I_{IN} - I_{OUT}$



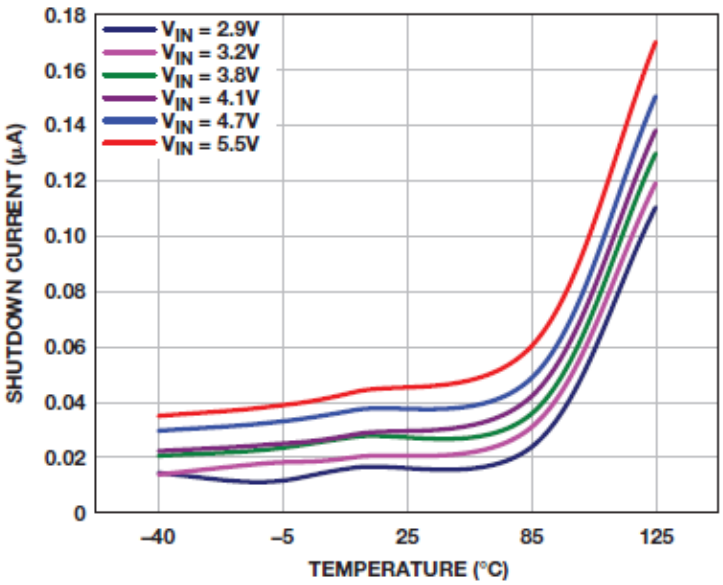
Σχήμα 3.43: Το ρεύμα ηρεμίας σε σχέση με τη τάση εισόδου του LDO.



Σχήμα 3.44: Το ρεύμα γείωσης σε σχέση με το ρεύμα φορτίου του LDO.

Ρεύμα απενεργοποίησης - Shutdown Current

Το ρεύμα απενεργοποίησης, είναι το ρεύμα εισόδου που καταναλώνει το LDO όταν η έξοδος είναι απενεργοποιημένη. Τα υψηλότερα ρεύματα διαρροής προκαλούν αύξηση του ρεύματος απενεργοποίησης με τη θερμοκρασία, όπως φαίνεται στο Σχήμα 3.45.



Σχήμα 3.45: Το ρεύμα απενεργοποίησης σε σχέση με τη θερμοκρασία του LDO.

Αποδοτικότητα - Efficiency

Η απόδοση ενός LDO καθορίζεται από το ρεύμα γείωσης και τις τάσεις εισόδου/εξόδου:

$$\text{Efficiency} = I_{OUT} / (I_{OUT} + I_{GND}) \times V_{OUT} / V_{IN} \times 100\%$$

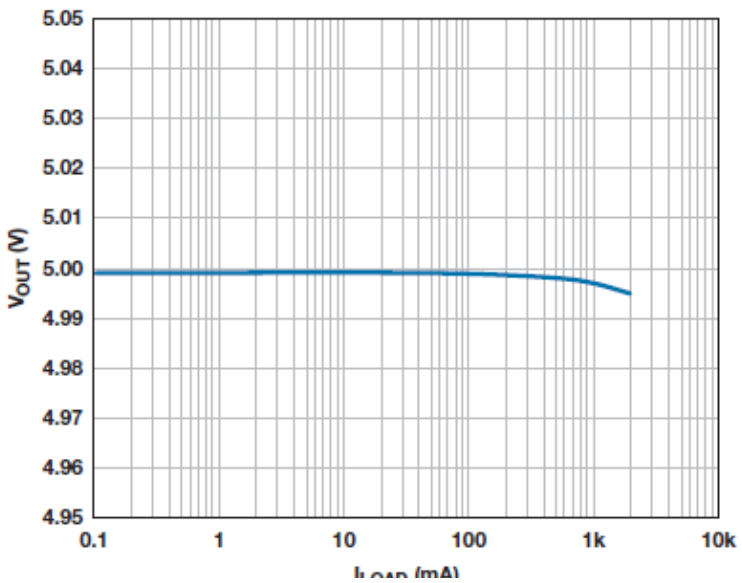
Για υψηλή απόδοση, πρέπει να ελαχιστοποιηθεί η τάση του περιθωρίου και το ρεύμα γείωσης. Επιπλέον, η διαφορά τάσης μεταξύ εισόδου και εξόδου πρέπει να ελαχιστοποιηθεί. Η διαφορά τάσης εισόδου-εξόδου είναι εγγενής παράγοντας για τον προσδιορισμό της απόδοσης, ανεξάρτητα από τις συνθήκες φορτίου. Για παράδειγμα, η απόδοση ενός LDO 3,3V δεν θα υπερβεί ποτέ το 66% όταν τροφοδοτείται από 5V, αλλά θα αυξηθεί στο μέγιστο 91,7% όταν η τάση εισόδου πέσει στα 3,6V. Η κατανάλωση ισχύος ενός LDO είναι:

$$(V_{IN} - V_{OUT}) \times I_{OUT}$$

Ρύθμιση φορτίου DC - DC - Load Regulation

Η ρύθμιση φορτίου είναι ένα μέτρο της ικανότητας του LDO να διατηρεί την καθορισμένη τάση εξόδου υπό διαφορετικές συνθήκες φορτίου. Η ρύθμιση φορτίου, που φαίνεται στο Σχήμα 3.46, ορίζεται ως

$$\text{Load Regulation} = \Delta V_{OUT} / \Delta I_{OUT}$$



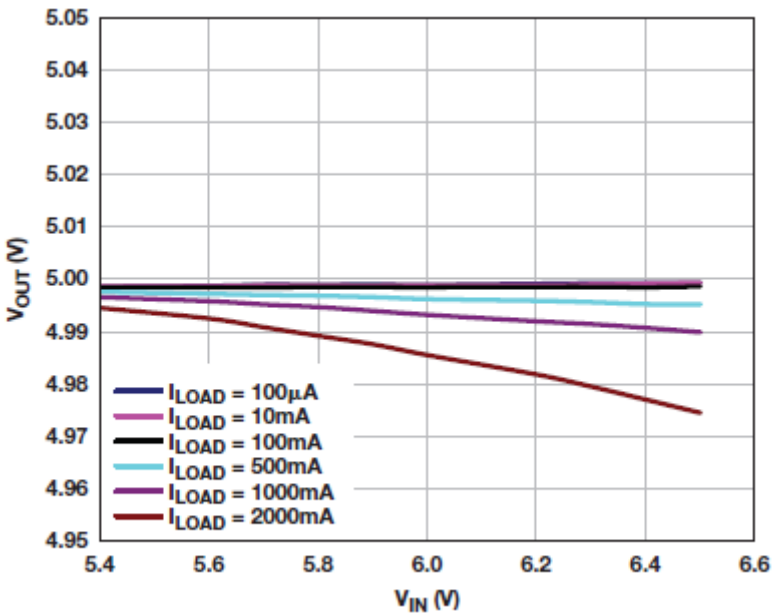
Σχήμα 3.46: Η τάση εξόδου σε σχέση με το ρεύμα φορτίου του LDO.

Ρύθμιση γραμμής τροφοδοσίας - DC - Line Regulation

Η ρύθμιση γραμμής τροφοδοσίας είναι ένα μέτρο της ικανότητας του LDO να διατηρεί την καθορισμένη τάση εξόδου με ποικίλη τάση εισόδου. Ο κανονισμός γραμμής ορίζεται ως

Line Regulation = $\Delta V_{OUT} / \Delta V_{IN}$

Το Σχήμα 3.47 δείχνει την τάση εξόδου του LDO, έναντι της τάσης εισόδου σε διαφορετικά ρεύματα φορτίου. Ο κανονισμός γραμμής επιδεινώνεται καθώς αυξάνεται το ρεύμα φόρτωσης επειδή μειώνεται το συνολικό κέρδος βρόχου του LDO.



Σχήμα 3.47: Η τάση εξόδου σε σχέση με τη τάση εισόδου του LDO.

DC ακρίβεια – Accuracy

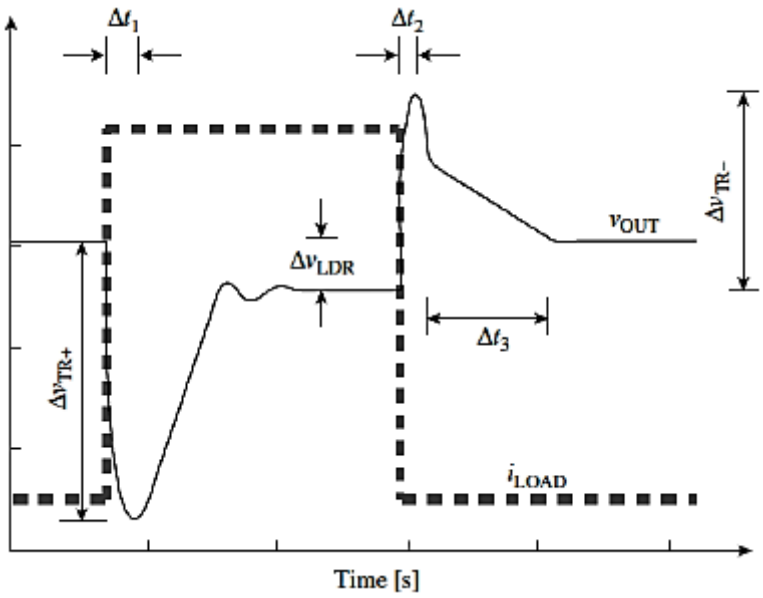
Η συνολική ακρίβεια, λαμβάνει υπόψη τα αποτελέσματα της ρύθμισης γραμμής και φορτίου, της απόκλισης της τάσης αναφοράς και της απόκλισης της τάσης του ενισχυτή σφάλματος. Η διακύμανση της τάσης εξόδου σε μια ρυθμιζόμενη παροχή τροφοδοσίας, οφείλεται κυρίως στη διακύμανση της θερμοκρασίας της τάσης αναφοράς και του ενισχυτή σφάλματος. Εάν χρησιμοποιούνται διακριτές αντιστάσεις για τη ρύθμιση της τάσης εξόδου, η ανοχή των αντιστάσεων μπορεί κάλλιστα να είναι ο μεγαλύτερος συντελεστής στη συνολική ακρίβεια. Η ρύθμιση γραμμής και φορτίου και οι αντισταθμίσεις(offsets) του ενισχυτή σφάλματος, αντιστοιχούν κανονικά στο 1% έως 3% της συνολικής ακρίβειας. [35]

Παροδικές Διακυμάνσεις - Transient Variations

Η χειρότερη περίπτωση διακυμάνσεων στην έξοδο του σταθεροποιητή είναι αυτή που οφείλεται στην μετάβαση του ρεύματος φορτίου από την ελάχιστη του τιμή στην μέγιστη και το αντίστροφο (Σχήμα 3.48). Έστω λοιπόν ότι το ρεύμα μεταβαίνει στην μέγιστή του τιμή. Τότε η πρώτη αντίδραση προέρχεται από τον πυκνωτή εξόδου. Αυτός για το χρονικό διάστημα Δt1 εκφορτίζεται, ρίχνοντας απότομα το δυναμικό της εξόδου (undershoot), παρέχοντας ρεύμα στο φορτίο . Με το πέρας του χρονικού διαστήματος αρχίζει να αντιδράει ο βρόχος του σταθεροποιητή ρυθμίζοντας κατάλληλα την διαγωγιμότητα του στοιχείου διέλευσης καλύπτοντας κατά αυτόν τον τρόπο τις ανάγκες του φορτίου για ρεύμα.

Στην πραγματικότητα το ρεύμα μέσω του στοιχείου διέλευσης είναι λίγο παραπάνω από αυτό που επιτάσσει τα φορτίο αφού ένα μέρος του φορτίζει τον πυκνωτή εξόδου.

Η τιμή της εξόδου για το μέγιστο ρεύμα είναι όση η αναμενόμενη τιμή της ελαττωμένη κατά την ποσότητα της ρύθμισης του φορτίου (Load Regulation). Το επόμενο φαινόμενο που λαμβάνει χώρα είναι η μετάβαση στην ελάχιστη τιμή του ρεύματος φορτίου. Και πάλι για ένα χρονικό διάστημα ο βρόχος εμφανίζει μία αδράνεια, το στοιχείο διέλευσης δεν μπορεί να σταματήσει την αγωγή ρεύματος ακαριαία. Έτσι το περίσσιο ρεύμα φορτίζει επιπλέον τον πυκνωτή εξόδου ανεβάζοντας την τάση εξόδου πιο ψηλά από τη επιθυμητή τιμή (overshoot). Μετά από Δt2 δευτερόλεπτα το στοιχείο διέλευσης κλείνει και ο πυκνωτής «αδειάζει» το επιπλέον φορτίο του μέσω του δικτυώματος της ανάδρασης. Αυτή η εκφόρτιση διαρκεί Δt3 δευτερόλεπτα και με το πέρας τους η έξοδος επιστρέφει στην αναμενόμενη τιμή της.



Σχήμα 3.48: Αντίδραση της τάσης εξόδου σε εναλλαγές ρεύματος μεταξύ μέγιστης και ελάχιστης τιμής και το αντίθετο.

Απόρριψη τροφοδοσίας - Power Supply Rejection

Με απλά λόγια, το PSRR είναι ένα μέτρο του πόσο καλά ένα κύκλωμα καταστέλλει τα ξένα σήματα (θόρυβος και κυματισμός) στην είσοδο τροφοδοσίας για να τα εμποδίζει να αλλοιώσουν την έξοδο. Το PSRR ορίζεται ως

$$PSRR = 20 \times \log (VE_{IN} / VE_{OUT})$$

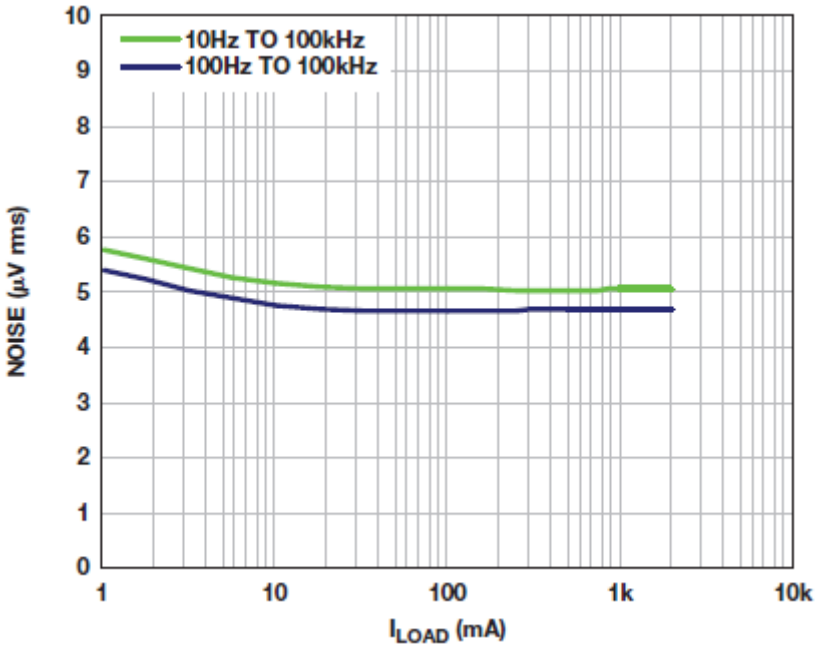
όπου V_{EIN} και V_{EOUT} είναι τα ξένα σήματα που εμφανίζονται στην είσοδο και την έξοδο, αντίστοιχα. Το PSRR έχει την ίδια σχέση με τη ρύθμιση γραμμής dc, αλλά περιλαμβάνει ολόκληρο το φάσμα συχνοτήτων.

Η απόρριψη τροφοδοσίας στην περιοχή των 100 kHz έως 1 MHz είναι πολύ σημαντική, καθώς τα switching τροφοδοτικά χρησιμοποιούνται συχνά σε συστήματα υψηλής απόδοσης, με τους LDO να καθαρίζουν τις θορυβώδεις τροφοδοσίες για τα ευαίσθητα αναλογικά κυκλώματα. Ο βρόχος ελέγχου του LDO τείνει να είναι ο κυρίαρχος παράγοντας στον καθορισμό της απόρριψης τροφοδοσίας. Οι πυκνωτές χαμηλού ESR μπορούν να είναι ευεργετικοί, ειδικά σε συχνότητες πέρα από το εύρος ζώνης κέρδους του βρόχου ελέγχου.

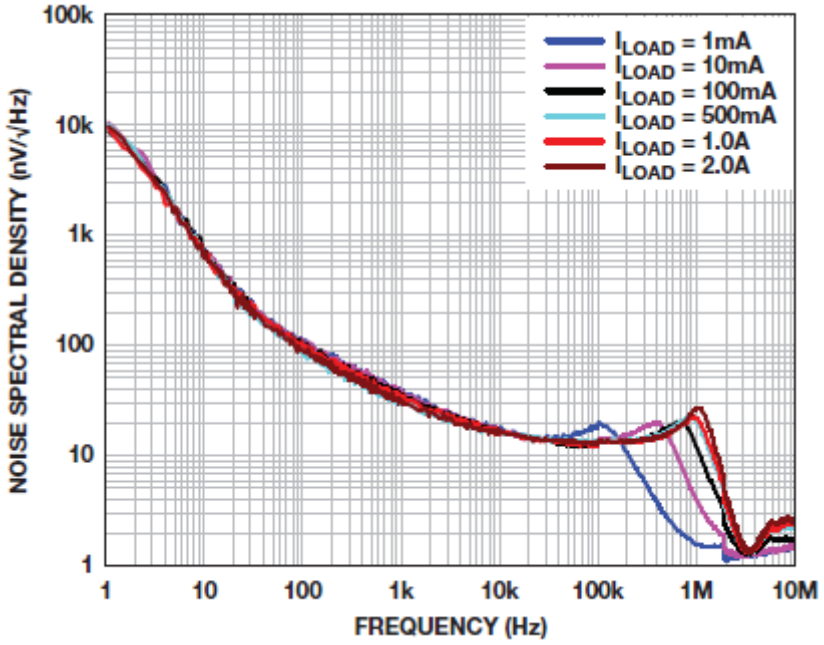
Τάση θορύβου στην έξοδο - Output Noise Voltage

Η τάση θορύβου στην έξοδο, είναι η rms τάση σε ένα δεδομένο εύρος συχνοτήτων (συνήθως 10 Hz ή 100 Hz έως 100 kHz) υπό τις συνθήκες ενός συνεχούς ρεύματος εξόδου και μιας τάσης εισόδου χωρίς κυματισμούς. Οι κύριες πηγές θορύβου στην έξοδο σε LDOs είναι η εσωτερική τάση αναφοράς και ο ενισχυτής σφάλματος. Τα σύγχρονα LDO λειτουργούν με εσωτερικά ρεύματα πόλωσης μερικών δεκάδων nanoamps για να επιτύχουν ρεύματα ηρεμίας 15 μA ή λιγότερα. Αυτά τα ρεύματα χαμηλής πόλωσης απαιτούν τη χρήση αντιστάσεων πόλωσης έως και GΩ. Ο θόρυβος εξόδου κυμαίνεται συνήθως από 5 μV rms έως 100 μV rms. Το Σχήμα 3.49 δείχνει το θόρυβο εξόδου έναντι του ρεύματος φορτίου.

Ένας άλλος τρόπος για να εκφραστεί ο θόρυβος εξόδου ενός LDO είναι η πυκνότητα φασματικού θορύβου. Το Σχήμα 3.50 δείχνει την φασματική πυκνότητα θορύβου από 1 Hz έως 10 MHz. Αυτές οι πληροφορίες μπορούν στη συνέχεια να χρησιμοποιηθούν για τον υπολογισμό του θορύβου rms σε μια συγκεκριμένη συχνότητα για ένα δεδομένο εύρος ζώνης. [35]



Σχήμα 3.49: Ο θόρυβος εξόδου σε σχέση με το ρεύμα φορτίου.



Σχήμα 3.50: Η φασματική πυκνότητα θορύβου σε σχέση με το ρεύμα φορτίου.

Εφαρμογές των low dropout ρυθμιστών

Σε price-sensitive εφαρμογές προτιμώνται οι γραμμικοί ρυθμιστές και η απόφαση σχεδιασμού είναι ιδιαίτερα ξεκάθαρη για τους κατασκευαστές των:

- εξοπλισμός επικοινωνιών
- μικρές συσκευές
- συστήματα με μπαταρία
- συσκευές χαμηλού ρεύματος
- μικροεπεξεργαστές υψηλής απόδοσης με κατάσταση sleep mode. [37]

ΒΙΒΛΙΟΓΡΑΦΙΑ 3ου ΚΕΦΑΛΑΙΟΥ

[1] CMOS DIGITAL INTERGRATED CIRCUITS, ANALYSIS AND DESIGN, Sung-Mo (Steve) Kang, University of Illinois at Urbana- Champaign, Yusuf Leblebigi, Worcester Polytechnic Institute, Swiss Federal Institute of Technology-Lausanne, McGraw-Hill Education, 3rd Edition, 2002.

[2] CMOS ΛΟΓΙΚΗ ΟΙΚΟΓΕΝΕΙΑ ΜΕ ΧΡΗΣΗ ΔΙΟΔΙΚΟΥ ΤΡΑΝΖΙΣΤΟΡ ΓΙΑ ΧΑΜΗΛΗ ΔΥΝΑΜΙΚΗ ΚΑΤΑΝΑΛΩΣΗ ΕΝΕΡΓΕΙΑΣ, Αικατερίνη Κατσαρού, Μεταπτυχιακή Εργασία, Πανεπιστήμιο Ιωαννίνων, Σχολή Θετικών Επιστημών, Τμήμα Μηχανικών Η/Υ & Πληροφορικής, Ιούνιος 2012.

[3] ΚΥΚΛΩΜΑΤΑ CMOS ΧΑΜΗΛΗΣ ΚΑΤΑΝΑΛΩΣΗΣ ΙΣΧΥΟΣ ΚΑΙ ΧΑΜΗΛΗΣ ΤΑΣΗΣ ΤΡΟΦΟΔΟΣΙΑΣ, Διδακτορική Διατριβή, Ιωάννης Χ. Μωυσιάδης, ΕΘΝΙΚΟ & ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ, Σχολή Θετικών Επιστημών Τμήμα Πληροφορικής, Αθήνα 2000.

[4] COMPUTER ORGANIZATION AND ARCHITECTURE DESIGNING FOR PERFORMANCE, Tenth Edition, William Stallings, With contribution by Peter Zeno, University of Bridgeport, Copyright © 2016, 2013, 2010 Pearson Education, Inc., Hoboken, NJ 07030.

[5] ENERGY MANAGEMENT TECHNIQUES FOR SOC DESIGN, Chapter 6, Hiroto YASUURA, Tohru ISHIHARA, Masanori MUROYAMA, System LSI Research Center, Kyushu University, 3-8-33, Momochihama, Sawara-ku, Fukuoka, 814-0001, JAPAN

[6] DESIGN TECHNIQUES FOR LOW POWER SYSTEMS, Paul J.M. Havinga, Gerard J.M. Smit, University of Twente, department of Computer Science, P.O. Box 217, 7500 AE Enschede, the Netherlands

[7] ΠΡΟΣΑΡΜΟΓΗ ΣΥΧΝΟΤΗΤΑΣ ΚΑΙ ΤΑΣΗΣ ΛΕΙΤΟΥΡΓΙΑΣ ΓΙΑ ΤΗ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΚΑΤΑΝΑΛΩΣΗΣ ΕΝΕΡΓΕΙΑΣ ΕΠΕΞΕΡΓΑΣΤΩΝ, Διπλωματική Εργασία, Σπηλιόπουλος Βασίλειος, Πανεπιστήμιο Πατρών Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών, Πάτρα, Οκτώβριος 2009

[8] ΥΛΟΠΟΙΗΣΗ ΤΕΧΝΙΚΩΝ ΔΙΑΧΕΙΡΙΣΗΣ ΤΟΥ ΡΟΛΟΓΙΟΥ ΜΕ ΣΤΟΧΟ ΤΗΝ ΜΕΙΩΣΗ ΤΗΣ ΚΑΤΑΝΑΛΩΣΗΣ ΕΝΕΡΓΕΙΑΣ ΣΕ ΠΟΛΥΝΗΜΑΤΙΚΟΥΣ ΚΑΙ ΠΟΛΥΠΥΡΗΝΟΥΣ ΕΝΣΩΜΑΤΩΜΕΝΟΥΣ ΕΠΕΞΕΡΓΑΣΤΕΣ ΓΡΑΦΙΚΩΝ,

Μεταπτυχιακή Διπλωματική Εργασία, Κεϊβάνη Ιωάννα, Πανεπιστήμιο Πατρων Τμήμα Μηχανικών Η/Υ & Πληροφορικής Μεταπτυχιακό Πρόγραμμα Σπουδών, Πάτρα, Φεβρουάριος 2017

[9] THRESHOLD-VOLTAGE CONTROL SCHEMES THROUGH SUBSTRATE-BIAS FOR LOW-POWER HIGH-SPEED CMOS LSI DESIGN, Tadahiro Kuroda And Taka Yasu Sakurai, Semiconductor Device Engineering Laboratory, Toshiba Corp. Journal of VLSI Signal Processing Systems 13, 191-201 (1996)

[10] INTERVAL-BASED MODELS FOR RUN-TIME DVFS ORCHESTRATION IN SUPERSCALAR PROCESSORS, Georgios Keramidas, Industrial Systems Institute Patras, Greece, Vasileios Spiliopoulos, Department of Electrical and Computer Engineering, University of Patras, Greece and Stefanos Kaxiras, Department of Electrical and Computer Engineering, University of Patras, Greece.

[11] MANAGING POWER AND PERFORMANCE FOR SYSTEM-ON-CHIP DESIGNS USING VOLTAGE ISLANDS, David E. Lackey, Paul S. Znchowski, Thomas R. Eednar, Douglas W. Stout, Scott W. G o a John M. Cobn, IBM Microelectronics Division, Essex Junction, Vermont 05452, USA. 0-7803-7607-2/02 @ 2002 IEEE

[12] COMPUTER ARCHITECTURE TECHNIQUES FOR POWER-EFFICIENCY, Stefanos Kaxiras, University of Patras, Greece and Margaret Martonosi, Princeton University. A Publication in the Morgan & Claypool Publishers series. Copyright © 2008 by Morgan & Claypool

[13] DCG: DETERMINISTIC CLOCK-GATING FOR LOW-POWER MICROPROCESSOR DESIGN, Hai Li, Swarup Bhunia, Yiran Chen, Kaushik Roy, Fellow, IEEE, and T. N. Vijaykumar, Member, IEEE. IEEE Transactions On Very Large Scale Integration (VLSI) Systems, Vol. 12, No. 3, March 2004.

[14] GATED CLOCK ROUTING FOR LOW-POWER MICROPROCESSOR DESIGN, Jaewon Oh and Massoud Pedram, Senior Member, IEEE. IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems, Vol. 20, No. 6, June 2001

[15] ΛΟΓΑΡΙΘΜΙΚΟ ΑΡΙΘΜΗΤΙΚΟ ΣΥΣΤΗΜΑ ΚΑΙ ΤΕΧΝΙΚΕΣ ΧΑΜΗΛΗΣ ΚΑΤΑΝΑΛΩΣΗΣ ΜΕ ΕΦΑΡΜΟΓΗ ΣΤΗΝ ΨΗΦΙΑΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΣΗΜΑΤΩΝ, Μεταπτυχιακή εργασία, Κασιώλας Βασίλειος, Πανεπιστήμιο Πατρών, Πολυτεχνική Σχολή, Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών & Πληροφορικής, Πρόγραμμα

Μεταπτυχιακών Σπουδών στα Ολοκληρωμένα Συστήματα Υλικού & Λογισμικού. Πατρα 2014

[16] ΤΕΧΝΙΚΗ ΧΑΜΗΛΗΣ ΣΤΑΤΙΚΗΣ ΚΑΤΑΝΑΛΩΣΗΣ ΨΗΦΙΑΚΩΝ ΟΛΟΚΛΗΡΩΜΕΝΩΝ ΚΥΚΛΩΜΑΤΩΝ ΜΕ ΤΗΝ ΕΙΣΑΓΩΓΗ ΠΟΛΛΑΠΛΩΝ ΚΑΤΑΣΤΑΣΕΩΝ ΥΠΝΟΥ, Μεταπτυχιακή εργασία, Κάτσικος Παναγιώτης, Τμήμα Πληροφορικής, Ιούλιος 2009.

[17] LOW-POWER DIGITAL VLSI DESIGN - CIRCUITS AND SYSTEMS, A. Bellaouar, M. I. Elmasry, Kluwer Academic Publishers, USA.

[18] HIGH SPEED CMOS DESIGN STYLES, K. Bernstein, Kluwer Academic Publishers, 1999

[19] CMOS DIGITAL INTEGRATED CIRCUITS, S.-M. Kang, Y. Leblebici, McGraw-Will, 1999

[20] LOW POWER DESIGN METHODOLOGIES, J. M. Rabaey, M. Pedram, Kluwer Academic Publishers, 1996, USA.

[21] MINIMIZING POWER CONSUMPTION IN DIGITAL CMOS CIRCUITS, A. P. Chandrakasan, R. W. Brodersen, Proceedings of the IEEE, voi. 83, no. 4, p.p. 498-523, April 1995

[22] A 450MHZ 64B RISC PROCESSOR USING MULTIPLE THRESHOLD VOLTAGE CMOS, T. Yamashita, et al., IEEE Int. Solid-State Circuits Conference, p.p. 414-415, 2000

[23] DESIGN AND OPTIMIZATION OF LOW VOLTAGE HIGH PERFORMANCE DUAL THRESHOLD CMOS CIRCUITS, L. Wei, et al., ACM/IEEE Design Automation Conf., p.p. 489-494, 1998

[24] MULTIPLE CLOCK DOMAIN CYCLE SKIPPING UTILIZING OPTIMAL MASK TO MINIMIZE VOLTAGE NOISE, Patent US20140095909, 2014

[25] DCG: DETERMINISTIC CLOCK GATING FOR LOW POWER MICROPROCESSOR DESIGN, Hai Li, S. Bhunia , Yiran Chen, K. Roy and T.N. Vijaykumar., IEEE Trans. VLSI Systems,Volume 12, Issue 3, March 2004 Page(s):245 254

[26] NEW CLOCK GATING TECHNIQUES FOR LOW POWER FLIP FLOPS, A.G.M. Stroll o, E. Napoli and D. De Caro., IEEE ISLPED'00, Rapallo, Italy , 2000, Page(s):114 119

[27]POWER COMPILER MANUAL, Synopsys Inc, https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/power-compiler.html

[28] POWER MANAGEMENT OF DIGITAL CIRCUITS IN DEEP SUB-MICRON TECHNOLOGIES, S. Henzler, Springer Series in Advanced Microelectronics, 2006.

[29] CLUSTERED VOLTAGE SCALING TECHNIQUE FOR LOW-POWER DESIGN, K. Usami, M. Horowitz, in Proc. Int. Symp. on Low-Power Electronics Design (ISPLED), pp. 3-8, 1995.

[30] ENERGY PRECISION TRADEOFFS IN THE GRAPHICS PIPELINE, J. Pool. Διδακτορική Διατριβή, University of North Carolina, Chaper Hill, 2012.

[31] INTERVAL BASED MODELS FOR RUN TIME DVFS ORCHESTRATION IN SUPERSCALAR PROCESSORS, G. Keramidas, V. Spiliopoulos and S. Kaxiras, In: Proceedings of 7 th ACM International Conference on Computer Frontiers (CF'10), 2010

[32] POWER REDUCTION THROUGH RTL CLOCK GATING, By Frank Emnett and Mark Biegel, SNUG San Jose 2000. Automotive Integrated Electronics Corporation.

[33] PIPELINING TO REDUCE THE POWER, Chapter 3, David Chinnery, Kurt Keutzer, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720, USA

[34] COMPUTER ARCHITECTURE, chapter 3, Dr.Wen-Chi Hou Professor Ph.D. Computer Engineering and Science, Case Western Reserve University, 1989, Department of Computer Science, Southern Illinois University, Carbondale, Carbondale, IL 62901

[35] UNDERSTAND LOW-DROPOUT REGULATOR (LDO) CONCEPTS TO ACHIEVE OPTIMAL DESIGNS, By Glenn Morita, Analog Dialogue 48-12, December 2014.

[36] UNDERSTANDING LOW-DROPOUT (LDO) REGULATORS, Michael Day, Texas Instruments.

[37] MICREL'S GUIDE TO DESIGNING WITH LOW-DROPOUT VOLTAGE REGULATORS, Bob Wolbert, Applications Engineering Manager, Revised Edition, December 1998.

[38] CMOS POWER MANAGEMENT UNIT ALONG WITH LOAD REGULATION USING SWITCHED CAPACITOR CONVERTERS, Purvi Patel and Biswajit Mishra∗, VLSI and Embedded Systems Research Group, DA-IICT, Gandhinagar 382007, India, Journal of Low Power Electronics, Vol. 15, 224–233, 2019, Copyright © 2019 American Scientific Publishers

[39] LOW POWER DESIGN FOR SOC WITH POWER MANAGEMENT UNIT, Daying Sun, Shen Xu, Weifeng Sun, Shengli Lu, Longxing Shi, National ASIC System Engineering Research Center, Southeast University, Nanjing 210096, China, 978-1-61284-193-9/11 ©2011 IEEE

[40] POWER MANAGEMENT FOR INTERNET OF THINGS (IOT) SYSTEM ON A CHIP (SOC) DEVELOPMENT, By Stephen M. Nolan, Vidatronic, Inc.

[41] POWER MANAGEMENT AND CALCULATION FOR CROSSLINK DEVICES, Lattice Semiconductor, Preliminary Technical Note, FPGA-TN-02018 Version 1.0, July 2016

ΚΕΦΑΛΑΙΟ 4:
Μνήμες και μείωση της κατανάλωσης ισχύος

4.1 Εισαγωγή

Οι μνήμες ημιαγωγών ικανές να αποθηκεύουν μεγάλες ποσότητες ψηφιακών πληροφοριών, είναι ουσιώδεις για όλα τα ψηφιακά συστήματα. Η ποσότητα μνήμης που απαιτείται σε ένα συγκεκριμένο σύστημα εξαρτάται από τον τύπο εφαρμογής, αλλά γενικά ο αριθμός των τρανζίστορς που χρησιμοποιούνται για τη λειτουργία αποθήκευσης δεδομένων, είναι πολύ μεγαλύτερος από τον αριθμό που χρησιμοποιούνται σε λογικές πράξεις και για άλλους σκοπούς. Η διαρκώς αυξανόμενη ζήτηση για μεγαλύτερη χωρητικότητα αποθήκευσης, οδήγησε την τεχνολογία κατασκευής και ανάπτυξης μνημών προς πιο συμπαγείς υλοποιήσεις και κατά συνέπεια, προς υψηλότερες πυκνότητες αποθήκευσης δεδομένων με σημαντική συμβολή στην αύξηση της κατανάλωσης ισχύος η οποία αποδίδεται σε αυτές. Η μνήμη αποτελεί έναν από τους μεγαλύτερους καταναλωτές ισχύος, με μερίδιο περίπου 20% έως και 25% του συνόλου της κατανάλωσης. [1] Το ποσοστό αυτό μεταβάλλεται ανάλογα με το είδος και την οργάνωση της μνήμης που χρησιμοποιείται, με τη καταναλισκόμενη ισχύ να εξαρτάται από τον αριθμό των αναγνώσεων και εγγραφών που πραγματοποιούνται, από τη δραστηριότητα που απαιτείται για τη διατήρηση των δεδομένων (ειδικά αυτών που για παράδειγμα σε ένα πολυεπεξεργαστικό σύστημα, χρησιμοποιούνται από περισσότερα cores), αλλά και από το leakage για τη διατήρηση των δεδομένων.

4.2 Βασικές αρχές σχεδίασης των τεχνικών μείωσης της κατανάλωσης ισχύος

Η σχεδίαση μνημών χαμηλής κατανάλωσης έχει δύο όψεις: τη μείωση της δυναμικής(active) ισχύος και τη μείωση της standby ισχύος.

Για τη μείωση της κατανάλωσης active ισχύος προτείνονται πολλές λύσεις, όπως ο κατακερματισμός (partitioning) των μεγάλων κρυφών μνημών σε μικρότερες προκειμένου να μειωθεί η δυναμική ενέργεια, αλλά και η χρήση κατάλληλης ιεραρχίας μνημών (memory hierarchy) ώστε οι προσβάσεις στη μνήμη να περιορίζονται κατά το δυνατόν σε ένα μόνο τμήμα της. Μία ακόμη τεχνική είναι αυτή της cache – way prediction, για να μειωθεί η κατανάλωση των κρυφών μνημών.

Μία ενδιαφέρουσα μέθοδος είναι αυτή της τοποθέτησης ενός φίλτρου στο interface της κρυφής μνήμης πρώτου επιπέδου, ώστε να αλλάζει κάθε φορά ο τρόπος οργάνωσης της μνήμης ανάλογα με τις απαιτήσεις. Το φίλτρο αυτό λέγεται filter cache και η τεχνική συνολικά ορίζεται σαν selective way caches. Τέλος, προτείνεται επίσης η οργάνωση του λογισμικού με τέτοιο τρόπο, ώστε να μειώνεται η πολυπλοκότητα

πράξεων μεταξύ δεδομένων και να ελαχιστοποιούνται έτσι κατά το δυνατόν οι προσβάσεις στη μνήμη. [15]

Για τη μείωση της κατανάλωσης standby ισχύος, η σχεδίαση εστιάζεται στο κυριότερο πρόβλημα που στην περίπτωση των μνημών είναι αυτό της διαχείρισης του leakage, μιας και τα στοιχεία μνήμης πάντα καταναλώνουν ασχέτως αν χρησιμοποιούνται ή όχι και για όσο υπάρχει τάση τροφοδοσίας σε αυτά.

Η τεχνική που μπορεί να εφαρμοστεί είναι η gated-V_{DD}, δηλαδή αυτή της απενεργοποίησης ενός τμήματος της μνήμης όταν αυτό δεν χρησιμοποιείται, διακόπτοντας την τάση τροφοδοσίας. Η χρήση της προτείνεται να γίνεται σε συνδυασμό με κατάλληλες παρεμβάσεις στο λογισμικό, μιας και θεωρείται πως τα δεδομένα σε κάθε τμήμα που απενεργοποιείται χάνονται. Το παραπάνω μειονέκτημα της απώλειας δεδομένων μπορεί να αναιρεθεί αν αντί της ολοκληρωτικής απουσίας τάσης χρησιμοποιηθεί μία χαμηλή τάση τροφοδοσίας, όση δηλαδή χρειάζεται για τη διατήρηση της κατάστασης της μνήμης όσο το τμήμα μνήμης δεν χρησιμοποιείται.

Για να εφαρμοστεί κάθε μία από τις παραπάνω προσεγγίσεις, θα πρέπει το σύστημα να είναι σε θέση να γνωρίζει ποιο τμήμα μνήμης δεν χρησιμοποιείται σε κάθε χρονική στιγμή. Για το σκοπό αυτό υπάρχει πληθώρα τεχνικών, με περισσότερο γνωστές αυτές της αυξομείωσης του μεγέθους της κρυφής μνήμης ανάλογα με τον όγκο δεδομένων που χρησιμοποιούνται, του περάσματος από τάση τροφοδοσίας σε χαμηλότερη τάση αν ένα τμήμα μνήμης δεν έχει χρησιμοποιηθεί για ένα προκαθορισμένο χρονικό διάστημα και την απενεργοποίηση τμημάτων με ποσοστό αποτυχιών (miss rate) πάνω από ένα όριο (hypothetical vs real miss rate). [15]

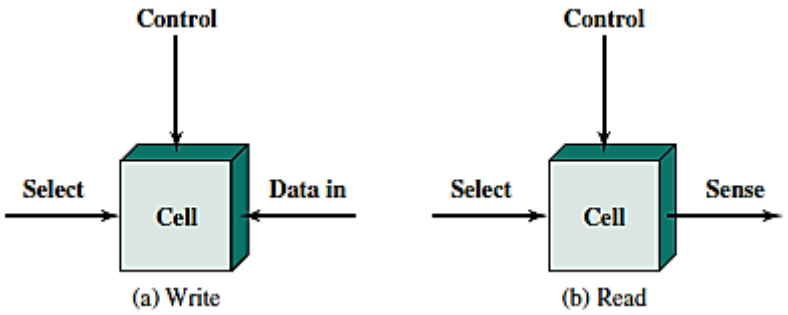
4.3 Το κελί μνήμης (memory cell)

Το βασικό στοιχείο μιας μνήμης ημιαγωγών είναι το κελί μνήμης. Αν και υπάρχουν πολλές τεχνολογίες υλοποίησης, όλες οι κυψέλες μνήμης μοιράζονται ορισμένες ιδιότητες:

- παρουσιάζουν δύο σταθερές (ή ημισταθερές) καταστάσεις, οι οποίες μπορούν να χρησιμοποιηθούν για δυαδικό 1 και 0.
- μπορούν να γράφονται (τουλάχιστον μία φορά) για να ορίσουν την κατάσταση.
- μπορούν να διαβάζουν την κατάσταση.

Το Σχήμα 4.1 απεικονίζει τη λειτουργία ενός κελιού μνήμης. Συχνότερα, το κελί διαθέτει τρεις εισόδους-εξόδους που μπορούν να μεταφέρουν ηλεκτρικό σήμα. Η είσοδος επιλογής(select), όπως υποδηλώνει το όνομα επιλέγει ένα κελί μνήμης για μια λειτουργία ανάγνωσης ή εγγραφής. Η είσοδος ελέγχου(control) υποδεικνύει την ανάγνωση ή την εγγραφή. Για την εγγραφή, η είσοδος Data in παρέχει ηλεκτρικό σήμα

που ορίζει την κατάσταση του κελιού σε 1 ή 0 και για την ανάγνωση η ίδια γραμμή χρησιμοποιείται για την έξοδο της κατάστασης του κελιού. [4]

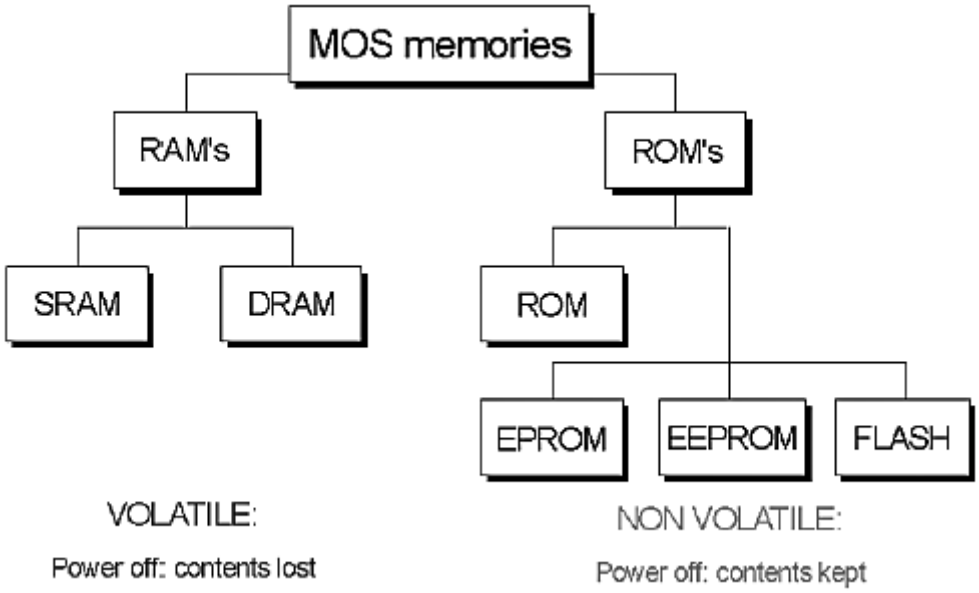


Σχήμα 4.1: Η λειτουργία του κελιού μνήμης.

4.4 Ταξινόμηση μνημών

Τα κυκλώματα μνήμης ταξινομούνται γενικά ανάλογα με τον τύπο αποθήκευσης δεδομένων και τον τύπο πρόσβασης δεδομένων. Τα κυκλώματα μνήμης μόνο για ανάγνωση (ROM-Read Only Memory) επιτρέπουν, όπως υποδηλώνει το όνομα, μόνο ανάκτηση των προηγουμένως αποθηκευμένων δεδομένων και δεν επιτρέπουν τροποποιήσεις των αποθηκευμένων πληροφοριών κατά την κανονική λειτουργία. Οι ROM δεν είναι πτητικές και η αποθήκευση δεδομένων δεν χάνεται ακόμη και όταν η τάση τροφοδοσίας είναι απενεργοποιημένη. Ανάλογα με τον τύπο της μεθόδου αποθήκευσης δεδομένων (εγγραφή δεδομένων), οι ROM ταξινομούνται ως mask-programmed ROMs, Programmable ROMs (PROM), Erasable PROMs (EPROM) και Electrically Erasable PROMs (EEPROM). [1]

Τα κυκλώματα μνήμης ανάγνωσης-εγγραφής (R/W) επιτρέπουν την τροποποίηση (εγγραφή) των bit δεδομένων που είναι αποθηκευμένα στη συστοιχία μνήμης, καθώς και την ανάκτησή τους (ανάγνωση) όταν αυτό ζητηθεί. Αυτό απαιτεί η λειτουργία αποθήκευσης δεδομένων να είναι πτητική, δηλαδή τα αποθηκευμένα δεδομένα θα χαθούν κατά την απενεργοποίηση της τάσης τροφοδοσίας. Το κύκλωμα μνήμης ανάγνωσης-εγγραφής ονομάζεται μνήμη τυχαίας προσπέλασης (RAM-Random Access Memory), κυρίως για ιστορικούς λόγους. Με βάση τον τύπο λειτουργίας των κυψελών αποθήκευσης δεδομένων, οι RAM ταξινομούνται σε δύο κύριες κατηγορίες: τη στατική RAM (SRAM-Static RAM) και τη δυναμική RAM (DRAM-Dynamic RAM). Στον Πίνακα 4.1 και στο Σχήμα 4.2 παρουσιάζονται οι διαφορετικοί τύποι μνημών και τη ταξινόμησή τους αντίστοιχα.



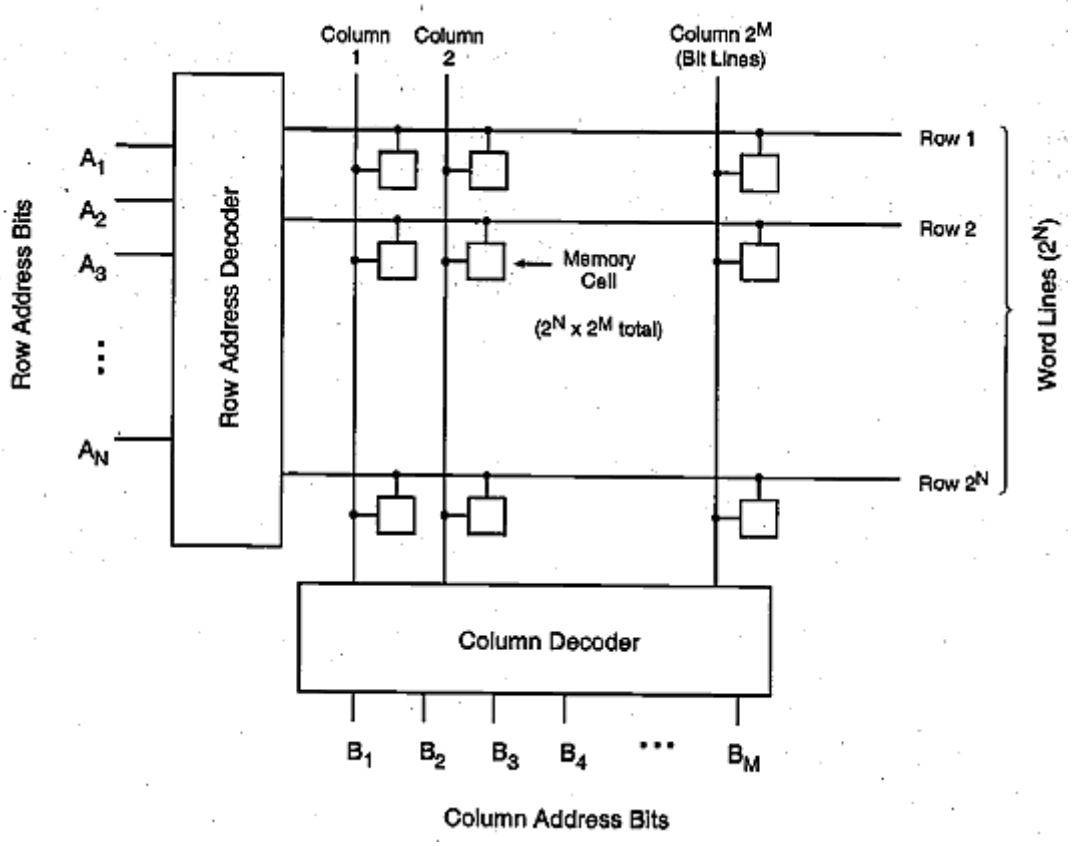
Σχήμα 4.2: Πτητικές και μη πτητικές μνήμες.

Πίνακας 4.1: Τύποι μνημών.

Memory Type	Category	Erasure	Write Mechanism	Volatility
Random-access memory (RAM)	Read-write memory	Electrically, byte-level	Electrically	Volatile
Read-only memory (ROM)	Read-only memory	Not possible	Masks	Nonvolatile
Programmable ROM (PROM)			Electrically	
Erasable PROM (EPROM)	Read-mostly memory	UV light, chip-level		
Electrically Erasable PROM (EEPROM)		Electrically, byte-level		
Flash memory		Electrically, block-level		

Μία τυπική οργάνωση συστοιχίας μνήμης φαίνεται στο σχήμα 4.3. Η δομή αποθήκευσης των δεδομένων ή πυρήνας, αποτελείται από μεμονωμένα κελιά μνήμης τακτοποιημένα σε μια συστοιχία οριζόντιων γραμμών και κατακόρυφων στηλών. Κάθε κελί έχει τη δυνατότητα να αποθηκεύει ένα μέρος των δυαδικών πληροφοριών. Επίσης, κάθε κελί μνήμης μοιράζεται μια κοινή σύνδεση με τα άλλα κελιά στην ίδια γραμμή, και μια άλλη κοινή σύνδεση με τα άλλα κελιά στην ίδια στήλη. Σε αυτή τη δομή, υπάρχουν 2^N γραμμές οι οποίες ονομάζονται γραμμές λέξεων(word lines) και 2^M στήλες, οι οποίες

ονομάζονται γραμμές bit (bit lines). Επομένως, το ο συνολικός αριθμός κελιών μνήμης σε αυτήν τη συστοιχία είναι $2^M x 2^N$. [1]



Σχήμα 4.3: Τυπική οργάνωση random-access memory συστοιχίας.

4.5 Κριτήρια απόδοσης μνημών

Η απόδοση περιοχής(area efficiency) της μνήμης, δηλαδή ο αριθμός των αποθηκευμένων bit δεδομένων ανά μονάδα περιοχής(stored data bits per unit area), είναι ένα από τα βασικά κριτήρια σχεδιασμού που καθορίζουν τη συνολική χωρητικότητα αποθήκευσης και επομένως, το κόστος μνήμης ανά bit. Ένα άλλο σημαντικό ζήτημα είναι ο χρόνος πρόσβασης στη μνήμη, δηλαδή, ο χρόνος που απαιτείται για την αποθήκευση ή/και την ανάκτηση ενός συγκεκριμένου bit δεδομένων στη μνήμη. Ο χρόνος πρόσβασης καθορίζει την ταχύτητα της μνήμης, η οποία αποτελεί σημαντικό κριτήριο απόδοσης. Τέλος, η στατική και δυναμική κατανάλωση ισχύος της μνήμης, αποτελεί σημαντικό παράγοντα που πρέπει να λαμβάνεται υπόψη στο σχεδιασμό, λόγω της αυξανόμενης σπουδαιότητας των εφαρμογών χαμηλής κατανάλωσης. [3]

4.6 Οι μνήμες DRAM και SRAM

Στον Πίνακα 4.1, το κοινό χαρακτηριστικό των μνημών αυτών, είναι η αναφορά τους ως μνήμη τυχαίας προσπέλασης (RAM). Πρόκειται στην πραγματικότητα για κατάχρηση του όρου, επειδή όλοι οι τύποι που αναφέρονται στον πίνακα είναι τυχαίας προσπέλασης.

Ένα χαρακτηριστικό που διακρίνει τις μνήμες που έχουν οριστεί ως RAM, είναι ότι είναι δυνατή και η ανάγνωση δεδομένων από τη μνήμη και η εγγραφή νέων δεδομένων στη μνήμη, εύκολα και με μεγάλη ταχύτητα. Το άλλο χαρακτηριστικό της RAM είναι ότι είναι πτητική(volatile), οπότε πρέπει να παρέχεται στη RAM με σταθερή τάση τροφοδοσίας. Εάν διακοπεί η τροφοδοσία, τότε τα δεδομένα χάνονται.

Επομένως, η μνήμη RAM μπορεί να χρησιμοποιηθεί μόνο για προσωρινή αποθήκευση. Οι δύο παραδοσιακές μορφές RAM που χρησιμοποιούνται στους υπολογιστές, είναι η DRAM και η SRAM. [4]

4.6.1 Η κατανάλωση ισχύος στις μνήμες DRAM και SRAM

Σε κάθε εφαρμογή χαμηλής ισχύος, η μείωση της τάσης λειτουργίας αποτελεί πάντα μια καλή μέθοδο για χαμηλότερη κατανάλωση δυναμικής ισχύος. Ωστόσο, η απόδοση του συστήματος και άλλες απαιτήσεις, συνήθως περιορίζουν αυτή τη μέθοδο και πρέπει να αναζητηθούν άλλες τεχνικές.

Υπάρχουν διάφορες πηγές ενεργής(active) ισχύος και ισχύος σε αναμονή (standby) για τη διατήρηση των δεδομένων(data retention) στις μνήμες. Η ενεργός ισχύς είναι το άθροισμα της ισχύος που καταναλώνεται από το τα ακόλουθα στοιχεία:

- αποκωδικοποιητές
- συστοιχία μνήμης
- ενισχυτές αίσθησης(sense amplifiers) και
- περιφερειακά κυκλώματα(κύκλωμα I/O, κύκλωμα εγγραφής κ.λπ.)

Όσον αφορά την standby κατανάλωση ισχύος, ο μεγαλύτερος παράγοντας που συμβάλλει σε αυτή είναι τα ρεύματα διαρροής από τη συστοιχία μνήμης, αφού τα στατικά ρεύματα από τα υπόλοιπα κυκλώματα είναι πρακτικά ελάχιστα(για παράδειγμα οι sense amplifiers απενεργοποιούνται στη standby κατάσταση λειτουργίας). [37]

4.6.2 SRAM vs DRAM

Οι στατικές και οι δυναμικές RAM είναι πτητικές, δηλαδή η τροφοδοσία πρέπει να παρέχεται συνεχώς στη μνήμη για τη διατήρηση των τιμών bit. Το κελί μίας δυναμικής μνήμης, είναι απλούστερο και μικρότερο από ένα κελί στατικής μνήμης και έτσι, μία DRAM είναι πιο πυκνή κατασκευαστικά (μικρότερα κελιά = περισσότερα κελιά ανά unit area) και λιγότερο δαπανηρή από μία αντίστοιχη SRAM. Από την άλλη πλευρά, μία DRAM απαιτεί την ύπαρξη κυκλώματος ανανέωσης(refresh) των δεδομένων.

Για μεγαλύτερες μνήμες, το σταθερό κόστος του κυκλώματος ανανέωσης αντισταθμίζεται από το μικρότερο κόστος των κυψελών DRAM. Έτσι, οι DRAM προτιμώνται όταν υπάρχουν μεγάλες απαιτήσεις μνήμης. Ένα τελευταίο σημείο είναι ότι οι SRAM είναι κάπως ταχύτερες από τις DRAM και γι'αυτό το λόγο, η SRAM χρησιμοποιείται για μνήμη cache (τόσο on-chip όσο και off-chip) και η DRAM χρησιμοποιείται για κύρια μνήμη. [4]

4.6.3 Δυναμική RAM - DRAM

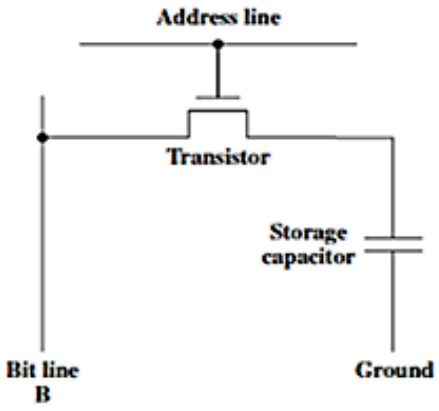
Η δυναμική RAM (DRAM) υλοποιείται με κελιά που αποθηκεύουν τα δεδομένα ως φόρτιση σε πυκνωτές. Η παρουσία ή η απουσία φόρτισης σε έναν πυκνωτή, ερμηνεύεται ως δυαδικό 1 ή 0. Επειδή οι πυκνωτές έχουν τη φυσική τάση για εκφόρτιση, οι RAM απαιτούν περιοδική ανανέωση της φόρτισης για τη διατήρηση των αποθηκευμένων δεδομένων. Ο όρος δυναμική αναφέρεται σε αυτή ακριβώς την τάση εκφόρτισης, ακόμη και όταν η τροφοδοσία εφαρμόζεται συνεχώς.

Το Σχήμα 4.4 φαίνεται μια τυπική δομή DRAM για ένα μεμονωμένο κελί που αποθηκεύει ένα bit. Η γραμμή διεύθυνσης ενεργοποιείται όταν πρόκειται να γίνει ανάγνωση της τιμής bit από αυτό το κελί. Το τρανζίστορ λειτουργεί ως διακόπτης που είναι κλειστός (επιτρέποντας την παροχή ρεύματος) εάν η τάση εφαρμόζεται στη γραμμή διεύθυνσης και είναι ανοιχτός (χωρίς ροές ρεύματος) εάν δεν υπάρχει τάση παρουσία στη γραμμή διεύθυνσης.

Για τη λειτουργία εγγραφής, εφαρμόζεται σήμα τάσης στη γραμμή bit. Υψηλή τάση αντιπροσωπεύει το 1 και μια χαμηλή τάση αντιπροσωπεύει το 0. Στη συνέχεια εφαρμόζεται ένα σήμα στη γραμμή διεύθυνσης, επιτρέποντας τη φόρτιση του πυκνωτή.

Για τη λειτουργία ανάγνωσης, όταν η γραμμή διεύθυνσης είναι επιλεγμένη, το τρανζίστορ είναι σε κατάσταση on και το φορτίο που είναι αποθηκευμένο στον πυκνωτή, τροφοδοτείται στη γραμμή bit. Ακολούθως, συγκρίνεται η τάση του πυκνωτή με μια τιμή αναφοράς και καθορίζει αν το κελί περιέχει μια λογική 1 ή μια λογική 0.

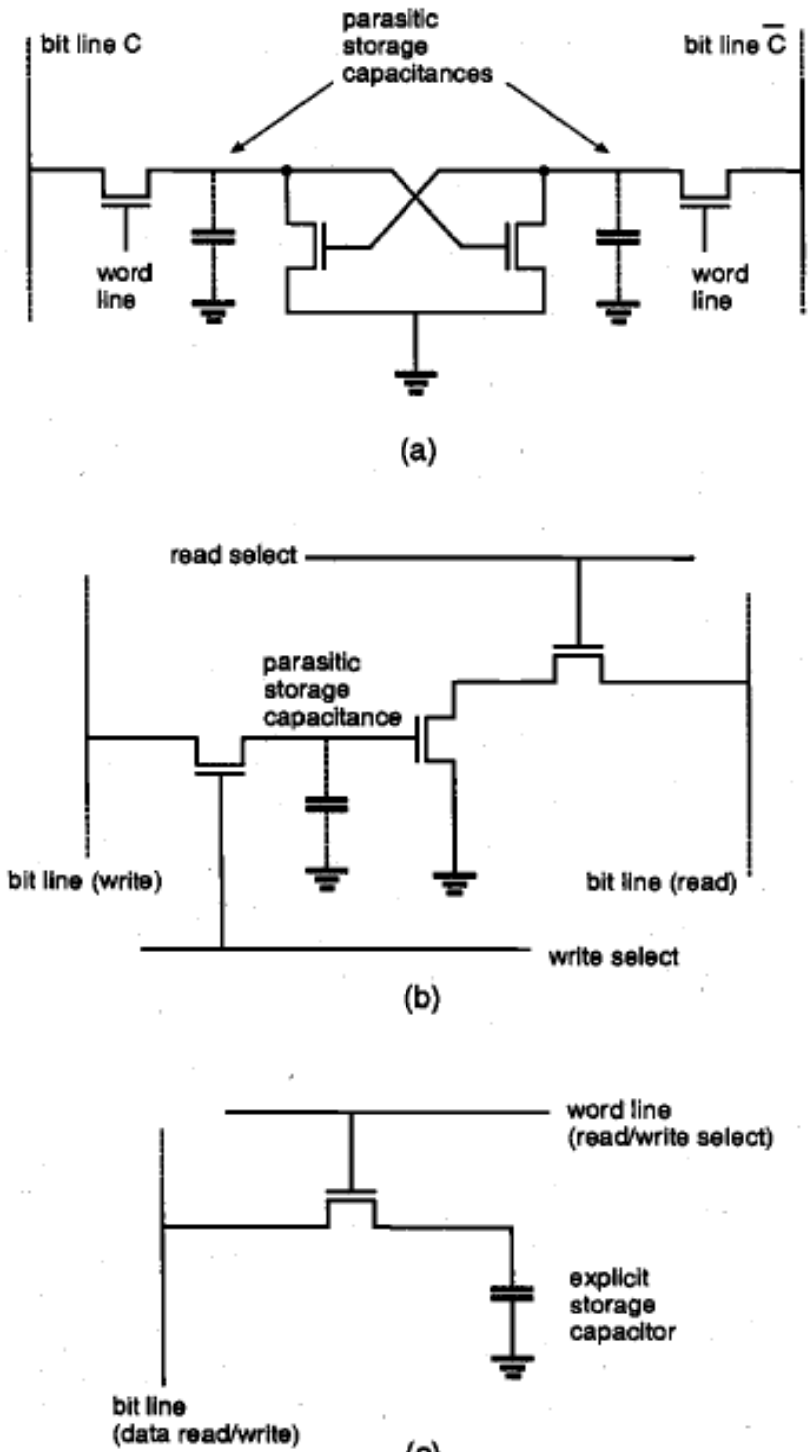
Αν και το κελί DRAM χρησιμοποιείται για την αποθήκευση ενός μόνο bit (0 ή 1), είναι ουσιαστικά αναλογική συσκευή, γιατί ο πυκνωτής μπορεί να αποθηκεύσει οποιαδήποτε τιμή φόρτισης εντός ενός εύρους και μία τιμή κατωφλίου καθορίζει εάν η φόρτιση ερμηνεύεται ως 1 ή 0. [4]



Σχήμα 4.4: DRAM memory cell.

4.6.4 Κυκλώματα κελιών δυναμικής μνήμης ανάγνωσης-εγγραφής (DRAM)

Στο Σχήμα 4.5, φαίνονται οι διαφορετικές υλοποιήσεις DRAM κελιού [3].

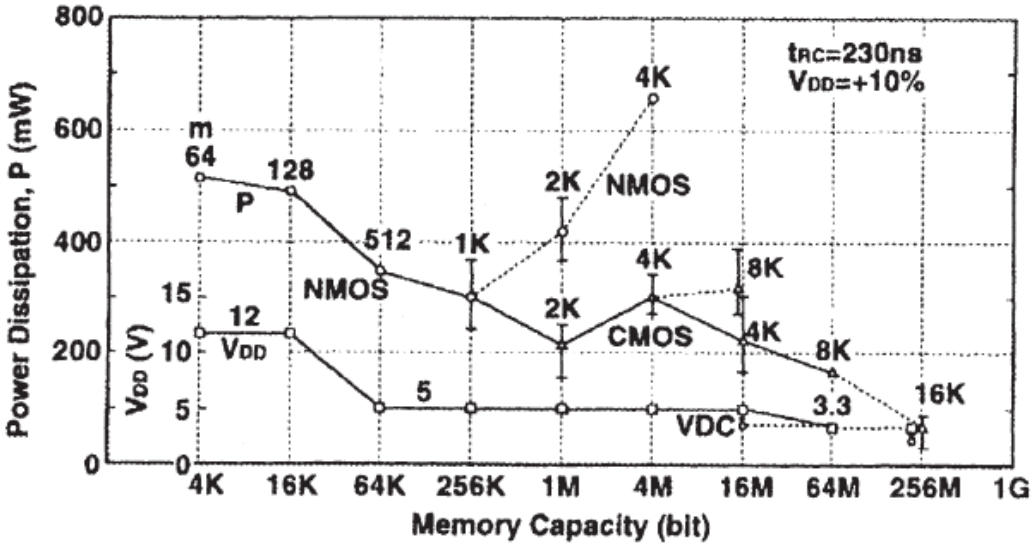


Σχήμα 4.5: Διαφορετικές υλοποιήσεις dynamic RAM cell. (a) Four-transistor DRAM cell με δύο κόμβους αποθήκευσης. (b) Three-transistor DRAM cell με δύο bit lines και δύο word lines. (c) One-transistor DRAM cell με μία bit line και μία word line.

4.6.5 Τεχνικές χαμηλής ισχύος DRAM

Η σταδιακή μείωση της standby ισχύος και της δυναμικής(active) ισχύος, παρά την αύξηση της χωρητικότητας της μνήμης, φαίνεται στο Σχήμα 4.6. Αυτό οφείλεται στα κυκλώματα χαμηλής κατανάλωσης που αναπτύσσονται σε κάθε γενιά. Για μία συγκεκριμένη χωρητικότητα μνήμης, η διαδοχική εξέλιξη έχει προκαλέσει μείωση ισχύος ισοδύναμη με 2 έως και 3 τάξεις μεγέθους. Αυτή η μείωση ισχύος οφείλεται στην υλοποίηση πολλών εξελιγμένων κυκλωμάτων όπως για παράδειγμα:

- μερική ενεργοποίηση μιας πολυδιαιρεμένης (multi-divided) γραμμής δεδομένων που μειώνει τη συνολική χωρητικότητα των γραμμών δεδομένων.
- Χρήση αποκωδικοποιητή CMOS NAND με μειωμένη χωρητικότητα.
- Μείωση της εξωτερικής τάσης τροφοδοσίας V_{DD} από 5 σε 3,3 V και χαμηλότερα.
- on-chip μείωση της τάσης που μειώνει τη V_{DD} κ.α.



Σχήμα 4.6: Τάση μείωσης της κατανάλωσης ισχύος στις μνήμες DRAM.

Μείωση δυναμικής ισχύος(active power)

Για τη μείωση της δυναμικής(active) ισχύος, αναφέρονται οι ακόλουθες τεχνικές:

- **Μείωση χωρητικότητας φόρτισης**

α. Partial Activation of Multi-divided Data Lines

β. Partial Activation of Multi-divided Word Lines

γ. Refresh Time Increase

- **Μείωση τάσης λειτουργίας**

α. Half-VDD Data Line Precharge

β. On-Chip Voltage-Down Converter (VDC)

γ. Signal-to-Noise-Ratio Improvement

- **Μείωση ρεύματος DC**

Μείωση ισχύος αναμονής(standby) για τη διατήρηση των δεδομένων(data retention)

Για τη μείωση της standby ισχύος, αναφέρονται οι ακόλουθες τεχνικές:

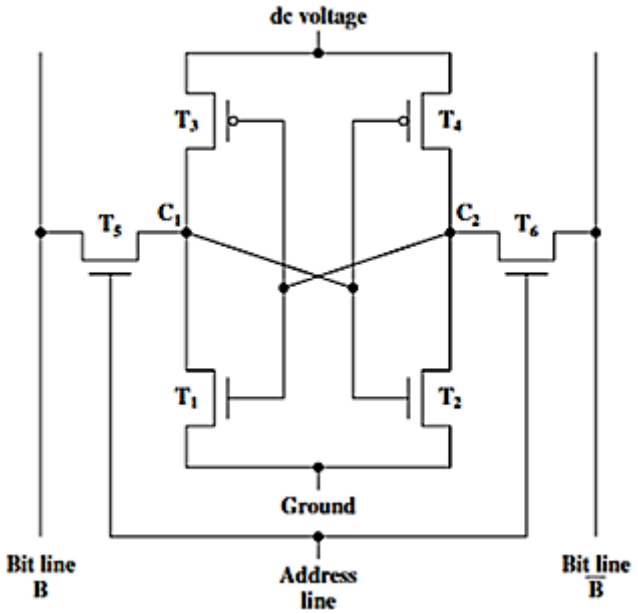
α. Voltage Conversion Circuits

β. Refresh Time Extension

γ. Refresh Charge Reduction. [38]

4.7 Στατική RAM - SRAM

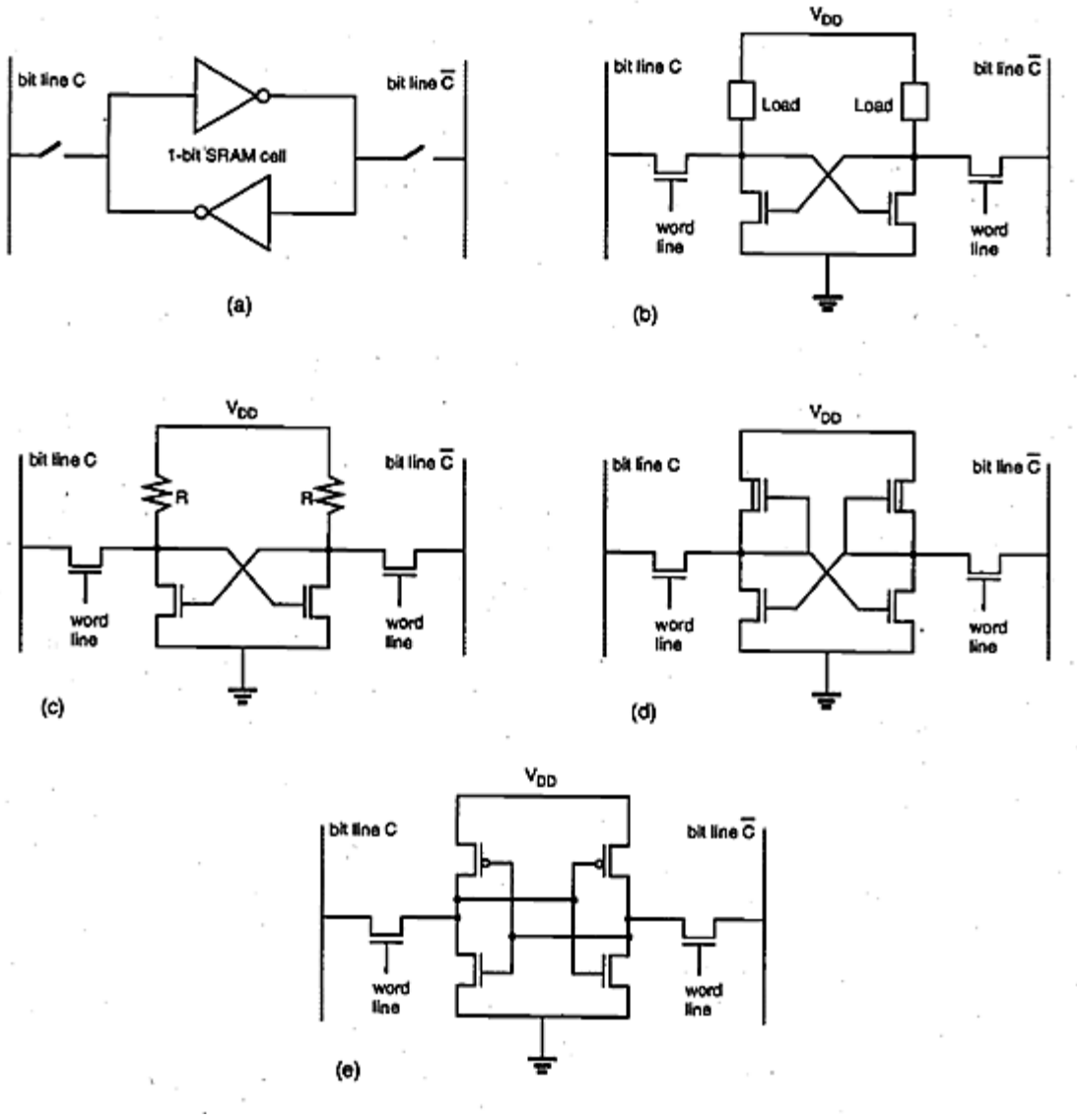
Σε μία SRAM, αποθηκεύονται δυαδικές τιμές με τη χρήση παραδοσιακών υλοποιήσεων λογικών πυλών και flip-flop. Μια στατική μνήμη RAM διατηρεί τα δεδομένα της για όσο χρόνο της παρέχεται τροφοδοσία. Το Σχήμα 4.7 παρουσιάζεται μια τυπική δομή SRAM για ένα μεμονωμένο κελί, όπου τέσσερα τρανζίστορς T1, T2, T3 και T4 συνδέονται μεταξύ τους σε διάταξη που παράγει σταθερή λογική κατάσταση. [4]



Σχήμα 4.7: SRAM memory cell.

4.7.1 Κυκλώματα κελιών Στατικής μνήμης ανάγνωσης-εγγραφής (SRAM)

Διαφορετικά κελιά μνήμης στατικής RAM, φαίνονται στο Σχήμα 4.8. [3]



Σχήμα 4.8: Διαφορετικές υλοποιήσεις SRAM cell (a) Συμβολική αναπαράσταση. (b) MOS SRAM cell. (c) Resistive-load SRAM cell. (d) Depletion-load nMOS SRAM cell. (e) Full CMOS SRAM cell.

4.7.2 Τεχνικές χαμηλής κατανάλωσης ισχύος SRAM

Η σπουδαιότητα των τεχνικών μείωσης της ισχύος εξαρτάται από την εφαρμογή. Στα περισσότερα συστήματα με τροφοδοσία μπαταρίας, η SRAM συνήθως δαπανά το μεγαλύτερο μέρος του χρόνου σε αδράνεια, ενώ σε συστήματα υψηλής απόδοσης η standby ισχύς ενδέχεται να μην είναι τόσο σημαντική.

Καθώς οι SRAM έχουν σχετικά σταθερή αρχιτεκτονική, οι τεχνικές μείωσης της δυναμικής ισχύος ταξινομούνται εύκολα. [37]

Μείωση της δυναμικής ισχύος(active power)

Για τη μείωση της δυναμικής(active) ισχύος, αναφέρονται οι ακόλουθες τεχνικές:

▪ **Μείωση ρεύματος DC**

α. Partial Activation of Multi-divided Word Lines

β. Pulse Operation of Word-Line Circuitry

γ. Pulse Operation of Column/Sense Circuitry

▪ **Μείωση τάσης λειτουργίας**

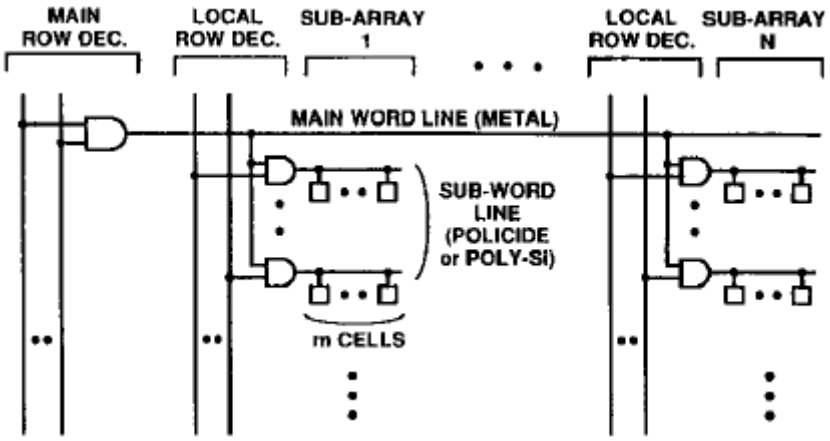
α. Low Voltage Sense Amplifier

β. On-chip Voltage Down Conversion

▪ **Μείωση χωρητικότητας φόρτισης**. [38]

4.7.3 Μείωση χωρητικότητας

Τα μεγαλύτερα χωρητικά στοιχεία σε μια μνήμη είναι οι word-line, bitlines και datalines, το καθένα με έναν αριθμό κελιών συνδεδεμένα με αυτά. Επομένως, η μείωση του μεγέθους αυτών των γραμμών, έχουν σημαντικό αντίκτυπο στην μείωση της κατανάλωσης ισχύος. Μια κοινή τεχνική που χρησιμοποιείται συχνά σε μεγάλες μνήμες λέγεται Divided Word Line (DWL) που υιοθετεί τη δομή ενός αποκωδικοποιητή δύο σταδίων όπως εμφαίνεται στο σχήμα 4.9. [37]



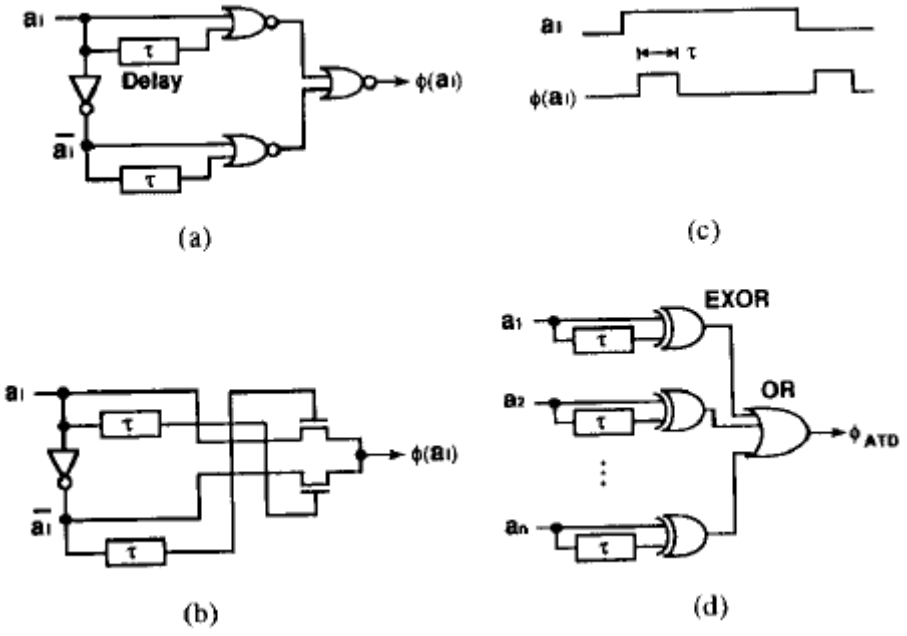
Σχήμα 4.9: Divided Word-Line Structure (DWL).

4.7.4 Η τεχνική Memory Bank Partitioning

Μια γνωστή τεχνική που χρησιμοποιείται για τη βελτίωση της ταχύτητας, είναι η κατάτμηση μιας συστοιχίας μνήμης σε μικρότερα τμήματα(banks) μνήμης, ώστε να ενεργοποιηθεί μόνο το τμήμα στο οποίο απευθύνεται το αίτημα. Ενδιαφέρον παρουσιάζει το γεγονός ότι η τεχνική αυτή μπορεί επίσης να βελτιώσει την ενεργειακή απόδοση της SRAM επειδή η χωρητικότητα της γραμμής λέξης(wordline) που αλλάζει και ο αριθμός των bit κελιών που ενεργοποιούνται μειώνονται.

Η διαμερισματοποίηση μπορεί να εφαρμοστεί ιεραρχικά για την παραγωγή μικρότερων τμημάτων μνήμης. Ωστόσο, κάποια στιγμή, η καθυστέρηση και η επιβάρυνση στη κατανάλωση που σχετίζονται με το κύκλωμα αποκωδικοποίησης των τμημάτων μνήμης κυριαρχεί και η περαιτέρω κατάτμηση γίνεται προβληματική. Συνήθως, δύο έως οκτώ τμήματα ενδείκνυνται. [36]

Άλλες τεχνικές είναι οι Pulse operation τεχνικές όπου και χρησιμοποιούνται Address Transition Detection (ATD) γεννήτριες παλμών (Σχήμα 4.10), οι τεχνικές Reduced Bitline Swing και οι τεχνικές Low-power sensing.



Σχήμα 4.10: Κυκλώματα Address transition detection. a) και b) ATD γεννήτριες παλμών, c) κυματομορφές ATD, d) κύκλωμα αθροιστή των παλμών ATD.

4.7.5 Μείωση της ισχύος αναμονής (standby)

Στις CMOS SRAM χαμηλής κατανάλωσης, το ρεύμα διαρροής των στατικών κυττάρων μνήμης, είναι η κύρια πηγή κατανάλωσης ισχύος που φτάνει έως και το 30%

της συνολικής κατανάλωσης. Καθώς, η πυκνότητα των κελιών της μνήμης SRAM είναι χαμηλή και τα περισσότερα δεδομένα επεξεργασίας μνήμης παραμένουν σταθερά κατά τη διάρκεια της λειτουργίας διατήρησης των δεδομένων, τα αποθηκευμένα δεδομένα μνήμης επηρεάζονται περισσότερο από τα φαινόμενα διαρροής στο κύκλωμα. Όσο αυξάνεται το μέγεθος της SRAM, το στατικό ρεύμα πρέπει να να μειωθεί, διαφορετικά το συνολικό στατικό ρεύμα της SRAM καθίσταται μη ανεκτό.

Η χρήση κελιών SRAM που χρησιμοποιούν 6 τρανζίστορς (PMOS φορτία) μπορεί να μειώσει το στατικό ρεύμα αλλά χρειάζεται περισσότερη επιφάνεια για τη δημιουργία του κελιού. Η τάση κατωφλίου συχνά μειώνεται για να αντισταθμιστεί την υποβάθμιση της ταχύτητας λόγω χαμηλής τάσης λειτουργίας. Η μείωση όμως της τάσης κατωφλίου οδηγεί σε εκθετική αύξηση των στατικών ρευμάτων διαρροής του SRAM κελιού και μπορεί να καταστεί περιοριστικός παράγοντας. Μία λύση είναι η χρήση της τεχνικής πολλαπλών κατωφλίων στο chip μνήμης.

4.8 Οι μνήμες ROM

Όπως υποδηλώνει το όνομα, μια μνήμη μόνο για ανάγνωση (ROM) περιέχει ένα μόνιμο μοτίβο δεδομένων που δεν μπορούν να αλλάξουν. Η ROM δεν είναι πτητική, δηλαδή δεν απαιτείται πηγή ενέργειας για τη διατήρηση των τιμών bit στη μνήμη. Ενώ είναι δυνατή η ανάγνωση μίας ROM, δεν είναι δυνατή η εγγραφή νέων δεδομένων σε αυτή. Το πλεονέκτημα της ROM είναι ότι τα δεδομένα ή το πρόγραμμα βρίσκεται μόνιμα στην κύρια μνήμη και δεν χρειάζεται ποτέ να φορτωθεί από δευτερεύουσα συσκευή αποθήκευσης.

Μία ROM δημιουργείται όπως κάθε άλλο ολοκληρωμένο κύκλωμα, με τα δεδομένα ήδη προγραμματισμένα στο τσιπ ως μέρος της διαδικασίας κατασκευής. Αυτό παρουσιάζει δύο προβλήματα:

■ Η εισαγωγή δεδομένων περιλαμβάνει ένα σχετικά μεγάλο σταθερό κόστος, ακόμα και αν κατασκευάζονται χιλιάδες αντίγραφα μιας συγκεκριμένης ROM.

■ Δεν υπάρχει περιθώριο λάθους. Εάν ένα bit είναι λάθος, ολόκληρη η παρτίδα των ROM αχρηστεύεται.

Όταν απαιτείται μόνο ένας μικρός αριθμός ROM με συγκεκριμένο περιεχόμενο μνήμης, μια λιγότερο δαπανηρή εναλλακτική είναι η προγραμματιζόμενη ROM (PROM). Όπως η ROM και η PROM δεν είναι πτητική και μπορεί να γραφτεί μόνο μία φορά. Για τη PROM, η διαδικασία εγγραφής γίνεται ηλεκτρικά και μπορεί να εκτελεστεί από τον προμηθευτή ή τον πελάτη κάποια στιγμή αργότερα από την αρχική κατασκευή του chip και απαιτείται ειδικός εξοπλισμός για τη διαδικασία εγγραφής ή "προγραμματισμού". Οι PROM παρέχουν ευελιξία και πρακτικότητα, ενώ η ROM εξακολουθεί να είναι ελκυστική για μεγάλες ποσότητες παραγωγής.

Μια άλλη παραλλαγή μνήμης μόνο για ανάγνωση, είναι η μνήμη η οποία είναι χρήσιμη για εφαρμογές στις οποίες οι λειτουργίες ανάγνωσης είναι πολύ πιο συχνές

από την εγγραφή και για τις οποίες απαιτείται μη πτητική αποθήκευση. Υπάρχουν τρείς τέτοιες μορφές μνήμης: η EPROM, η EEPROM και μνήμη flash.

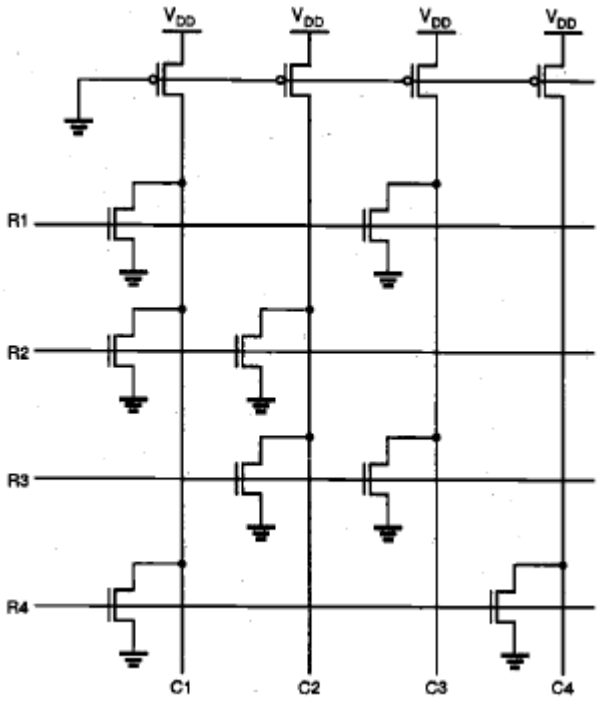
Η προγραμματιζόμενη μνήμη μόνο για ανάγνωση EPROM, διαβάζεται και γράφεται ηλεκτρικά, όπως και η PROM. Ωστόσο, πριν από μια λειτουργία εγγραφής, τα παλαιά δεδομένα πρέπει να διαγράφονται με την έκθεση του τσιπ στην υπεριώδη ακτινοβολία μέσω ενός παραθύρου που έχει σχεδιαστεί πάνω στο chip μνήμης. Αυτή η διαδικασία διαγραφής μπορεί να εκτελεστεί επανειλημμένα και κάθε διαγραφή μπορεί να διαρκέσει έως και 20 λεπτά. Έτσι, η EPROM μπορεί να τροποποιηθεί πολλές φορές και, όπως η ROM και η PROM, κρατούν τα δεδομένα τους πρακτικά επ' αόριστο. Για συγκρίσιμες ποσότητες αποθήκευσης, η EPROM είναι περισσότερο ακριβή από τη PROM, αλλά έχει το πλεονέκτημα της δυνατότητας πολλαπλής ενημέρωσης. [4]

Μια πιο ελκυστική μορφή μνήμης είναι η ηλεκτρικά διαγραφόμενη και προγραμματιζόμενη μνήμη μόνο για ανάγνωση EEPROM. Αυτή είναι μια μνήμη που μπορούν να γράφονται ανά πάσα στιγμή νέα δεδομένα χωρίς να σβηστούν τα προηγούμενα παρά μόνο το byte ή τα bytes που ενημερώθηκαν. Η εργασία εγγραφής διαρκεί πολύ περισσότερο από ό,τι η λειτουργία ανάγνωσης. Η EEPROM συνδυάζει το πλεονέκτημα της μη πτητικότητας με την ευελιξία της δυνατότητας πολλαπλής ενημέρωσης, όντας όμως πιο ακριβή από την EPROM και μικρότερης πυκνότητας, υποστηρίζοντας λιγότερα bit ανά chip.

Μια άλλη μορφή μνήμης ημιαγωγών είναι η μνήμη flash (που πήρε το όνομά της επειδή το μικροτσίπ είναι οργανωμένο έτσι ώστε τα κελιά ενός τμήματος μνήμης διαγράφονται με μία μόνο ενέργεια ή "flash"). Πρωτοπαρουσιάστηκε στα μέσα της δεκαετίας του 1980 και η μνήμη flash είναι μία ενδιάμεση κατάσταση μεταξύ EPROM και EEPROM τόσο ως προς το κόστος όσο και ως προς τη λειτουργικότητα. Όπως και η EEPROM, η μνήμη flash χρησιμοποιεί τεχνολογία ηλεκτρικής διαγραφής, ωστόσο δεν παρέχει διαγραφή σε επίπεδο byte. Μια ολόκληρη flash μνήμη μπορεί να διαγραφεί σε ένα ή λίγα δευτερόλεπτα, πολύ πιο γρήγορα από τη μνήμη EPROM. Επιπλέον, είναι δυνατή η διαγραφή μπλοκ μνήμης αντί ενός ολόκληρου τσιπ. Όπως και με την EPROM, η μνήμη flash χρησιμοποιεί μόνο ένα τρανζίστορ ανά bit και έτσι επιτυγχάνει την υψηλή πυκνότητα της EPROM(σε σύγκριση με την EEPROM). [4]

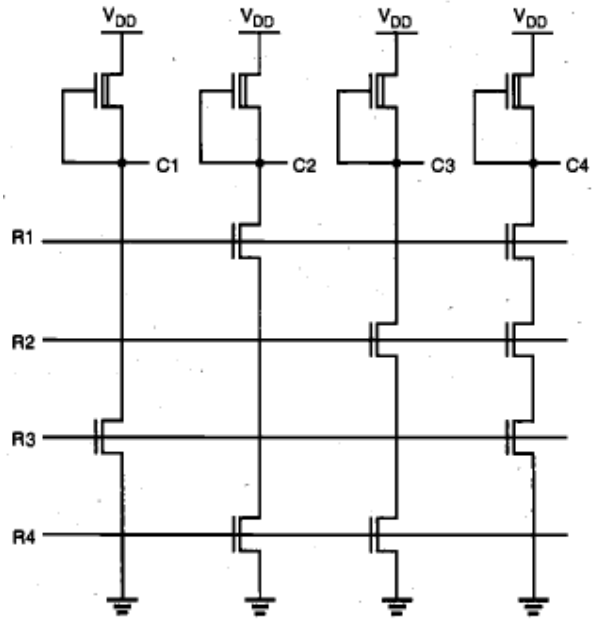
4.8.1 Κυκλώματα μνήμης μόνο για ανάγνωση (ROM)

Η συστοιχία μνήμης μόνο για ανάγνωση μπορεί επίσης να θεωρηθεί ως ένα απλό Boolean δικτύωμα, το οποίο παράγει μια καθορισμένη τιμή εξόδου για κάθε συνδυασμό εισόδου, δηλαδή για κάθε διεύθυνση(Σχήματα 4.11, 4.12). [3]



R1	R2	R3	R4	C1	C2	C3	C4
1	0	0	0	0	1	0	1
0	1	0	0	0	0	1	1
0	0	1	0	1	0	0	1
0	0	0	1	0	1	1	0

Σχήμα 4.11: 4-bit x 4-bit NOR-based μνήμη ROM.



R1	R2	R3	R4	C1	C2	C3	C4
0	1	1	1	0	1	0	1
1	0	1	1	0	0	1	1
1	1	0	1	1	0	0	1
1	1	1	0	0	1	1	0

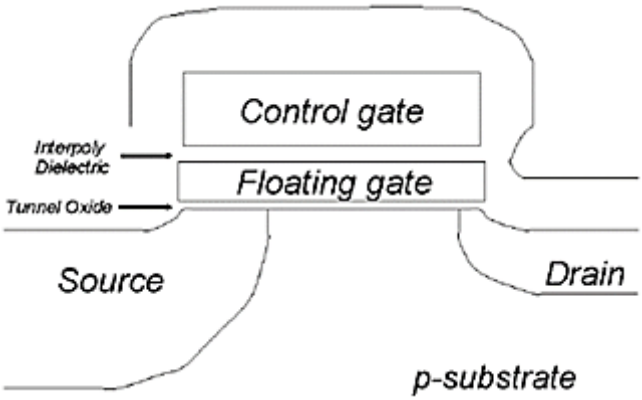
Σχήμα 4.12: 4-bit x 4-bit NAND-based μνήμη ROM.

4.8.2 FLASH memory

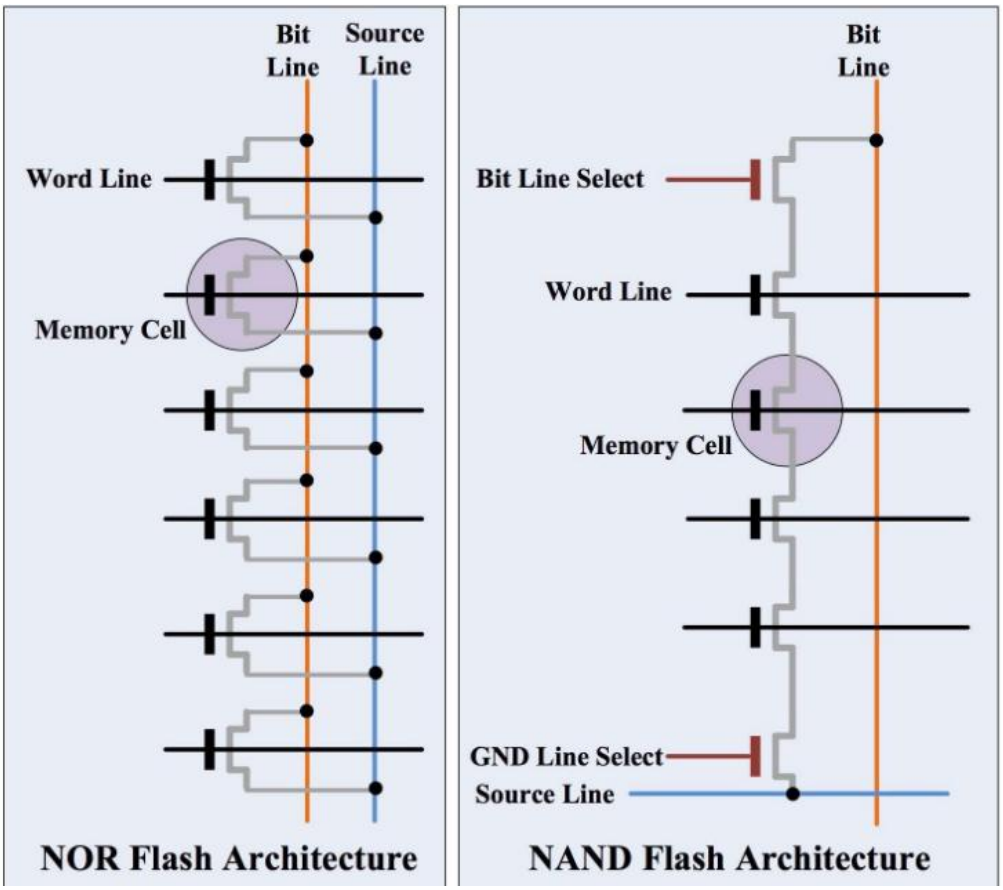
Από την πρώτη φορά που αναφέρθηκε η ιδέα της flash μνήμης, πολλές διαφορετικές παραλλαγές κυττάρων μνήμης και πολλές τεχνικές κυκλώματος έχουν αναπτυχθεί και εμπορευματοποιηθεί πολλαπλασιάζοντας με τη σειρά τους τις χρήσεις τους. Τα κελιά μνημών flash, διαιρούνται σε δύο ομάδες. Σε αυτά της αρχιτεκτονικής NOR Flash και σε αυτά της NAND flash.

4.8.3 Flash Memory Architecture

Οι μνήμες Flash αποθηκεύουν πληροφορίες σε κελιά μνήμης κατασκευασμένα από floating gate τρανζίστορς(Σχήμα 4.13). Τα ονόματα των τεχνολογιών εξηγούν τον τρόπο οργάνωσης των κυττάρων μνήμης. Στη NOR flash, το ένα άκρο κάθε κελιού μνήμης συνδέεται με τη γραμμή source και το άλλο άκρο απευθείας σε μια γραμμή bit που μοιάζει με πύλη NOR. Στη NAND Flash, αρκετά κελιά μνήμης (συνήθως οκτώ κελιά) συνδέονται σε σειρά παρόμοια με μια πύλη NAND (βλ. Σχήμα 4.14).



Σχήμα 4.13: Κελί μνήμης flash



Σχήμα 4.14: NOR και NAND flash αρχιτεκτονικές.

Η NOR flash αρχιτεκτονική, παρέχει αρκετές γραμμές διευθύνσεων για τη διευθυνσιοδότηση ολόκληρου του εύρους μνήμης. Αυτό δίνει το πλεονέκτημα της τυχαίας πρόσβασης και των σύντομων χρόνων ανάγνωσης, γεγονός που το καθιστά ιδανικό για εκτέλεση κώδικα. Στα μειονεκτήματα περιλαμβάνονται το μεγαλύτερο μέγεθος κελιού με αποτέλεσμα υψηλότερο κόστος ανά bit και οι πιο αργές ταχύτητες εγγραφής και διαγραφής. Αντίθετα, η αρχιτεκτονική NAND flash έχει πολύ μικρότερο μέγεθος κελιού που της δίνει το πλεονέκτημα της high-bit πυκνότητας, με το κόστος ανα-bit να είναι το χαμηλότερο μεταξύ των μνημών flash και πολύ υψηλότερες ταχύτητες εγγραφής και διαγραφής σε σύγκριση με τη NOR flash. Στα μειονεκτήματα περιλαμβάνονται η χαμηλότερη ταχύτητα ανάγνωσης και η αδυναμία τυχαίας προσπέλασης. [42]

4.8.4 Κατανάλωση ισχύος στις μνήμες flash

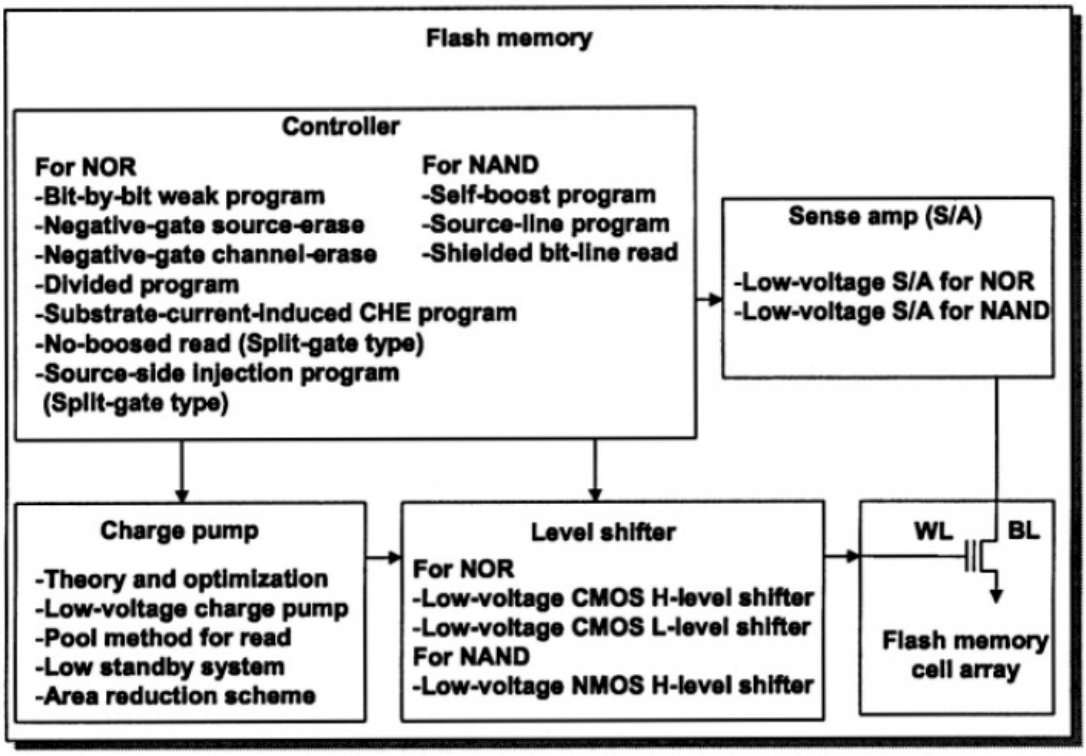
Οι μνήμες NOR flash συνήθως απαιτούν περισσότερο ρεύμα από τη NAND flash κατά την αρχική ενεργοποίηση. Ωστόσο, το ρεύμα αναμονής για τη NOR flash είναι

πολύ χαμηλότερο από της NAND flash. Η ενεργή ισχύς εξαρτάται από τη χρονική διάρκεια για την οποία είναι ενεργή η μνήμη. Η NOR flash διατηρεί ένα πλεονέκτημα όταν πρόκειται για τυχαίες αναγνώσεις, ενώ η NAND flash καταναλώνει συγκριτικά πολύ χαμηλότερη ισχύ για εργασίες διαγραφής, εγγραφής και διαδοχικής ανάγνωσης.

Γενικά, η μνήμη flash NOR είναι μια εξαιρετική επιλογή για εφαρμογές που απαιτούν χαμηλότερη χωρητικότητα, γρήγορη πρόσβαση τυχαίας ανάγνωσης και υψηλότερη αξιοπιστία δεδομένων, όπως απαιτείται για την εκτέλεση κώδικα. Η NAND Flash από την άλλη, είναι ιδανική για εφαρμογές όπως η αποθήκευση δεδομένων όπου απαιτείται μεγαλύτερη χωρητικότητα μνήμης και γρηγορότερες λειτουργίες εγγραφής και διαγραφής.

4.8.5 Τεχνικές χαμηλής κατανάλωσης ισχύος για τις μνήμες flash

Στο Σχήμα 4.15, φαίνονται διάφορα σχήματα ελέγχου και κυκλώματα που είναι ειδικά για μνήμες flash NOR και NAND χαμηλής ισχύος, όπως pump circuits, level shifters, and sense amps.

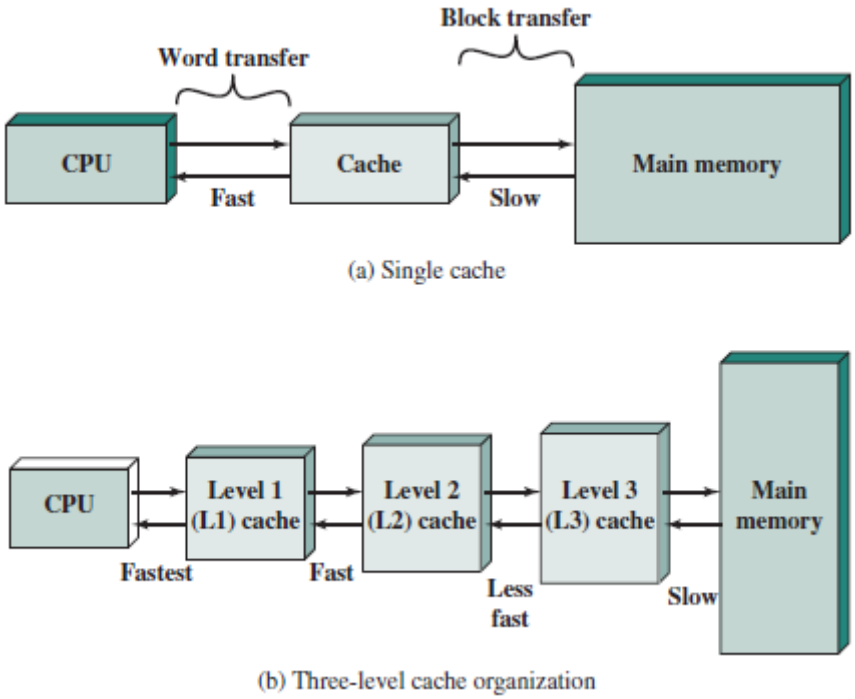


Σχήμα 4.15: Μπλόκ διάγραμμα flash μνήμης και τεχνικές σχεδίασης χαμηλής κατανάλωσης.

4.9 Η CACHE memory

Η μνήμη cache έχει σχεδιαστεί για να συνδυάζει το χρόνο πρόσβασης ακριβών, υψηλής ταχύτητας μνημών, σε συνδυασμό με μεγάλου μεγέθους, λιγότερο δαπανηρών και μικρότερης ταχύτητας μνημών. Η έννοια αυτή απεικονίζεται στο Σχήμα 4.16(α) όπου υπάρχει μία σχετικά μεγάλη και αργή κύρια μνήμη, μαζί με τη μικρότερη και ταχύτερη μνήμη cache. Αναλύοντας τη λειτουργία της cache μνήμης, θα μπορούσε να ονομαστεί και ως προσωρινή ή κρυφή μνήμη. Η προσωρινή αυτή μνήμη, περιέχει αντίγραφο τμημάτων της κύριας μνήμης και όταν ο επεξεργαστής προσπαθεί να διαβάσει μια λέξη από τη μνήμη, γίνεται έλεγχος για να προσδιοριστεί εάν η λέξη αυτή βρίσκεται στην προσωρινή μνήμη. Εάν ναι, η λέξη παραδίδεται στον επεξεργαστή. Εάν όχι, ένα τμήμα της κύριας μνήμης, που αποτελείται από κάποιο σταθερό αριθμό λέξεων, μεταφέρεται στην προσωρινή μνήμη και στη συνέχεια η λέξη παραδίδεται στον επεξεργαστή. Λόγω του φαινομένου locality of reference, όταν ένα σύνολο δεδομένων έχει εισαχθεί στην προσωρινή μνήμη για να ικανοποιήσει μία μόνο αναφορά στη μνήμη, είναι πιθανό ότι θα υπάρξουν μελλοντικές αναφορές σε αυτήν την ίδια θέση μνήμης ή σε άλλες λέξεις στο ίδιο μπλοκ. [4]

Το Σχήμα 4.16(β) απεικονίζει τη χρήση πολλαπλών επιπέδων cache. Η προσωρινή μνήμη L2 είναι πιο αργή και συνήθως μεγαλύτερη από την προσωρινή μνήμη L1, και η προσωρινή μνήμη L3 είναι πιο αργή και τυπικά μεγαλύτερη από την cache L2.



Σχήμα 4.16: Cache και κύρια μνήμη.

4.9.1 Αριθμός caches

Όταν εισήχθησαν αρχικά οι caches, ένα τυπικό σύστημα είχε μία μόνο προσωρινή μνήμη. Περισσότερο πρόσφατα, η χρήση πολλαπλών caches έγινε ο κανόνας. Δύο πτυχές αυτής της σχεδίασης, αφορούν στον αριθμό των επιπέδων των caches και στη χρήση ενοποιημένων έναντι των χωριστών caches.

4.9.2 Multilevel caches

Καθώς η λογική πυκνότητα της σχεδίασης έχει αυξηθεί, έχει γίνει δυνατό να συνυπάρχει cache μνήμη στο ίδιο chip με τον επεξεργαστή: είναι η on-chip cache. Συγκρινόμενη με τη cache η οποία είναι προσβάσιμη μέσω εξωτερικού διαύλου, η on-chip cache, μειώνει τη δραστηριότητα του επεξεργαστή με τον εξωτερικό δίαυλο και ως εκ τούτου, επιταχύνει τους χρόνους εκτέλεσης και αυξάνει συνολικά την απόδοση συστήματος.

Όταν η ζητούμενη οδηγία ή τα δεδομένα βρίσκονται στην on-chip cache, η πρόσβαση στον εξωτερικό δίαυλο καταργείται. Λόγω των σύντομων διαδρομών των δεδομένων εσωτερικά στον επεξεργαστή σε σύγκριση με τις μεγάλες διαδρομές του διαύλου, η πρόσβαση στην on-chip cache θα είναι αισθητά γρηγορότερη. Επιπλέον, κατά τη κατά την περίοδο αυτή ο εξωτερικός δίαυλος είναι ελεύθερος να υποστηρίξει άλλες μεταφορές.

Η προσθήκη μιας on-chip cache, αφήνει ανοικτό το ερώτημα εάν η off-chip ή η εξωτερική cache παραμένει μιά επιθυμητή σχεδιαστική λύση. Συνήθως, η απάντηση είναι ναι και οι περισσότεροι σύγχρονοι σχεδιασμοί περιλαμβάνουν και τις δύο υλοποιήσεις cache. Η απλούστερη υλοποίηση cache, είναι με εσωτερικού επιπέδου 1 (L1-level one cache) και εξωτερική cache που ορίζεται ως επίπεδο 2 (L2-level two cache). Ο λόγος για τη ύπαρξη της μνήμης L2 είναι ο ακόλουθος: εάν δεν υπάρχει προσωρινή μνήμη L2 και ο επεξεργαστής υποβάλει αίτημα πρόσβασης για μια θέση μνήμης που δεν βρίσκεται στη L1 cache, τότε ο επεξεργαστής πρέπει να έχει πρόσβαση στη DRAM ή ROM διαμέσω του διαύλου. Λόγω της τυπικά αργής ταχύτητας διαύλου και του αργού χρόνου πρόσβασης της μνήμης, αυτό έχει ως αποτέλεσμα τη κακή απόδοση.

Από την άλλη πλευρά, εάν χρησιμοποιείται SRAM L2 cache, τότε πολύ συχνά οι πληροφορίες που λείπουν μπορούν να ανακτηθούν γρήγορα. Εάν η SRAM cache είναι αρκετά γρήγορη ώστε να ταιριάζει με την ταχύτητα του διαύλου, τότε η πρόσβαση στα δεδομένα μπορεί να γίνει με zero-wait states, ο οποίος είναι και ο πιο γρήγορος τύπος μεταφοράς διαύλου.

Αξίζει να επισημανθούν δύο χαρακτηριστικά του σύγχρονου σχεδιασμού της cache για cache πολλαπλών επιπέδων. Πρώτον, για off-chip L2 cache, πολλά σχέδια δεν χρησιμοποιούν το δίαυλο του συστήματος ως τη διαδρομή μεταφοράς μεταξύ της cache L2 και του επεξεργαστή, αλλά χρησιμοποιούν μιά ξεχωριστή διαδρομή δεδομένων ώστε να μειωθεί ο φόρτος του διαύλου συστήματος. Δεύτερον, με τη συνεχή μείωση μεγέθους

των εξαρτημάτων του επεξεργαστή, ενσωματώνεται πλέον η L2 cache στο chip του επεξεργαστή, βελτιώνοντας την απόδοση.

Οι πιθανές εξοικονομήσεις λόγω της χρήσης της L2 cache, εξαρτάται από τα ποσοστά επιτυχίας(hit rate) και στις δύο caches L1 και L2. Πολλές μελέτες έχουν δείξει ότι, η χρήση ενός δεύτερου επιπέδου cache βελτιώνει την απόδοση, ωστόσο η χρήση πολυεπίπεδων cache περιπλέκει όλο το σχεδιασμό του συστήματος.

4.9.3 Unified vs Split caches

Όταν η on-chip cache μνήμη εμφανίστηκε για πρώτη φορά, πολλά από τα σχέδια αποτελούνταν από μία μνήμη cache που χρησιμοποιούταν για την αποθήκευση αναφορών τόσο για δεδομένα όσο και για εντολές. Πιο πρόσφατα, έχει γίνει σύνηθες να χωρίζεται η cache μνήμη σε δύο: μία για οδηγίες και μία για δεδομένα. Και οι δύο αυτές μνήμες υπάρχουν στο ίδιο επίπεδο, συνήθως ως δύο L1 caches. Όταν ο επεξεργαστής επιχειρεί να λάβει μια εντολή από την κύρια μνήμη, πρώτα συμβουλεύεται την προσωρινή μνήμη L1 και όταν επιχειρήσει να ανακτήσει δεδομένα από την κύρια μνήμη, πρώτα συμβουλεύεται την προσωρινή μνήμη L1 δεδομένων.

Υπάρχουν δύο πιθανά πλεονεκτήματα μιας ενοποιημένης(unified cache) προσωρινής μνήμης:

■ Για δεδομένο μέγεθος cache, μια ενοποιημένη(unified) προσωρινή μνήμη έχει υψηλότερο ποσοστό επιτυχίας από τις διαιρεμένες(split) caches επειδή εξισορροπεί αυτόματα το φορτίο μεταξύ εντολών και ροών δεδομένων. Δηλαδή, αν ένα μοτίβο εκτέλεσης περιλαμβάνει πολύ περισσότερες εντολές από δεδομένα, τότε η προσωρινή μνήμη θα τείνει να γεμίζει με εντολές, και εάν ένα μοτίβο εκτέλεσης περιλαμβάνει σχετικά περισσότερες λήψεις δεδομένων, θα συμβεί το αντίθετο.

■ Μόνο μία cache χρειάζεται να σχεδιαστεί και να υλοποιηθεί.

Η τάση είναι προς διαιρεμένες(split) caches στο L1 και ενοποιημένες caches για τα υψηλότερα επίπεδα, ιδιαίτερα για συστήματα τα οποία δίνουν έμφαση στην παράλληλη εκτέλεση εντολών.

4.9.4 Μείωση της κατανάλωσης ισχύος στις μνήμες cache

Ο επεξεργαστής είναι ένας ιδιαίτερα σημαντικός στόχος στις σχεδιάσεις χαμηλής κατανάλωσης ισχύος. Οι επεξεργαστές υψηλής απόδοσης συνήθως περιλαμβάνουν on-chip και off-chip caches. Οι on-chip caches, καταλαμβάνουν περίπου μέχρι και το 40% της επιφάνειας του chip και αποτελούν μία από τις μεγάλες καταναλώσεις του μικροεπεξεργαστή. Επομένως, είναι πολύ σημαντικό να μειωθεί η κατανάλωση ενέργειας στην προσωρινή μνήμη των cpu.

Όπως προαναφέρθηκε, η μνήμη cache είναι μνήμη τύπου SRAM, οπότε οι τεχνικές μείωσης της κατανάλωσης ισχύος , εμπίπτουν σε αυτές της μνήμης SRAM. Επιπρόσθετα όμως, υπάρχουν τεχνικές μείωσης της κατανάλωσης που αφορούν

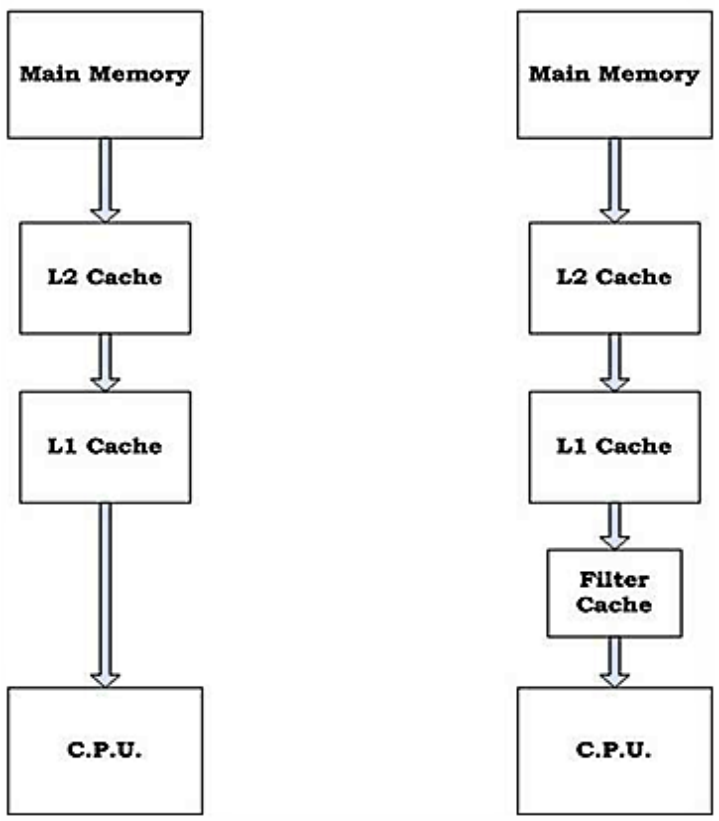
αποκλειστικά τις μνήμες cache όπως για παράδειγμα η filter cache αρχιτεκτονική, η cache decay τεχνική, η μείωση της snoop energy, η χρήση μίας ενιαίας multi – ported cache μνήμης, η χρήση cache μνήμης για κάθε επεξεργαστή ξεχωριστά και η τεχνική Crossbar Connected Cache (CCC). [39]

Η filter cache αρχιτεκτονική

Ο σχεδιασμός της filter cache αρχιτεκτονικής, βασίζεται στο γεγονός ότι η μείωση της κατανάλωσης ενέργειας θα αντισταθμίσει την απώλεια απόδοσης . Η filter cache αντιμετωπίζει το πρόβλημα της μεγάλης κατανάλωσης ενέργειας των L1 caches με την προσθήκη μιας μικρής και επομένως χαμηλής κατανάλωσης μνήμης cache, μεταξύ της CPU και της L1 cache. [40]

Η μικρή αυτή L1 cache μνήμη ονομάζεται filter cache, έτσι ώστε να διαχωρίζεται από τις παραδοσιακές caches που είναι σχεδιασμένες αποκλειστικά για υψηλή απόδοση. Ωστόσο, ενώ η σχεδίαση διαφέρει από εκείνες μιας παραδοσιακής cache, η βασική δομή είναι η ίδια.

Η βασική οργάνωση της filter cache απεικονίζεται στο Σχήμα 4.17, όπου συγκρίνεται με μια παραδοσιακή cache υλοποίηση. Η L1 cache είναι πιθανό να έχει τα ίδια χαρακτηριστικά για τις δύο υλοποιήσεις, ωστόσο με τη νέα σχεδίαση η L1 cache είναι προσβάσιμη μόνο εφόσον δεν προσπελαστεί πρώτα η filter cache, αλλιώς παραμένει σε κατάσταση αναμονής. Επειδή η filter cache είναι μικρότερη από την L1 cache, γενικά θα έχει γρηγορότερο χρόνο προσπέλασης και ενώ αυτό το φαινόμενο μπορεί να αποτελέσει ευκαιρία για αύξηση του ρολογιού του επεξεργαστή, θα οδηγήσει αναγκαστικά σε ένα αύξηση της καθυστέρησης πρόσβασης στην L1 cache. [41]



Σχήμα 4.17: Τυπική 2-level cache και filter cache αρχιτεκτονικές.

Η cache decay τεχνική

Η cache decay τεχνική, χρησιμοποιείται για τη μείωση των ρευμάτων διαρροής. Σε αυτή τη τεχνική, απενεργοποιείται η cache μνήμη όταν διατηρεί δεδομένα που δεν είναι πιθανό να χρησιμοποιηθούν ξανά. Συγκεκριμένα στη cache μνήμη, γίνεται μιά εκτεταμένη χρήση όταν εισάγονται τα δεδομένα για πρώτη φορά και έπειτα ακολουθεί μια περίοδος νεκρού χρόνου μεταξύ της τελευταίας πρόσβασής τους και όταν νέα στοιχεία δεδομένων εισάγονται ξανά.

Η απενεργοποίηση της cache μνήμης κατά τη διάρκεια αυτής της νεκρής περιόδου, μειώνει τα leakage ρεύματα. Με την επινόηση αποτελεσματικών και χαμηλής κατανάλωσης τρόπων μείωσης του νεκρού χρόνου, σε πολλές περιπτώσεις μπορούμε να μειώσουμε την κατανάλωση ισχύος λόγω διαρροής της L1 cache κατά τέσσερις φορές. [12]

Μείωση snoop energy

Στα πολυπεξεργαστικά συστήματα, οι ελεγκτές μεταξύ των κρυφών μνημών και των διαύλων, παρακολουθούν τους διαύλους (bus snooping) προκειμένου να εξασφαλίζεται η συνέπεια των δεδομένων (coherence). Η δραστηριότητα αυτή συμμετέχει στις

ενεργειακές απαιτήσεις ενός συστήματος μνήμης και στη βιβλιογραφία αναφέρεται σαν snoop energy.

Η σημαντικότερη τεχνική που προτείνεται για τη μείωση της snoop δραστηριότητας είναι η εισαγωγή ενός module ανάμεσα στην cache μνήμη και τον δίαυλο που ονομάστηκε JETTY. Το JETTY λειτουργεί σαν φίλτρο και ελέγχει ποια αιτήματα απευθύνονται στην cache μνήμη με την οποία είναι συνδεδεμένο. Αν το αίτημα δεν αφορά στη συγκεκριμένη μνήμη, τότε δεν γίνεται αποδεκτό. Αυτό έχει σαν αποτέλεσμα τη μείωση της δραστηριότητας της κάθε cache μνήμης κατά 29% περίπου και τη μείωση της συνολικής bus snooping δραστηριότητας γενικά κατά 74% κατά μέσο όρο. [15]

Χρήση μίας ενιαίας multi – ported cache μνήμης

Η χρήση μίας ενιαίας multi – ported cache μνήμης, χρησιμοποιείται από όλους τους επεξεργαστές που μπορεί να υπάρχουν στο ολοκληρωμένο (Σχήμα 4.18(α)). Τα πλεονεκτήματα στη συγκεκριμένη περίπτωση είναι η εύκολη υλοποίηση της επικοινωνίας μεταξύ των επεξεργαστών και η μείωση του ρυθμού αποτυχιών (miss rate). Από την άλλη, η χρήση ενιαίας cache μνήμης παρουσιάζει μεγάλη κατανάλωση και πολύ μειωμένη δυνατότητα κλιμάκωσης (scalability).

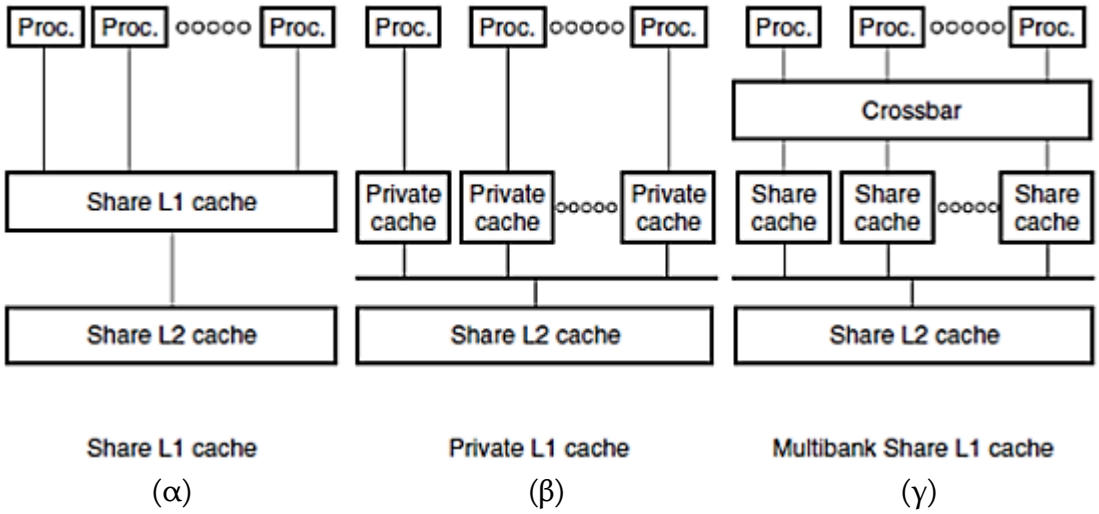
Χρήση cache μνήμης για κάθε επεξεργαστή ξεχωριστά

Η χρήση κρυφής μνήμης για κάθε επεξεργαστή ξεχωριστά (Σχήμα 4.18(β)), έχει για κυριότερα πλεονεκτήματα αυτά της εύκολης κλιμάκωσης (scalability), των μικρότερων χρόνων πρόσβασης στη κάθε μνήμη και των μικρότερων ενεργειακών απαιτήσεων ανά πρόσβαση (μιας και ο φόρτος επιμερίζεται). Ένα σημαντικό μειονέκτημα, ωστόσο, της οργάνωσης αυτής είναι η ανάγκη οι εντολές και τα δεδομένα να χρειάζεται πολλές φορές να αντιγράφονται σε περισσότερες από δύο επιμέρους κρυφές μνήμες. Επιπλέον, απαιτείται η ύπαρξη μηχανισμού για τον έλεγχο των τμημάτων και για τη διατήρηση της ομαλής λειτουργίας του συστήματος.

Η τεχνική Crossbar Connected Cache (CCC)

Η τεχνική με το όνομα Crossbar Connected Cache (CCC), στόχο έχει να συνδυάσει τα πλεονεκτήματα των δύο προηγούμενων τεχνικών, αμβλύνοντας παράλληλα τα μειονεκτήματα που παρουσιάζουν. Μία ενιαία κρυφή μνήμη χωρίζεται εσωτερικά σε πολλαπλά τμήματα (banks), χρησιμοποιώντας ένα crossbar nxm, όπου n ο αριθμός των επεξεργαστών και m ο αριθμός των banks (Σχήμα 4.18(γ)). Με αυτό τον τρόπο, εξασφαλίζεται η δυνατότητα κλιμάκωσης (scalability), δεν απαιτείται πλέον η αντιγραφή εντολών και δεδομένων σε πολλά διαφορετικά τμήματα και τέλος,

καταργείται η ανάγκη ύπαρξης ελεγκτικού μηχανισμού. Τα παραπάνω είναι ιδιαίτερα χρήσιμα, μιας και αποτελούν μια βάση που, σε συνδυασμό με άλλες τεχνικές, μπορούν να οδηγήσουν σε σημαντικά μικρότερες ενεργειακές απαιτήσεις. [15]



Σχήμα 4.18: Τρεις διαφορετικές τακτικές οργάνωσης on-chip cache μνήμης.

4.10 Μείωση της κατανάλωσης ισχύος με τη χρήση Direct Memory Access (DMA)

4.10.1 Εισαγωγή

Η λειτουργία της απευθείας πρόσβασης στη μνήμη (DMA-Direct Memory Access), παρέχει έναν αποτελεσματικό τρόπο μεταφοράς δεδομένων, χωρίς να παρεμβάλλεται η CPU και απαιτεί την ύπαρξη μίας μονάδας ελεγκτή DMA (DMA controller) στο σύστημα. Σε εφαρμογές περιορισμένων πόρων, η μονάδα DMA μπορεί να επιτρέψει στην CPU να χειρίζεται μια σημαντική εργασία, ενώ ταυτόχρονα διαβάζει για παράδειγμα τα δεδομένα από ένα I/O port και να τα τοποθετεί στη μνήμη για μεταγενέστερη επεξεργασία. Για έναν προγραμματιστή που ενδιαφέρεται να ελαχιστοποιήσει την κατανάλωση ενέργειας, η CPU μπορεί να τεθεί σε λειτουργία χαμηλής κατανάλωσης, επιτρέποντας στην DMA να μεταφέρει δεδομένα σε πολύ χαμηλότερο επίπεδο ενέργειας και έπειτα να επαναφέρει την CPU σε ενεργή κατάσταση για να τα επεξεργαστεί. [35]

Αυτή η μονάδα (Σχήμα 4.19), είναι ικανή να μιμείται τον επεξεργαστή και να αναλαμβάνει τον έλεγχο του συστήματος από αυτόν, έτσι ώστε να μεταφέρει δεδομένα από και προς τη μνήμη διαμέσω του διαύλου συστήματος. Για το σκοπό αυτό, η μονάδα DMA πρέπει να χρησιμοποιεί το δίαυλο μόνο όταν ο επεξεργαστής δεν τον

χρειάζεται ή πρέπει να αναγκάσει τον επεξεργαστή να αναστείλει τη λειτουργία του προσωρινά. Η τεχνική αυτή είναι πιο συνήθης και αναφέρεται ως κλοπή κύκλου(cycle stealing), διότι η μονάδα DMA στην πραγματικότητα κλέβει έναν κύκλο διαύλου.

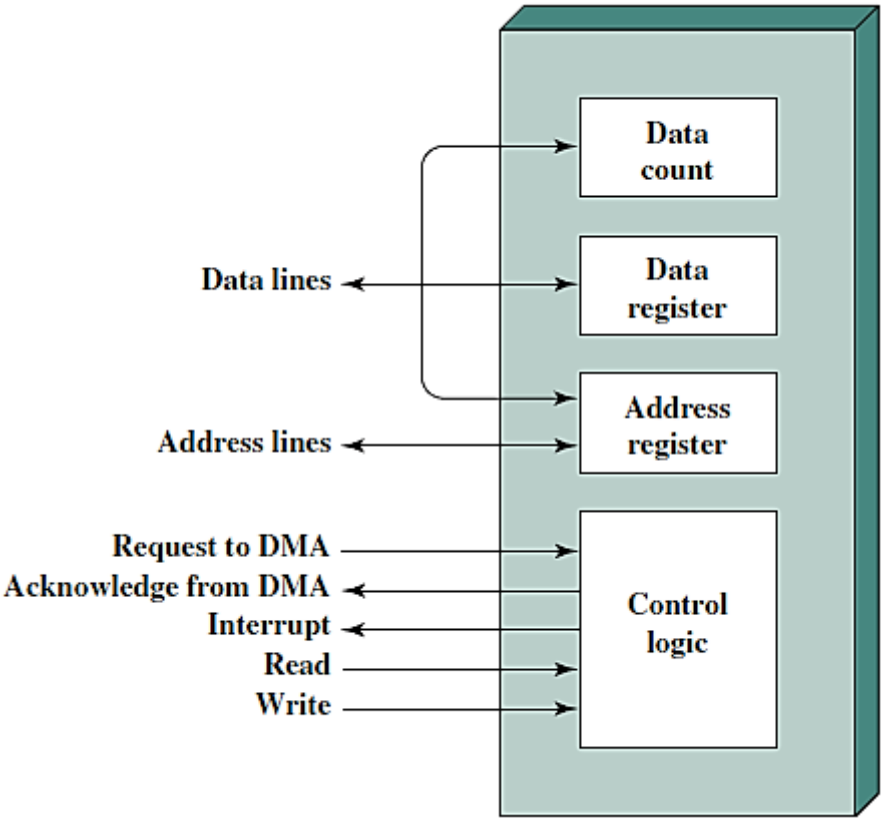
Όταν ο επεξεργαστής επιθυμεί να διαβάσει ή να γράψει ένα μπλοκ δεδομένων, δίνει μία εντολή στη μονάδα DMA, αποστέλλοντας σε αυτή τις ακόλουθες πληροφορίες:

■ Αν απαιτείται ανάγνωση ή εγγραφή, με χρήση της γραμμής ελέγχου ανάγνωσης ή εγγραφής μεταξύ του επεξεργαστή και της μονάδας DMA.

■ Τη διεύθυνση της σχετικής συσκευής I/O, η οποία κοινοποιείται στις γραμμές δεδομένων.

■ Την αρχική θέση στη μνήμη για ανάγνωση ή εγγραφή, που κοινοποιείται στις γραμμές διευθύνσεων και αποθηκεύονται από την μονάδα DMA στον καταχωρητή διευθύνσεών της.

■ Τον αριθμό των λέξεων που θα διαβαστούν ή θα γραφούν και θα κοινοποιηθούν ξανά μέσω των γραμμών δεδομένων και αποθηκευτούν στον καταχωρητή καταμέτρησης δεδομένων.



Σχήμα 4.19: DMA μπλόκ διάγραμμα.

Στη συνέχεια, ο επεξεργαστής συνεχίζει με άλλες εργασίες αφού έχει αναθέσει αυτή τη λειτουργία εισόδου/εξόδου στη μονάδα DMA. Η μονάδα DMA μεταφέρει ολόκληρο το μπλοκ δεδομένων, μία λέξη τη φορά, απευθείας από ή προς τη μνήμη, χωρίς να περνά από τον επεξεργαστή.

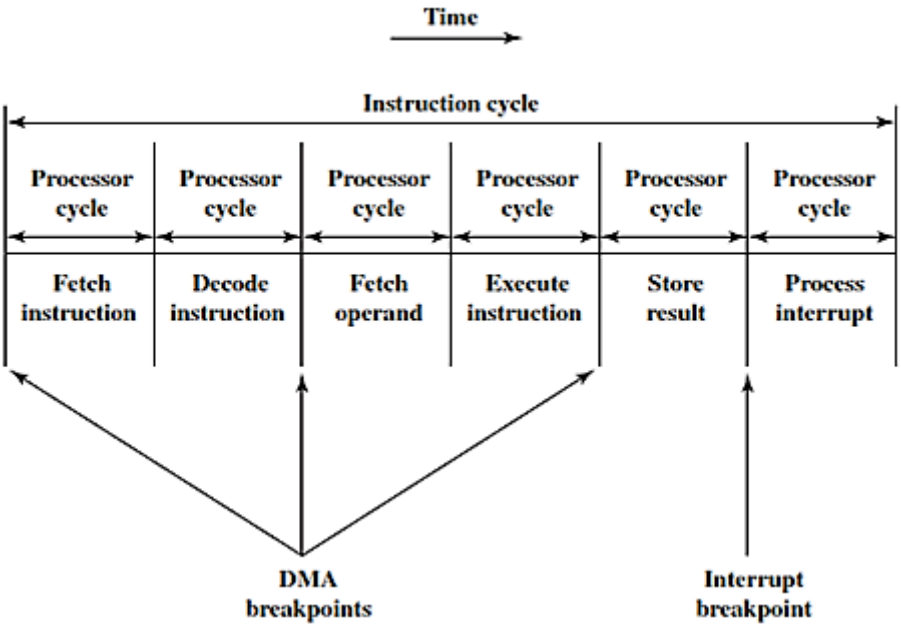
Όταν ολοκληρωθεί η μεταφορά, η μονάδα DMA στέλνει ένα σήμα διακοπής στον επεξεργαστή. 'Ετσι, ο επεξεργαστής συμμετέχει μόνο στην αρχή και στο τέλος της μεταφοράς.

Το σχήμα 4.20 δείχνει σε ποιό σημείο στον κύκλο λειτουργίας, μπορεί να ανασταλεί ο επεξεργαστής. Σε κάθε περίπτωση, ο επεξεργαστής αναστέλλεται αμέσως πριν χρειαστεί να χρησιμοποιηθεί ο δίαυλος. Η μονάδα DMA στη συνέχεια μεταφέρει μία λέξη και επιστρέφει τον έλεγχο στον επεξεργαστή. Να σημειωθεί ότι δεν πρόκειται για διακοπή. Ο επεξεργαστής δεν αποθηκεύει τα δεδομένα και δεν κάνει κάτι άλλο. Αντίθετα, διακόπτει προσωρινά τη λειτουργία του για έναν κύκλο διαύλου. Το αποτέλεσμα είναι να προκαλείται η πιο αργή λειτουργία του επεξεργαστή. Ωστόσο, για τη μεταφορά πολλαπλών λέξεων I/O, η DMA είναι πολύ πιο αποτελεσματική από ότι με interrupts.

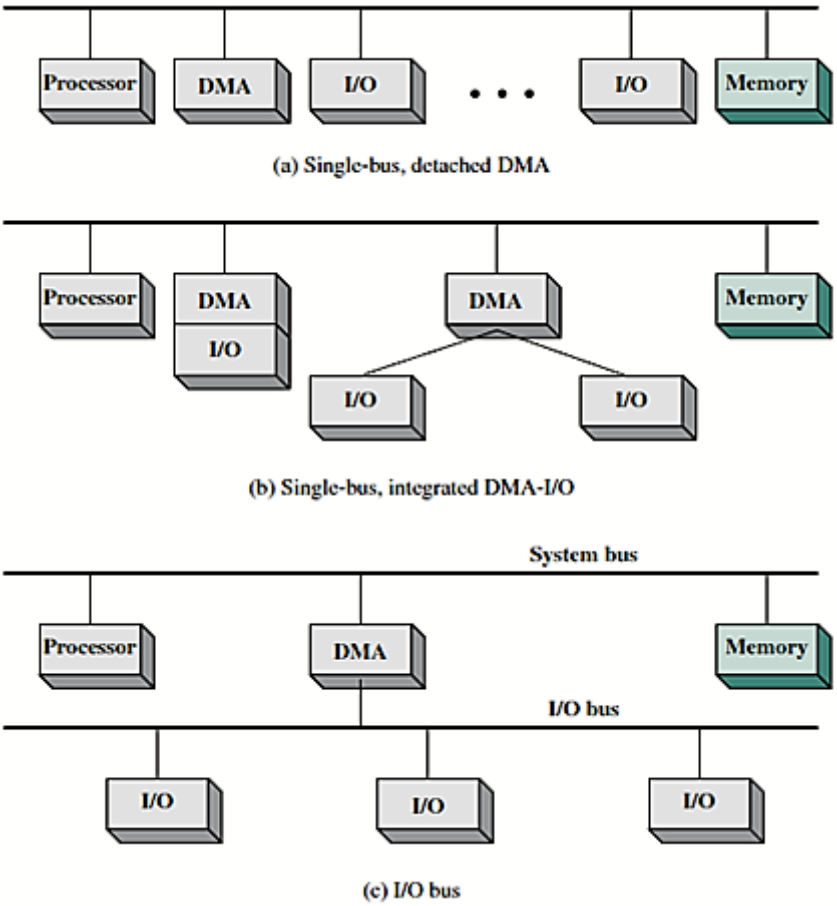
Ο μηχανισμός DMA μπορεί να υλοποιηθεί με διάφορους τρόπους. Ορισμένες δυνατότητες παρουσιάζονται στο σχήμα 4.21. Στο πρώτο παράδειγμα(Σχήμα 4.21(a)), όλες οι λειτουργικές μονάδες μοιράζονται τον ίδιο δίαυλο συστήματος. Η μονάδα DMA, ενεργώντας ως υποκατάστατος επεξεργαστής, χρησιμοποιεί προγραμματιζόμενη I/O για να ανταλλαγή δεδομένων μεταξύ μνήμης και της μονάδας I/O μέσω της μονάδας DMA.

Αυτή η υλοποίηση, αν και μπορεί να είναι οικονομική, είναι σαφώς αναποτελεσματική. Όπως και με την ελεγχόμενη από τον επεξεργαστή προγραμματιζόμενη I/O, κάθε μεταφορά μιας λέξης καταναλώνει δύο κύκλους διαύλου. [4]

Ο αριθμός των απαιτούμενων κύκλων διαύλου, μπορεί να μειωθεί σημαντικά με την ενσωμάτωση της DMA και των λειτουργιών I/O. Όπως δείχνει το σχήμα 4.21(b), αυτό σημαίνει ότι υπάρχει μία διαδρομή μεταξύ της μονάδας DMA και μίας ή περισσότερων μονάδων I/O που δεν περιλαμβάνουν τον δίαυλο του συστήματος.



Σχήμα 4.20: DMA και interrupt breakpoints στη διάρκεια ενός κύκλου εντολής.



Σχήμα 4.21: Εναλλακτικές DMA υλοποιήσεις.

Η λογική DMA μπορεί να είναι μέρος μιας μονάδας I/O ή μπορεί να είναι ξεχωριστή μονάδα που ελέγχει μία ή περισσότερες μονάδες I/O. Η έννοια αυτή μπορεί να πάει ένα βήμα παραπέρα, συνδέοντας τις μονάδες I/O στη μονάδα DMA χρησιμοποιώντας έναν δίαυλο I/O (σχήμα 4.21.(c)). Έτσι μειώνεται ο αριθμός των διασυνδέσεων I/O στη DMA μονάδα σε μία και παρέχει μιά υλοποίηση με δυνατότητα εύκολης επέκτασης.

Και στις δύο αυτές περιπτώσεις (Σχήματα 4.21.(b) και 4.21(c)), ο δίαυλος συστήματος τον οποίο η μονάδα DMA μοιράζεται με τον επεξεργαστή και τη μνήμη, χρησιμοποιείται από τη μονάδα DMA μόνο για την ανταλλαγή δεδομένων με μνήμη. Η ανταλλαγή δεδομένων μεταξύ της μονάδας DMA και I/O πραγματοποιείται εκτός του διαύλου συστήματος. [4]

4.10.2 DMA και χαμηλή κατανάλωση ισχύος

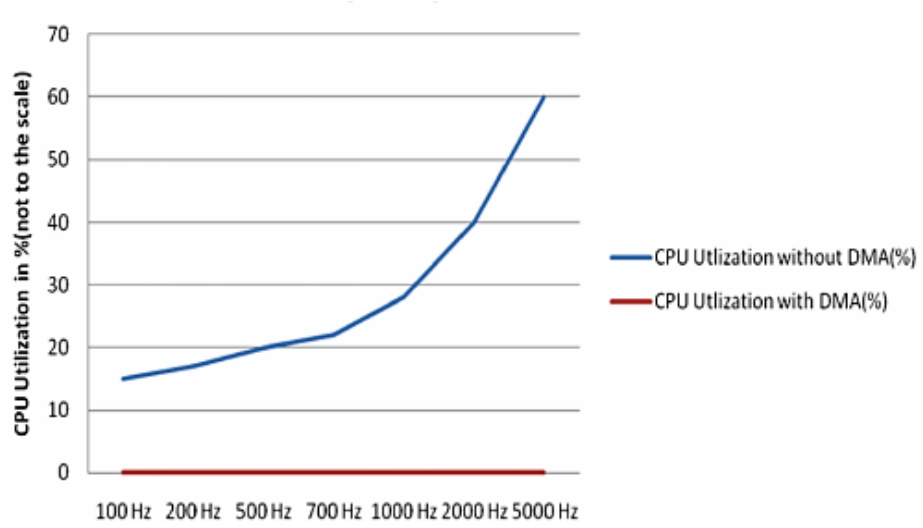
Όπως αναφέρθηκε, οι μεταφορές δεδομένων ενδέχεται να καταλαμβάνουν ένα μεγάλο ποσοστό χρόνου χρήσης της CPU σε ένα σύστημα, οπότε για να μειωθεί αυτός ο χρόνος, πολλά συστήματα περιλαμβάνουν μία μονάδα DMA. Αυτή η μονάδα, μπορεί να κάνει τη μεταφορά των δεδομένων χωρίς να παρεμβάλλεται η CPU, αφήνοντάς την να χειρίζεται όλες τις άλλες εργασίες. Ο ελεγκτής DMA μπορεί να χρησιμοποιηθεί για τους ακόλουθους τύπους μεταφοράς δεδομένων, με τη διαθεσιμότητα να διαφέρει από ελεγκτή σε ελεγκτή:

- Μεταφορά δεδομένων από και προς τη μνήμη
- Μεταφορά δεδομένων από τη μνήμη προς τις περιφερειακές μονάδες
- Μεταφορά δεδομένων από τις περιφερειακές μονάδες προς τη μνήμη
- Μεταφορά δεδομένων από και προς τις περιφερειακές μονάδες

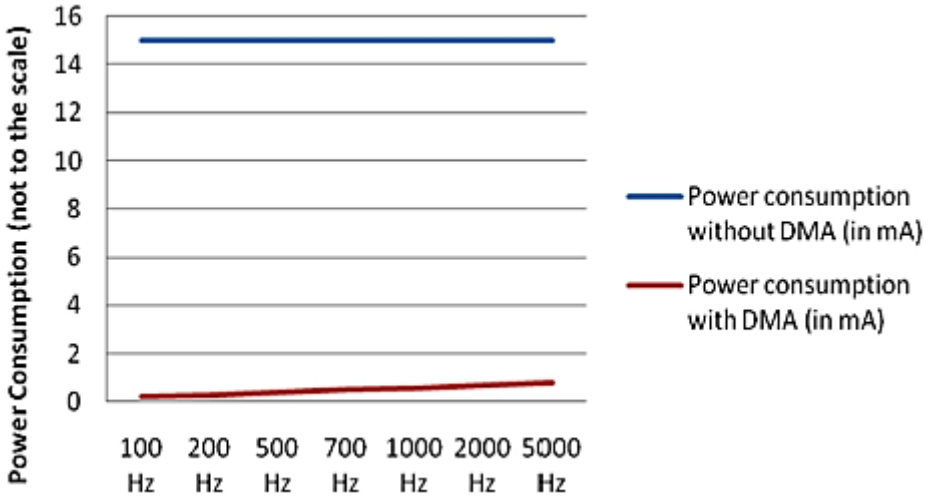
4.10.3 Πλεονεκτήματα της χρήσης DMA

i. Όσο περισσότερο η CPU χρησιμοποιείται, τόσο περισσότερη ενέργεια καταναλώνεται. Η χρήση του DMA ελεγκτή για τη μείωση του φόρτου επεξεργασίας της CPU, μειώνει την ισχύ που καταναλώνεται.
ii. Ο DMA ελεγκτής, λειτουργεί παράλληλα με την CPU προσομοιώνοντας έτσι ένα περιβάλλον πολλαπλών επεξεργασιών, αυξάνοντας αποτελεσματικά την απόδοση της CPU.
iii. Η μείωση του φόρτου επεξεργασίας της CPU, έχει ως αποτέλεσμα περισσότερο χρόνο αδράνειας(idle time) για τη CPU με αποτέλεσμα να αυξάνει τη διαθέσιμη ικανότητα επεξεργασίας.

Στα Σχήματα 4.22 και 4.23, φαίνεται η χρήση της CPU (CPU Utilization) και η κατανάλωση ισχύος αντίστοιχα, σε μία εφαρμογή ενός συστήματος SoC που παράγει κυματομορφές συχνοτήτων. [43]



Σχήμα 4.22: Χρήση(%) της CPU με και χωρίς χρήση του DMA.



Σχήμα 4.23: Κατανάλωση ισχύος της CPU με και χωρίς χρήση του DMA.

Αν κοιτάξουμε την κατανάλωση ισχύος, γενικά η CPU καταναλώνει περισσότερη ισχύ σε σύγκριση με έναν ελεγκτή DMA. Όταν η CPU όμως δεν χρησιμοποιείται, μπορεί να απενεργοποιηθεί εξοικονομώντας τη στατική και την δυναμική ισχύ που καταναλώνει. Έτσι, το DMA βοηθά στη βελτιστοποίηση της κατανάλωσης ισχύος του συστήματος όπου το μεγαλύτερο μέρος της εργασίας είναι η μεταφορά δεδομένων. [43]

ΒΙΒΛΙΟΓΡΑΦΙΑ 4ου ΚΕΦΑΛΑΙΟΥ

[1] CMOS DIGITAL INTERGRATED CIRCUITS, ANALYSIS AND DESIGN, Sung-Mo (Steve) Kang, University of Illinois at Urbana- Champaign, Yusuf Leblebigi, Worcester Polytechnic Institute, Swiss Federal Institute of Technology-Lausanne, McGraw-Hill Education, 3rd Edition, 2002.

[2] ΚΥΚΛΩΜΑΤΑ CMOS ΧΑΜΗΛΗΣ ΚΑΤΑΝΑΛΩΣΗΣ ΙΣΧΥΟΣ ΚΑΙ ΧΑΜΗΛΗΣ ΤΑΣΗΣ ΤΡΟΦΟΔΟΣΙΑΣ, Διδακτορική Διατριβή, Ιωάννης Χ. Μωυσιάδης, ΕΘΝΙΚΟ & ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ, Σχολή Θετικών Επιστημών Τμήμα Πληροφορικής, Αθήνα 2000.

[3] COMPUTER ORGANIZATION AND ARCHITECTURE DESIGNING FOR PERFORMANCE, Tenth Edition, William Stallings, With contribution by Peter Zeno, University of Bridgeport, Copyright © 2016, 2013, 2010 Pearson Education, Inc., Hoboken, NJ 07030.

[4] COMPUTER ARCHITECTURE TECHNIQUES FOR POWER-EFFICIENCY, Stefanos Kaxiras, University of Patras, Greece and Margaret Martonosi, Princeton University. A Publication in the Morgan & Claypool Publishers series. Copyright © 2008 by Morgan & Claypool

[5] ΛΟΓΑΡΙΘΜΙΚΟ ΑΡΙΘΜΗΤΙΚΟ ΣΥΣΤΗΜΑ ΚΑΙ ΤΕΧΝΙΚΕΣ ΧΑΜΗΛΗΣ ΚΑΤΑΝΑΛΩΣΗΣ ΜΕ ΕΦΑΡΜΟΓΗ ΣΤΗΝ ΨΗΦΙΑΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΣΗΜΑΤΩΝ, Μεταπτυχιακή εργασία, Κασιώλας Βασίλειος, Πανεπιστήμιο Πατρών, Πολυτεχνική Σχολή, Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών & Πληροφορικής, Πρόγραμμα Μεταπτυχιακών Σπουδών στα Ολοκληρωμένα Συστήματα Υλικού & Λογισμικού. Πατρα 2014

[6] DECREASING ENERGY CONSUMPTION USING DIRECT MEMORY ACCESS, JACOB BENINGO, https://www.embedded.com, January 23, 2018

[7] PRACTICAL LOW POWER DIGITAL VLSI DESIGN by Gary Yeap, Copyright © 1998 by Springer Science+Business Media New York

[8] LOW-POWER SRAM CIRCUIT DESIGN Martin Margala, Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Alberta, Canada T6G 2G7

[9] TRENDS IN LOW-POWER RAM CIRCUIT TECHNOLOGIES, K. Itoh, K. Sasaki and Y. Nakagome, Proc IEEE.

[10] POWER CONSIDERATIONS FOR MEMORY - RELATED MICROARCHITECTURE DESIGNS, Zhichun Zhu, College of William & Mary - Arts & Sciences, 2003

[11] FILTER CACHE: FILTERING USELESS CACHE BLOCKS FOR A SMALL BUT EFFICIENT SHARED LAST-LEVEL CACHE, Han Jun Bae1 · Lynn Choi1 © Springer Science+Business Media, LLC, part of Springer Nature 2020

[12] THE FILTER CACHE: AN ENERGY EFFICIENT MEMORY STRUCTURE, Johnson Kin, Munish Gupta and William H. Mangione-Smith, The Department of Electrical Engineering, UCLA Electrical Engineering, 1997

[13] LOW POWER MEMORY DESIGN, Chapter 3, Yukihito Oowaki and Tohru Tanzawa, SoC R&D Center, Toshiba Corp., 2007

[14] OPTIMIZING EMBEDDED APPLICATIONS USING DMA, by Sachin Gupta, Applications Engineer Sr and Lakshmi Natarajan, Applications Engineer Sr, Cypress Semiconductor Corp, November 2010.

ΚΕΦΑΛΑΙΟ 5:
Τεχνικές χαμηλής κατανάλωσης ισχύος σε επίπεδο software

5.1 Εισαγωγή

Στα προϊόντα και τις υπηρεσίες του τομέα της πληροφορικής, η αυξανόμενη ζήτηση για υπολογιστική ισχύ συνδέεται στενά με την όλο και μεγαλύτερη κατανάλωση ενέργειας. Αυτή η κατανάλωση η οποία αυξάνεται ταχύτερα από ότι είχε αρχικά προβλεφθεί, αναμένεται να φτάσει το 15% της συνολικής κατανάλωσης ενέργειας του κόσμου έως το 2020. Παραδοσιακά, η έρευνα για την ενεργειακή απόδοση έχει επικεντρωθεί στη μείωση της κατανάλωσης ενέργειας σε επίπεδο υλικού. Όμως στους μικροεπεξεργαστές και τους μικροελεγκτές, είναι το λογισμικό που κατευθύνει μεγάλο μέρος της δραστηριότητας του υλικού και παρέχει πολλές ευκαιρίες για βελτίωση της ενεργειακής απόδοσης. [18], [19]

Η κατανάλωση ενέργειας λογισμικού, αποτελεί έναν πρόσφατο τομέα εντατικής έρευνας όπου αναπτύσσονται διάφορες τεχνικές ανάλυσης και βελτιστοποίησης. Τέτοιες τεχνικές παρέχουν ήδη γνώσεις σχετικά με την ενεργειακή απόδοση των δομών δεδομένων, τον ενεργειακό αντίκτυπο διαφορετικών πρακτικών προγραμματισμού τόσο σε κινητές όσο και σε desktop εφαρμογές, ακόμα και τον τρόπο πρόβλεψης της κατανάλωσης ενέργειας σε διάφορα συστήματα λογισμικού. Ένα ενδιαφέρον ερώτημα που προκύπτει συχνά στον τομέα της ενεργειακής απόδοσης του λογισμικού είναι εάν ένα ταχύτερο πρόγραμμα, είναι επίσης ενεργειακά αποδοτικότερο ή όχι. Εάν η απάντηση είναι ναι, τότε η βελτιστοποίηση ενός προγράμματος για ταχύτητα σημαίνει επίσης τη βελτιστοποίηση του για ενέργεια και αυτό ακριβώς κάνει η κοινότητα δημιουργίας μεταγλωττιστών από την απαρχή των γλωσσών προγραμματισμού. Ωστόσο, η κατανάλωση ενέργειας δεν εξαρτάται μόνο από το χρόνο εκτέλεσης, όπως φαίνεται στην εξίσωση Ενέργεια = Χρόνος × Ισχύς. Στην πραγματικότητα, υπάρχουν πολλά ερευνητικά έργα που δείχνουν διαφορετικά αποτελέσματα σχετικά με αυτό το θέμα. [2]

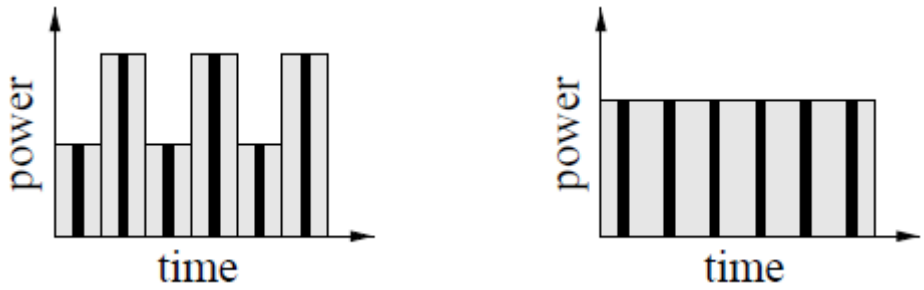
Η σύνταξη βιώσιμου, αποδοτικού και πράσινου λογισμικού(green software), απαιτεί την κατανόηση της συμπεριφοράς κατανάλωσης ενέργειας ενός προγράμματος υπολογιστή και ένα από τα οφέλη είναι το γεγονός, ότι οι προγραμματιστές βελτιώνοντας τις υλοποιήσεις του πηγαίου κώδικα τους, μπορούν να βελτιστοποιήσουν την κατανάλωση ενέργειας ενός λογισμικού. [6], [8]

5.2 Power vs Energy

Η βελτιστοποίηση για την ελάχιστη κατανάλωση ισχύος ή την ελάχιστη ενεργειακή χρήση, μπορεί να έχει διαφορετικές μετρήσεις και ως εκ τούτου να οδηγήσει σε διαφορετικές στρατηγικές βελτιστοποίησης. Μια πιθανή μετρική για την ισχύ και την

ενέργεια, είναι αυτή του επιπέδου δραστηριότητας σε οποιοδήποτε δεδομένο σημείο κατά την εκτέλεση του προγράμματος και το συνολικό ποσό δραστηριότητας για ένα τμήμα του προγράμματος. Όσο περισσότερη επεξεργασία γίνεται σε ένα σημείο προγράμματος, τόσο περισσότερη ισχύς καταναλώνεται. Δεδομένων αυτών των μετρήσεων, η βελτιστοποίηση για την ισχύ είναι η ίδια με τη βελτιστοποίηση για ενέργεια; Η απάντηση θα εξαρτηθεί από την ιδιαίτερη κατάσταση της επεξεργασίας.

Ένας μεταγλωττιστής βελτιστοποίησης (optimizing compiler), μπορεί να ορίσει σαν επεξεργασία, τον αριθμό των εντολών που εκτελούνται σε ένα δεδομένο χρονικό σημείο. Αυτό το μοντέλο υποθέτει ότι, (1) μια καθορισμένη ποσότητα ισχύος συσχετίζεται με κάθε εκτελεσθείσα εντολή και ότι (2) η κατανάλωση ισχύος μιας εντολής, είναι ανεξάρτητη από τις συγκεκριμένες τιμές τελεστή ή άλλες εντολές που εκτελούνται. Το σχήμα 5.1 απεικονίζει αυτή την περίπτωση. Με αναδιάταξη ή χρονοπρογραμματισμό εντολών, ένα τμήμα προγράμματος όπως φαίνεται στα αριστερά του σχήματος 5.1, μπορεί ιδανικά να μετατραπεί σε αυτό που φαίνεται στα δεξιά. Ενώ η μέγιστη κατανάλωση ισχύος είναι διαφορετική και για τα δύο σχεδιαγράμματα, η ενεργειακή χρήση είναι η ίδια. Με άλλα λόγια, η δραστηριότητα ή ο χρονοπρογραμματισμός της εργασίας, μπορεί να είναι ένας αποτελεσματικός τρόπος για να μειωθεί η μέγιστη κατανάλωση ισχύος ενώ δεν έχει καμία επίδραση στη χρήση της ενέργειας. Επομένως, η μείωση μέγιστης ισχύος μπορεί να είναι ένας στόχος βελτιστοποίησης διαφορετικός από τη μείωση της ενέργειας.



Σχήμα 5.1: Βελτιστοποίηση ισχύος vs βελτιστοποίηση ενέργειας

Για μοντέλα ισχύος που βασίζονται σε δραστηριότητες μεταγωγής σε επίπεδο δυαδικών ψηφίων ως έννοια επεξεργασίας, ο χρονοπρογραμματισμός των εντολών μπορεί να στοχεύει στη συνολική χρήση ενέργειας, ομαδοποιώντας τις εντολές με βάση συγκεκριμένα μοτίβα των δυαδικών ψηφίων τους. Εκτός από τον χρονοπρογραμματισμό εντολών, μια προσεκτική επιλογή της ονοματοδοσίας των καταχωρητών στη φάση της παραγωγής του κώδικα ενός μεταγλωττιστή, μπορεί να οδηγήσει σε ακολουθίες κώδικα που έχουν μοτίβα δυαδικών ψηφίων με λιγότερες μεταβάσεις, για παράδειγμα λόγω της επαναχρησιμοποίησης καταχωρητών με παρόμοια ονόματα.

5.3 Power/Energy vs Performance

Αρχικά, οι βελτιστοποιήσεις για μεγαλύτερη απόδοση επισκίαζαν αυτές για τη κατανάλωση της ισχύος και τη διαχείριση της ενέργειας. Επομένως, η ισχύς/ενέργεια δεν αποτελούσε σκοπό της βελτιστοποίησης. Παραδοσιακές βελτιστοποιήσεις όπως η αφαίρεση πλεονάζοντα κώδικα ή η αφαίρεση dead code, αύξησε την απόδοση ενός προγράμματος μειώνοντας την υπολογιστική εργασία που πρέπει να εκτελεστεί και η μείωση του φόρτου εργασίας επέφερε εξοικονόμηση ισχύος/ενέργειας. Βελτιστοποιήσεις της ιεραρχίας μνήμης (memory hierarchy) όπως το loop tiling και το register allocation, βοηθούν στο να κρατηθούν τα δεδομένα πιο κοντά στον επεξεργαστή, δεδομένου ότι θα έχουν ταχύτερη πρόσβαση. Η διατήρηση δεδομένων σε μια on-chip cache μνήμη αντί για την off-chip ή σε έναν καταχωρητή αντί της cache μνήμης, εξοικονομεί επίσης ισχύ/ενέργεια λόγω της μειωμένης δραστηριότητας μεταβάσης. Ωστόσο, υπάρχει θεμελιώδης διαφορά στα μοντέλα και τις μετρικές που χρησιμοποιούνται για τις βελτιστοποιήσεις των επιδόσεων και εκείνων για τις ισχύος/ενέργειας.

Ένα παράδειγμα όπου οι βελτιστοποιήσεις για ισχύ/ενέργεια μπορεί να είναι διαφορετικές από αυτές για την απόδοση, είναι η εικαζόμενη εκτέλεση (speculative execution), η οποία εκτελείται νωρίτερα, βασιζόμενη σε κάποιες υποθέσεις σχετικά με τη μελλοντική συμπεριφορά του προγράμματος. Εάν αυτές οι υποθέσεις αποδειχθούν ψευδείς, μπορεί να υπάρξει πρόσθετη επεξεργασία που θα είναι απαραίτητη για να αναιρεθούν οι επιπτώσεις της εικαζόμενης εκτέλεσης. Το software prefetching, είναι ένα παράδειγμα μιάς τέτοιας βελτιστοποίησης, όπου ο μεταγλωττιστής εισάγει προανακληθείσες (prefetch) εντολές για τη πρόσβαση στη μνήμη. Στο πλαίσιο των βελτιστοποιήσεων ισχύος/ενέργειας κάθε επιπλέον εικαζόμενη δραστηριότητα, πρέπει να αντισταθμίζεται από το συνολικό όφελος ισχύος/ενέργειας της βελτιστοποίησης. Με άλλα λόγια, το όφελος για τη βελτιστοποίηση ισχύος/ενέργειας, πρέπει να είναι μεγαλύτερο από το όφελος της βελτιστοποίησης της απόδοσης. Αυτό δεν σημαίνει ότι το speculative execution δεν μπορεί να εφαρμοστεί για βελτιστοποιήσεις ισχύος/ενέργειας, αλλά προτείνεται μια λιγότερο επιθετική εφαρμογή της, περιορίζοντάς την στις περιπτώσεις όπου το όφελος είναι πιθανό. [3]

5.4 Οι παράγοντες κατανάλωσης της ισχύος

Υπάρχουν τουλάχιστον τρεις παράγοντες που συνεισφέρουν στη κατανάλωση ισχύος σε ένα υπολογιστικό σύστημα και που μπορούν να επηρεαστούν σημαντικά από το λογισμικό. **Το σύστημα μνήμης (memory system)** συνεισφέρει ένα σημαντικό κλάσμα της κατανάλωσης ισχύος (της τάξης του ενός δέκατου έως ενός τέταρτου) και ειδικά σε φορητά συστήματα μπορεί να είναι η κυρίαρχη πηγή κατανάλωσης ισχύος σε

απαιτητικές εφαρμογές. **Οι δίαυλοι συστήματος (system busses – instruction, address και data)** είναι στοιχεία για τα οποία η δραστηριότητα μεταγωγής (switching activity) καθορίζεται σε μεγάλο βαθμό από το λογισμικό. **Οι διαδρομές των δεδομένων (data paths)** π.χ. σε μία CPU οι μονάδες ALU, απαιτούν ισχύ ειδικά εάν όλες οι λειτουργικές μονάδες (π.χ. αθροιστής, πολλαπλασιαστής) ενεργοποιούνται ανά πάσα στιγμή. Ακόμη και η ισχύς που απαιτείται για τη λογική ελέγχου και τη διανομή ρολογιού είναι σημαντική, καθώς η συμβολή της στη κατανάλωση ενέργειας ενός προγράμματος, είναι ανάλογη με τον αριθμό των κύκλων εκτέλεσης του προγράμματος.

Οι διαδρομές δεδομένων για τις μονάδες αριθμητικής λογικής (ALU's) και τις μονάδες κινητής υποδιαστολής (FPU's), καταναλώνουν ένα μεγάλο μέρος ισχύος σε μια CPU. Ακόμη και αν οι ακριβείς ακολουθίες εισόδου δεδομένων στην ALU και την FPU είναι δύσκολο να προβλεφθούν, η ακολουθία των λειτουργιών και των εξαρτήσεων των δεδομένων, αποφασίζονται κατά τη διάρκεια του σχεδιασμού και της σύνταξης του λογισμικού. Ο κακός προγραμματισμός αυτών των λειτουργιών μπορεί να οδηγήσει σε περιττή σπατάλη ενέργειας.

Ορισμένες πηγές κατανάλωσης ισχύος, όπως η διανομή ρολογιού και η λογική ελέγχου, μπορεί να φαίνεται να μην έχουν καμία σχέση με τις αποφάσεις της σχεδίασης του λογισμικού. Ωστόσο, κάθε κύκλος εκτέλεσης ενός προγράμματος συνεπάγεται ενεργειακό κόστος και όσο περισσότερο χρόνο ένα πρόγραμμα απαιτεί για να εκτελεστεί, τόσο μεγαλύτερο θα είναι αυτό το ενεργειακό κόστος. Και σε καμία περίπτωση δεν πλεονεκτεί η χαμηλότερη μέση κατανάλωση ισχύος μιας ελαφρώς μεγαλύτερης ακολουθίας κώδικα, της επιπρόσθετης κατανάλωσης ενέργειας που έχει σχέση με τους επιπλέον κύκλους εκτέλεσης.

Οι δίαυλοι(busses), έχουν συνήθως υψηλά χωρητικά φορτία λόγω του αριθμού των μονάδων που συνδέονται με κάθε δίαυλο και του μήκους των διαδρομών των διαύλων. Η δραστηριότητα μεταγωγής σε αυτούς τους διαύλους καθορίζεται σε μεγάλο βαθμό από το λογισμικό. Η μεταγωγή σε ένα δίαυλο εντολών καθορίζεται από την ακολουθία των op-codes εντολών που πρέπει να εκτελεστούν. Ομοίως, η μεταγωγή ενός διαύλου διευθύνσεων, καθορίζεται από την ακολουθία και τις προσβάσεις δεδομένων και εντολών. Και οι δύο αυτές περιπτώσεις, μπορούν να ληφθούν υπόψη κατά τη μεταγλώττιση(compilation). Η μεταγωγή που σχετίζεται με δεδομένα είναι πολύ πιο δύσκολο να προβλεφθεί κατά τη μεταγλώττιση, καθώς τα περισσότερα δεδομένα εισόδου σε ένα πρόγραμμα δεν παρέχονται μέχρι το χρόνο εκτέλεσης.

Οι προσβάσεις στη μνήμη είναι ενεργειακά δαπανηρές για διάφορους λόγους. Η ανάγνωση ή η εγγραφή μιας θέσης μνήμης περιλαμβάνει switching activity στις εξαιρετικά υψηλές χωρητικά γραμμές των δεδομένων και διευθύνσεων που πηγαίνουν στη μνήμη. Η χρήση πολλαπλών memory banks, έτσι ώστε να είναι δυνατή η παράλληλη φόρτωση πολλαπλών λέξεων δεδομένων, όχι μόνο βελτιώνουν την απόδοση, αλλά είναι πιο αποδοτικά ενεργειακά. Οι τρόποι πρόσβασης της μνήμης από ένα πρόγραμμα, μπορεί να επηρεάσει σημαντικά την απόδοση της κρυφής μνήμης(cache) ενός συστήματος. Η λανθασμένη πρόσβαση στη μνήμη (ή μια κρυφή

μνήμη που είναι πολύ μικρή) θα οδηγήσει σε cache misses και κατανάλωση ισχύος από τις πολλές προσβάσεις. Ο συμπαγής κώδικας μηχανής μειώνει τη δραστηριότητα στη μνήμη, μειώνοντας τον αριθμό των εντολών που πρέπει να ληφθούν και μειώνοντας την πιθανότητα των cache misses. Οι προσβάσεις στη cache μνήμη είναι ενεργειακά πιο αποδοτικές από τις προσπελάσεις της κύριας μνήμης. Η cache μνήμη είναι πιο κοντά στην CPU από ό, τι είναι η κύρια μνήμη, με αποτέλεσμα μικρότερες και λιγότερο χωρητικές γραμμές διευθύνσεων και δεδομένων. [6]

5.5 Η εκτίμηση της κατανάλωσης ισχύος του λογισμικού και η CPU

Το πρώτο βήμα προς τη βελτιστοποίηση του λογισμικού για χαμηλή ισχύ, είναι η εκτίμηση της κατανάλωσης ισχύος ενός κομματιού κώδικα. Αυτό διαμορφώνεται σε δύο βασικά επίπεδα. Το χαμηλότερο επίπεδο είναι να χρησιμοποιηθούν τα υπάρχοντα εργαλεία προσομοίωσης και εκτίμησης ισχύος. Μια προσέγγιση υψηλότερου επιπέδου είναι η εκτίμηση της ισχύος με βάση τη συχνότητα εκτέλεσης κάθε τύπου εντολής ή ακολουθίας εντολών, δηλαδή το ενεργειακό profile εκτέλεσης. Το ενεργειακό profile εκτέλεσης μπορεί να χρησιμοποιηθεί με διάφορους τρόπους. Στην εκτίμηση ισχύος αρχιτεκτονικής, καθορίζει ποια βασικά συστατικά ενός επεξεργαστή θα είναι ενεργά κατά τη διάρκεια κάθε κύκλου εκτέλεσης ενός προγράμματος. Οι εκτιμήσεις ισχύος για κάθε ενεργό συστατικό στη συνέχεια λαμβάνονται από ένα πίνακα και προστίθενται στην εκτίμηση ισχύος για το πρόγραμμα. Μια άλλη προσέγγιση βασίζεται στην παραδοχή ότι η δραστηριότητα μεταγωγής στους διαύλους (διευθύνσεων, εντολών και δεδομένων) είναι αντιπροσωπευτική της δραστηριότητας μεταγωγής (και της κατανάλωσης ισχύος) σε ολόκληρο τον επεξεργαστή. Η δραστηριότητα μεταγωγής διαύλου, μπορεί να εκτιμηθεί με βάση την ακολουθία των op-codes και των διευθύνσεων μνήμης. Και τελευταία, αναφέρεται η προσέγγιση ως ανάλυση ισχύος σε επίπεδο εντολής και σχετίζεται με μεμονωμένες εντολές και ορισμένες ακολουθίες εντολών. Αυτές οι καταναλώσεις, μπορούν να εφαρμοστούν στην ακολουθία εκτέλεσης εντολών ενός προγράμματος για να ληφθεί μια συνολική εκτίμηση ισχύος.

5.5.1 Εκτίμηση ισχύος σε επίπεδο πύλης

Η εκτίμηση ισχύος σε επίπεδο πύλης ενός επεξεργαστή που εκτελεί ένα πρόγραμμα, είναι η πιο ακριβής μέθοδος υποθέτοντας ότι μια λεπτομερής περιγραφή επιπέδου πύλης είναι διαθέσιμη για τον επεξεργαστή. Μια τέτοια λεπτομερής περιγραφή του επεξεργαστή δεν είναι πιθανό να είναι διαθέσιμη σε έναν προγραμματιστή λογισμικού, ειδικά εάν ο επεξεργαστής δεν είναι in-house σχεδιασμός. Ακόμη και αν οι λεπτομέρειες είναι διαθέσιμες, αυτή η προσέγγιση θα είναι πολύ αργή για τη βελτιστοποίηση χαμηλής ισχύος ενός προγράμματος.

5.5.2 Εκτίμηση ισχύος σε επίπεδο αρχιτεκτονικής

Η εκτίμηση ισχύος σε επίπεδο αρχιτεκτονικής, είναι λιγότερο ακριβής αλλά πολύ ταχύτερη από την εκτίμηση επιπέδου πύλης. Αυτή η προσέγγιση απαιτεί ένα μοντέλο του επεξεργαστή στο επίπεδο κυρίων μονάδων (ALU, register file, κλπ.) μαζί με τις εκτιμήσεις κατανάλωσης ισχύος για κάθε μονάδα. Απαιτεί επίσης ένα μοντέλο του οποίου οι μονάδες θα είναι ενεργές ως συνάρτηση των οδηγιών που εκτελούνται. Η προσέγγιση σε επίπεδο αρχιτεκτονικής εφαρμόζεται σε έναν προσομοιωτή εκτίμησης ισχύος που ονομάζεται ESP(Early design Stage Power and performance simulator). Το ESP προσομοιώνει την εκτέλεση ενός προγράμματος, καθορίζοντας ποιες μονάδες του συστήματος είναι ενεργές σε κάθε κύκλο εκτέλεσης και προσθέτοντας τη συμβολή ισχύος κάθε μονάδας. [6]

5.5.3 Ανάλυση της ισχύος σε επίπεδο εντολής (Instruction-Level Power Analysis)

Η ανάλυση ισχύος σε επίπεδο εντολής (ILPA-Instruction-Level Power Analysis), αποτελεί μια εμπειρική μέθοδο για τον χαρακτηρισμό της κατανάλωσης ισχύος των σύντομων ακολουθιών εντολών και για τη χρήση αυτών των αποτελεσμάτων για την εκτίμηση της κατανάλωσης ισχύος (ή ενέργειας) ενός προγράμματος. Δεν απαιτείται λεπτομερής περιγραφή του εσωτερικού σχεδιασμού του επεξεργαστή, ωστόσο κάποια κατανόηση της αρχιτεκτονικής του επεξεργαστή είναι σημαντική για την κατάλληλη επιλογή των τύπων ακολουθιών των εντολών που θα χαρακτηριστούν. Αυτή η κατανόηση καθίσταται ακόμη πιο σημαντική όταν εξετάζονται τρόποι βελτιστοποίησης ενός προγράμματος με βάση την εκτίμηση ισχύος.

Η πρώτη απαίτηση για το ILPA, είναι να εκτιμηθεί η μέση κατανάλωση ισχύος που σχετίζεται με κάθε εντολή ή ακολουθία εντολών. Τρεις προσεγγίσεις έχουν τεκμηριωθεί για την επίτευξη αυτής της εκτίμησης. Η πιο απλή μέθοδος αν το επιτρέπει η διάταξη του hardware, είναι η άμεση μέτρηση του ρεύματος που καταναλώνει ο επεξεργαστής, καθώς εκτελεί διάφορες ακολουθίες εντολών. Εάν αυτό δεν είναι πρακτικό, μια άλλη μέθοδος είναι να χρησιμοποιηθεί μία γλώσσα περιγραφής υλικού (όπως Verilog ή VHDL) του επεξεργαστή, για να προσομοιωθεί η εκτέλεση των ακολουθιών των εντολών. Ένας πραγματικός επεξεργαστής μπορεί στη συνέχεια να χρησιμοποιηθεί και να γίνει έλεγχος των αποτελεσμάτων της προσομοίωσης με ένα αμπερόμετρο. Μια τρίτη προσέγγιση είναι η χρήση προσομοίωσης ισχύος επιπέδου πύλης του επεξεργαστή για τη λήψη εκτιμήσεων επιπέδου εντολής. Η επιλογή των ακολουθιών εντολών για αυτή την τοεκτίμηση, είναι κρίσιμη για την επιτυχία αυτής της μεθόδου. Σαν ελάχιστο, είναι απαραίτητο να καθοριστεί το βασικό ενεργειακό κόστος των μεμονωμένων εντολών, το οποίο αφορά στο τμήμα της κατανάλωσης ισχύος μιας εντολής που είναι ανεξάρτητη από την προηγούμενη κατάσταση του επεξεργαστή. Το βασικό ενεργειακό κόστος, αποκλείει την ύπαρξη καταστάσεων όπως η pipeline καθυστέρηση, οι cache misses και το switching στους διαύλους λόγω της ύπαρξης op-codes για διαδοχικές εντολές και μπορεί να προσδιοριστεί με τον έλεγχο αρκετών περιπτώσεων αυτής της εντολής μέσα σε έναν βρόχο infinite loop, όπου η μέση τιμή του ρεύματος τροφοδοσίας

μετριέται ενώ εκτελείται ο βρόχος. Ο βρόχος πρέπει να διαρκέσει όσο το δυνατόν περισσότερο, ώστε να ελαχιστοποιηθεί το σφάλμα εκτίμησης που οφείλεται στην επιστροφή του βρόχου (με την εντολή jump στο τέλος του), ωστόσο δεν πρέπει να γίνει τόσο μεγάλος ώστε να προκαλέσει cache misses. Οι εκτιμήσεις ισχύος και ενέργειας για την εντολή, υπολογίζονται από τη μέση κατανάλωση ρεύματος, την τάση τροφοδοσίας και τον αριθμό κύκλων εκτέλεσης ανά εντολή.

Οι βασικές καταναλώσεις για κάθε εντολή, δεν είναι πάντα επαρκείς για μία ακριβή εκτίμηση ισχύος του λογισμικού και απαιτούνται πρόσθετες ακολουθίες εντολών, προκειμένου να ληφθεί υπόψη η επίδραση της προηγούμενης κατάστασης του επεξεργαστή στην κατανάλωση ισχύος κάθε εντολής. Οι pipeline καθυστερήσεις, οι buffer καθυστερήσεις και οι cache misses είναι προφανή γεγονότα κατανάλωσης ενέργειας, των οποίων η εμφάνιση εξαρτάται από την προηγούμενη κατάσταση του επεξεργαστή. Μπορούν να δημιουργηθούν ακολουθίες εντολών που προκαλούν κάθε ένα από αυτά τα γεγονότα, έτσι ώστε να μπορούν να γίνουν οι μετρήσεις των καταναλώσεων. Ωστόσο, οι καθυστερήσεις και οι αστοχίες της κρυφής μνήμης, είναι αποτελέσματα που απαιτούν ανάλυση μεγαλύτερης κλίμακας ή προσομοίωση της εκτέλεσης του προγράμματος, προκειμένου να ληφθούν υπόψη κατάλληλα για την ενεργειακή εκτίμηση ενός προγράμματος.

Υπάρχουν και άλλα ενεργειακά κόστη που μπορούν να αποδοθούν άμεσα στις αλλαγές κατάστασης του επεξεργαστή που προκύπτουν από την εκτέλεση ενός ζεύγους εντολών. Τα κόστη αυτά αναφέρονται ως circuit state effects. Στην πιο κάτω ακολουθία κώδικα για παράδειγμα:

ADD A ← B, C
MULT C ← A, B

Το κύριο circuit state effect είναι πιθανώς το ενεργειακό κόστος που συνδέεται με τη μεταβολή της κατάστασης του διαύλου εντολών, καθώς ο op-code μεταβαίνει από τη λειτουργία πρόσθεσης, στον op-code για μια λειτουργία πολλαπλασιασμού. Άλλες circuit state effects για αυτό το παράδειγμα, θα μπορούσαν να περιλαμβάνουν την εναλλαγή των γραμμών ελέγχου για να απενεργοποιήσουν την πρόσθεση και να ενεργοποιήσουν τον πολλαπλασιασμό, τις αλλαγές λειτουργίας εντός της ALU και την εναλλαγή των γραμμών δεδομένων για την αναδρομολόγηση σημάτων μεταξύ της ALU και του register file. Αν και αυτό το παράδειγμα εξετάζει ένα ζεύγος παρακείμενων εντολών, είναι επίσης δυνατό τα circuit state effects να εκτείνονται σε έναν αυθαίρετο αριθμό κύκλων εκτέλεσης. [6]

5.6 Η δραστηριότητα μεταγωγής διαύλου (Bus switching activity)

Η δραστηριότητα μεταγωγής διαύλων, είναι άλλος ένας δείκτης κατανάλωσης της ισχύος του λογισμικού. Όπως περιγράφηκε στα προηγούμενα κεφάλαια, σε επίπεδο υλικού και αρχιτεκτονικής οι δύο μεγάλες καταναλώσεις ισχύος είναι η δυναμική (active ή switching) κατανάλωση ισχύος και η στατική (standby ή static). Αυτές οι καταναλώσεις, σχετίζονται με τη μείωση της δραστηριότητας μεταγωγής, και πρέπει με τεχνικές σε επίπεδο software, π.χ. bit, εντολής, compiler, λειτουργικού συστήματος (OS - Operating System), εφαρμογής (application) να ελαχιστοποιηθούν. [6]

Η μοντελοποίηση της δραστηριότητας μεταγωγής διαύλων, απαιτεί τη γνώση της αρχιτεκτονικής διαύλου ενός επεξεργαστή, των op-codes του σετ των εντολών, ένα αντιπροσωπευτικό σύνολο (ή στατιστικά στοιχεία) των δεδομένων εισόδου σε ένα πρόγραμμα και του τρόπου με τον οποίο ο κώδικας και τα δεδομένα αντιστοιχίζονται στο χώρο διευθύνσεων. Με την προσομοίωση της εκτέλεσης ενός προγράμματος, μπορεί κανείς να καθορίσει την ακολουθία των op-codes, των διευθύνσεων και των δεδομένων που εμφανίζονται στους διάφορους διαύλους. Στατιστικά της δραστηριότητας μεταγωγής, μπορούν να υπολογιστούν απευθείας από αυτές τις ακολουθίες δυαδικών τιμών. [6], [19], [20]

5.7 Low power τεχνικές στο λογισμικό

Η μείωση της δραστηριότητας μεταγωγής και κατά συνέπεια της δυναμικής κατανάλωσης ισχύος, πετυχαίνεται εισάγοντας στην υλοποίηση του λογισμικού προσεγγίσεις όπως, η ρύθμιση-βελτιστοποίηση του datapath δηλαδή του πλάτους των δυαδικών ψηφίων (bit width) των διαύλων(busses), η βελτιστοποίηση στη μεταγλώττιση (compilation) με βάση τον χρονοπρογραμματισμό και την επαναταξινόμηση των εντολών (instruction scheduling-reordering), το operand swapping, η δημιουργία κώδικα μέσω αντιστοίχισης μοτίβων (code generation through loop matching), [1] το cold scheduling κ.α. Η χρήση μεθόδων και πρακτικών όπως, το approximate computing, ο παράλληλος προγραμματισμός, η χρήση κατάλληλων γλωσσών προγραμματισμού, η ανάλυση του πηγαίου κώδικα, οι πρακτικές κωδικοποίησης και οι δομές δεδομένων επιτυγχάνουν αύξηση της ενεργειακής απόδοσης. Πληροφορίες στους προγραμματιστές κατά τη διάρκεια της φάσης ανάπτυξης σχετικά με την αξιολόγηση της κατανάλωσης ενέργειας του λογισμικού, μπορούν να παρέχουν τα software & hardware energy monitoring εργαλεία και η χρήση benchmarks. [19], [20]

5.7.1 Οι βελτιστοποιήσεις στον κώδικα

Δεδομένου ότι υπάρχουν πολλές πιθανές ακολουθίες κώδικα που εκτελούν την ίδια εργασία, θα πρέπει να είναι δυνατή η επιλογή μιας ακολουθίας κώδικα ή ενός μείγματος εντολών που να ελαχιστοποιεί την ισχύ ή την ενέργεια. [22] Στην πραγματικότητα, πολλές τεχνικές βελτιστοποίησης για την ταχύτητα εκτέλεσης ή το μέγεθος του κώδικα θα πρέπει να εφαρμόζονται εξίσου στη βελτιστοποίηση ισχύος ή ενέργειας.

Λαμβάνοντας υπόψη τα οφέλη του κώδικα υψηλής απόδοσης σε σχέση με την εξοικονόμηση ενέργειας, οι βελτιστοποιήσεις που βελτιώνουν την ταχύτητα είναι ιδιαίτερα χρήσιμες. Η ελαχιστοποίηση του μεγέθους του κώδικα μπορεί να είναι απαραίτητη, ειδικά σε ενσωματωμένα συστήματα, αλλά δεν είναι τόσο άμεσα συνδεδεμένη με την εξοικονόμηση ενέργειας και όσον αφορά την απόδοση και την ελαχιστοποίηση της ενέργειας, η απόδοση της κρυφής μνήμης αποτελεί μεγαλύτερο πρόβλημα. Ο μεγάλος κώδικας και μια μικρή κρυφή μνήμη μπορούν να οδηγήσουν σε συχνά cache misses με αποτέλεσμα την υψηλή κατανάλωση ισχύος. Το μέγεθος του κώδικα και η απόδοση της κρυφής μνήμης είναι προβλήματα που απαιτούν προσπάθεια ώστε να μεγιστοποιηθεί η πυκνότητα του κώδικα για τους ενσωματωμένους επεξεργαστές. [6]

Το **instruction packing,** είναι ένα χαρακτηριστικό ορισμένων επεξεργαστών το οποίο επιτρέπει μια ALU λειτουργία και μια μεταφορά δεδομένων στη μνήμη, να «συσκευαστεί» σε μια ενιαία εντολή. Έτσι το ενεργειακό κόστος κάθε μεμονωμένης εντολής δεν επαναλαμβάνεται όταν οι λειτουργίες εκτελούνται παράλληλα. Κατά συνέπεια, μπορεί να υπάρξει σχεδόν 50% μείωση της ενέργειας που σχετίζεται με τη «συσκευασία» δύο εντολών. Παρόμοια οφέλη μπορούν να επιτευχθούν με τη **παράλληλη φόρτωση πολλαπλών λέξεων** από διαφορετικά τμήματα της μνήμης. Η ταυτόχρονη εκτέλεση ακέραιων και κινητής υποδιαστολής λειτουργιών, αντιπροσωπεύει ακόμα μια άλλη ευκαιρία για τη μείωση της συνολικής ενέργειας μιας ομάδας λειτουργιών. Όλες αυτές οι τεχνικές είναι ειδικές περιπτώσεις των τύπων κωδικοποίησης που μπορούν να γίνουν σε VLIW (very long instruction word) και στις υπερβαθμωτές (superscalar) αρχιτεκτονικές, όπου οι ευκαιρίες για instruction packing είναι πολύ πιο εκτεταμένες.

Το **instruction reordering – scheduling,** χρησιμοποιείται παραδοσιακά για τη μείωση της δραστηριότητας μεταγωγής στις διαδρομές των δεδομένων που είναι και ο κύριος καθοριστικός παράγοντας της κατανάλωσης ενέργειας.

Το instruction reordering για την επίτευξη χαμηλής κατανάλωσης ισχύος, σχετίζεται με τα circuit state effects. Το circuit state effect, είναι η ενέργεια που καταναλώνεται ως αποτέλεσμα της μετάβασης του επεξεργαστή από την εκτέλεση ενός τύπου εντολής σε ένα άλλο, οπότε η ενέργεια που καταναλώνεται κατά την εκτέλεση μιας συγκεκριμένης εντολής, θα ποικίλει ανάλογα με το ποιά ήταν η προηγούμενη εντολή. Έτσι, μια κατάλληλη αναδιάταξη των εντολών σε ένα πρόγραμμα, μπορεί να οδηγήσει σε κατανάλωση χαμηλότερης ενέργειας. [23]

Σε ορισμένους τύπους επεξεργαστών, ειδικά DSP, το μέγεθος του circuit state effect μπορεί να ποικίλει σημαντικά ανάλογα με το ζεύγος οδηγιών που εμπλέκονται. Δεδομένου ενός πίνακα με τα ενεργειακά κόστη των circuit state effects, μπορεί κανείς να εκτιμήσει το συνολικό κόστος των circuit state effects για μια ακολουθία οδηγιών. Η ταξινόμηση των εντολών στη συνέχεια, περιλαμβάνει την αναδιάταξη των εντολών για την ελαχιστοποίηση του συνολικού ενεργειακού κόστους των circuit state effects, χωρίς να μεταβάλλεται η συμπεριφορά του προγράμματος. [6] Επίσης, με βάση το ενεργειακό κόστος κάθε εντολής, ο αλγόριθμος χρονοπρογραμματισμού εντολών cold scheduling, δίνει προτεραιότητα στην επιλογή και τοποθέτησης της κάθε εντολής.

Μέσω πειραματισμών που έγιναν, διαπιστώθηκε ότι μιά τεχνική προγραμματισμού επικεντρωμένη στη βελτίωση της απόδοσης, δεν δημιουργεί απαραιτήτως τον πιο ενεργειακά αποδοτικό κώδικα. Από την άλλη, ένας προσανατολισμένος στην εξοικονόμηση ενέργειας προγραμματισμός, ήταν μόνο επιπλέον 6% χειρότερος από τον προγραμματισμό απόδοσης. Αντί της χωριστής βελτιστοποίησης για την ενέργεια και την απόδοση, μπορούν να σχεδιαστούν τεχνικές χρονοπρογραμματισμού των εντολών, που εξετάζουν ταυτόχρονα την ενέργεια και την απόδοση. Μία από τις δυσκολίες του χρονοπρογραμματισμού εντολών για την ενέργεια, είναι η εκτίμηση του ενεργειακού αντίκτυπου των τμημάτων δεδομένων των εντολών. Έτσι, εκτός από τον χρονοπρογραμματισμό εντολών, τεχνικές όπως η αναδιάταξη τελεστών (π.χ. εναλλαγή τελεστών σε λειτουργίες ALU ή κινητής υποδιαστολής) μπορεί να είναι πολύ αποτελεσματικές στη μείωση της δραστηριότητας μεταγωγής στους διαύλους. [5]

Η **εναλλαγή τελεστών (operand swapping)**, επιχειρεί να εναλλάξει (swap) τελεστές σε λειτουργίες της ALU ή της floating point unit, με τέτοιο τρόπο που να ελαχιστοποιεί τη δραστηριότητα μεταγωγής που σχετίζεται με τη λειτουργία. Ένας τρόπος με τον οποίο η εναλλαγή τελεστών θα μπορούσε να βοηθήσει, είναι να ελαχιστοποιηθεί η εναλλαγή που σχετίζεται με την ακολουθία των τιμών εισόδου. Εάν βρίσκονται latches στις εισόδους μιας λειτουργικής μονάδας για να διατηρήσουν τις εισόδους από την εναλλαγή κατά τη διάρκεια περιόδων αδράνειας, τότε θα μπορούσε να υπάρξει ένα πλεονέκτημα για την εναλλαγή τελεστών όσον αφορά την ελαχιστοποίηση της μεταγωγής στις εισόδους της λειτουργικής μονάδας. Για παράδειγμα, εάν δύο διαδοχικές προσθέσεις έχουν έναν τελεστή που παραμένει ο ίδιος (ας πούμε x + 7 και y + 7), τότε η εναλλαγή θα ελαχιστοποιηθεί εάν ο κοινός τελεστής εφαρμοστεί στην ίδια είσοδο αθροιστή και τις δύο φορές. Η εναλλαγή επίσης των τελεστών στην είσοδο του πολλαπλασιαστή, αφήνει αναλλοίωτο το αποτέλεσμα, αλλά μεταβάλλει σημαντικά την ισχύ.

Η **ταξινόμηση των τελεστών (Operand ordering),** μπορεί να έχει ακόμη μεγαλύτερη επίδραση εάν οι τελεστές σε μια μεταθετική λειτουργία δεν αντιμετωπιστούν συμμετρικά. Για παράδειγμα, ο αριθμός των προσθέσεων και αφαιρέσεων που εκτελούνται σε έναν Booth πολλαπλασιαστή, εξαρτάται από το μοτίβο των δυαδικών ψηφίων του δεύτερου τελεστή. Ο αριθμός των προσθέσεων και αφαιρέσεων που

απαιτούνται, αναφέρεται ως το **βάρος επανακωδικοποίησης (recoding weight)** του δεύτερου τελεστή. Σε ένα δεδομένο ζεύγος τιμών που πρέπει να πολλαπλασιαστούν, η ισχύς θα μειωνόταν αν τοποθετούνταν η τιμή με το χαμηλότερο βάρος επανακωδικοποίησης στη δεύτερη είσοδο του πολλαπλασιαστή Booth. Σπάνια γνωρίζουμε από πριν την αξία και των δύο τελεστών, αλλά δεν είναι ασυνήθιστο να γνωρίζουμε την αξία (και το βάρος κωδικοποίησης) του ενός τελεστή, οπότε εάν ένας τελεστής είναι γνωστό ότι έχει υψηλό βάρος κωδικοποίησης, θα τοποθετηθεί στην πρώτη θέση τελεστή. Εάν ένας τελεστής είναι γνωστό ότι έχει χαμηλό βάρος κωδικοποίησης, τότε θα τοποθετηθεί στη δεύτερη θέση. Έχει αποδειχτεί ότι στην πλειονότητα των περιπτώσεων, πραγματοποιήθηκε σημαντική εξοικονόμηση ενέργειας ανά λειτουργία, της τάξης του 10% έως 30%, ενώ για τις λίγες περιπτώσεις κατά τις οποίες η εναλλαγή δεν ήταν επωφελής, η επιπλέον κατανάλωση ήταν μέτρια, μικρότερη από 6%. [6]

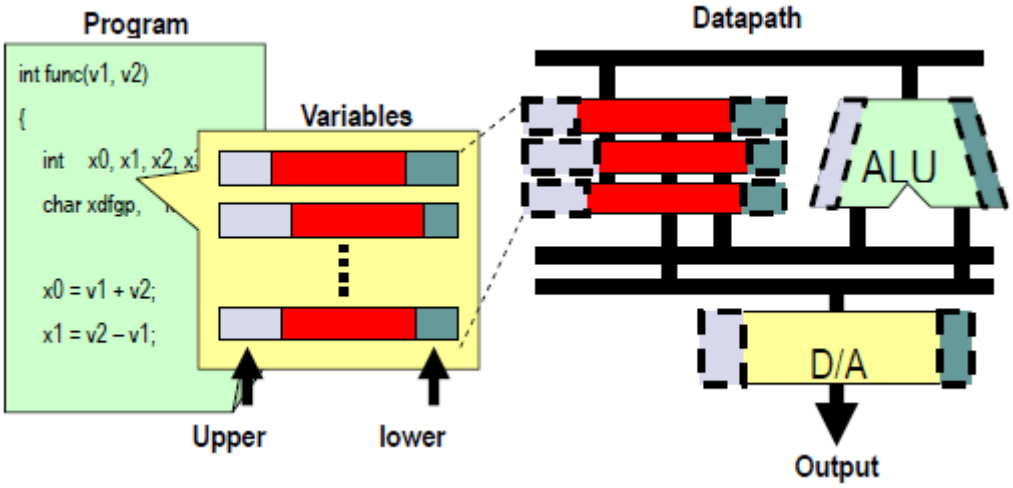
Η **χρήση processor-specific εντολών,** επίσης βοηθά στη μείωση της δραστηριότητα μεταγωγής. Για παράδειγμα στον επεξεργαστή ARM, ο προεπιλεγμένος τύπος δεδομένων int, είναι περίπου 20% πιο αποτελεσματικός ενεργειακά από τον char ή τον short. Επίσης στον ARM είναι αποτελεσματικότερες οι υπό όρους εντολές(conditional instructions) αντί των διακλαδώσεων. [22]

5.7.2 Η βελτιστοποίηση του datapath bit-width

Μια τεχνική για τη μείωση της δραστηριότητας μεταγωγής είναι η ρύθμιση του πλάτους δεδομένων (datapath bitwidth). Δεδομένου ότι το πλάτος του datapath, δηλαδή το bitwidth των διαύλων και των μονάδων σε ένα σύστημα, επηρεάζει έντονα το μέγεθος των κυκλωμάτων και των μνημών σε ένα σύστημα, η κατανάλωση ενέργειας ενός συστήματος εξαρτάται επίσης από το πλάτος του datapath. Το bitwidth των δεδομένων, το μήκος δηλαδή των δεδομένων, που επεξεργάζονται στο σύστημα, είναι μία από τις σημαντικότερες παραμέτρους σχεδιασμού που σχετίζονται με την απόδοση, το κόστος και την ισχύ του συστήματος. Η παροχή μεγαλύτερου πλάτους δεδομένων για την επεξεργασία από ό,τι απαιτείται, θα καταναλώνει περισσότερη δυναμική και στατική ισχύ από τα επιπλέον δυαδικά ψηφία, από ό,τι είναι απαραίτητο.

Οι τυπικοί αλγόριθμοι που ορίζονται για παράδειγμα στη C / C++, δεν περιέχουν αρχικά ορισμούς του πραγματικού bit-width των εντολών. Η επιλογή αλγορίθμου, συχνά βασίζεται σε υπολογισμούς κινητής υποδιαστολής και ακέραιων αριθμών. Ανάλογα με τους στόχους που τίθενται για την βελτιστοποίηση της εφαρμογής, οι προγραμματιστές μπορούν να αξιολογήσουν τις ελάχιστες και μέγιστες τιμές σε συγκεκριμένες εντολές και στη συνέχεια να επιλέξουν ανάλογα το βέλτιστο bit-width. Αυτό τους επιτρέπει να κατανοήσουν την επίπτωση του bit-width στην κατανάλωση ενέργειας. Σε μιά ποιοτική σχεδίαση και τα υψηλότερα και τα χαμηλότερα bits μπορούν

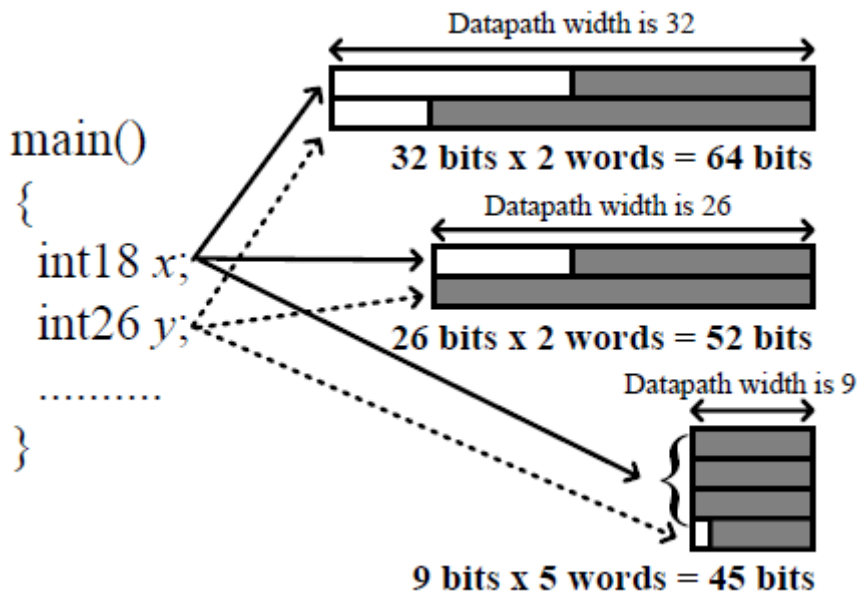
να μειωθούν(Σχήμα 5.2) και αυτό σημαίνει ότι υπάρχει δυνατότητα για περαιτέρω μείωση της ενέργειας μειώνοντας όμως την ακρίβεια υπολογισμού. [17]



Σχήμα 5.2: Ρύθμιση datapath bit-width.

Γενικά, ο περιορισμός του datapath width, μειώνει την ισχύ του επεξεργαστή, αλλά υποβαθμίζει την απόδοση. Ο αριθμός των κύκλων εκτέλεσης αυξάνεται, καθώς ορισμένες λειτουργίες μονής ακρίβειας (single-precision), πρέπει να αντικατασταθούν με διπλής ακριβείας (double precision) λειτουργίες, προκειμένου να διατηρηθεί η ακρίβεια του υπολογισμού. Οι single-precision λειτουργίες είναι εκείνες των οποίων η ακρίβεια είναι μικρότερη από αυτή του datapath width. Για παράδειγμα, μια προσθήκη δύο δεδομένων 32-bit είναι μια λειτουργία μονής ακρίβειας, των οποίων το datapath width είναι ίσο ή μεγαλύτερο από 32 bits, ενώ είναι μια λειτουργία διπλής ακρίβειας σε επεξεργαστές 16-bit.

Στο σχήμα 5.3, δίνεται ένα πρόγραμμα που περιλαμβάνει δύο μεταβλητές x και y και υποθέτουμε ότι δύο μεταβλητές x και y απαιτούν το πολύ 18 bits και 26 bits, αντίστοιχα. Όταν το πλάτος της διαδρομής δεδομένων είναι 32 bits, απαιτούνται δύο λέξεις για την αποθήκευση αυτών των δύο μεταβλητών και η ποσότητα της μνήμης δεδομένων είναι 64 bits. Δεδομένου ότι το ελάχιστο μέγεθος bit που απαιτείται για την αποθήκευση των μεταβλητών είναι μόνο 44 bits (18+26), 20 bits της μνήμης (περίπου 30%) παραμένουν αχρησιμοποίητα. Με τη μείωση του πλάτους datapath σε 26 bits, μπορεί κανείς να μειώσει τα αχρησιμοποίητα bits σε 8. Τα αχρησιμοποίητα bits ωστόσο αυξάνονται σε 31, αν υιοθετηθεί ένα datapath των 25 bits, επειδή το y απαιτεί τώρα δύο λέξεις. Όταν το πλάτος της διαδρομής δεδομένων είναι 9 bits, απαιτούνται δύο λέξεις και τρεις λέξεις για τα x και y, αντίστοιχα, και η αχρησιμοποίητη περιοχή είναι μόνο 1 bit. Πολλά αχρησιμοποίητα bits στη μνήμη δεδομένων μπορούν να εξαλειφθούν με βελτιστοποίηση πλάτους datapath. [17]



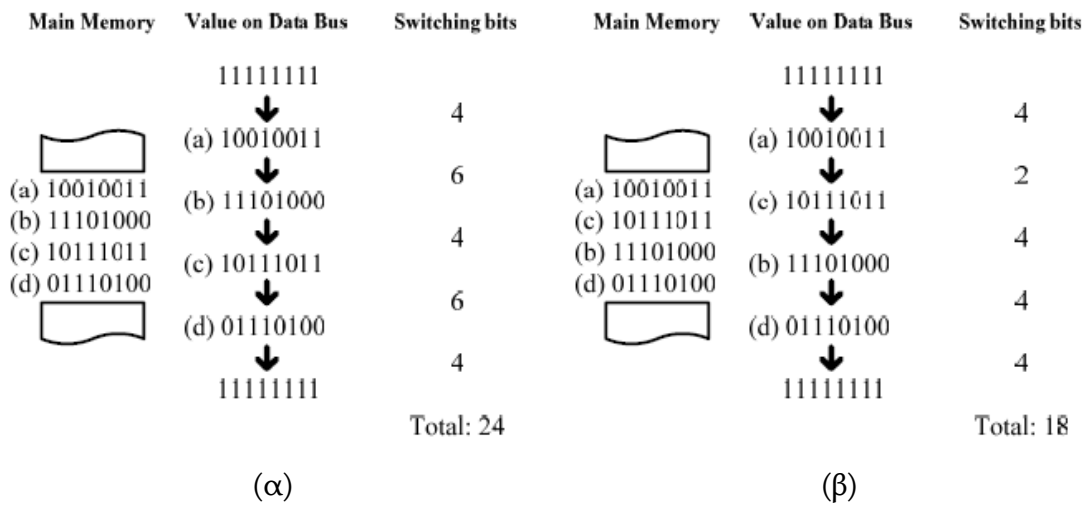
Σχήμα 5.3: Παράδειγμα ρύθμισης του datapath width.

Η τεχνική βελτιστοποίησης του bitwidth για τη μείωση όχι μόνο της δυναμικής αλλά και της στατικής ισχύος, έδωσε για τον αλγόριθμο Lempel-Ziv, εξοικονόμηση δυναμικής ισχύος κατά 59.2% και την στατικής ισχύος κατά 64.3% στο βέλτιστο πλάτος datapath των 15bits. Για τον κωδικοποιητή ADPCM, η εξοικονόμηση δυναμικής ισχύος είναι 44.2% και και της στατικής ισχύος είναι 4.74% στο βέλτιστο πλάτος datapath των 19bits. Για τον αποκωδικοποιητή ήχου MPEG-2 AAC, η εξοικονόμηση δυναμικής ισχύος είναι 14,5% και η εξοικονόμηση στατικής ισχύος είναι 18,1% στο βέλτιστο πλάτος datapath των 24bits και στον MPEG2 αποκωδικοποιητή βίντεο, η εξοικονόμηση δυναμικής ισχύος είναι 18,3% και της στατικής ισχύος είναι 19,1% στο βέλτιστο πλάτος datapath των 28bits. Για διαφορετικές εφαρμογές, ο αριθμός των μεταβλητών είναι διαφορετικός και το ενεργειακά αποδοτικό μέγεθος των μεταβλητών είναι επίσης διαφορετικό, επομένως το βέλτιστο datapath bitwidth της ελάχιστης ισχύος, θα είναι διαφορετικό. [20]

5.7.3 Instruction scheduling-based βελτιστοποίηση για τη μείωση της δραστηριότητας μεταγωγής

Μία instruction scheduling τεχνική που μειώνει τις μεταβάσεις σε ένα δίαυλο δεδομένων μεταξύ μιας on-chip cache μνήμης και της κύριας μνήμης (και ως εκ τούτου μειώνει την ισχύ που καταναλώνεται), φαίνεται στο σχήμα 5.4 όπου το πλάτος εντολής είναι 8-bit και η cache μνήμη 32-bit. Υπάρχουν τέσσερις εντολές (a) έως (d) στο μπλοκ μνήμης. Όταν το μπλοκ μνήμης αποστέλλεται στην cache μνήμη, αποστέλλεται πρώτα η εντολή (a) και τέσσερα bits αλλάζουν από 1 σε 0. Στον επόμενο κύκλο, η (b) αποστέλλεται στην cache μνήμη και έξι δυαδικά ψηφία αλλάζουν. Ως αποτέλεσμα, ένα cache miss μπορεί να συμβεί για συνολικά είκοσι τέσσερις μεταβάσεις στο δίαυλο

δεδομένων. Με την αλλαγή της θέσης των δύο εντολών (b) και (c) και θεωρώντας ότι διατηρείται η συνέχεια του προγράμματος, μειώνονται οι μεταβάσεις διαύλου κατά 25%, δηλαδή από είκοσι τέσσερις σε δεκαοκτώ. Έτσι, το instruction scheduling μπορεί να μειώσει τις μεταβάσεις στον δίαυλο. Ο αλγόριθμος χρονοπρογραμματισμού, επιτυγχάνει σημαντική μείωση έως και 28% των μεταβάσεων στο δίαυλο δεδομένων, λειτουργώντας αποτελεσματικά.



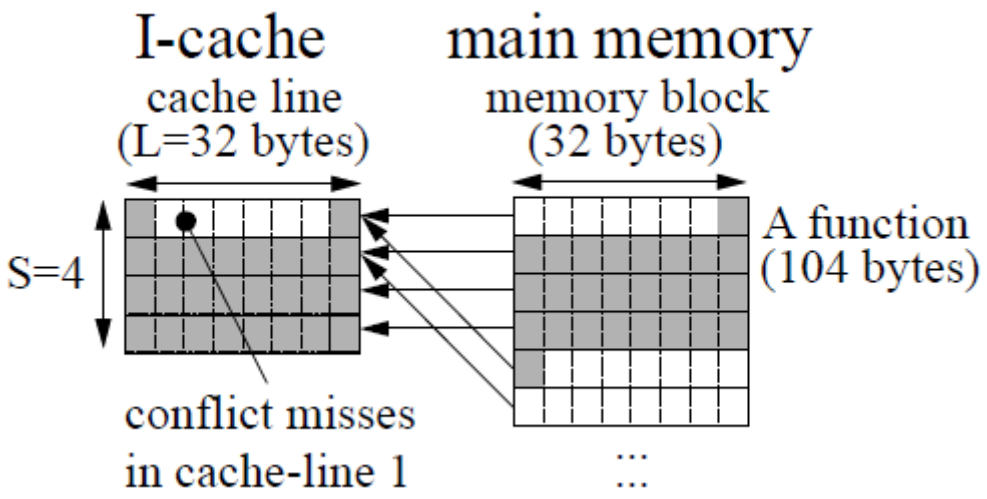
Σχήμα 5.4: (α) Μεταβάσεις του διαύλου χωρίς βελτιστοποίηση και (β) μεταβάσεις του διαύλου με instruction scheduling.

5.7.4 Compiler-based προσέγγιση για τη μείωση των cache misses

Μια προσέγγιση για τη μείωση του αριθμού των cache misses και κατά συνέπεια της δραστηριότητας μεταγωγής του διαύλου, είναι μία compiler-based προσέγγιση. Η ιδέα είναι να τροποποιηθεί η τοποθέτηση των βασικών μπλοκ, των διαδικασιών ή των μεταβλητών στο χώρο διευθύνσεων, ώστε ο αριθμός των cache conflict misses να ελαττωθεί. Αυτό μπορεί να μειώσει σημαντικά τον αριθμό των cache misses και την κατανάλωση ενέργειας των υποσυστημάτων μνήμης.

Σε μιά instruction cache μνήμη με 4 γραμμές, όπου κάθε γραμμή κρυφής μνήμης είναι 32 bytes όπως φαίνεται στο σχήμα 5.5, οι λειτουργίες A, B, C και D είναι τοποθετημένες στην κύρια μνήμη όπως φαίνεται στην αριστερή πλευρά του σχήματος. Εάν οι λειτουργίες A, B και D υλοποιούνται σε βρόχο, εμφανίζονται conflict misses, επειδή οι A και D λειτουργίες τοποθετούνται στην ίδια γραμμή κρυφής μνήμης. Εάν οι θέσεις των C και D αλλαχτούν όπως φαίνεται στη δεξιά πλευρά του σχήματος 5.3, το conflict miss επιλύεται. Οι τεχνικές τοποθέτησης κώδικα (Code placement techniques) τροποποιούν την τοποθέτηση βασικών μπλοκ ή λειτουργιών στο χώρο διευθύνσεων έτσι ώστε να ελαχιστοποιείται ο συνολικός αριθμός των conflict misses στη μνήμη cache. Παρόμοια

με τις τεχνικές τοποθέτησης κώδικα, οι τεχνικές τοποθέτησης δεδομένων (Data placement techniques) τροποποιούν την τοποθέτηση των μεταβλητών στο χώρο διευθύνσεων έτσι ώστε να μειωθεί ο αριθμός των data cache misses.



Σχήμα 5.5: Παράδειγμα τοποθέτησης κώδικα(Code placement).

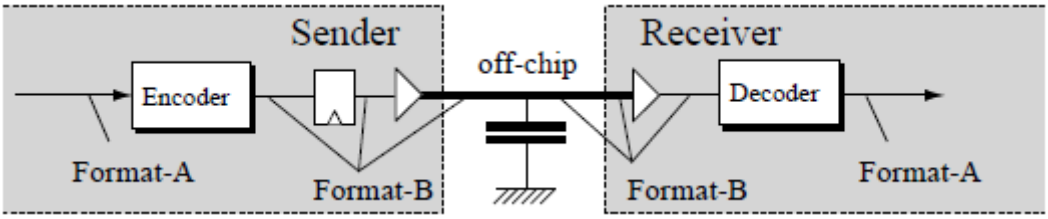
Σε έναν άλλο αλγόριθμο τοποθέτησης δεδομένων που βρίσκει τις βέλτιστες θέσεις των μεταβλητών στην κύρια μνήμη, ο αλγόριθμος διερευνά τα διαφορετικά μεγέθη cache μνήμης, λαμβάνοντας υπόψη τις αντισταθμίσεις μεταξύ της απόδοσης και της κατανάλωσης ενέργειας. Σαν πρώτο βήμα, μετρούνται τα cache miss ποσοστά για διαφορετικά μεγέθη cache μνήμης. Μόλις ληφθούν τα ποσοστά, ο αλγόριθμος εκτελεί τη τοποθέτηση των δεδομένων για κάθε μέγεθος cache μνήμης και υπολογίζει την κατανάλωση ενέργειας, συμπεριλαμβανομένων των καταναλώσεων για on-chip και off-chip προσβάσεις. [17]

5.7.5 Κωδικοποίηση διαύλου (Bus encoding)

Οι τεχνικές κωδικοποίησης διαύλου, μειώνουν την κατανάλωση ισχύος της επικοινωνίας. Η βασική στρατηγική είναι να μειωθεί η δραστηριότητα μεταγωγής των off-chip διαύλων με την κωδικοποίηση των δεδομένων που μεταδίδονται μεταξύ ενός επεξεργαστή και μιας μνήμης, αλλά πρέπει να υπάρξει εξισορρόπηση του ενεργειακού κόστους μεταξύ encoding και decoding. [22] Ας υποθέσουμε ότι έχουμε μια αρχική μορφή δεδομένων με τη μορφή format-A και μία χαμηλής μεταγωγής τη μορφή format-B, όπως φαίνεται στο σχήμα 5.6. Η κατανάλωση ενέργειας για την αποστολή δεδομένων χρησιμοποιώντας το format-A και το format-B είναι EA και EB, αντίστοιχα. Η επιπλέον ενέργεια για την κωδικοποίηση και αποκωδικοποίηση (μετατρέποντας δηλαδή το format-A σε format-B και αντίστροφα) είναι $E_{overhead}$. Οι τεχνικές

κωδικοποίησης διαύλου, είναι αποτελεσματικές μόνο όταν ισχύει η ακόλουθη ανισότητα,

$$EA > EB + Eoverhead$$



Σχήμα 5.6: Κωδικοποίηση διαύλου χαμηλής κατανάλωσης.

Η κωδικοποίηση αναστροφής διαύλου(bus-invert), είναι μία από τις πιο δημοφιλείς προσεγγίσεις. Στην κωδικοποίηση αναστροφής διαύλου, εάν ο αριθμός των αλλαγμένων (switched) δυαδικών ψηφίων μεταξύ του νέου μοτίβου που πρόκειται να μεταφερθεί και του παλιού που βρίσκεται επί του παρόντος στο δίαυλο είναι μεγαλύτερος από το μισό του πλάτους του διαύλου, το νέο μοτίβο μεταφέρεται με κάθε bit ανεστραμμένο. Ένα επιπρόσθετο ανεστραμμένο bit, χρησιμοποιείται για να ενημερώσει την πλευρά του δέκτη εάν το σχέδιο έχει αναστραφεί ή όχι. Οι μετρήσεις κατέδειξαν ότι η τεχνική κωδικοποίησης bus-invert, μειώνει τη μέγιστη κατανάλωση ισχύος εισόδου/εξόδου κατά 50% και τη μέση κατανάλωση ισχύος εισόδου/εξόδου κατά 25%. [17]

Άλλες πρακτικές χαμηλής κατανάλωσης στους διαύλους αναφέρονται οι ακόλουθοι:

- Mixed bus κωδικοποίηση.
- Κωδικοποίηση βάσει στατιστικής ανάλυσης της δραστηριότητας μεταγωγής των διαύλων.
- Συμπίεση για την αφαίρεση πλεονάζοντα κώδικα.
- Χρήση κώδικα Gray στα address busses. [22]

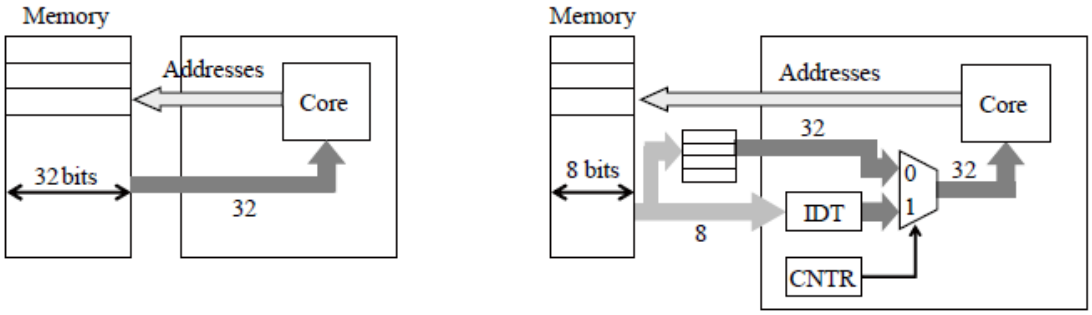
5.7.6 Επιλεκτική συμπίεση κώδικα, μία εναλλακτική προσέγγιση στην κωδικοποίηση διαύλου

Η βασική στρατηγική είναι να χρησιμοποιηθούν κωδικοί εντολών με μικρό εύρος bit(narrow bit-width) για τη μείωση της δραστηριότητας μεταγωγής στον δίαυλο, όταν οι εντολές μεταδίδονται από μια μνήμη προγράμματος σε μια CPU.

Μία από τις πιο γνωστές προσεγγίσεις συμπίεσης εντολών είναι το σύνολο εντολών "Thumb" της οικογένειας μικροεπεξεργαστών ARM. Οι πυρήνες ARM μπορούν να

προγραμματιστούν χρησιμοποιώντας ένα μειωμένο σύνολο εντολών 16-bit αντί των τυπικών risc εντολών 32-bit, το οποίο μειώνει τη μνήμη που καταλαμβάνουν οι εντολές κατά τον συντελεστή 2.

Η τεχνική της επιλεκτικής συμπίεσης εντολών, βασίζεται στη συμπίεση μόνο ενός σταθερού υποσυνόλου 256 στοιχείων των εντολών που χρησιμοποιούνται από ένα πρόγραμμα, δηλαδή εκείνων που εκτελούνται πιο συχνά. Αυτή η προσέγγιση βασίζεται στη παρατήρηση ότι οι 256 πιο συχνά χρησιμοποιούμενες εντολές, πάντα εκτελούνται για τουλάχιστον 50% έως και 99,99% του χρόνου. Η τεχνική μπορεί να εφαρμοστεί όπως φαίνεται στο σχήμα 5.7. [17]



Σχήμα 5.7: Επιλεκτική συμπίεση κώδικα.

5.7.7 Ελαχιστοποίηση του ενεργειακού κόστους των προσβάσεων στη μνήμη

Δεδομένου ότι η μνήμη αντιπροσωπεύει συχνά ένα μεγάλο τμήμα της καταναλισκόμενης ισχύος ενός συστήματος, υπάρχει ένα σαφές κίνητρο για την ελαχιστοποίηση του κόστους ισχύος της μνήμης. Οι τεχνικές σχεδιασμού λογισμικού που έχουν χρησιμοποιηθεί για τη βελτίωση της απόδοσης του συστήματος μνήμης, μπορεί επίσης να είναι χρήσιμες στη μείωση της κατανάλωσης ισχύος ενός προγράμματος. Εάν οι προσβάσεις στη μνήμη είναι τόσο αργές όσο και απαιτητικές σε ισχύ, τότε οι τεχνικές λογισμικού που ελαχιστοποιούν τις προσβάσεις στη μνήμη, τείνουν να έχουν και απόδοση και ένα όφελος ισχύος. Οι χαμηλότερες απαιτήσεις μνήμης, βοηθούν στο να είναι η συνολική μνήμη RAM μικρότερη και βελτιώνουν το ποσοστό των λειτουργιών που χρειάζονται πρόσβαση μόνο στους καταχωρητές ή τη μνήμη cache. Οι τεχνικές ελαχιστοποίησης της ισχύος που σχετίζονται με τη μνήμη, επικεντρώνονται σε έναν ή περισσότερους από τους ακόλουθους στόχους:

- **Ελαχιστοποίηση του αριθμού των προσβάσεων (read-write) στη μνήμη** που απαιτούνται από έναν αλγόριθμο.

- **Ελαχιστοποίηση της συνολικής μνήμης** που απαιτείται από έναν αλγόριθμο.

- **Υλοποίηση των προσβάσεων στη μνήμη όσο το δυνατόν πιο κοντά στον επεξεργαστή**, πρώτα στους καταχωρητές, στη συνέχεια στη cache (με βελτιστοποίηση των ποσοστών επιτυχίας του cache hit ratio) και τέλος στη RAM.

- **Βελτιστοποιήσεις βρόχων** με τη χρήση του **loop unrolling**, - όπου στόχος είναι να μειωθεί ο συνολικός αριθμός των συγκρίσεων σε ένα πρόγραμμα. Το αποτέλεσμα που αναφέρθηκε ήταν η χαμηλή χρήση της ALU -απόδοση 20%- με μειονέκτημα την αύξηση του μεγέθους του κώδικα πάνω από 10% - και του **loop fusion** που επίσης μειώνει την κατανάλωση ισχύος της πρόσβασης στη μνήμη. [1], [22]

- **Αποτελεσματική χρήση του διαθέσιμου bandwidth μνήμης** με όσο το δυνατό μεγαλύτερη παράλληλη φόρτωση πολλαπλών λέξεων αντί της φόρτωσης μονών λέξεων. Η μεγιστοποίηση της **παράλληλης φόρτωσης πολλαπλών λέξεων** από τη μνήμη είναι μία από τις τεχνικές για τη μείωση του ενεργειακού κόστους. Η διπλή φόρτωση πλεονεκτεί όσον αφορά την ενεργειακή μείωση αλλά όχι απαραίτητα για τη μείωση ισχύος, επειδή η στιγμιαία κατανάλωση ισχύος μιας διπλής φόρτωσης είναι κάπως υψηλότερη από τη κατανάλωση δύο μονών φορτώσεων. Έχει αποδειχτεί, ότι η εξοικονόμηση ενέργειας φτάνει μέχρι και 47% με τη μεγιστοποίηση των διπλών φορτώσεων.

Όσον αφορά τις πρακτικές που ελαχιστοποιούν τις απαιτήσεις για το bandwidth της μνήμης, αυτές είναι:

- **Κατανομή των καταχωρητών (register allocation)** ώστε να ελαχιστοποιηθούν οι εξωτερικές αναφορές μνήμης.

- **Αποκλεισμός της κρυφής μνήμης (cache blocking),** δηλαδή μετασχηματισμοί στους υπολογισμούς πινάκων έτσι ώστε μπλοκ στοιχείων των πινάκων να διαβάζονται μόνο μία φορά στην κρυφή μνήμη.

- **Αποκλεισμός καταχωρητών (register blocking),** παρόμοιος με το cache blocking, εκτός του ότι οι πλεονάζουσες φορτώσεις στοιχείων των πινάκων στον καταχωρητή διαγράφονται.

- **Συμπαγείς πολλαπλές αναφορές στη μνήμη σε μια ενιαία αναφορά**. [6]

5.8 Πρακτικές και μέθοδοι για την υλοποίηση low power λογισμικού

Αύξηση της ενεργειακής απόδοσης επιτυγχάνεται με τη χρήση μεθόδων και πρακτικών όπως, ο παράλληλος προγραμματισμός **(Parallel Programming),** το **Approximate Computing**, η ανάλυση πηγαίου κώδικα **(Source Code Analysis),** η χρήση των κατάλληλων γλωσσών προγραμματισμού **(Programming Languages**), των δομών δεδομένων **(Data Structures)** και των πρακτικών κωδικοποίησης **(Coding Practices).** Επιπλέον, η χρήση software & hardware energy monitoring εργαλείων και των benchmarks για την αξιολόγηση της κατανάλωσης ενέργειας του λογισμικού, μπορούν να παρέχουν πληροφορίες στους προγραμματιστές κατά τη διάρκεια της φάσης ανάπτυξης. Επιπρόσθετα, κατά τη συντήρηση (maintenance) του λογισμικού, εφαρμόζονται refactoring πρότυπα και τεχνικές, με στόχο τη βελτίωση όσον αφορά τη μείωση της κατανάλωσης ενέργειας διατηρώντας παράλληλα τη λειτουργικότητά του. [19], [20]

5.8.1 Παράλληλος προγραμματισμός (Parallel programming)

Ο παράλληλος προγραμματισμός, είναι η διαδικασία του διαχωρισμού ενός μεγάλου προβλήματος σε μικρότερα για την ταυτόχρονη επίλυσή τους. Με άλλα λόγια μια διαδικασία (process), σπάει σε μικρότερα τμήματα-νήματα (threads) για την ταυτόχρονη επεξεργασία τους. Στον παράλληλο προγραμματισμό, αναπτύσονται εφαρμογές που εκμεταλλεύονται την ύπαρξη πολλαπλών επεξεργαστικών μονάδων σε έναν πολυεπεξεργαστή για να επιτύχουν αύξηση των υπολογιστικών επιδόσεων και μείωση του απαιτούμενου χρόνου εκτέλεσης της εφαρμογής. [15] Η κατανάλωση ενέργειας αυξάνεται καθώς αυξάνεται ο αριθμός των νημάτων και αυτό συμβαίνει έως ότου ο αριθμός των νημάτων φτάσει στον αριθμό των πυρήνων της cpu. [19]

Ένας αλγόριθμος παραλληλοποιείται με τη διάσπασή του σε πολλαπλά τμήματα, τα οποία ανατίθενται σε ξεχωριστά νήματα ή διεργασίες και έτσι εκτελούνται παράλληλα σε διαφορετικές επεξεργαστικές μονάδες. Ωστόσο δεν είναι βέβαιο ότι ένας αλγόριθμος, υλοποιημένος σε κάποιο πρόγραμμα μπορεί να παραλληλοποιηθεί πάντα: μία υπορουτίνα η οποία υπολογίζει την ακολουθία Φιμπονάτσι μέσω ενός επαναληπτικού βρόχου δεν μπορεί να διασπαστεί σε πολλαπλά τμήματα με διαμοίραση των επαναλήψεων του βρόχου σε μικρότερους υποβρόχους, αφού ο υπολογισμός ο οποίος γίνεται σε κάθε επανάληψη εξαρτάται από αυτούς των δύο προηγούμενων επαναλήψεων και επομένως όλες οι επαναλήψεις πρέπει να εκτελεστούν σειριακά στο εσωτερικό μίας μόνο διεργασίας. Τέτοιου τύπου εξαρτήσεις είναι το σημαντικότερο εμπόδιο για την παραλληλοποίηση. Ωστόσο ακόμα και στην περίπτωση μη παραλληλοποιήσιμου προγράμματος η χρήση διαφορετικών νημάτων ή διεργασιών μπορεί να είναι ευεργετική αν ο προγραμματιστής επιθυμεί να επικαλύψει υπολογισμούς με επικοινωνίες (π.χ. ένα νήμα να συνεχίζει τους υπολογισμούς όσο το άλλο αναμένει είσοδο από το δίκτυο ή από τον χρήστη).

Στις περισσότερες περιπτώσεις ένα παράλληλο πρόγραμμα έχει σχεδιαστεί κατάλληλα και ο πηγαίος κώδικας έχει γραφεί από τον προγραμματιστή έτσι ώστε να υποστηρίζει παραλληλισμό. Επειδή η διαδικασία αυτή είναι επίπονη, χρονοβόρα και απαιτεί ειδικές γνώσεις εκ μέρους του προγραμματιστή, χρησιμοποείται η αυτόματη παραλληλοποίηση τμημάτων σειριακού πηγαίου κώδικα (συνήθως επαναληπτικών βρόχων) από κατάλληλους μεταγλωττιστές. Στην περίπτωση αυτή ο προγραμματιστής συγγράφει τυπικό σειριακό κώδικα και κατά τη μεταγλώττιση εντοπίζονται αυτομάτως τα σημεία που επιδέχονται παραλληλισμό. Οι περισσότεροι μεταγλωττιστές σήμερα υποστηρίζουν ανάλογες μεθόδους, ωστόσο ο κώδικας ο οποίος προκύπτει συνήθως δεν εκτελείται εξίσου γρήγορα σε σχέση με τον κώδικα που έχει εξαρχής γραφτεί να υποστηρίζει παραλληλισμό –ίσως μάλιστα να εκτελείται πιο αργά και από τον σειριακό κώδικα– ή παράγει λανθασμένα αποτελέσματα. [15]

Πίνακας 5.1 Η επιρροή του parallel programming σε σχέση με την ενέργεια και την απόδοση

Thread Management Strategies	Adjustments	Implications (in %) Energy	Performance	Application Type	Source
Effective Parallelization[c]	From 1 to 16 Threads	55 avg.	69 avg.	CPU & GPU bound[d]	(Kambadur and Kim 2014)
Effective Parallelization & Thread Management Constructs	Explicit Threading Work Stealing	– (−50) – 30	– (−23) – 10	I/O-bound[e] Embarrassingly Parallel[f]	(Pinto et al. 2014)
Pack & Cap[a]	Criticality level DVFS	56 avg.	Unaffected	CPU-bound[b]	(Cai et al. 2011)
Work Stealing[g]	Load-base DVFS	11–12	(−3)–(−4)	CPU & GPU bound[h]	(Ribic and Liu 2014a)

[a] Collecting threads under the same core and reducing its frequency.
[b] Recognition-Mining-Synthesis (RMS), an Intel's application (Chen et al. 2008).
[c] Statically selecting the effective number of threads.
[d] Parsec 3.0, SPLASH-2x, SPEC CPU 2006, DaCapo 9.12, SPEC JBB 2013.
[e] Application such as *Largestimage*.
[f] Applications such as *Sunflow, Spectralnorm, N-Queens, Tomcat*.
[g] *"Underutilized processors take the initiative: they attempt to steal threads from other processors"* (Blumofe and Leiserson 1999).
[h] Applications from Problem-based benchmark suite (Shun et al. 2012).

Εμπειρικές μελέτες

Με την εκτέλεση μιας εμπειρικής μελέτης, οι Kambadur και Kim (2014) προσδιόρισαν τις διαμορφώσεις και παράμετρους που μπορούν να μειώσουν την κατανάλωση ενέργειας των παράλληλων εφαρμογών.

Οι συγγραφείς συνέκριναν εννέα υπάρχουσες στρατηγικές διαχείρισης της ενέργειας, χρησιμοποιώντας μία τυποποιημένη αρχιτεκτονική συστήματος: λειτουργικό σύστημα (os), εργαλείο μέτρησης και πέντε benchmark σουίτες. Οι στρατηγικές (π.χ. processor frequency tuning, overclocking, parallelism και βελτιστοποιήσεις στον μεταγλωττιστή), εκτελέστηκαν σε 220 πειραματικά configurations και δοκιμάστηκαν σε 41 εφαρμογές, με

περισσότερες από 200.000 εκτελέσεις. Τα ληφθέντα αποτελέσματα υπογραμμίζουν τη σημασία του αποτελεσματικού παραλληλισμού στον πηγαίο κώδικα, χρησιμοποιώντας τον κατάλληλο αριθμό νημάτων (πίνακας 5.1).

Για να αξιολογηθεί η ενεργειακή απόδοση των διαφορετικών υλοποιήσεων διαχείρισης νημάτων, ο Pinto (2014) πραγματοποίησε μια εμπειρική μελέτη που συνέκρινε τρείς υλοποιήσεις (explicit thread creation, fixed-size thread pooling και work stealing) σε εννέα benchmarks, με τη ρύθμιση του αριθμού των νημάτων, το επίπεδο κοκκιότητας της εργασίας, το μέγεθος των δεδομένων και τη φύση της πρόσβασης στα δεδομένα. Αυτό που παρατηρήθηκε ήταν ότι η κατανάλωση ενέργειας των υλοποιήσεων, ποικίλλει σε διαφορετικές καταστάσεις. Επίσης παρατηρήθηκε ότι η κατανάλωση ενέργειας αυξάνεται καθώς ο αριθμός των νημάτων αυξάνεται και αυτό συμβαίνει έως ότου ο αριθμός των νημάτων φτάσει στον αριθμό των πυρήνων της cpu. Στη συνέχεια, παρατηρείται μείωση της κατανάλωσης ενέργειας για τις περισσότερες υπό δοκιμή εφαρμογές. [19]

Όπως φαίνεται από τον πίνακα 5.1, τα κέρδη και οι ζημίες που αφορούν την κατανάλωση ενέργειας και την απόδοση του χρόνου εκτέλεσης είναι εφικτά επιλέγοντας τον κατάλληλο αριθμό νημάτων. Για παράδειγμα, τόσο ο Pinto και οι Kambadur & Kim, ρυθμίζουν τον αριθμό των νημάτων για να αξιολογήσουν τις εφαρμογές τους. Χρησιμοποιώντας διαφορετικές υλοποιήσεις διαχείρισης νήματος και διαφορετικό αριθμό νημάτων, ο Pinto τροποποίησε την κατανάλωση ενέργειας από -50% έως 30%, ενώ οι Kambadur & Kim τη μείωσαν κατά 55% κατά μέσο όρο. Η σημαντική διαφορά μεταξύ των δύο μελετών που μπορεί να εξηγήσει τις αποκλίσεις στα αποτελέσματα, είναι ότι ο Pinto χρησιμοποίησε εφαρμογές Java, ενώ οι Kambadur & Kim χρησιμοποίησαν Java και C.

Όπως φαίνεται και από τα παραπάνω, πραγματοποιώντας πειράματα με διαφορετικό αριθμό νημάτων ή υλοποιήσεις διαχείρισης νημάτων, μπορούν να βοηθηθούν οι προγραμματιστές στο να προσδιορίσουν εκείνες τις ρυθμίσεις που είναι πιθανότερο να μειώσουν την κατανάλωση ενέργειας και τον χρόνο εκτέλεσης των εφαρμογών τους.

Αλγόριθμοι (Algorithms)

Προκειμένου να αναπτυχθούν αποδοτικότεροι αλγόριθμοι -από άποψη κατανάλωσης ενέργειας- είναι απαραίτητο να κατανοηθούν εκ των προτέρων οι πιθανές καταστάσεις που μπορεί να προκύψουν κατά τη διάρκεια του σταδίου ανάπτυξης και μπορεί να προκαλέσουν είτε σχετική αύξηση είτε μείωση της κατανάλωσης ενέργειας του λογισμικού. Ωστόσο, δεδομένης της μεγάλης πολυπλοκότητας των μεταβλητών, που μπορούν να επηρεάσουν αρνητικά ή θετικά την κατανάλωση, είναι απαραίτητο να χρησιμοποιηθούν προσεκτικές προσεγγίσεις που επιτρέπουν αξιόπιστες συγκρίσεις και καλά συμπεράσματα κατά τη διάρκεια ή ακόμα και πριν από τη φάση ανάπτυξης. Η μέτρηση της κατανάλωσης ενέργειας που προκύπτει από διάφορες συμπεριφορές ενός αλγορίθμου δεν είναι μια τετριμμένη

εργασία. Υπάρχουν πολλά σχετικά ζητήματα που πρέπει να εξεταστούν, από την απομόνωση των μεταβλητών στο πείραμα, μέχρι τα προβλήματα που προκαλούνται από την παρεμβολή της ίδιας της διαδικασίας μέτρησης. Οποιαδήποτε από αυτές τις καταστάσεις μπορεί να οδηγήσει σε ασαφή αποτελέσματα. Εκ πρώτης όψεως, θα μπορούσε κανείς να πει ότι το υψηλό κόστος χρόνου εκτέλεσης, θα είχε αναλογικά και υψηλότερο κόστος κατανάλωσης ενέργειας. Ωστόσο, η βιβλιογραφία δείχνει ότι υπάρχουν παράγοντες που μπορούν να κάνουν ένα σύστημα να αντιδρά διαφορετικά όσον αφορά την κατανάλωση ενέργειας. [10]

Για να είναι επωφελής ο παράλληλος προγραμματισμός, οι Cai (2011) και οι Ribic & Liu (2014b) ανέπτυξαν αλγόριθμους για την ελαχιστοποίηση της κατανάλωσης ενέργειας, με αποτελεσματική διαχείριση του φόρτου εργασίας των νημάτων. Το μεγαλύτερο μέρος των DVFS τεχνικών, δεν είναι σχεδιασμένες για να τρέχουν σε πολυνηματικούς επεξεργαστές και επομένως, δεν είναι σε θέση να εξοικονομήσουν μια σημαντικά μεγάλη ποσότητα ενέργειας. Για το σκοπό αυτό, προτάθηκε η χρησιμοποίηση τεχνικών όπως το thread shuffling και η Hermes.

Η τεχνική thread shuffling, συνδυάζει τεχνικές όπως το thread migration και τη dvfs, για να μειώσει την κατανάλωση ενέργειας μια εφαρμογής χωρίς να διακυβεύεται η απόδοση του χρόνου εκτέλεσης. Η βασική ιδέα πίσω από το thread shuffling είναι να προσδιοριστούν τα νήματα με τον ίδιο βαθμό κρισιμότητας νήματος (αργή εκτέλεση νημάτων) χρησιμοποιώντας ένα αλγόριθμο πρόβλεψης και η αντιστοίχισή τους (map) κάτω από τον ίδιο πυρήνα μέσω toy thread migration. Στη συνέχεια, ο αλγόριθμος εφαρμόζει δυναμικά την τάση για να κλιμακώσει τη συχνότητα εκείνων των πυρήνων που επεξεργάζονται μη-κρίσιμα νήματα (γρήγορη εκτέλεση νημάτων).

Η τεχνική Hermes των Ribic and Liu, χρησιμοποιεί μια στρατηγική για την κλοπή εργασίας (work-stealing), εφαρμόζοντας μία thief-victim προσέγγιση. Ως κλέφτης αναφέρεται το νήμα που ολοκληρώνει τα καθήκοντά του και κλέβει εργασία από άλλα νήματα, τα λεγόμενα θύματα.

Και στις δύο παραπάνω τεχνικές, χρησιμοποιείται η τεχνική dvfs με το workload migration για την επίτευξη εξοικονόμησης ενέργειας. Το thread shuffling, ομαδοποιεί όλα τα νήματα με παρόμοιο επίπεδο κρισιμότητας κάτω από συγκεκριμένους πυρήνες και ρυθμίζει έπειτα τη συχνότητα ρολογιών των πυρήνων αναλόγως, ενώ η τεχνική Hermes μεταβάλλει τη συχνότητα ρολογιού κάθε πυρήνα με βάση τον τρέχοντα φόρτο εργασίας των νημάτων. Ιδιαίτερα, η προσέγγιση του work-stealing της τεχνικής Hermes, είχε ως αποτέλεσμα μικρή εξοικονόμηση ενέργειας σε σχέση με το thread shuffling που οδήγησε σε 56% εξοικονόμηση ενέργειας κατά μέσο όρο, όπως απεικονίζεται στον πίνακα 5.1. Συμπερασματικά, η τεχνικλη Hermes παρείχε εξοικονόμηση ενέργειας με μια μικρή ποινή στην απόδοση του χρόνου εκτέλεσης (δηλ., 3-4%), ενώ το thread shuffling πέτυχε δραστική εξοικονόμηση ενέργειας, χωρίς να υπάρξει συμβιβασμός στην απόδοση του χρόνου εκτέλεσης (βλέπε πίνακα 3). [19]

5.8.2 Approximate computing (AC)

Το approximate computing (and storage), βασίζεται στην παρατήρηση ότι ενώ η εκτέλεση ακριβούς υπολογισμού απαιτεί υψηλό ποσό πόρων, επιτρέποντας μια πιο επιλεκτική προσέγγιση, μπορεί να προσφέρει μεγαλύτερα κέρδη στην αποδοτικότητα. Είναι μια τεχνική που θυσιάζει την ακρίβεια υπολογισμού -όταν το επιτρέπει η εφαρμογή- για να αυξήσει την απόδοση του χρόνου εκτέλεσης ή / και την εξοικονόμηση ενέργειας. [14], [19] Για παράδειγμα, ο αλγόριθμος ομαδοποίησης k-means, μπορεί να πετύχει έως και 50x εξοικονόμηση ενέργειας, επιτρέποντας απώλεια στην ακρίβεια ταξινόμησης 5% και το loop perforation (πίνακας 5.2) το οποίο δίνει τη δυνατότητα για την τακτική παράλειψη ορισμένων επαναλήψεων ενός βρόχου, επιτρέπει τη διαχείριση της απόδοσης του χρόνου εκτέλεσης και την ακρίβεια στο επιθυμητό αποτέλεσμα. [19] Έτσι το AC αξιοποιεί την παρουσία περιοχών κώδικα ανεκτικών σε σφάλματα στις εφαρμογές και τους περιορισμούς των χρηστών, για να επιτύχει κέρδη απόδοσης ή ενεργειακά κέρδη. Εν συντομία, το AC, εκμεταλλεύεται το κενό μεταξύ του επιπέδου ακρίβειας που απαιτείται από τις εφαρμογές/χρήστες και εκείνου που παρέχεται από το σύστημα επεξεργασίας, για την επίτευξη ποικίλων βελτιστοποιήσεων.

Ωστόσο, αν και πολλά υποσχόμενο, το AC δεν είναι πανάκεια. Η αποτελεσματική χρήση του AC απαιτεί τη συνετή επιλογή τμημάτων προσεγγιστικού κώδικα/δεδομένων και κάποιας προσεγγιστικής στρατηγικής, καθώς το ομοιόμορφο AC, μπορεί να προκαλέσει απώλεια ποιότητας. Ακόμη χειρότερα, το AC στις λειτουργίες ελέγχου ροής ή πρόσβασης μνήμης, μπορεί να οδηγήσει σε καταστροφικά αποτελέσματα, όπως το σφάλμα τμηματοποίησης (segmentation fault). [13]

Programming Frameworks

Για την έρευνά τους, ο Misailovic (2014) και οι Baek & Chilimbi (2010), εισήγαγαν μη ενεργοβόρα frameworks, για την επίτευξη εξοικονόμησης ενέργειας μέσω προσεγγιστικών υπολογισμών. Συγκεκριμένα, ο Misailovic πρότεινε το Chisel, ένα βελτιστοποιημένο framework, που λειτουργεί με αυτοματοποιημένο τρόπο επιλέγοντας κατά προσέγγιση λειτουργίες του πυρήνα, που οδηγούν σε βελτιστοποιήσεις ενέργειας, αξιοπιστίας και ακρίβειας. Οι Baek και Chilimbi πρότειναν το Green, ένα framework με στόχο τη βελτιστοποίηση των «ακριβών» ενεργειακά βρόχων και λειτουργιών, λαμβάνοντας υπόψη τις QoS απαιτήσεις. Το Green, επιτυγχάνει εξοικονόμηση ενέργειας μέσω προσεγγιστικών υπολογισμών για τις λειτουργίες και την πρόωρη λήξη βρόχων. Επιπλέον, αντισταθμίζει το QoS και την κατανάλωση ενέργειας, με την εφαρμογή προσεγγιστικών τεχνικών προγραμματισμού μόνο όταν οι QoS απαιτήσεις πληρούνται. Το κοινό στοιχείο τόσο του Green όσο και του Chisel framework, είναι η σύγκριση μεταξύ του ακριβούς και του κατά προσέγγιση στιγμιότυπου (instance), για τον υπολογισμό της αξιοπιστίας των αποτελεσμάτων (βλέπε πίνακα 5.2).

Annotation - based extensions

Οι Sampson (2011), Esmaeilzadeh (2012) και Yazdanbakhsh (2015) πρότειναν τα EnerJ, Parrot και Axilog, αντίστοιχα. Όλα είναι επεκτάσεις που επιτυγχάνουν εξοικονόμηση ενέργειας εκτελώντας προσεγγιστικά annotated source code. [19]

Και τα τρία annotation συστήματα, προσφέρουν έναν μηχανισμό ασφαλείας για την απομόνωση των κατά προσέγγιση από τα ακριβή τμήματα του κώδικα, εξασφαλίζοντας έτσι την κύρια λειτουργικότητα μιας εφαρμογής. Σε αντίθεση με τα Green, Chisel και Axilog, με τα EnerJ και Parrot οι προγραμματιστές καθορίζουν τις εφαρμοζόμενες τεχνικές προσέγγισης, επιλέγοντας ποια τμήματα κώδικα θα εκτελεστούν επακριβώς.

Directive - based extensions

Μια άλλη προσέγγιση για την εφαρμογή τεχνικών προσέγγισης στις υπολογιστικές εργασίες, είναι να εξαρτάται από το επίπεδο σπουδαιότητάς τους (ή κρισιμότητας). Σε αυτή τη γραμμή, ο Vassiliadis (2016a, 2016b) πρότεινε directive-based προσεγγίσεις, επεκτείνοντας το OpenMP που χρησιμοποιεί τεχνικές προσέγγισης για τη μείωση της κατανάλωσης ενέργειας των εφαρμογών. Ο Vassiliadis (2016a), πρότεινε ένα μοντέλο προγραμματισμού με στόχο την επίτευξη του υψηλότερου επίπεδου ακρίβειας για μια εφαρμογή, σύμφωνα με το energy-budget που καθορίζεται από το χρήστη. Ως εκ τούτου, εισήγαγε ένα runtime σύστημα που είναι υπεύθυνο για την επιλογή των κατάλληλων configurations (δηλαδή αριθμός πυρήνων, συχνότητα ρολογιών) για συγκεκριμένη είσοδο. Οι κατάλληλες διαμορφώσεις, συνάγονται από ένα μοντέλο που έχει "εκπαιδευτεί" για να εντοπίζει τις διαμορφώσεις εκείνες που παρέχουν την υψηλότερη δυνατή ακρίβεια εξόδου για ένα δεδομένο budget ενέργειας.

Μια άλλη προσέγγιση είναι η DCO / SCORPIO, ένα framework που προτείνεται από τον Vassiliadis (2016b), που υποστηρίζει αυτοματοποιημένη ανάλυση για τον προσδιορισμό του επιπέδου σημαντικότητας του κώδικα. [19]

Memoization

Μια άλλη προσέγγιση της εξοικονόμησης ενέργειας μέσω του κατά προσέγγιση υπολογισμού, είναι η χρήση της απομνημόνευσης (memoization) για την αποθήκευση αποτελεσμάτων από την επεξεργασία αργών και ενεργειακά «ακριβών» κλήσεων μιας συνάρτησης, για την αποφυγή περιττών υπολογισμών. Η απομνημόνευση, λειτουργεί με την αποθήκευση των αποτελεσμάτων των συναρτήσεων σε ένα πίνακα, για μεταγενέστερη επαναχρησιμοποίηση τους σε μιά επόμενη κλήση με τα ίδια ορίσματα, τα οποία ανακτώνται από τη μνήμη cache για να επιταχυνθεί ο χρόνος εκτέλεσης των προγραμμάτων, [11], [12], [13] υπό την προϋπόθεση ότι ο χρόνος αναζήτησης μιας προηγουμένως υπολογισμένης τιμής, θα είναι μικρότερος από τον χρόνο για τον

υπολογισμό της. Ο στόχος της απομνημόνευσης είναι να μειωθεί ο χρόνος εκτέλεσης με αντάλλαγμα τον χρησιμοποιούμενο χώρο μνήμης. [12]

Ο Agosta (2011) ανέπτυξε ένα μοντέλο απόδοσης για να επιλέγονται και να απομνημονεύονται εντατικές υπολογιστικές λειτουργίες. Σε σύγκριση με τις παραπάνω προσεγγίσεις, η τεχνική του memoization φαίνεται να έχει την υψηλότερη εξοικονόμηση ενέργειας και απόδοση χρόνου εκτέλεσης (βλέπε πίνακα 5.2).

Πίνακας 5.2. Η επιρροή του approximate computing σε σχέση με την ενέργεια και την απόδοση

Tool Name	Approximate Optimization Focus	Implications (in %) Energy	Performance	Precision Loss	Source
GREEN	Computations and loop termination	14 avg.	21 avg.	0.27	(Baek and Chilimbi 2010)
–	Function memoization	74 avg.	79 avg.	<3	(Agosta et al. 2011)
Parrot	Optimizes regions of imperative code	66 avg.	56 avg.	10 avg.	(Esmaeilzadeh et al. 2012)
Chisel	Computational kernel operations	9–20	–	<3	(Misailovic et al. 2014)
EnerJ	Computations, Data Storage, and Algorithmic[a]	10–50	–	–	(Sampson et al. 2011)
Axilog	Source code parts	54 avg.	–	10 avg.	(Yazdanbakhsh et al. 2015)
–	Selected group of tasks[b]	–	–	–	(Vassiliadis et al. 2016a)
DCO SCORPIO	Selected group of tasks[b]	56 avg.	–	–	(Vassiliadis et al. 2016b)

[a]Programmer can write two different implementations, one is invoked when the data are precise and the other when they are approximate.
[b]Allows the developer to set which computation tasks, from a group, are going to be executed approximately/precise.

5.8.3 Ανάλυση του πηγαίου κώδικα (Source code analysis)

Η ανάλυση πηγαίου κώδικα, είναι μια διαδικασία δοκιμών που επικεντρώνεται στην ανεύρεση ελαττωμάτων και τρωτών σημείων σε ένα πρόγραμμα υπολογιστή πριν από τη φάση της ανάπτυξής του. Ο πίνακας 5.3 παρουσιάζει τα αποτελέσματα εργασιών της ανάλυσης πηγαίου κώδικα και παρέχει τις πληροφορίες για την πλατφόρμα χρήσης, τη συσχέτιση της μετρούμενης ενέργειας και το ποσοστό του περιθωρίου σφάλματος.

Κλήσεις συστήματος (System calls)

Το GreenAdvisor είναι ένα σύστημα ανάλυσης, το οποίο προβλέπει τη συμπεριφορά, την απόδοση του χρόνου εκτέλεσης και τις τροποποιήσεις που σχετίζονται με την

ενέργεια μιας εφαρμογής (Aggarwal 2015). Για να προβλέψει αλλαγές που σχετίζονται με την ενέργεια, το GreenAdvisor συγκρίνει τον αριθμό των κλήσεων συστήματος σε μια τρέχουσα έκδοση ενός λογισμικού και στη προηγούμενη. Εάν η κατανάλωση ενέργειας αυξάνεται, τότε εντοπίζει τα ενεργειακά hot-spots που προκάλεσαν τις αλλαγές, βοηθώντας έτσι τους προγραμματιστές στην ανάλυση και κατανόηση των συνεπειών των αποφάσεών τους.

Για τον εντοπισμό ενεργειακών bottlenecks σε επίπεδο πηγαίου κώδικα, ο Pathak (2012) παρουσίασε το Eprof, ένα εργαλείο μοντελοποίησης της ισχύος βασισμένο στις κλήσεις συστήματος για εφαρμογές smartphones. [19]

Σε γενικές γραμμές, τα παραπάνω εργαλεία χρησιμοποιούν ξεχωριστά μοντέλα εκτίμησης της ενέργειας για να παρουσιάσουν μετρήσεις ενέργειας. Για παράδειγμα, το Eprof χρησιμοποιεί ένα μοντέλο που υπολογίζει την κατανάλωση ενέργειας μιάς εφαρμογής σε επίπεδο ρουτίνας και στα διάφορα hardware στοιχεία, ενώ το GreenAdvisor συγκρίνει τη κατανάλωση ενέργειας των διαφόρων εκδόσεων ενός προϊόντος και επισημαίνει τις κλήσεις συστήματος που προκάλεσαν την αλλαγή.

Εργαλεία βελτιστοποίησης (Optimization tools)

Οι Honig (2014) και Manotas (2014) πρότειναν εργαλεία για τη δυναμική ανάλυση του πηγαίου κώδικα. Και οι δύο προτείνουν energy-aware προγραμματιστικές προσεγγίσεις, για την καθοδήγηση των προγραμματιστών κατά τη διάρκεια της φάσης υλοποίησης, παρέχοντας συμβουλές σχετικές με την ενέργεια. Ο Honig παρουσίασε το Proactive Energy-awarE development Kit (PEEK), ενώ ο Manotas προώθησε το Software Engineer's Energy-optimization Decision Support (SEEDS) framework. Περιοδικά, και οι δύο προσεγγίσεις αναλύουν τον υπό ανάπτυξη πηγαίο κώδικα και δημιουργούν συνεχώς πολλά διαφορετικά στιγμιότυπα (instances) από τον τρέχοντα πηγαίο κώδικα για να εντοπίσουν τις βέλτιστες τροποποιήσεις. Ωστόσο, μια σημαντική διαφορά μεταξύ των δύο προσεγγίσεων είναι ότι οι σχετικές με την ενέργεια υποδείξεις του PEEK, σχετίζονται με μηχανισμούς διαχείρισης ενέργειας (π. χ. κατάσταση ύπνου, DVFS, κατάσταση αδράνειας, λογική κώδικα προγράμματος, βιβλιοθήκες) ενώ οι προτάσεις του SEEDS, περιορίζονται στις Java Collection Libraries (JCL), σε αλγόριθμους ή refactoring τμήματα του πηγαίου κώδικα.

Tests generation framework

Ένας άλλος τρόπος για τον εντοπισμό σφαλμάτων που σχετίζονται με την κατανάλωση ενέργειας, είναι μέσω ενός αυτοματοποιημένου πλαισίου παραγωγής δοκιμών, που εισήχθη από τον Banerjee (2014). Η προαναφερθείσα έρευνα, επισημαίνει τα hot-spots που σχετίζονται με την ενέργεια σε τέσσερις κατηγορίες εφαρμογών smartphones που είναι: 1) οι hardware πόροι, 2) οι μεταβάσεις κατάστασης ύπνου, 3) οι υπηρεσίες παρασκηνίου και 4) η ελαττωματική λειτουργικότητα. Με την ενεργειακή αναγνώριση των hot-spots, το εργαλείο εκδίδει αναφορές δοκιμών για τους

προγραμματιστές. Ωστόσο, σε σύγκριση με το έργο των Honig και Manotas, το έργο του Banerjee δεν παρέχει συμβουλές για τη βελτιστοποίηση της ενέργειας, αλλά βρίσκει μόνο τα ενεργοβόρα μέρη μιας εφαρμογής. [19]

Line by line

Για τη μέτρηση της κατανάλωσης ενέργειας εφαρμογών και για την αύξηση της ενεργειακής ευαισθητοποίησης κατά τη διάρκεια της φάσης ανάπτυξης, ο Hao πρότεινε το eLens. Το eLens, εκτιμά την κατανάλωση ενέργειας μέσω της ανάλυσης του προγράμματος και της μοντελοποίησης της ισχύος ανά εντολή. Χρησιμοποιείται η ανάλυση προγράμματος για να αποκτηθούν πληροφορίες σχετικές με την εκτέλεση, όπως bytecode ή api κλήσεις από διάφορα στοιχεία του smartphone (π.χ. cpu, ram, gps). Στη συνέχεια, οι πληροφορίες που συλλέγονται μεταβιβάζονται στο στοιχείο μοντελοποίησης ισχύος ανά εντολή, για την εκτίμηση της κατανάλωσης ενέργειας μιας εφαρμογής. Έτσι το eLens, παρέχει μετρήσεις ενέργειας σε διάφορα επίπεδα όπως: εφαρμογή, μέθοδος, class, διαδρομή και γραμμές πηγαίου κώδικα. Σε σύγκριση με όλα τα εργαλεία που παρουσιάζονται, το eLens είναι το μόνο που παρέχει μετρήσεις ενέργειας σε όλα τα επίπεδα του πηγαίου κώδικα. [19]

Πίνακας 5.3: Source code analysis

Tool Name	Target Platform	Energy Measurement Correlation	Error Margin (in %)	Source
Eprof	Android & Windows os	Process, Threads, System calls, Routines	6<	(Pathak et al. 2012)
eLens	Android	Full Source Code Granularity	10	(Hao et al. 2013)
GreenAdvisor	Android	Routines, System-calls	–	(Aggarwal et al. 2015)
PEEK	Embedded	Function Level Systems	–	(Honig et al. 2014)
SEEDS	Any Java-based Platform	User can set which portion to analyze	–	(Manotas et al. 2014)
–	Smart-phones	Function Level	–	(Banerjee et al. 2014)

5.8.4 Οι γλώσσες προγραμματισμού (Programming languages)

Οι γλώσσες προγραμματισμού (πίνακας 5.4) προσφέρουν ένα σύνολο οδηγιών που επιτρέπουν στους προγραμματιστές να χρησιμοποιούν πόρους του συστήματος για να λύσουν ένα πρόβλημα. Οι γλώσσες διαφέρουν ως προς τα χαρακτηριστικά και τον τρόπο κατανομής των υπολογιστικών πόρων και όσον αφορά τη σύγκριση των γλωσσών προγραμματισμού, τα ερωτήματα που εγείρουν είναι: μία ταχύτερη γλώσσα, είναι και μία ενεργειακά αποδοτικότερη; Ο τρόπος διαχείρισης της μνήμης μέσω των γλωσσών προγραμματισμού, επηρεάζει την κατανάλωση ενέργειας; Απάντηση σε αυτά τα ερωτήματα, δίνονται στον πίνακα 5.5 όπου φαίνονται τα συγκριτικά αποτελέσματα των γλωσσών προγραμματισμού, σε σχέση με την κατανάλωση ενέργειας, την ταχύτητα και τη χρήση της μνήμης. Η σύγκριση των γλωσσών προγραμματισμού, ωστόσο, είναι ένα εξαιρετικά περίπλοκο έργο, καθώς η απόδοση μιας γλώσσας

επηρεάζεται από την ποιότητα του μεταγλωττιστή, της virtual machine, των διαθέσιμων βιβλιοθηκών κλπ. Πράγματι, ένα πρόγραμμα λογισμικού μπορεί να γίνει γρηγορότερο βελτιώνοντας τον πηγαίο κώδικά του, αλλά και βελτιστοποιώντας τις βιβλιοθήκες του ή/και τον μεταγλωττιστή του. [2]

Πίνακας 5.4: Τα πρότυπα των γλωσσών προγραμματισμού [2]

Paradigm	Languages
Functional	Erlang, F#, Haskell, Lisp, Ocaml, Perl, Racket, Ruby, Rust;
Imperative	Ada, C, C++, F#, Fortran, Go, Ocaml, Pascal, Rust;
Object-Oriented	Ada, C++, C#, Chapel, Dart , F#, Java, JavaScript, Ocaml, Perl, PHP, Python, Racket, Rust, Smalltalk, Swift, TypeScript;
Scripting	Dart, Hack, JavaScript, JRuby, Lua, Perl, PHP, Python, Ruby, TypeScript;

Πίνακας 5.5: Κανονικοποιημένα συγκριτικά αποτελέσματα των γλωσσών προγραμματισμού σε σχέση με την κατανάλωση ενέργειας, την ταχύτητα και τη χρήση της μνήμης.

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

	Mb
(c) Pascal	1.00
(c) Go	1.05
(c) C	1.17
(c) Fortran	1.24
(c) C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	2.71
(i) Python	2.80
(c) Ocaml	2.82
(v) C#	2.85
(i) Hack	3.34
(v) Racket	3.52
(i) Ruby	3.97
(c) Chapel	4.00
(v) F#	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84

Εμπειρικές μελέτες της επίπτωσης στη κατανάλωση ενέργειας και την απόδοση του χρόνου εκτέλεσης των γλωσσών προγραμματισμού

Ακολούθως εξετάζονται κάποιες εμπειρικές μελέτες, που διερευνούν τις επιπτώσεις στη κατανάλωση ενέργειας και την απόδοση του χρόνου εκτέλεσης των γλωσσών προγραμματισμού. Στον πίνακα 5.6, παρουσιάζεται η σχέση της καταναλισκώμενης ενέργειας και της απόδοσης του χρόνου εκτέλεσης των γλωσσών προγραμματισμού.

Οι μελέτες που παρουσιάζονται βασίζονται σε διαφορετικές πειραματικές πλατφόρμες. Η μελέτη Abdulsalam (2014) πραγματοποιήθηκε σε σταθμό εργασίας(workstation), η μελέτη Rashid (2015) σε ένα ενσωματωμένο σύστημα, και η μελέτη Chen & Zong (2016) σε ένα smartphone.

Πίνακας 5.6: Η επιρροή των γλωσσών προγραμματισμού σε σχέση με την ενέργεια και την απόδοση

Comparing X to Y	Implications (avg. in %) Energy	Performance	Opt. Flag	Run-Time Env.	Selected Application	Source
C++ to C	8.4	8.67				
C++ to Java	47.4	38.12				
C++ to Python	166.28	195.17				
C to Java	38.39	29.08	–O3	–	All Apps[a]	(Abdulsalam et al. 2014)
C to Python	106.4	195.5				
Java to Python	195.87	194.77				
C++ to Java	166.14	159.13		Dalvik[b]	Quick Sort	
C++ to Java	Identical	Identical	–O3	ART[c]		(Chen and Zong 2016)
C to C++	Identical	3.38		–	Fibonacci	
ARM-assembly to C	15.38	10.52	–	–	Counting Sort	(Rashid et al. 2015)
ARM-assembly to Java	58.84	90.90			Counting Sort	

[a]Fast Fourier, Quick Sort, Linked List.
[b]https://source.android.com/devices/tech/dalvik/.
[c]Android Run-Time (ART), https://source.android.com/devices/tech/dalvik/.

Πίνακας 5.7: Τα configurations των γλωσσών προγραμματισμού

Programming Languages	Optimization Flags	Target Platforms	Test Cases	Source
C, C++, Java, and Python	–O{1,2,3}	Server system	Fast Fourier, Linked List, Quick Sort	(Abdulsalam et al. 2014)
ARM assembly, C, and Java	–	Embedded system	Bubble, Merge Quick, Counting	(Rashid et al. 2015)
C, C++, and Java	–O{1,2,3}	Android devices	Fibonacci, Tower of Hanoi, Pi calculation	(Chen and Zong 2016)

Στις μελέτες των Abdulsalam και Chen & Zong, αποδείχθηκε ότι οι C και C++, πέτυχαν σημαντική εξοικονόμηση ενέργειας και μείωση του χρόνου εκτέλεσης έναντι των άλλων γλωσσών προγραμματισμού.

Στο πλαίσιο των ενσωματωμένων συστημάτων, (μελέτη Rashid) πραγματοποιήθηκε ένα πείραμα για να συγκριθεί η επίπτωση στη κατανάλωση ενέργειας και στην απόδοση του χρόνου εκτέλεσης τεσσάρων αλγορίθμων ταξινόμησης, γραμμένων σε Arm assembly, C και Java. Εκτελώντας τα πειράματα σε ένα Raspberry Pi, αποδείχτηκε ότι οι υλοποιήσεις σε arm-assembly πέτυχαν τα πιο ενεργειακά αποδοτικά αποτελέσματα σε σχέση με τη C και τη Java. Ομοίως στη μελέτη αυτή, η Java αποδεικνύεται ως η πιο ενεργοβόρα μεταξύ των επιλεγμένων γλωσσών προγραμματισμού.

Ο πίνακας 5.6, απεικονίζει την ανωτερότητα των compiled γλωσσών προγραμματισμού έναντι των interpreted και semi-compiled, όσον αφορά την κατανάλωση ενέργειας και την απόδοση του χρόνου εκτέλεσης. Η Java και η Python υποφέρουν περισσότερο από υψηλή κατανάλωση ενέργειας και χαμηλή απόδοση χρόνου εκτέλεσης. Η χρήση interpreter, κάνει την Python πιο αργή και λιγότερο ενεργειακά αποδοτική, επειδή πρέπει να ερμηνεύσει τον πηγαίο κώδικα για κάθε εντολή.

Επιπλέον, οι Chen & Zong έδειξαν ότι η χρήση του Android Run Time (ART) περιβάλλοντος, θα μπορούσε να μειώσει την κατανάλωση ενέργειας και να αυξήσει την απόδοση του χρόνου εκτέλεσης των εφαρμογών Android. [19]

5.8.5 Δομές δεδομένων (Data structures)

Μια δομή δεδομένων είναι ένας τρόπος για να οργανωθούν, διαχειριστούν και να αποθηκευτούν δεδομένα για περαιτέρω διαδικασία ή ανάλυση. Πειραματικές μελέτες και εργαλεία υποστήριξης για τις δομές δεδομένων, βοηθούν στον προσδιορισμό εκείνων των δομών δεδομένων που είναι οι πιο ενεργειακά αποδοτικές για συγκεκριμένες περιπτώσεις. Ο πίνακας 5.8 παρουσιάζει τα αποτελέσματα εργασιών σχετικά με τις δομές δεδομένων και τη κατανάλωση ενέργειας.

Πειραματικές μελέτες (Experimental studies)

Με τη διεξαγωγή πειραματικών μελετών, οι Pereira (2016), Hasan (2016) και Pinto (2016), εντόπισαν ορισμένες θετικές και αρνητικές περιπτώσεις σχετικά με την κατανάλωση ενέργειας διαφόρων δομών δεδομένων. Οι συγγραφείς αξιολόγησαν την κατανάλωση ενέργειας των δομών δεδομένων από διαφορετικά interfaces και κυρίως από το Java Collection Framework (JCF).

Με τη χειροκίνητη αντικατάσταση δομών δεδομένων σε εφαρμογές, οι συγγραφείς έλαβαν σημαντική εξοικονόμηση ενέργειας όπως απεικονίζεται στον πίνακα 5.8. Επειδή οι συντάκτες της παραπάνω εργασίας αναφέρουν έναν εκτεταμένο κατάλογο αποτελεσμάτων, έγινε σύγκριση της κατανάλωσης ενέργειας του περισσότερο και λιγότερο ενεργειακά αποδοτικού interface και παρουσιάζεται στον πίνακα 5.8 η δομή δεδομένων με τις πιο ενεργειακά αποδοτικές μεθόδους για κάθε interface. Εκτός από την JCF, οι συγγραφείς χρησιμοποίησαν δομές δεδομένων από τις Apache Commons Collections (ACC) και Trove. Ως αποτέλεσμα, οι συγγραφείς απέδειξαν ότι η κατανάλωση ενέργειας αρχίζει να αποκλίνει μεταξύ των δομών δεδομένων μόνο όταν ο αριθμός των στοιχείων που περιέχουν είναι πάνω από 500. Επιπλέον, οι συγγραφείς σημείωσαν για τις δομές δεδομένων με απλουστευμένους τύπους δεδομένων (που βρέθηκαν στη συλλογή Trove) ότι, ενώ καταναλώνουν λιγότερη μνήμη από τα αντικείμενα, είναι πιο ενεργειακά αναποτελεσματικά στις περισσότερες περιπτώσεις.

Ομοίως, ο Pinto χρησιμοποίησε 13 thread-safe και τρεις non-thread-safe υλοποιήσεις της JCF. Οι συγγραφείς εξέτασαν τις παραπάνω δομές δεδομένων χρησιμοποιώντας διακριτές διαμορφώσεις όπως ο αριθμός των νημάτων, η αρχική ικανότητα, και ο παράγοντας φορτίο. Ως αποτέλεσμα, οι συγγραφείς παρατήρησαν ότι η σωστή επιλογή δομής δεδομένων και ο αριθμός των νημάτων (για τις περισσότερες από τις thread-safe και non-thread-safe υλοποιήσεις) μπορούν να μειώσουν την κατανάλωση ενέργειας των εφαρμογών. Στον πίνακα 5.8, μπορούμε να δούμε ότι ο Pereira πέτυχε την υψηλότερη εξοικονόμηση ενέργειας στα πειράματά του. Ωστόσο, σε σύγκριση με τους Hasan και Pinto, ο Pereira χρησιμοποίησε micro-benchmarks και όχι real-world εφαρμογές. Ο

Pinto εξέτασε τις ενεργειακές επιπτώσεις των real-world εφαρμογών με (i) την αντικατάσταση των non-thread-safe με thread-safe δομές δεδομένων και (ii) την αλλαγή του αριθμού των νημάτων. Αυτό βοήθησε να επιτευχθεί σημαντική εξοικονόμηση ενέργειας όπως απεικονίζεται στον πίνακα 5.8. Επιπλέον, τα αποτελέσματα από τον Hasan, αποκαλύπτουν τις αρνητικές επιπτώσεις της επιλογής δομής δεδομένων που μπορεί να επηρεάσει σημαντικά την κατανάλωση ενέργειας (βλέπε πίνακα 5.8).

Υποστήριξη εργαλείων (Tooling support)

Η πρόβλεψη των πιο ενεργειακά αποδοτικών δομών δεδομένων για ένα δεδομένο πρόβλημα, μπορεί να μειώσει την κατανάλωση ενέργειας. Σε αυτή τη γραμμή, ο Michanan (2016) εισήγαγε το GreenC5, ένα εργαλείο που μπορεί να προβλέψει ποιες δομές δεδομένων μεταξύ της συλλογής Copenhagen Comprehensive Collection Classes για τη C# (C5), μπορούν να μειώσουν την κατανάλωση ενέργειας της εφαρμογής με βάση το φόρτο εργασίας του συστήματος. Όπως φαίνεται στον πίνακα 5.8, ο Michanan επιτυγχάνει εξοικονόμηση ενέργειας για real-world εφαρμογές. Αυτά τα εργαλεία, μπορούν να βοηθήσουν έναν προγραμματιστή να επιλέξει και να πειραματιστεί με διάφορες δομές δεδομένων για να επιτύχει επαρκή εξοικονόμηση ενέργειας. Επιπλέον, η περαιτέρω υποστήριξη σε διάφορους τύπους δομής δεδομένων, μπορεί να κάνει τα εργαλεία αυτά πιο επωφελή και ελκυστικά για τους προγραμματιστές.

Πίνακας 5.8: Η επιρροή των δομών δεδομένων σε σχέση με την ενέργεια και την απόδοση

Collection Interface	Collection Library	Data Structure	Avg. Energy Impact (%)	Selected Apps	Source
	C5	HashedLinkedList	23.27	Apps[a]	(Michanan et al. 2016)
	JCF	AttributeList	24.88	CEB[b]	(Pereira et al. 2016)
List	JCF	ArrayList	38	Gson	
		LinkedList	−309	SETS[c]	(Hasan et al. 2016)
	JCF	LinkedHashMap	50.16	CEB[b]	(Pereira et al. 2016)
Map	JCF	ConcurrentHashMapV8	17.8	XALAN	
	JCF	ConcurrentHashMapV8	9.32	TOMCAT	(Pinto et al. 2016)
	C5	HashSet	31.44	Apps[a]	(Michanan et al. 2016)
Set	JCF	LinkedHashSet	12.5	CEB[b]	(Pereira et al. 2016)
Queue	JCF	PriorityQueue, LinkedTransferQueue, ConcurrentLinkedDeque, LinkedBlockingDeque, ConcurrentLinkedQueue	7.5	Apps[d]	(Manotas et al. 2014)
Bag	C5	HashBag	16.93	Apps[a]	(Michanan et al. 2016)

[a] A* Path Finder, Huffman Encoder, Genetic Algorithm.
[b] https://github.com/greensoftwarelab/Collections-Energy-Benchmark.
[c] Stock Exchange Trading Simulator.
[d] Barbecue, Jdepend, Apache-xml-security, Joda-Time, Commons Lang, Commons CLI.

5.8.6 Πρακτικές κωδικοποίησης (Coding practices)

Οι βέλτιστες πρακτικές κωδικοποίησης, είναι σύνολα κανόνων επίσημα ή ανεπίσημα, που έχουν θεσπιστεί από διάφορες κοινότητες που βοηθούν τους επαγγελματίες του λογισμικού να βελτιώσουν την ποιότητα του λογισμικού. Η κατανάλωση ενέργειας των πρακτικών κωδικοποίησης εξετάζεται βασιζόμενη σε εμπειρική αξιολόγηση και στον Πίνακα 5.9 παρουσιάζονται τα αποτελέσματα που προσδιορίζονται ως "καλές" και "κακές" πρακτικές κωδικοποίησης. Με τον όρο "καλό" και "κακό", αναφερόμαστε σε πρακτικές κωδικοποίησης που μπορεί να επηρεάσουν την αναγνωσιμότητα ενός προγράμματος, τη συντηρησιμότητα, την αποτελεσματικότητα και χρηστικότητα θετικά ή αρνητικά, αντίστοιχα.

Όσον αφορά τα ενσωματωμένα συστήματα, ο Groschedl (2007) αξιολόγησε την απόδοση του χρόνου εκτέλεσης, την κατανάλωση ενέργειας, τη χρήση μνήμης και το μέγεθος του κώδικα σε πέντε μπλοκ κρυπτογράφησης (δηλαδή RC6, Rijndael, Serpent, Twofish και XTEA), σε επεξεργαστή Strongarm SA-1100. Οι συγγραφείς τροποποίησαν τον υπάρχοντα πηγαίο κώδικα των αλγορίθμων κρυπτογράφησης (για να μειώσουν τις γραμμές κώδικα τους) με:

- Αντικατάσταση μακροεντολών με function calls.

- Χρησιμοποιώντας loop unrolling στις λειτουργίες κρυπτογράφησης και αποκρυπτογράφησης.

- Αντικατάσταση του t-lookup με forward και inverse S-box πίνακες και τη μείωση των μεγεθών τους.

Τα αποτελέσματά τους δείχνουν ότι τα μπλοκ κρυπτογράφησης XTEA και RC6 (τα οποία είχαν το μικρότερο μέγεθος κώδικα) ήταν τα πιο ενεργειακά αποδοτικά,

προσέφεραν την καλύτερη απόδοση στο χρόνο εκτέλεσης και χρησιμοποίησαν λιγότερη κύρια μνήμη για τις εργασίες κρυπτογράφησης και αποκρυπτογράφησης.

Στο πείραμά του, ο Tonini διερεύνησε την ενεργειακή απόδοση των βέλτιστων πρακτικών για την ανάπτυξη σε Android. Τα αποτελέσματά τους δείχνουν ότι η σωστή χρήση του for loop και των getters/setters, μπορεί να βελτιώσει την κατανάλωση ενέργειας (βλέπε πίνακα 5.9).

Ομοίως, οι Li και Halfond έλεγξαν πρακτικές όπως η ομαδοποίηση αιτήσεων http με συγκεκριμένο μέγεθος και χρήση μνήμης. Η εφαρμογή των πρακτικών κωδικοποίησης βοήθησε τους Li και Halfond να επιτύχουν αξιοσημείωτη εξοικονόμηση ενέργειας όπως απεικονίζεται στον πίνακα 5.9. Η πρακτική της αποφυγής υπολογισμού του μήκους μιας δομής δεδομένων σε έναν βρόχο αποδείχθηκε ευεργετική, αφού η ύπαρξη της κατάστασης τερματισμού του βρόχου σε μια μεταβλητή εξοικονομεί ενέργεια παρακάμπτοντας τον υπολογισμό του μήκους σε κάθε επανάληψη.

Το Android προσφέρει μια ποικιλία κλήσεων API (Application Programming Interface) και αν δεν χρησιμοποιηθούν αποτελεσματικά μπορούν να συμβάλουν στην αύξηση της κατανάλωσης ενέργειας (Linares-Vásquez 2014). Στη μελέτη τους, οι Linares-Vásquez πραγματοποίησαν μια ανάλυση σε 55 εφαρμογές Android από διάφορους τομείς και απαρίθμησαν τις πιο ενεργειακά αναποτελεσματικές μεθόδους API. Επιπλέον, οι συγγραφείς πρότειναν έναν κατάλογο πρακτικών που μπορούν να αποφέρουν εξοικονόμηση ενέργειας χρησιμοποιώντας αποτελεσματικά κλήσεις API. Μετά την ανάλυση των αποτελεσμάτων τους, εντόπισαν 133 ενεργοβόρα API από το σύνολο των 807. Από τα ενεργοβόρα API, το 61% σχετίζεται με το GUI και τον χειρισμό εικόνων, ενώ το υπόλοιπο 39% εμπίπτει στην κατηγορία της βάσης δεδομένων.

Δύο πρακτικές προτάθηκαν και αξιολογήθηκαν από τον Procaccianti (2016). Οι επιλεγμένες πρακτικές έβαλαν την εφαρμογή σε ύπνο, δηλαδή έβαλαν διαδικασίες ή νήματα σε κατάσταση ύπνου εάν περιμένουν λειτουργίες I/O ή δεν είναι πλέον ενεργές και αποφεύγουν τη χρήση δαπανηρών ενεργειακών λειτουργιών. Ως αποτέλεσμα, οι συγγραφείς πέτυχαν ενεργειακή απόδοση χρησιμοποιώντας και τις δύο πρακτικές και επίσης αυξημένη απόδοση του χρόνου εκτέλεσης (βλέπε πίνακα 5.9).

Τα αποτελέσματα στον πίνακα 5.9 δείχνουν ότι η χρήση του for με δεδομένο μέγεθος μήκους, συμβάλλει στην εξοικονόμηση ενέργειας από 36% έως 52% σύμφωνα με τους Tonini και Li & Halfond, αντίστοιχα. Επιπλέον, και στα δύο έργα οι συγγραφείς απέφυγαν τη χρήση getters/setters και εξοικονόμησαν ενέργεια ίση με 24-27% (Tonini 2013) και 30-35% (Li και Halfond 2014). Παρόλο που και οι δύο μελέτες χρησιμοποίησαν micro-benchmarks, τα αποτελέσματά τους προσφέρουν διαφορετική κλίμακα εξοικονόμησης ενέργειας. Αυτό μπορεί να συνέβη επειδή χρησιμοποίησαν διαφορετικό hardware, ξεχωριστές εκδόσεις Android και διαφορετικά εργαλεία για τις μετρήσεις ενέργειας. Από τον πίνακα 5.9, μπορούμε επίσης να παρατηρήσουμε ότι ο Grossschadl επιτυγχάνει σημαντική εξοικονόμηση ενέργειας (δηλαδή 179%) χρησιμοποιώντας τις αντίστοιχες πρακτικές κωδικοποίησης. Ωστόσο, πρέπει να σημειωθεί ότι οι συγγραφείς συνέκριναν τα μπλοκ κρυπτογράφησης μεταξύ τους, αντί

των αρχικών και βελτιστοποιημένων εκδόσεων. Συνοψίζοντας, υπάρχουν ευκαιρίες για βελτίωση της κατανάλωσης ενέργειας και της απόδοσης του χρόνου εκτέλεσης ακόμη και με την εφαρμογή μικρών αλλαγών κώδικα. Επιπλέον, η σωστή επιλογή API που απαιτεί λιγότερους πόρους συστήματος, μπορεί επίσης να μειώσει την κατανάλωση ενέργειας, σύμφωνα με τους Linares-Vásquez. Επιπλέον, ο Procaccianti βελτίωσε την κατανάλωση ενέργειας και την απόδοση του χρόνου εκτέλεσης, μειώνοντας τη χρήση δαπανηρών λειτουργιών βάσης δεδομένων (όταν αυτές δεν ήταν υποχρεωτικές) και θέτοντας τις εφαρμογές σε ύπνο (όταν δεν εκτελούσαν καμία ενέργεια). [19]

Πίνακας 5.9: Η επιρροή των πρακτικών κωδικοποίησης σε σχέση με την ενέργεια και την απόδοση

Coding Practice	Practice Implication (in %) Practices	Energy	Performance	Operating System	Source
	For loop with length	36–52	33–38	Android [a]	(Tonini et al. 2013)
		10 avg.	–	Android 4.2	(Li and Halfond 2014)
Good	Efficient query usage	25.1 avg.	24.9 avg.	Linux distro	(Procaccianti et al. 2016)
	Put application to sleep	8.48 avg.	6 avg.		
	Change macros to function calls, loop unrolling, reducing lookUp tables sizes[b]	179.3 avg.	–	–	(Grossschadl et al. 2007)
	Avoid setters & setters	24–27	24–30	Android [a]	(Tonini et al. 2013)
		30–35	–	Android 4.2	
	Invoke static methods	15 avg.	–	Android 4.2	(Li and Halfond 2014)
Bad	Use of relational database				
	Use of unnecessary views and widgets	–	–	Android 4.2	(Linares-Vásquez et al. 2014)

[a] Used Android 2.3.6 for a Samsung Galaxy Y Pro Duo, Android 4.0.3 for an Asus Tablet, and Android 4.0.4 for a Motorola Tablet.

[b] Comparing RC6 against Twofish, also runtime performance was calculated in clock cycles instead of time, therefore, we did not adding it in our results.

5.8.7 Εργαλεία μέτρησης της καταναλισκόμενης ισχύος και ενέργειας

Σε αυτή την ενότητα, παρουσιάζονται τα εργαλεία εκείνα που χρησιμοποιούνται για τη μέτρηση και τον έλεγχο της καταναλισκόμενης ισχύος και ενέργειας, μετά την ανάπτυξη της εφαρμογής. Τα εργαλεία αυτά χωρίζονται στα Benchmarks και στα Monitoring Tools.

Benchmarks

Τα benchmarks, είναι εργαλεία που αποτελούνται από δύο βασικά στοιχεία: (i) ένα εργαλείο δημιουργίας προφίλ, που χρησιμοποιείται σε μια συγκεκριμένη εκτέλεση και (ii) ένα benchmark επιδόσεων που δημιουργεί φόρτο εργασίας για ένα σύστημα. Τα παραπάνω συστατικά σε συνδυασμό, εκτελούν μετρήσεις κατανάλωσης ενέργειας. Ο πίνακας 5.10 συνοψίζει τις πληροφορίες που συλλέγονται όσον αφορά τη πλατφόρμα, τη συσχέτιση των ενεργειακών μετρήσεων με τα hardware ή software στοιχεία, των βελτιστοποιήσεων που εφαρμόζονται και των σχετικών πόρων.

Πίνακας 5.10: Οι πληροφορίες των benchmarks

Benchmark Suite	Target Platform	Energy Correlcation With	Optimization Level	Source
GBench	Linux	Hardware Components	Data memory movement, block size, number of cores	(Subramaniam and Feng 2012)
ALEA	Linux	Basic code blocks	Power capping, DVFS, compiler optimization, thread throttling	(Mukhanov et al. 2015)
AxBench	Linux	Hardware components	Source code via approximation	(Yazdanbakhsh et al. 2017)
PowerBench	Tinyos	Each node of the test-bed	–	(Haratcherev et al. 2008)

Monitoring Tools

Για να εξαχθεί η κατανάλωση ενέργειας ενός υπολογιστικού συστήματος, υπάρχουν σήμερα δύο προσεγγίσεις: (1) έμμεσες μετρήσεις ενέργειας μέσω μοντέλων εκτίμησης ή μετρητών απόδοσης ή (2) άμεσες μετρήσεις, μέσω αναλυτών ενέργειας του hardware και αισθητήρων. Παρουσιάζονται τα εργαλεία που χρησιμοποιούν έμμεσες και άμεσες προσεγγίσεις για την εκτέλεση μετρήσεων, με μερικά από αυτά να αναλύουν τις τρέχουσες εφαρμογές ή τις κλήσεις συστήματος για να εκτιμήσουν την κατανάλωση ενέργειας. Ωστόσο, σε σύγκριση με τα εργαλεία ανάλυσης πηγαίου κώδικα, τα εργαλεία παρακολούθησης της ενέργειας αναφέρουν μόνο την κατανάλωση ενέργειας μιας εφαρμογής χωρίς να επισημαίνουν τα ενεργειακά hot-spots ή να παρέχουν συμβουλές για τη βελτίωση των εντοπισμένων ελλείψεων. Οι πίνακες 5.11 και 5.12 παρουσιάζουν τα εργαλεία παρακολούθησης ενέργειας και απεικονίζουν μια ποικιλία πληροφοριών, όπως ονόματα εργαλείων, πλατφόρμας, τύπους μέτρησης, διαστήματα δειγματοληψίας και διάμεσους ρυθμούς σφάλματος. [19]

Software Energy Monitoring Tools

Χρησιμοποιούμε τον όρο " Software Energy Monitoring Tools", για αναλυτές βασισμένους σε λογισμικό, που χρησιμοποιούν μετρητές απόδοσης ή μοντέλα εκτίμησης για τη μέτρηση της κατανάλωσης ενέργειας των εφαρμογών που εκτελούνται. Τα εργαλεία παρακολούθησης διαχωρίζονται σχετικά με τα χαρακτηριστικά τους, τους

περιορισμούς, την αρχιτεκτονική (μοντέλο ενεργειακής εκτίμησης) και το υποστηριζόμενο λειτουργικό σύστημα. Επιπλέον, περαιτέρω ταξινομούνται τα εργαλεία παρακολούθησης ενέργειας λογισμικού, σύμφωνα με την πλατφόρμα που στοχεύουν σε τρεις κατηγορίες: (1) στους σταθμούς εργασίας και διακομιστές, (2) στα VMS και (3) στα smartphones.

Hardware Energy Monitoring Tools

Οι hardware αναλυτές ενέργειας ή οι αισθητήρες, μπορούν επίσης να ανακτήσουν μετρήσεις κατανάλωσης ενέργειας από ένα σύστημα υπολογιστή. Τα hardware εργαλεία παρακολούθησης ενέργειας είναι συνήθως μη-αυτόνομα εργαλεία που απαιτούν πρόσθετα εξαρτήματα hardware, όπως μια εξωτερική συσκευή για τη λήψη μετρήσεων ισχύος ή ενέργειας. Σε αυτή την ενότητα, περιγράφονται ορισμένα τέτοια εργαλεία για σταθμούς εργασίας, διακομιστές και έξυπνα τηλέφωνα.

Πίνακας 5.11: Πληροφορίες των Software-based Monitoring Tools

Tool's Name	Target Platform	Measurement Type	Sampling Rate (msec)	Median Error Rate	Source
Jalen	Linux	Energy	500	–	(Noureddine et al. 2012a)
PowerAPI	Linux	Energy	500	0.5–3	(Bourdon et al. 2012)
jRAPL	Linux	Energy	1	1.13	(Liu et al. 2015)
Jolinar	Linux	Energy	500	3	(Noureddine and Rajan 2015)
RAPL	Linux	Energy	1	3	(David et al. 2010)
SoCWatch	Windows, Linux, & Android	Power	1–1000	–	(Pantels 2015)
PowerGadget	Windows & Linux	Power	1–1000	–	(Pantels et al. 2014)
JouleMeter	VMS & Windows	Power	1000	5	(Liu et al. 2010)
VMeter	VMS	Power	–	6	(Bohra and Chaudhary 2010)
BitWatts	VMS	Energy	500	2	(Colmant et al. 2015)
PowerBooter	Android	Power	–	0.8	(Zhang et al. 2010)
GreenOracle	Android	Energy	–	10	(Chowdhury and Hindle 2016)
AEP	Android	Power	–	–	(Chen and Zong 2016)
PETTA	Android	Energy	1000	0.04	(Di Nucci et al. 2017)

Πίνακας 5.12: Οι πληροφορίες των Hardware-based Monitoring Tools

Test-bed's Name	Target Platform	Measurement Type	Sampling Rate (in sec.)	Median Error Rate	Source
Atom LEAP	Linux	Energy	1	–	(Peterson et al. 2011)
SEFLab	Windows	Power, Energy	1	1%	(Ferreira et al. 2013)
GreenMiner	Android	Power	1	insignificant	(Hindle et al. 2014)

5.8.8 Η συντήρηση του λογισμικού (maintenance)

Η συντήρηση, είναι η διαδικασία βελτίωσης ή διόρθωσης των σφαλμάτων σε ένα λογισμικό μετά την ανάπτυξή του. Στη φάση της συντήρησης, υλοποιούνται τεχνικές και εργαλεία για την ανακατασκευή (refactoring) του πηγαίου κώδικα με στόχο την εξοικονόμηση ενέργειας. Στο πλαίσιο της ανάπτυξης λογισμικού για την ενεργειακή απόδοση, το refactoring είναι η πρακτική που στοχεύει στη βελτιστοποίηση της κατανάλωσης ενέργειας των εφαρμογών μέσω τροποποιήσεων στο επίπεδο του κώδικα χωρίς να μεταβάλλεται η υποκείμενη δομή του. Χρησιμοποιείται επίσης για την ανάλυση του πηγαίου κώδικα κατά τη διάρκεια της ανάπτυξης λογισμικού, για την ανίχνευση ενεργειακών hot-spots ή σφαλμάτων (bugs).

Εμπειρική αξιολόγηση του ανακατασκευασμένου κώδικα

Εδώ, εξετάζονται εμπειρικές αξιολογήσεις των τεχνικών refactoring και πρότυπα για την βελτιστοποίηση της ενέργειας/απόδοσης, που μπορούν να χρησιμοποιήσουν οι προγραμματιστές για τη μείωση της κατανάλωσης ενέργειας των εφαρμογών. Ο πίνακας 5.13, απεικονίζει μερικά από τα συλλεχθέντα αποτελέσματα των συγγραφέων σχετικά με τις επιπτώσεις στην απόδοση/ενέργεια και στον χρόνο εκτέλεσης της μεθόδου refactoring, καθώς και περιπτώσεις δοκιμών τους. Ο Fowler (1999), εισήγαγε την έννοια του refactoring κώδικα για τη βελτίωση της σαφήνειας, της συντηρησιμότητας και της επεκτασιμότητας του υπάρχοντος πηγαίου κώδικα.

Για το σκοπό αυτό, οι Sahin (2014) και Park (2014) σε εμπειρικές τους μελέτες, εξετάζουν πώς ορισμένα μοτίβα refactoring κώδικα επηρεάζουν την κατανάλωση ενέργειας μιας εφαρμογής. Και οι δύο συγγραφείς, χρησιμοποίησαν διακριτά ενσωματωμένα συστήματα, παραμέτρους και μια σειρά από code smells για τις εμπειρικές μελέτες τους. Τα code smells, είναι δομές στον κώδικα που υποδηλώνουν παραβίαση των θεμελιωδών αρχών σχεδιασμού και επηρεάζουν αρνητικά την ποιότητα του σχεδιασμού. Συνήθως δεν είναι σφάλματα, δεν είναι τεχνικά λανθασμένα και δεν εμποδίζουν τη λειτουργία του προγράμματος. Αντ 'αυτού, δείχνουν αδυναμίες στο σχεδιασμό που μπορεί να επιβραδύνουν την ανάπτυξη ή να αυξήσουν τον κίνδυνο σφαλμάτων ή προβλημάτων στο μέλλον.

Συγκεκριμένα, ο Sahin αξιολόγησε έξι ευρέως γνωστές τεχνικές refactoring δύο Java Virtual Machines (JVM), πάνω σε εννέα εφαρμογές. Ο Park αξιολόγησε 63 από τις 68

τεχνικές refactoring πάνω από στα δείγματα κώδικα που προτάθηκαν από τον Fowler. Ο Sahin κατέληξε στο συμπέρασμα ότι κάθε refactoring μέθοδος, μπορεί να αυξήσει ή να μειώσει τη χρήση ενέργειας της εφαρμογής εκτός από τις Extend Local Variables, που πάντα μειώνουν τη κατανάλωση ενέργειας. Ομοίως, ο Park διαπίστωσε ότι μερικά από τα code smells του refactoring κώδικα, μπορούν να μεταβάλλουν θετικά ή αρνητικά την κατανάλωση ενέργειας. Ιδιαίτερα, από τα λαμβανόμενα αποτελέσματα, ο Park παρουσίασε ότι 33 από τις τεχνικές refactoring, οδηγούν σε εξοικονόμηση ενέργειας ενώ οι υπόλοιπες 30 δεν το κάνουν. Εκτός αυτού, οι συγγραφείς συμπέραναν ότι η κατανάλωση ενέργειας μεταξύ των εκδόσεων Java στο πλαίσιο των τεχνικών refactoring, δεν είναι σταθερή.

Ο όρος Energy Code Smells που δηλώνεται από τον Morisio (2013), σχετίζεται με την αναποτελεσματική εφαρμογή προγραμματιστικών επιλογών που αυξάνουν την κατανάλωση ενέργειας. Μέσω της μελέτης του, ο Morisio στοχεύει στον προσδιορισμό ενός αριθμού code smells που μπορούν να μεταβάλουν την ενέργεια, την απόδοση του χρόνου εκτέλεσης, ή ακόμα και τα δύο. Ως εκ τούτου, πραγματοποίησαν το πείραμά τους σε υπάρχοντα code smells που βρέθηκαν από τα εργαλεία CppCheck35 και FindBugs36.

Τα αποτελέσματα που απεικονίζονται στον πίνακα 5.13, δείχνουν ότι οι Morisio, Sahin, και Park, προκάλεσαν και κέρδη και απώλειες στην κατανάλωση ενέργειας όπως 1%, -7% έως 5% και -8% έως 7, αντίστοιχα. Παρατηρείται, ότι τα αποτελέσματα των Sahin και Park, χρησιμοποιούν αρκετά παρόμοια μοτίβα refactoring, δεν διαφέρουν πολύ παρόλο που χρησιμοποίησαν διαφορετικές γλώσσες για να τις αναπτύξουν (Java και C++ αντίστοιχα). Ο Morisio επίσης, χρησιμοποίησε refactoring μοτίβα που εισήγαγε ο Fowler (1999), ωστόσο επέδειξαν μικρή μόνο εξοικονόμηση ενέργειας (λιγότερο από 1%). Επιπλέον ο Morisio, έδειξε ότι τα energy code smells, δεν επηρεάζουν τις performance smells και το αντίστροφο. Τα παραπάνω αποτελέσματα υποδηλώνουν ότι η εξοικονόμηση ενέργειας που επιτυγχάνεται με τις τεχνικές refactoring, είναι πολύ χαμηλή. Το γεγονός αυτό υπογραμμίζει ότι οι εφαρμοζόμενες τεχνικές refactoring, επικεντρώνονται κυρίως στη βελτίωση της συντηρησιμότητας του κώδικα, της επεκτασιμότητας και της σαφήνειας. Έτσι, δεν μπορούν να προσφέρουν σημαντικά κέρδη όσον αφορά την ενεργειακή απόδοση. [19]

Refactoring Tools

Οι εργασίες σε αυτόν τον τομέα, έχουν ως στόχο την παροχή συντηρησιμότητας με τη μείωση της κατανάλωσης ενέργειας, μέσω μοτίβων του κώδικα refactoring που ενσωματώνονται σε εργαλεία. Ένα πλαίσιο βελτιστοποίησης ενέργειας για εφαρμογές Android, εισήγαγαν οι Li και Gallagher (2016) το οποίο επικεντρώνεται στη παροχή ενεργειακών βελτιστοποιήσεων στον αναπτυγμένο πηγαίο κώδικα μέσω ενός συνόλου refactoring στρατηγικών. Το προτεινόμενο πλαίσιο παίρνει τον πηγαίο κώδικα ως είσοδο και τον αναλύει, ανακτώντας τα δεδομένα που σχετίζονται με την ενέργεια για να τα συσχετίσει με βασικά μπλοκ κώδικα. Στη συνέχεια, εντοπίζονται τα ενεργειακά

hot-spots του πηγαίου κώδικα και εφαρμόζονται στρατηγικές refactoring (π.χ. loop unrolling, loop unswitching, method in-line). Η χρήση ενός τέτοιου εργαλείου refactoring, μπορεί να βοηθήσει έναν χρήστη να τροποποιήσει τον πηγαίο κώδικα υφιστάμενων ή παλαιών συστημάτων, έτσι ώστε να μειωθεί η κατανάλωση ενέργειάς τους και να αυξηθεί η απόδοση του χρόνου εκτέλεσης. Οι συγγραφείς αξιολόγησαν τα εργαλεία τους σε real-world εφαρμογές και πέτυχαν εξοικονόμηση ενέργειας που κυμαινόταν από 6% έως 50%.

Πίνακας 13: Η επιρροή του refactoring σε σχέση με την ενέργεια και την απόδοση

Refactoring Techniques	Energy (%)	Implications Performance	Test Cases	Source
Dead Local Variable, Non Short Circuit,[a] Parameter By Value, Repeated Conditionals, Self Assignment Variable	< 1 avg.	Energy Code Smells are not affecting execution time	Authors' Apps[e]	(Morisio et al. 2013)
Convert Local Variable to Field,[b] Extract Local Variable,[c] Extract Method, Introduce Indirection,[d] In line Method,[g] Introduce Parameter Object[h]	(−7.5)–4.54	There is no rough correlation between runt time performance and energy consumption for JVM 6 and 7	Selected Apps[f]	(Sahin et al. 2014)
Replace Method with Method Object and Encapsulate Collection	−7.91–6.99	–	M.Fowler's Code Samples[i]	(Park et al. 2014)
Loop Unrolling, Loop Unswitching, Method Inline	6.4–50.21	Observed boost on runtime performance	Cocos2d game engine	(Li and Gallagher 2016)

[a]Using "&" and "||" cause all statements evaluation, in contrast to "&&" and "|| ||".
[b]Refactoring local variables to public class fields.
[c]Occurrences of the same expression can be replaced from a variable.
[d]Redirecting all the method invocations to a newly created static method.
[e]Use of mirco-benchmarks.
[f]Commons-{Beanutils, CLI, Collections, IO, Lang, Math}, Joda-Convert, Joda-Time, Sudoku.
[g]In place of the method's invocation its source code is added instead.
[h]Created a new class at the top level (super class).
[i]Code Samples by Fowler et al. (1999).

5.9 Το λειτουργικό σύστημα (OS), οι εφαρμογές (applications) και η κατανάλωση ενέργειας

Τα λειτουργικά συστήματα και οι εφαρμογές, πρέπει να λαμβάνουν υπόψη την κατανάλωση ενέργειας και να χρησιμοποιούν τα χαρακτηριστικά εξοικονόμησης ενέργειας που παρέχονται από το υλικό όσο το δυνατόν πιο αποτελεσματικά. Σε αντίθεση με τη διαχείριση μνήμης η οποία υπήρξε ένα βασικό συστατικό του σχεδιασμού των λειτουργικών συστημάτων για δεκαετίες, η κατανάλωση ενέργειας σε μεγάλο βαθμό δεν συμπεριλαμβανόταν στο σχεδιασμό λογισμικού πριν από το 1998. Σε λειτουργικά συστήματα γενικής χρήσης, η ενεργειακή αποδοτικότητα μπορεί να αυξηθεί με τη ρύθμιση της κατανάλωσης ισχύος από μία διαδικασία κατά το χρόνο εκτέλεσης. Η πρόβλεψη ή η μέτρηση της κατανάλωσης ενέργειας ενός εκτελούμενου προγράμματος, είναι μια απαίτηση για τη λήψη σωστών αποφάσεων διαχείρισης ενέργειας. Για προφανείς λόγους, η πρόβλεψη θα πρέπει να προσθέσει τη λιγότερη δυνατή υπέρβαση στην κατανάλωση ενέργειας.

Επιπλέον, απαιτείται να υπάρξει εικόνα για το πού και πώς η ισχύς χρησιμοποιείται σε έναν υπολογιστή και σε επίπεδο λογισμικού αυτό σημαίνει να βρεθούν ποια μοτίβα λειτουργικής συμπεριφοράς είναι τα πιο αναποτελεσματικά ενεργειακά.

Ακολούθως, παρουσιάζονται κάποιες προσεγγίσεις που μπορούν να εφαρμοστούν στο επίπεδο του λειτουργικού συστήματος και των εφαρμογών για τη μείωση της κατανάλωσης ενέργειας.

5.9.1 Χρονοπρογραμματισμός (Scheduling)

Σε ένα σύστημα, ο χρονοπρογραμματισμός είναι απαραίτητος όταν πολλαπλές λειτουργικές μονάδες πρέπει να έχουν πρόσβαση στο ίδιο αντικείμενο και χρησιμοποιείται από το λειτουργικό σύστημα για να παρέχει σε κάθε μονάδα ένα μερίδιο χρήσης του αντικειμένου. Ο χρονοπρογραμματισμός εφαρμόζεται σε διάφορα μέρη ενός συστήματος όπως για παράδειγμα για το χρόνο χρήσης του επεξεργαστή, την επικοινωνία, την πρόσβαση στο σκληρό δίσκο και γίνεται με κριτήρια όπως η προτεραιότητα, η καθυστερήση (latency) ή οι χρονικές απαιτήσεις. [21]

Οι πιό κάτω μηχανισμοί scheduling, παρουσιάζουν πλεονεκτήματα όσον αφορά την χαμηλή κατανάλωση ενέργειας.

Χρονοπρογραμματισμός χρόνου επεξεργαστή (processor time scheduling)

Τα περισσότερα συστήματα ξοδεύουν μόνο ένα κλάσμα του χρόνου για την εκτέλεση χρήσιμων υπολογισμών. Ο υπόλοιπος χρόνος καταναλώνεται σε αδρανή κατάσταση. Ο διαχειριστής ενέργειας του λειτουργικού συστήματος, θα πρέπει να παρακολουθεί τις περιόδους επεξεργασίας, έτσι ώστε όταν εισάγεται μια περίοδος αδράνειας, να μπορεί αμέσως να απενεργοποιηθεί μέρη του συστήματος που δεν

χρειάζονται πλέον. Ο βασικός στόχος του scheduler είναι να συλλέγει αιτήματα για επεξεργασία και να συμπυκνώνει τα χρονικά τμήματα της ενεργής επεξεργασίας σε «εκρήξεις επεξεργασίας» (computation bursts). Ακόμα περισσότερο, ταξινομώντας τις εργασιές και τις διαδικασίες (jobs and processes) σε κατηγορίες παρασκηνίου (background), περιοδικές (periodic) και προσκηνίου (foreground), ο επεξεργαστής μπορεί να λειτουργεί σε χαμηλότερη ταχύτητα όταν εκτελεί χαμηλής προτεραιότητας background εργασίες.

Σύστημα αρχείων

Το σύστημα αρχείων είναι ένα άλλο ζήτημα που αφορά την αλληλεπίδραση μεταξύ των μονάδων και το λογισμικό του συστήματος. Είναι θετικό να απενεργοποιείται ένας σκληρός δίσκος όταν είναι αδρανής για ένα λεπτό, αλλά είναι πολύ καλύτερο να σχεδιαστεί ένα σύστημα αρχείων με τέτοιο τρόπο, ώστε να εκμεταλλεύεται τη δυνατότητα απενεργοποίησης του δίσκου. Για παράδειγμα, ο χρονοπρογραμματισμός στο σύστημα αρχείων του λειτουργικού συστήματος, συλλέγει λειτουργίες του δίσκου σε μια προσωρινή μνήμη και μεταθέτει χρονικά την εκτέλεση αυτών που έχουν χαμηλή προτεραιότητα, μόνο έως ότου ο σκληρός δίσκος ήδη λειτουργεί ή έχει αρκετά δεδομένα.

Επικοινωνία

Η επικοινωνία ενός συστήματος μπορεί να ρυθμιστεί με τέτοιο τρόπο ώστε να ελαχιστοποιείται η μετάβαση καταστάσεων και να λειτουργεί με μαζική μεταφορά δεδομένων. Σε μια κατάσταση με ποικίλες και πολλαπλές διασυνδέσεις δικτύου, είναι έξυπνο να γίνεται προφόρτωση (prefetch) κάποιων πληροφοριών ή να μετατίθεται χρονικά η επικοινωνία, μέχρι να είναι διαθέσιμο ένα πιο οικονομικό όσον αφορά την ισχύ, δίκτυο. Για παράδειγμα, μιά εφαρμογή μπορεί να προγραμματίσει την ενεργοποίηση του επεξεργαστή όταν το σύστημα είναι συνδεμένο σε ενσύρματο δίκτυο, έτσι ώστε να γίνεται λήψη πληροφοριών όταν καταναλώνει λιγότερη ενέργεια ή δεν κάνει χρήση της τροφοδοσίας από τις μπαταρίες.

5.9.2 Ο διαχειριστής ενέργειας (Power Manager – PM)

Η απενεργοποίηση των αχρησιμοποίητων μονάδων είναι μια προσέγγιση που χρησιμοποιείται για τη μείωση κατανάλωσης της ενέργειας. Για να γίνει εκμετάλλευση των καταστάσεων χαμηλής κατανάλωσης ισχύος των συσκευών, είτε το λειτουργικό σύστημα πρέπει να κατευθύνει τη συσκευή (ή τμήμα της) στην απενεργοποίησή της όταν προβλέπεται ότι η εξοικονόμηση ενέργειας θα πλεονεκτεί σε σχέση με τον χρόνο και την ενέργεια που θα απαιτηθεί για την απενεργοποίηση και της επανεκκίνησή της, ή οι μονάδες θα χρησιμοποιήσουν έναν αλγόριθμο με γνώμονα τη ζήτηση δεδομένων, για την αυτόματη μεταγωγή τους σε κατάσταση ύπνου ή αδράνειας.

Η διάκριση του συστήματος σε μονάδες πρέπει να είναι τέτοια ώστε να παρέχουν ένα σύμπλεγμα λειτουργικότητας. Για παράδειγμα, στο σύστημα μνήμης το locality of reference (φαινόμενο κατά το οποίο ένα πρόγραμμα κατά την εκτέλεση του, τείνει να έχει πρόσβαση στο ίδιο σύνολο θέσεων μνήμης για μια συγκεκριμένη χρονική περίοδο), μπορεί να αξιοποιηθεί για τη μείωση της κατανάλωσης ισχύος σε μεγάλα μπλοκ μνήμης. Αυτό δείχνει για άλλη μια φορά, ότι μια στενή συνεργασία μεταξύ του λειτουργικού συστήματος που εκτελεί τη κατανομή μνήμης, του διαχειριστή ενέργειας που ελέγχει τις καταστάσεις ισχύος των μονάδων, μαζί με μια κατάλληλη αρχιτεκτονική υλικού μπορεί να είναι επωφελής για τη μείωση της κατανάλωσης της ενέργειας του συστήματος. [21]

Μία από τις βασικές πτυχές όσον αφορά την πολιτική διαχείρισης της ενέργειας, είναι ο συντονισμός και η λήψη αποφάσεων να περάσουν στο λειτουργικό σύστημα. Το λειτουργικό σύστημα θα ελέγχει τις καταστάσεις ισχύος των συσκευών του συστήματος και θα μοιράζεται αυτές οι πληροφορίες με τις εφαρμογές και τους χρήστες.

5.9.3 Εφαρμογές (Applications)

Οι εφαρμογές διαδραματίζουν τον πιο κρίσιμο ρόλο στην διαχείριση της ισχύος σε ένα σύστημα. Επειδή οι εφαρμογές έχουν άμεση γνώση του τρόπου με τον οποίο ο χρήστης χρησιμοποιεί το σύστημα για να εκτελέσει κάποια λειτουργία, αυτή η γνώση πρέπει να ενσωματωθεί στη λήψη αποφάσεων της διαχείρισης ενέργειας. Προφανώς, η απρόσεκτη χρήση του επεξεργαστή και του σκληρού δίσκου επηρεάζει δραστικά την κατανάλωση ισχύος. Για παράδειγμα, η εκτέλεση μη βασικών εργασιών στο παρασκήνιο και σε κατάσταση αδράνειας, εμποδίζει τον επεξεργαστή από το να τεθεί σε κατάσταση χαμηλής κατανάλωσης ισχύος. Επομένως, δεν αρκεί να είναι μόνο το σύστημα χαμηλής ισχύος, αλλά και οι εφαρμογές που εκτελούνται στο σύστημα πρέπει να γραφτούν έχοντας κατά νου τη μείωση της κατανάλωσης ισχύος.

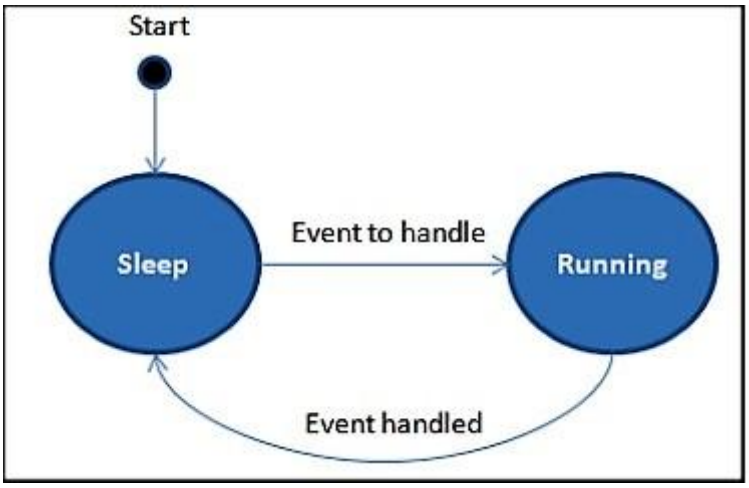
5.10 Real Time Operating System (RTOS) - Το λειτουργικό σύστημα πραγματικού χρόνου

5.10.1 Τα χαρακτηριστικά του RTOS

Η χαμηλή κατανάλωση ενέργειας, αποτελεί βασικό παράγοντα σε πολλές εφαρμογές ενσωματωμένων συστημάτων. Πολλές από αυτές τις εφαρμογές, ξοδεύουν το μεγαλύτερο μέρος του χρόνου τους περιμένοντας να συμβεί ένα γεγονός, όπως λήψη δεδομένων σε μια σειριακή θύρα, μια αλλαγή κατάστασης στα I/O's ή αναμονή για τη λήξη μιάς χρονικής καθυστέρησης.

Εάν ο επεξεργαστής εξακολουθεί να βρίσκεται στην ενεργό κατάσταση ακόμα και όταν είναι αδρανής, η διάρκεια ζωής της μπαταρίας μειώνεται. Η μετάπτωση της MCU σε λειτουργία χαμηλής κατανάλωσης κατά τη διάρκεια περιόδων αδράνειας, μπορεί να παρατείνει σημαντικά τη διάρκεια ζωής της μπαταρίας και αυτή η πολύ σημαντική απαίτηση, μπορεί να καλυφθεί με την χρήση ενός RTOS.

Με τη χρήση ενός τέτοιου λειτουργικού συστήματος, μπορούν να ανατεθούν διαφορετικές προτεραιότητες στις διάφορες διεργασίες, ανάλογα με τις απαιτήσεις της κάθε εφαρμογής για το χρόνο εκτέλεσής της. Επιπλέον χρησιμοποιώντας μηχανισμούς συγχρονισμού, μπορεί να σηματοδοτηθεί η εκτέλεση κάποιας ενέργειας από την ανίχνευση κάποιου γεγονότος, υπάρχει δηλαδή η δυνατότητα να γίνει το σύστημα οδηγούμενο από γεγονότα (event-driven), σχήμα 5.8. Αυτό, οδηγεί σε σημαντικό περιορισμό της δαπανούμενης ενέργειας, καθώς στο χρόνο που δεν συμβαίνουν γεγονότα το σύστημα μπορεί να βρίσκεται σε κατάσταση αδράνειας (idle). Είναι φανερό δηλαδή, ότι η ενσωμάτωση της διαχείρισης ισχύος και η χρήση ενός RTOS είναι εξίσου σημαντικός τρόπος εξοικονόμησης ενέργειας όσο και η hardware σχεδίαση. Παραδείγματα δοκιμών έδειξαν ότι ορισμένοι μικροεπεξεργαστές, ενδέχεται να βρίσκονται σε κατάσταση αδράνειας ακόμα και για το 60% – 80% του χρόνου λειτουργίας τους. [5], [7], [72]



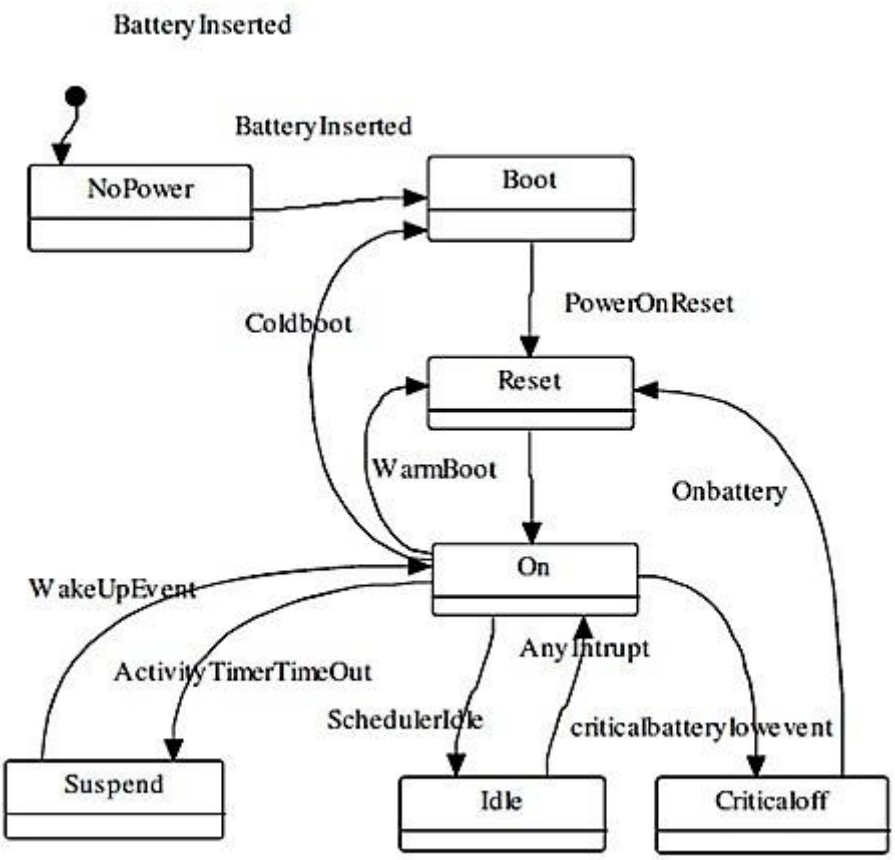
Σχήμα 5.8: Παράδειγμα event-driven λειτουργίας.

Όσον αφορά τις εφαρμογές, για τη μεγιστοποίηση του χρόνου κατά τον οποίο ο επεξεργαστής βρίσκεται σε αδρανή ή ύπνου κατάσταση, μπορεί να χρησιμοποιηθεί ένας μεταγλωττιστής που είναι αποτελεσματικός στη βελτιστοποίηση της ταχύτητας, έτσι ώστε όσο πιο γρήγορα εκτελεστεί μια συγκεκριμένη εργασία, τόσο λιγότερο χρόνο χρειάζεται να είναι σε ενεργό κατάσταση ο επεξεργαστής. [4] Μπορεί επίσης, να γίνει η δόμηση του κώδικα κατά τέτοιο τρόπο που να προσδιορίζει πότε το σύστημα δεν εκτελεί εργασίες. Μια προτεινόμενη προσέγγιση είναι να υπάρχει μία task-oriented σχεδίαση, στην οποία μια εργασία μπορεί να οριστεί με τη χαμηλότερη προτεραιότητα και να εκτελείται μόνο όταν δεν υπάρχει άλλη εργασία που απαιτείται να εκτελεστεί. Αυτή η εργασία, ονομάζεται αδρανής εργασία και είναι η ιδανική περίπτωση για να εφαρμοστεί η διαχείριση ενέργειας.

Ένα RTOS, παρέχει στον προγραμματιστή το πλαίσιο στο οποίο θα χτίσει και θα οργανώσει τα χαρακτηριστικά του συστήματος. Ακόμη και για συστήματα που δεν έχουν ανάγκη από δυνατότητες πραγματικού χρόνου, ο προκύπτων κώδικας μπορεί να είναι καθαρότερος και καλύτερα οργανωμένος. Η ενσωμάτωση ενός RTOS, μπορεί να λύσει μια ποικιλία προβλημάτων που μπορεί να προκύψουν στον κώδικα μιάς εφαρμογής, καθώς παρέχει τη δυνατότητα multitasking και επιτρέπει την κατάτμηση της εφαρμογής σε μικρότερα κομμάτια. Κάθε εργασία έχει τη δική της προτεραιότητα με βάση τη σημασία της και ο προληπτικός χρονοπρογραμματισμός (pre-emptive scheduling), εξασφαλίζει ότι θα εκτελεστεί η εργασία που έχει την υψηλότερη προτεραιότητα μεταξύ εκείνων που είναι έτοιμες να τρέξουν. [16]

5.10.2 Οι καταστάσεις ισχύος (Power states)

Σε οποιοδήποτε σύστημα, υπό ορισμένες συνθήκες κάποια hardware υλικά είναι πάντα αδρανή. Αυτό ισχύει για τις συσκευές, τον επεξεργαστή και για τις εφαρμογές και υπό αυτές τις συνθήκες, μπορούν να τεθούν σε μία χαμηλότερη κατάσταση ισχύος. Στο RTOS, υπάρχουν διαφορετικές καταστάσεις ισχύος που σχετίζονται με τις εφαρμογές, τις συσκευές και τον επεξεργαστή. Είναι πολύ πιθανό ότι όταν ένας επεξεργαστής βρίσκεται σε κατάσταση ύπνου, η συσκευή να βρίσκεται σε αδρανή κατάσταση και η εφαρμογή σε ενεργή κατάσταση. Το σχήμα 5.9, δείχνει τη μετάβαση της κατάστασης ισχύος για ένα πλήρες σύστημα, μεταξύ των διάφορων διαθέσιμων καταστάσεων.



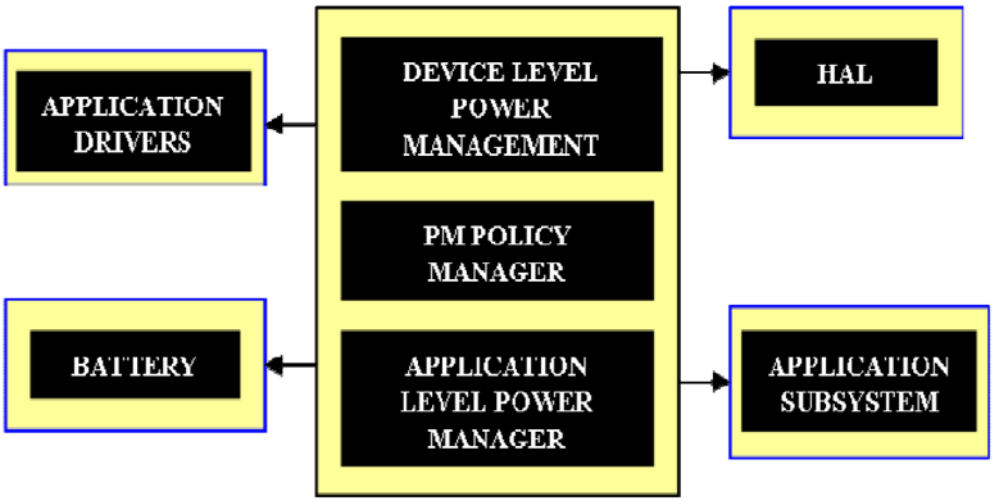
Σχήμα 5.9: Καταστάσεις ισχύος του συστήματος

Ένα σύστημα μπορεί να έχει οποιαδήποτε από τις επτά καταστάσεις (NoPower, Boot, On, Idle, Suspend, CriticalOff και Reset) που φαίνονται στο σχήμα 5.9, ανάλογα με την εργασία η οποία εκτελείται στη συσκευή. Ωστόσο, η μετάβαση μεταξύ των καταστάσεων, μπορεί να καταναλώσει κάποιο χρόνο ανάλογα με το νήμα που εκτελείται από τον power manager. [5], [7]

5.10.3 RTOS low power χρονοπρογραμματισμός

Οι τεχνικές διαχείρισης της ενέργειας σε επίπεδο λειτουργικού συστήματος που έχουν στο επίκεντρο τις I/O, εκμεταλλεύονται τις πληροφορίες σχετικά με τις εκτελούμενες εργασίες. Συγκεκριμένα, ανάλογα με τη χρήση μιάς συσκευής, ο διαχειριστής ενέργειας μπορεί να αποφασίσει ποιες συσκευές πρέπει να τοποθετηθούν στις χαμηλής ισχύος καταστάσεις που διατίθενται κατά το χρόνο εκτέλεσης. Προφανώς, η μεταβολή των καταστάσεων ισχύος της συσκευής συνεπάγεται να ληφθεί υπόψη η πιθανή επιβάρυνση στη κατανάλωση ενέργειας που σχετίζεται κυρίως με το νεκρό χρόνο (break-even time) (δηλαδή, τον ελάχιστο αδρανή χρόνο μέχρι να επιτευχθεί η εξοικονόμηση ενέργειας). Είναι κρίσιμο να βρεθεί μία ισορροπία μεταξύ της ενέργειας που εξοικονομείται διατηρώντας τις συσκευές σε καταστάσεις χαμηλής ισχύος και της ενέργεια που

καταναλώνεται κατά τη διάρκεια των μεταβάσεων κατάστασης. Για το λόγο αυτό, μπορεί να παρασχεθεί πολύτιμη βοήθεια από την αλληλεπίδραση με τον χρονοπρογραμματιστή διαδικασιών, ο οποίος μπορεί να χειριστεί την εκτέλεση εργασιών με τέτοιο τρόπο ώστε να μεγιστοποιήσει τις αδρανείς (idle) περιόδους για τις συσκευές I/O. Το σχήμα 5.10 δείχνει την αλληλεπίδραση των συσκευών, εφαρμογών και της μπαταρίας με τον PM. Ο PM ενεργεί ως μεσολαβητής μεταξύ των συσκευών, των εφαρμογών, και του επεξεργαστή. Πληροφορίες από τις διάφορες διεπαφές σχετικά με την κατάσταση ισχύος τους, συλλέγονται και διαχειρίζονται από τον PM. Με βάση αυτές τις πληροφορίες, ολόκληρο το σύστημα τίθεται στη χαμηλότερη δυνατό low power κατάσταση για τη συγκεκριμένη εφαρμογή.

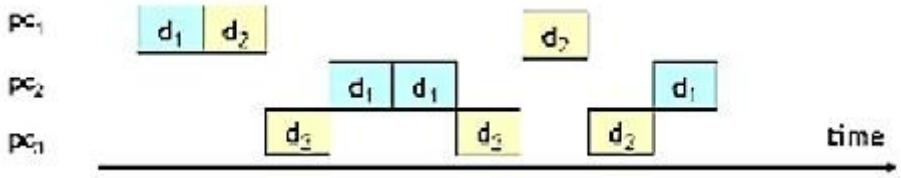


Σχήμα 5.10: Αλληλεπίδραση του διαχειριστή ισχύος με τις συσκευές και τις εφαρμογές

Device - oriented χρονοπρογραμματισμός

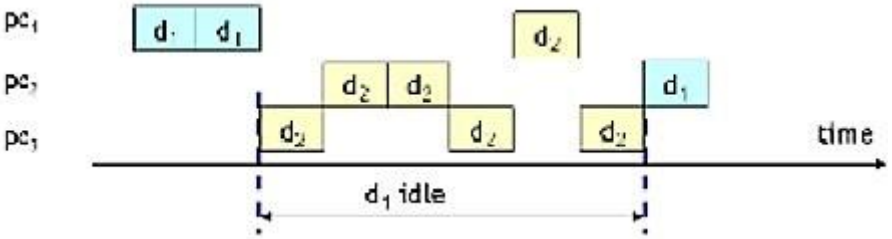
Ο device-oriented χρονοπρογραμματισμός, στοχεύει στην επιλογή των διαδικασιών ανάλογα με τη χρήση των συσκευών τους. Αυτή η τεχνική, προσπαθεί να οργανώσει την εκτέλεση των διαδικασιών για να επιτύχει μια ομαδοποίηση στη χρήση της συσκευής. Ας υποθέσουμε ότι τρεις διαδικασίες βρίσκονται σε κατάσταση ετοιμότητας σε ένα συγκεκριμένο σύστημα. Αυτές οι τρεις διαδικασίες, pc1, pc2 και pc3 χρησιμοποιούν συσκευές από ένα σύνολο αποτελούμενο από τις d1 και d2. Ένας κλασσικός αλγόριθμος χρονοπρογραμματισμού βάσει προτεραιότητας, θα μπορούσε να οργανώσει τη διαδικασία εκτέλεσης όπως απεικονίζεται στο σχήμα 5.11.

Σε μια τέτοια περίπτωση, τα χρονικά διαστήματα στα οποία μια συσκευή είναι αδρανής φαίνεται να είναι σύντομα και με πολύ κατακερματισμένη κατανομή με την πάροδο του χρόνου. Με τέτοιο τρόπο, θα υπάρξουν λίγες ευκαιρίες για εξοικονόμηση ενέργειας θέτοντας τις συσκευές σε καταστάσεις χαμηλής ισχύος. [9]



Σχήμα 5.11: Χρονοπρογραμματισμού βάσει προτεραιότητας

Αντίθετα, αν ο χρονοπρογραμματιστής επιλέξει την επόμενη διαδικασία για να εκτελεστεί λαμβάνοντας υπόψη τις συσκευές που απαιτούνται, το διάγραμμα χρήσης των συσκευών θα μπορούσε να παρουσιάσει μεγαλύτερα αδρανή χρονικά διαστήματα, με αποτέλεσμα περισσότερες ευκαιρίες για εξοικονόμηση ενέργειας, όπως φαίνεται στο σχήμα 5.12.



Σχήμα 5.12: Device-oriented χρονοπρογραμματισμός

Device - exclusion χρονοπρογραμματισμός

Με τον device-oriented χρονοπρογραμματισμό, οι επόμενες διαδικασίες επιλέγονται με βάση τις απαιτήσεις της συσκευής τους. Αυτή η προσέγγιση έχει ως στόχο την προσαρμογή της χρήσης της συσκευής δυναμικά, προκειμένου να δημιουργηθούν ευκαιρίες εξοικονόμησης ενέργειας. Μια αντίθετη προσέγγιση είναι να προσδιοριστεί ένα σταθερό μοτίβο χρήσης για ένα συγκεκριμένο σύνολο συσκευών. Το μοτίβο πρέπει να είναι γνωστό από τον χρονοπρογραμματιστή που θα επιλέξει τις επόμενες διαδικασίες, ανάλογα με το ποια συσκευή μπορεί να χρησιμοποιηθεί και ποια όχι, όπως καθορίζεται από το μοτίβο χρήσης της συσκευής. Ως εκ τούτου, η προσέγγιση του αποκλεισμού συσκευών(device exclusion) γνωρίζει εκ των προτέρων (δηλ. ο διαχειριστής ενέργειας θα μπορούσε να κοινοποιήσει μια τέτοια απόφαση στον χρονοπρογραμματιστή), ποιες συσκευές θα μεταβούν στην κατάσταση χαμηλής ισχύος και για πόσο χρόνο. Με αυτόν τον τρόπο, ο χρονοπρογραμματιστής θα επιλέξει μία διαδικασία για να τρέξει, βάσει του ποιές συσκευές επιτρέπεται να είναι σε busy state. Μια τέτοια ακραία προσέγγιση προκύπτει από την εκτίμηση ότι χρησιμοποιώντας έναν

device-oriented χρονοπρογραμματισμό, οι χρόνοι αδράνειας της συσκευής δεν είναι γνωστοί και δεν μπορούν να ελεγχθούν. Αντ'αυτού, με έναν device-exclusion χρονοπρογραμματισμό, τα αδρανή διαστήματα για μια συσκευή θα μπορούσαν να ρυθμιστούν ώστε να είναι μεγαλύτερα από τον νεκρό χρόνο (break-even time) (δηλ. τον ελάχιστο αδρανή χρόνο μέχρι να επιτευχθεί η εξοικονόμηση ενέργειας) αποφεύγοντας έτσι τη σπατάλη ενέργειας λόγω της επιπλέον κατανάλωσης κατά τη διάρκεια των μεταβάσεων κατάστασης. [9]

ΒΙΒΛΙΟΓΡΑΦΙΑ 5ου ΚΕΦΑΛΑΙΟΥ

[1] A SURVEY OF SOFTWARE OPTIMIZATION TECHNIQUES FOR LOW-POWER CONSUMPTION, Oscar Acevedo, Advisor: Dr. Manuel Jimenez, Electrical and Computer Engineering Department, University of Puerto Rico, Mayagüez Campus, Mayagüez, Puerto Rico, January 2002.

[2] ENERGY EFFICIENCY ACROSS PROGRAMMING LANGUAGES HOW DOES ENERGY, TIME, AND MEMORY RELATE, Rui Pereira, Marco Couto and Francisco Ribeiro, Rui Rua, HASLab/INESC TEC, Universidade do Minho, Portugal, Jácome Cunha, NOVA LINCS, DI, FCT, Univ. Nova de Lisboa, Portugal, João Paulo Fernandes, Release/LISP, CISUC, Universidade de Coimbra, Portugal, João Saraiva, HASLab/INESC TEC, Universidade do Minho, Portugal. SLE'17, October 23–24, 2017, Vancouver, BC, Canada © 2017 Association for Computing Machinery. ACM ISBN 978-1-4503-5525-4/17, https://doi.org/10.1145/3136014.3136031

[3] LOW POWER/ENERGY COMPILER OPTIMIZATIONS, Ulrich Kremer, Department of Computer Science, Rutgers University, Mar 21, 2006.

[4] POWER ANALYSIS OF EMBEDDED OPERATING SYSTEMS, Robert P. Dick and Niraj K. Jha, Department of Electrical Engineering, Princeton University, Princeton N,J Ganesh Lakshminarayana and Anand Raghunathan, CCRL-NEC USA, 4 Independence Way, Princeton NJ, 2000.

[5] POWER MANAGEMENT SYSTEM FOR EMBEDDED RTOS: AN OBJECT ORIENTED APPROACH, Ankur Agarwal, Saeed Rajput and A. S. Pandya, CSE Dept. FAU,Boca Raton, FL, IEEE CCECE/CCGEI, Ottawa, May 2006.

[6] SOFTWARE DESIGN FOR LOW POWER, Kaushik Roy And Mark C. Johnson, School of Electrical and Computer Engineering Purdue University, West Lafayette, Indiana, U.S.A.

[7] SYSTEM LEVEL POWER MANAGEMENT FOR EMBEDDED RTOS: AN OBJECT ORIENTED APPROACH, Ankur Agarwal and Eduardo Fernandez , Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL, USA, International Journal of Engineering (IJE), Volume (3): Issue (5).

[8] THE IMPACT OF SOURCE CODE IN SOFTWARE ON POWER CONSUMPTION, Hayri Acar, Gülfem I. Alptekin, Jean, Patrick Gelas and Parisa Ghodous. International Journal of Electronic Business Management, January 2016.

[9] DPM AT OS LEVEL: LOW-POWER SCHEDULING POLICIES, C. Brandolese, W. Fornaciari, F. Salice, L. Pomante, R. Zafalon, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, ITALY, proceedings of the 5th WSEAS Int. Conf. on Circuits, Systems, Electronics, Control & Signal Processing, Dallas, USA, November 1-3, 2006.

[10] COMPARATIVE ANALYSES OF POWER CONSUMPTION IN ARITHMETIC ALGORITHMS IMPLEMENTATION, RITA, Volume 18, Número 2, Alexandre Wagner C. Faria, Leandro P. de Aguiar, Daniel D. Lara, Antonio A. F. Loureiro, Federal University of Minas Gerais, UFMG, 2011.

[11] TOWARDS USING MEMOIZATION FOR SAVING ENERGY IN ANDROID, Rui Rua, Marco Couto, Adriano Pinto, Jacome Cunha and Joao Saraiva, Universidade do Minho, Universidade NOVA de Lisboa, HASLab/INESC TEC, NOVA LINCS, Portugal,2020.

[12] IMPROVING PERFORMANCE USING COMPUTATIONAL COMPRESSION THROUGH MEMOIZATION: A CASE STUDY USING A RAILWAY POWER CONSUMPTION SIMULATOR, Alejandro Calderon, Alberto Garcıa, Felix Garcia-Carballeira, Jesus Carretero and Javier Fernandez, The International Journal of High Performance Computing Applications 1–17, DOI: 10.1177/1094342016637813, 2016.

[13] A SURVEY OF TECHNIQUES FOR APPROXIMATE COMPUTING, Sparsh Mittal, Oak Ridge National Laboratory, ACM Comput. Surv., Vol. a, No. b, Article 1, Publicati. date: 2015.

[14] APPROXIMATE COMPUTING: AN EMERGING PARADIGM FOR ENERGY-EFFICIENT DESIGN, Jie Han, Department of Electrical and Computer Engineering, University of Alberta Edmonton, AB, Canada, Michael Orshansky, Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA, May 2013.

[15] ΠΑΡΑΛΛΗΛΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ, Βικιπαίδεια, https://el.wikipedia.org/wiki/

[16] ANALYSIS OF POWER DISSIPATION IN EMBEDDED SYSTEMS USING REAL TIME OPERATING SYSTEMS, Robert P. Dick, Ganesh Lakshminarayana, Anand Raghunathan, and Niraj K. Jha, IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems, Vol. 22, No. 5, May 2003.

[17] ENERGY MANAGEMENT TECHNIQUES FOR SOC DESIGN, Chapter 6, Hiroto YASUURA, Tohru ISHIHARA, Masanori MUROYAMA, System LSI Research Center, Kyushu University, 3-8-33, Momochihama, Sawara-ku, Fukuoka, 814-0001, JAPAN, 2006.

[18] SOFTWARE POWER OPTIMIZATION ANALYSIS AND OPTIMIZATION FOR ENERGY-EFFICIENT SOFTWARE, Manuj R Sabharwal, Software Solution Group, Intel Corporation, Folsom, USA. 978-1-61284-660-6/11 © 2011 IEEE.

[19] SOFTWARE DEVELOPMENT LIFECYCLE FOR ENERGY EFFICIENCY: TECHNIQUES AND TOOLS STEFANOS GEORGIOU, Athens University of Economics and Business, Singular Logic S.A., STAMATIA RIZOU, Singular Logic S.A., DIOMIDIS SPINELLIS, Athens University of Economics and Business. ACM Computing Surveys, Vol. 52, No. 4, Article 81. Publication date: August 2019.

[20] LEAKAGE POWER REDUCTION USING BITWIDTH OPTIMIZATION, Yun Cao Hiroto Yasuura, Department of Computer Science and Communication Engineering, Kyushu University, 6–1 Kasuga-koen, Kasuga-shi, Fukuoka 816-8580, Japan, July 2002.

[21] DESIGN TECHNIQUES FOR LOW POWER SYSTEMS, Paul J.M. Havinga, Gerard J.M. Smit, University of Twente, department of Computer Science, P.O. Box 217, 7500 AE Enschede, the Netherlands.

[22] POWER-AWARE DESIGN - PART II REDUCTION & MANAGEMENT, Lecture #8, Mani Srivastava, UCLA - EE Department, EE202A, Fall 2003.

[23] COMPILATION TECHNIQUES FOR LOW ENERGY: AN OVERVIEW, Vivek Tiwari Sharad, Malik Andrew Wolfe, Dept. of Electrical Engineering, Princeton University, Princeton, NJ, Presented at the 1994 Symposium on Low-Power Electronics, San Diego, CA, October 1994.

ΚΕΦΑΛΑΙΟ 6:
Συμπεράσματα

Ενώ αρχικά οι τεχνικές σχεδιασμού επικεντρώνονταν σχεδόν αποκλειστικά στη βελτίωση της απόδοσης, η κατανάλωση ενέργειας γινόταν ολοένα και πιο σημαντικός παράγοντας στη σχεδίαση των συστημάτων. Πολύ μεγαλύτερη προσοχή δίνεται πια στις τεχνικές σχεδιασμού χαμηλής κατανάλωσης ισχύος καθώς υλοποιούνται όλο και περισσότερο, φορητά συστήματα τροφοδοτούμενα από μπαταρία. Τόσο η αύξηση της δυναμικής ισχύος όσο και ακόμη περισσότερο η εκθετική αύξηση της ισχύος διαρροής σε συνδυασμό με την συνεχώς αυξανόμενη συχνότητα ρολογιού, έχουν φέρει σαρωτικές αλλαγές στον τρόπο σχεδιασμού και κατασκευής των ολοκληρωμένων κυκλωμάτων και των επεξεργαστών.

Σε τεχνολογικό και αρχιτεκτονικό επίπεδο, η κατανάλωση ενέργειας μπορεί να μειωθεί μειώνοντας την τάση τροφοδοσίας, το χωρητικό φορτίο και τη συχνότητα μεταγωγής. Μεγάλο κέρδος μπορεί να επιτευχθεί αποφεύγοντας την περιττή δραστηριότητα και σε επίπεδο συστήματος, όπου μεγάλα ενεργειακά οφέλη μπορούν να υπάρξουν από τις δυνατότητες διαχείρισης ισχύος όπου είναι διαθέσιμες, καθώς και τις τεχνικές προγραμματισμού για τη μείωση της κατανάλωσης ενέργειας.

Είναι αξιοσημείωτο ότι ορισμένες τεχνικές εξοικονόμησης ενέργειας όχι μόνο οδηγούν σε μειωμένη κατανάλωση ενέργειας, αλλά και σε περισσότερες επιδόσεις. Για παράδειγμα, ο βελτιστοποιημένος κώδικας τρέχει πιο γρήγορα, είναι μικρότερος και επομένως καταναλώνει λιγότερη ενέργεια. Η χρήση προσωρινής μνήμης σε ένα σύστημα όχι μόνο βελτιώνει την απόδοση, αλλά - παρόλο που απαιτεί περισσότερη επιφάνεια πυριτίου - χρησιμοποιεί λιγότερη ενέργεια καθώς τα δεδομένα διατηρούνται τοπικά. Η προσέγγιση της χρήσης συνεπεξεργαστών FPU και DMA controllers, δεν είναι μόνο πιο αποτελεσματική από την άποψη της κατανάλωσης ενέργειας, αλλά έχει επίσης αύξηση της απόδοσης επειδή οι συγκεκριμένοι συνεπεξεργαστές-ελεγκτές μπορούν να κάνουν την εργασία τους πιο αποτελεσματική σε σχέση με έναν επεξεργαστή γενικής χρήσης. Τα ενεργειακά αποδοτικά ασύγχρονα συστήματα, έχουν επίσης τη δυνατότητα αύξησης της απόδοσης, επειδή η ταχύτητα δεν υπαγορεύεται πλέον από ένα ρολόι, αλλά είναι τόσο γρήγορη όσο η ροή δεδομένων.

Ωστόσο, πρέπει να γίνουν ορισμένοι συμβιβασμοί, αφού οι περισσότερες τεχνικές συχνά οδηγούν σε μειωμένη απόδοση, η οποία μπορεί να βελτιωθεί μόνο με την προσθήκη περισσότερου υλικού. Επιπλέον, τα ενεργειακά αποδοτικά συστήματα μπορεί να είναι πιο περίπλοκα και ο χρόνος αδράνειας μπορεί να αυξηθεί, επειδή πρέπει να ξυπνήσει ένα σύστημα από την κατάσταση ύπνου.

Όπως αποδείχτηκε από τη μελέτη, ο πιο απλός και αποτελεσματικός τρόπος μείωσης της κατανάλωσης ισχύος είναι η μείωση της τάσης τροφοδοσίας (DVS) ή ο συνδυασμός της με μείωση της συχνότητας (DVFS). Ο λόγος για την επιτυχία των τεχνικών

κλιμάκωσης της τάσης ή/και της συχνότητας, πηγάζει από το γεγονός ότι οι μηχανισμοί αυτοί έχουν ενσωματωθεί σε πολλές υλοποιήσεις και μπορούν να χρησιμοποιηθούν σε όλο και πιο αποτελεσματικές πρακτικές για τη διαχείριση της κατανάλωσης ισχύος. Φαίνεται όμως απίθανο ότι η εξάρτηση από περαιτέρω τεχνικές DVFS θα προσφέρει σημαντικές περαιτέρω αποδόσεις στο μέλλον, δεδομένου ότι η ενέργεια διαρροής διαδραματίζει όλο και μεγαλύτερο ρόλο στην κατανάλωση ισχύος λόγω της επιζήμιας επίδρασης των τεχνικών αυτών στα ρεύματα διαρροής. Ένα σημαντικό σημείο επίσης είναι, ότι ενώ το DVFS υπήρξε η ραχοκοκαλιά της διαχείρισης ισχύος στους περισσότερους επεξεργαστές, αντιμετωπίζει ορισμένους σημαντικούς περιορισμούς που θα μειώσουν την αξία του σε μελλοντικούς σχεδιασμούς. Παρότι προσέφερε τετραγωνική μείωση ισχύος, η πρακτική του αξία στηρίζεται σε ένα ευρύ φάσμα τάσεων τροφοδοσίας, το οποίο συρρικνώνεται συνεχώς με τη μείωση του ανώτερου ορίου αλλά όχι με αντίστοιχη μείωση του κατώτερου ορίου. Επιπλέον, οι χαμηλότερες τάσεις τροφοδοσίας αυξάνουν σημαντικά την ευαισθησία των υλοποιήσεων σε σφάλματα και έτσι είναι πολύ πιθανό ότι σε μελλοντικούς σχεδιασμούς το DVFS να είναι είτε ανέφικτο, είτε άνευ ουσίας εάν η ισχύς που απαιτείται για τη διόρθωση σφαλμάτων υπερτερεί της ισχύος που εξοικονομείται με την κλιμάκωση της τάσης.

Όσον αφορά την ενέργεια διαρροής, παραμένει η πιο ανησυχητική από τα προβλήματα που έχουν ανακύψει. Σε επίπεδο αρχιτεκτονικής, έχουν βρεθεί λύσεις για τη μείωση της διαρροής πύλης (π.χ. διηλεκτρικά υψηλού k), αλλά η διαρροή κάτω από το κατώτατο όριο παραμένει πρόβλημα. Οι power gating κυκλωματικές τεχνικές είναι μία λύση, αλλά απαιτούν αυξημένες καλωδιώσεις και εισάγουν πρόσθετη πολυπλοκότητα στο κύκλωμα. Η κλιμάκωση της τάσης τροφοδοσίας μειώνει τόσο τη δυναμική ισχύ όσο και την ισχύ διαρροής (π.χ. τεχνικές μετάπτωσης σε κατάσταση ύπνου), αλλά όπως αναφέρθηκε παραπάνω, οι μειωμένες τάσεις τροφοδοσίας θέτουν σε κίνδυνο την αξιοπιστία των τεχνολογιών νανοκλίμακας. Συνολικά, η διαχείριση διαρροών αποτελεί έναν από τους βασικούς τομείς της μελλοντικής έρευνας σε επίπεδο αρχιτεκτονικής.

Τα κυκλώματα μνήμης αποτελώντας αναπόσπαστο μέρος της σχεδίασης, συμβάλλουν σημαντικά στην κατανάλωση ισχύος σε επίπεδο συστήματος. Ένα ποσοστό πάνω από 25% της συνολικής κατανάλωσης ενός συστήματος, αποδίδεται στα κυκλώματα μνήμης. Επομένως, η μείωση της κατανάλωσης ισχύος στις μνήμες, μπορεί να βελτιώσει σημαντικά την απόδοση, την αξιοπιστία και το συνολικό κόστος του συστήματος.

Όσον αφορά το software, υπάρχουν τεχνικές και εργαλεία για την επίτευξη ενεργειακής απόδοσης σε κάθε φάση ανάπτυξής του. Ο παράλληλος προγραμματισμός αποδεικνύεται επωφελής όσον αφορά την κατανάλωση ενέργειας, εφαρμόζοντας τις κατάλληλες παραμέτρους όπως ο αριθμός νημάτων, το μέγεθος δεδομένων και η τοποθεσία των δεδομένων. Είναι δυνατόν να επιτευχθεί εξοικονόμηση ενέργειας 50% και απόδοση χρόνου εκτέλεσης 65% εάν οι παράμετροι της εφαρμογής συντονιστούν αποτελεσματικά. Ομοίως, η χρήση κατά προσέγγιση τεχνικών, μπορεί να επιφέρει μείωση της ενέργειας από 10% έως και 50% για εφαρμογές με ανοχή στην ακρίβεια των

υπολογισμών. Ωστόσο, αποτελεί πρόκληση τόσο για παράλληλους όσο και κατά προσέγγιση υπολογισμούς, ο εντοπισμός τμημάτων πηγαίου κώδικα που μπορούν να βελτιστοποιηθούν και η επιλογή των κατάλληλων παραμέτρων για τη μείωση της κατανάλωσης ενέργειας. Μια πιθανή προσέγγιση θα μπορούσε να είναι ένας αναλυτής πηγαίου κώδικα που μπορεί να επισημάνει τα πιθανά ενεργοβόρα τμήματα κώδικα και να προτείνει βελτιστοποιήσεις όπως η επιλογή επαρκούς αριθμού νημάτων ή να εντοπίσει τμήματα που μπορούν να επωφεληθούν από τον κατά προσέγγιση υπολογισμό.

Η ανάπτυξη εφαρμογών σε διαφορετικές γλώσσες προγραμματισμού ποικίλει όσον αφορά τη χρήση ενέργειας και την απόδοση χρόνου εκτέλεσης. Γλώσσες προγραμματισμού όπως C και C ++ επιβραβεύουν τους προγραμματιστές με χαμηλότερη κατανάλωση ενέργειας και ταυτόχρονα καλύτερη απόδοση χρόνου εκτέλεσης. Ωστόσο, μόνο ένα μικρό μέρος των διαθέσιμων γλωσσών προγραμματισμού έχει διερευνηθεί όσον αφορά την αποτελεσματικότητά τους στη μείωση της κατανάλωσης ενέργειας.

Ζωτικής σημασίας είναι και η επιλογή ενεργειακά αποδοτικών δομών δεδομένων, διότι μπορεί να επηρεάσει σημαντικά το ενεργειακό κόστος μιας εφαρμογής. Σε ορισμένες περιπτώσεις, η επιλογή μιας πιο αποτελεσματικής δομής δεδομένων, είχε ως αποτέλεσμα την εξοικονόμηση ενέργειας έως 38%. Ωστόσο, το να επιλέγονται συγκεκριμένες δομές δεδομένων χωρίς την ανάγκη πειραματισμού είναι ένα αρκετά φιλόδοξο και δύσκολο έργο. Μια πιθανή ερευνητική κατεύθυνση θα ήταν να εξεταστούν περισσότεροι τύποι δομών δεδομένων, να προσδιοριστούν ποιοι είναι οι πιο ενεργειακοί αποδοτικά για επιλεγμένες περιπτώσεις και να συντεθούν αυτές οι γνώσεις με τη μορφή ενός refactoring εργαλείου που προτείνει τις απαιτούμενες αλλαγές.