Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων
Και Μηχανικων Υπολογιστων

Τομεας Επικοινωνιων, Ηλεκτρονικης &
Συστηματων Πληροφορικης

Κατάτμηση Υλικού Λογισμικού σε Ενσωματωμένα Τηλεπικοινωνιακά Συστήματα

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

του

ΦΩΤΙΟΥ Ε. ΑΝΔΡΙΤΣΟΠΟΥΛΟΥ

Διπλωματούχου Ηλεκτρολόγου Μηχανικού &
Μηχανικού Υπολογιστών Δ.Π.Θ. (1999)

Αθήνα, Ιούλιος 2003

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων & Μηχανικων Υπολογιστων
Τομεας Επικοινωνιων, Ηλεκτρονικης &
Συστηματων Πληροφορικης

Κατάτμηση Υλικού Λογισμικού σε Ενσωματωμένα Τηλεπικοινωνιακά Συστήματα

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

του

ΦΩΤΙΟΥ Ε. ΑΝΔΡΙΤΣΟΠΟΥΛΟΥ

Διπλωματούχου Ηλεκτρολόγου Μηχανικού &
Μηχανικού Υπολογιστών Δ.Π.Θ. (1999)

Συμβουλευτική Επιτροπή: Ε. Πρωτονοτάριος
Η. Αβραμόπουλος
Ι. Πουντουράκης

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την 7η Ιουλίου 2003.

...	...	...
Ε. Πρωτονοτάριος	Η. Αβραμόπουλος	Ι. Πουντουράκης
Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ε.Μ.Π.
...	...	...
Γ. Στασινόπουλος	Ν. Μήτρου	Μ. Θεολόγου
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

...
Δ. Ρεΐσης
Επ. Καθηγητής
Παν. Αθηνών

Αθήνα, Ιούλιος 2003

. . .

ΦΩΤΙΟΣ Ε. ΑΝΔΡΙΤΣΟΠΟΥΛΟΣ
Διδάκτωρ Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών
Ε.Μ.Π.

Περιληψη

Ο σχεδιασμός που βασίζεται σε συστήματα εντός ολοκληρωμένων κυκλωμάτων έρχεται να καλύψει το χάσμα μεταξύ της σταθερά αυξανόμενης χωρητικότητας των συστημάτων όσον αφορά τα τρανζίστορς και της σχεδιαστικής παραγωγικότητας όσον αφορά τον διαθέσιμο χρόνο ανάπτυξης. Ωστόσο, η ολοκλήρωση διαφόρων λειτουργιών ή πρωτοκόλλων σε τηλεπικοινωνιακά συστήματα σε ένα μόνο ολοκληρωμένο κύκλωμα, είτε σε υλικό είτε σε λογισμικό, δημιουργεί κατά την υλοποίηση τους ένα κυρίαρχο σχεδιαστικό πρόβλημα για τα ενσωματωμένα συστήματα. Για τον λόγο αυτό δημιουργείται η ανάγκη για την σωστή επιλογή των στοιχείων του συστήματος που υλοποιούν την εκάστοτε λειτουργία καθώς και η επιλογή της λειτουργίας που θα αντιστοιχισθεί στο εν λόγω στοιχείο.

Σε αυτήν την διατριβή παρουσιάζονται διάφορες τεχνικές που έρχονται να επιλύσουν το πρόβλημα της κατάτμησης των λειτουργιών μεταξύ υλικού και λογισμικού σε ενσωματωμένα τηλεπικοινωνιακά συστήματα, έτσι ώστε να επιτυγχάνεται ο συντομότερος κύκλος σχεδιασμού με την μεγαλύτερη δυνατή απόδοση. Οι προσεγγίσεις που προτείνονται αφορούν την κατάτμηση που πραγματοποιείται από τα αρχικά στάδια της σχεδίασης ενός συστήματος και συγκεκριμένα από το στάδιο των προδιαγραφών. Πέρα από τις κλασσικές δικτυακές εφαρμογές παρουσιάζεται και μία προσέγγιση κατάτμησης υλικού και λογισμικού σε συστήματα ψηφιακής επεξεργασίας σημάτων για την βελτίωση της απόδοσής τους, προβάλλοντας το κέρδος που αποκομίζεται από μία τέτοια διαδικασία. Τα αποτελέσματα που παρουσιάζονται προέρχονται τόσο από αυτοματοποιημένα εργαλεία σχετικά με την κατάτμηση υλικού και λογισμικού, όσο και από εμβόλιμες παρεμβάσεις του σχεδιαστή όπου αυτό κρίνεται απαραίτητο. Σημαντικό είναι το γεγονός ότι η επέμβαση του σχεδιαστή στα εργαλεία αυτά, αποτελεί πολλές φορές επίλυση προβλημάτων τα οποία υπό διαφορετικές συνθήκες οδηγούν σε παρερμηνεύσεις ή σχεδιαστικά λάθη.

Abstract

Design based on system on a chip (SoC) is emerging to close the gap between steadily increasing capacity, in terms of transistors on integrated devices, and design productivity, in terms of the number of transistors designed in a given period. However, the integration of several operations or protocols into a single system on one chip either in hardware or software, makes the implementation of such operations a dominant design problem for embedded systems. There is therefore a need for the right choice between the various components that are going to host the functionality, as well as for the definition of the operations that will be mapped on the specific component.

In this thesis various methods are presented that target to solve the problem of the partitioning between hardware and software in telecommunication embedded systems, so that to achieve short design cycle with the maximum potential performance. The proposed approaches concern partitioning which is performed during the early stages of design and this is the stage of the system's specifications. Except the classic networking applications, such as the network processors, an approach for hardware and software partitioning for digital signal processing systems is proposed, introducing the acquired gain by such procedures. The presented results derive from automated tools related on the hardware and software partitioning, as well as from manual interference by the designer on critical points of the design if needed. The fact that the designer's interference on such tools usually solves major problems that under circumstances could lead many times in deadlocks, is also considerable.

Περιεχόμενα

1 Εισαγωγή **3**

2 Ενσωματωμένα συστήματα **7**
2.1 Περιγραφή ενσωματωμένων συστημάτων 7
2.2 Χαρακτηριστικά διαφοροποίησης 11
2.3 Μοντέλα ενσωματωμένων συστημάτων 15
2.4 Κατηγορίες μοντέλων 17
2.5 Μοντέλα κατάστασης 19
2.5.1 Μηχανή πεπερασμένης κατάστασης 19
2.6 Μοντέλα δραστηριότητας 25
2.6.1 Γράφημα ροής δεδομένων 25
2.6.2 Γράφημα ροής flowchart 28
2.7 Μοντέλα δομής 30
2.7.1 Γράφημα διασυνδεσημότητας στοιχείων 30
2.8 Μοντέλα δεδομένων 33
2.8.1 Γράφημα σχέσης οντότητας 33
2.8.2 Γραφήματα Jackson 35
2.9 Μοντέλα ετερογένειας 36
2.9.1 Γράφημα δομής 40

3 Καταμερισμός Υλικού και Λογισμικού **43**
3.1 Δυικότητα υλικού-λογισμικού 43
3.2 Δομική και λειτουργική κατάτμηση 46
3.2.1 Δομική κατάτμηση 47
3.2.2 Λειτουργική κατάτμηση 51
3.3 Θέματα κατάτμησης 54
3.3.1 Αφαίρεση σε επίπεδο προδιαγραφών 55
3.3.2 Διακριτικότητα 57
3.3.3 Κατανομή στοιχείων συστήματος 59
3.3.4 Μέτρα και εκτιμήσεις 60
3.3.5 Συναρτήσεις αντικειμενικότητας και εγγύτητας 62

3.3.6 Αλγόριθμοι κατάτμησης 64
3.3.7 Αποτέλεσμα αλγορίθμου 66
3.3.8 Ροή ελέγχου και αλληλεπίδραση σχεδιαστή 67
3.3.9 Τυπικός σχηματισμός συστήματος 68
3.4 Βασικοί αλγόριθμοι κατάτμησης 69
3.4.1 Τυχαίας αντιστοίχησης 70
3.4.2 Ιεραρχικής ομαδοποίησης 70
3.4.3 Εξομοιωμένη ισχυροποίηση 74
3.4.4 Γενετικής εξέλιξης 77
3.5 Αλγόριθμοι κατάτμησης υλικού/λογισμικού 80
3.5.1 Πλεονεκτικοί αλγόριθμοι 81
3.5.2 Αλγόριθμοι hill-climbing 83
3.5.3 Δυαδικοί αλγόριθμοι δυαδικής αναζήτησης 85

4 Ο επεξεργαστής δικτύων PRO^3 87
4.1 Περιγραφή του ολοκληρωμένου κυκλώματος PRO^3 88
4.1.1 Βασικά χαρακτηριστικά 89
4.1.2 Εφαρμογές του συστήματος PRO^3 91
4.1.3 Μπλοκ διάγραμμα του PRO^3 96
4.1.4 Ενσωματωμένοι IP πυρήνες 102
4.2 Η αρχιτεκτονική του PRO^3 107
4.2.1 Λειτουργική αρχιτεκτονική του PRO^3 107
4.3 Αρχιτεκτονική 112
4.4 Εσωτερικές διαδρομές δεδομένων 116
4.4.1 IP εφαρμογή 116
4.4.2 ATM εφαρμογή 125
4.5 Ανάλυση και απόδοση του πρωτοκόλλου SSCOP 128
4.5.1 Περιγραφή του πρωτοκόλλου SSCOP 130
4.5.2 Ορισμοί και μεθοδολογία 132
4.5.3 Μετρήσεις επιδόσεων πρωτοκόλλου 135
4.5.4 Ανάλυση πρωτοκόλλου 138
4.5.5 Επιδόσεις πρωτοκόλλου 140
4.6 Ανάλυση του ATM στο PRO^3 144
4.6.1 Δέκτης και πομπός ATM/CPCS 144
4.7 Κατάτμηση υλικού λογισμικού στο PRO^3 149
4.8 Η γλώσσα SpecC 152
4.9 Το περιβάλλον SCE 154
4.10 Ανάλυση του ATM μέσω του SCE 157
4.11 Κατάτμηση της ATM εφαρμογής 170

5 Κατάτμηση σε συστήματα DSP **177**
5.1 Καταστολή ακουστικής ηχούς 178
5.1.1 Μοντέλο καταστολέα ηχούς 179
5.1.2 LMS και θεωρία καταστολής 181
5.2 Ανάλυση του αλγορίθμου LMS 187
5.2.1 Υλοποίηση σε λογισμικό 192
5.2.2 Κατάτμηση σε υλικό/λογισμικό 195

6 Συμπεράσματα **201**

Κατάλογος Σχημάτων

2.1 Συσκευές που χρησιμοποιούν ενσωματωμένα συστήματα 8
2.2 Τυπικό ενσωματωμένο σύστημα 10
2.3 Μοντέλο FSM για ενάν ελεγκτή ανελκτυστήρα 20
2.4 Μοντέλο κατάστασης FSM για τον ελεγκτή ανελκυστήρα . . . 22
2.5 Μοντέλο FSMD για τον ελεγκτή ανελκυστήρα 23
2.6 Γράφημα ροών δεδομένων (a) επιπέδου δραστηριότητας (b) επιπέδου λειτουργίας. 26
2.7 Γράφημα ροής για τη διαδικασία εύρεσης μεγίστων. 29
2.8 Μοντελοποίηση δομής.(a) Μπλοκ διάγραμμα συστήματος, (b) Σχηματικό σε επίπεδο καταχωρητών, (c) Σχηματικό σε επίπεδο πυλών 31
2.9 Παράδειγμα γραφήματος σχέσης οντοτήτων 33
2.10 Γράφημα Jackson 35
2.11 Γράφημα ροής ελέγχου/δεδομένων: (α) Κώδικας προγράμματος, (β) CDFG 38
2.12 CDFG σε επίπεδο δραστηριότητας 39
2.13 Παράδειγμα γραφήματος δομής 41

3.1 Βασικά θέματα κατάτμησης ενός συστήματος 54
3.2 Διαφυγή τοπικών ελαχίστων στην επαναληπτική κατάτμηση . . 65
3.3 Τυπικό σχηματισμός ενός συστήματος κατάτμησης 69
3.4 Ιεραρχική ομαδοποίηση 71

4.1 Επεξεργασία πρωτοκόλλων σε μη ταυτόχρονη εκπομπή και λήψη 95
4.2 Επεξεργασία πρωτοκόλλων σε ταυτόχρονη εκπομπή και λήψη . . 96
4.3 PRO^3 μπλοκ διάγραμμα 97
4.4 Μπλοκ διάγραμμα του E1-32XSR 106
4.5 Λειτουργική αρχιτεκτονική του PRO^3 108
4.6 Αρχιτεκτονική του PRO^3 113
4.7 Μπλοκ διάγραμμα του RPM 115
4.8 Ροές δεδομένων για εφαρμογές IP πάνω από SONET 118

4.9 Ροές δεδομένων για εφαρμογές ATM 127
4.10 Η θέση του SSCOP στη στοίβα σηματοδοσίας του ATM 130
4.11 Μοντέλο λειτουργάς της στοίβας σηματοδοσίας του ATM . . . 135
4.12 Συχνότητα κλήσεων των συναρτήσεων του SSCOP 138
4.13 Κλάδος σε μορφή SDL του SSCOP 139
4.14 Τυπική ροή για τον πομπό και τον δέκτη του SSCOP 142
4.15 Μπλοκ λήψης ATM/CPCS 145
4.16 Μπλοκ εκπομπής ATM/CPCS 148
4.17 Το περιβάλλον του SCE 155
4.18 Μηχανισμός λειτουργία του SCE 156
4.19 Δομή του ATM στο SCE 162
4.20 Ανάλυση απόδοσης του ATM 164
4.21 Τύποι εντολών στον επεξεργαστή Coldfire 166
4.22 Κατανομή εντολών στο DSP56600 167
4.23 Κατανομή εντολών του CRC32 σε υλικό 169
4.24 Σύγκριση του ATM-Rx στις διάφορες πλατφόρμες 170
4.25 Αντιστοίχιση επιπέδων του ATM στο PRO^3 172
4.26 Λειτουργία του SSCOP στο RPM 174
4.27 Λειτουργία του Q.2931 στο RPM 175

5.1 Μοντέλο καταστολέα ακουστικής ηχούς 180
5.2 Φίλτρο τύπου FIR 183
5.3 Γράφημα ροής της καταστολής με LMS 188
5.4 Το περιβάλλον CC Studio και η λήψη των μετρήσεων 193
5.5 Χρονική κατανομή εκτέλεσης στον LMS 197

Κατάλογος Πινάκων

4.1 Ροές δεδομένων και μηνυμάτων 109
4.2 Αριθμός κλήσεων ανά συνάρτηση 137
4.3 Τυπικά μήκη των ανταλλασσόμενων πακέτων 141
4.4 Ληφθέντες χρόνοι για την ροή μετάδοσης 143
4.5 Ληφθέντες χρόνοι για την ροή λήψης 143
4.6 Παρερχόμενος χρόνος για τα μηνύματα σηματοδοσίας 144
4.7 Λέξεις επικοινωνίας με την CAM 146
4.8 Χρονική αλληλουχία των λειτουργιών του ATM 163

5.1 Ενέργειες του LMS σύμφωνα με τους μετρητές 192
5.2 Χρόνοι εκτέλεσης των συναρτήσεων σε λογισμικό 194
5.3 Χρόνοι εκτέλεσης υπό κατάτμηση (`coef_update()`) 197
5.4 Χρόνοι εκτέλεσης υπό κατάτμηση (`coef_update()` και `FIR()`) . 198
5.5 Χρόνοι εκτέλεσης υπό κατάτμηση (`coef_update()` και `FIR()`) . 199
5.6 Ανάλυση υλικού για τις λειτουργίες `coef_update()` και `FIR()` σε FPGA Xilinx Vertex II 199

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

AAL5	ATM Adaptation Layer 5
ACK	Acknowledge
ACRX	ATM/CPCS Receiver
AEC	Acoustic Echo Cancellation
ALU	Arithmetic Logic Unit
ARB	Arbiter
ASIC	Application Specific Integrated Circuit
ATM	Asynchronous Transfer Mode
BCS	Binary Constraint Search
CAM	Content Addressable Memory
CCD	Component Connectivity Diagram
CDFG	Control Data Flow Graph
CFG	Control flow Graph
CPCS	Common Part Convergence Sublayer
CPU	Central Processing Unit
CRC	Cyclic Redunduncy Check
CRIM	Control RAM Interface Module
DDR	Double Data Rate (RAM)
DEB	Debug
DFG	Diagram Flow Graph
DMA	Direct Memory Access
DMM	Data Memory Manager
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processor
EPROM	Electrical Programmable ROM
FFT	Fast Fourier Transform
FIFO	First In First Out
FIR	Finit Impulse Response
FLOWID	Flow Identifier
FPU	Floating Point Unit
FSM	Finite State Machine
FTP	File Transfer Protocol
GNU	Gnu's Not Unix
HW	Hardware

HYRTK	Hyperstone Real Time Kernel
IE	Information Element
IN	Input
INEX	Insert - Extract
IP	Internet Protocol
ITU	International Telecommunication Union
LAPB	Link Access Procedure Balanced
LAPD	Link Access Procedure-D
LMS	Least Mean Square
MAC	Media Access Controller
MHY	Modified Hyperstone
NLMS	Normalised LMS
NLP	Non Linear Processor
OC-48	Optical Carrier 48 (2.488 Gbps)
OUT	Output
PC	Program Counter
PCM	Pulse Code Modulation
PDA	Personal Digital Assistant
PDU	Packet Data Unit
PID	Port Identifier
POS-PHY	Packet Over SOnet - PHYsical
PRO^3	Protocol Processor Project
PTI	Packet Type Identifier
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
ROM	Read Only Memory
RPM	Reconfigurable Pipeline Module
RT	Real Time
RTL	Register Transfer Level
RTOS	Real Time Operating System

SAAL	Signaling ATM Adaptation Layer
SAR	Segmentation and Reassembly
SCE	System on Chip Environment
SD	Send Data
SDL	Specification and Description Language
SDRAM	Synchronous Dynamic Random Access Memory
SDU	Service Data Unit
SoC	System on a Chip
SONET	Synchronous Optical Networking
SR	Status Register
SRAM	Static RAM
SSCOP	Service Specific Connection Oriented Protocol
STAT	Status
TCP	Transmission Control Protocol
TIM	Timers
TRS	Traffic Scheduler
TSC	Task Scheduler
UDP	User Datagram Protocol
UNI	User Network Interface
VC	Virtual Circuit
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VP	Virtual Path

Προλογος

...Επιτέλους! Μετά από 21 χρόνια "σχολείου" μπορώ να τραγουδήσω πια το School's out [Cooper72]. Φτάνοντας στο τέλος μίας διδακτορικής διατριβής και αναλογιζόμενος την ως τώρα πορεία, καταλαβαίνει κανείς τη βαρύτητα της απόφασης να ξεκινήσει μία τέτοια *δοκιμασία*. Το πλέον σίγουρο είναι ότι για την περάτωση της, χρειάζονται πάρα πολλοί άνθρωποι να βρίσκονται δίπλα σου, για να σε βοηθήσει ο καθένας με τον τρόπο του.

Συνεπώς, οι ευχαριστίες μου για την όλη πορεία μου και για αυτά που έμαθα σε αυτήν τη διατριβή θα έπρεπε να αποτελούν ένα ολόκληρο κεφάλαιο για να μην αδικήσω κανέναν. Πρώτα από όλα θέλω να ευχαριστήσω την οικογένεια μου για την συμπαράσταση και την υποστήριξη που μου παρείχε ώστε να ξεκινήσω αλλά και να τελειώσω αυτήν τη διατριβή. Επίσης, τον επιβλέποντα Καθηγητή μου Ε. Πρωτονοτάριο ο οποίος αποτελεί υπόδειγμα οργανωτή για τις ομάδες του Εργαστηρίου Τηλεπικοινωνιών του Ε.Μ.Π. αποδεικνύοντάς το έμπρακτα καθημερινά και τη συμβουλευτική μου επιτροπή που αποτελείται από του Αν. Καθηγητές Ι. Πουντουράκη και Η. Αβραμόπουλο. Ευχαριστώ πολύ τον Καθηγητή Γ. Στασσινόπουλο για την πολύ καλή υποδοχή που μου επεφύλαξε όταν ήρθα στο Ε.Μ.Π. αλλά και για την άριστη συνεργασία που είχαμε όλο αυτόν τον καιρό. Τη διεύρυνση των γνώσεων μου και τη δουλειά που έγινε στα θέματα ψηφιακής επεξεργασίας σημάτων την οφείλω αναμφισβήτητα στον Αν. Καθηγητή του Ε.Μ.Π. Κωνσταντίνο Παπαοδυσσεύς. Ευχαριστώ τον Δρ. Γρηγόριο Δουμένη για την άψογη συνεργασία που είχαμε αλλά και για το ενδιαφέρον που επέδειξε για όλη την ομάδα του Εργαστηρίου Τηλεπικοινωνιών, προτείνοντας πάντα ευρηματικότατες λύσεις όπου απαιτούνταν. Ακόμη, ευχαριστώ τον Επ. Καθηγητή Διονύσιο Ρεΐση για την συμβολή του στο Εργαστήριο και ειδικότερα στην ομάδα του ερευνητικού προγράμματος PRO^3.

Από τους "φίλους" θέλω φυσικά να ευχαριστήσω τον Ιωάννη Μητσό που

με ανέχτηκε (και τον ανέχτηκα!) για εννιά συναπτά έτη, κάνοντας ατελείωτες συζητήσεις επί παντός επιστητού. Τον Χρήστο Χαρόπουλο, ο οποίος τα τελευταία κυρίως χρόνια αναδείχθηκε σε έναν εξαιρετικό φίλο, τα οποία και συνεργαστήκαμε περισσότερο από ποτέ. Ευχαριστώ την Δρ. Ιωάννα Θεολογίτου για την παρέα της, τα σχόλια της, την εμπιστοσύνη της και την απίστευτη εντιμότητα της! Ακόμη, ευχαριστήσω τον Δρ. Φώτη Καρούμπαλη για την την ανταλλαγή τεχνικών και μουσικών απόψεων και τον Δρ. Βασίλη Καλούδη για την φοβερή υπομονή του και τις γνώσεις που μοιράστηκε μαζί μου. Τέλος θα ήθελα να ευχαριστήσω τον Andreas Gerstlauer από το Κέντρο Ενσωματωμένων Υπολογιστικών Συστημάτων στο Πανεπιστήμιο της Καλιφόρνια,(UCI, Irvine) για την άδεια χρήσης του λογισμικού SCE. Νομίζω πως θα έπρεπε να ευχαριστήσω πάρα πολύ κόσμο ακόμη, γιατί πιστεύω πως όλα αυτά τα χρόνια όλα τα άτομα που συνεργάστηκα μαζί τους μου προσέφεραν κάτι. Τους ευχαριστώ πραγματικά **όλους**.

Τελευταία και καλύτερη, αφήνω να ευχαριστήσω την Ελίνα Γεντέκου για την κατανόηση και την υπομονή της, αλλά και για την βοήθεια που μου προσέφερε όποτε απελπιζόμουν! Η συμπαράστασή της προς εμένα ξεπέρασε τη φαντασία μου!

Φώτης Ανδριτσόπουλος

fandrit@telecom.ntua.gr

Κεφάλαιο 1

Εισαγωγή

Η κατάτμηση αποτελεί έναν σχετικά νέο τομέα έρευνας στην επιστήμη του Μηχανικού και επικεντρώνεται σε πολύπλοκα συστήματα όπου οι εφαρμογές εμφανίζουν ιδιαίτερες απαιτήσεις κατά τη σχεδίαση τους. Τα πολύπλοκα αυτά συστήματα, που τείνουν να γίνουν ενσωματωμένα, δεν αναφέρονται πλέον σε μία μικρή μερίδα της αγοράς· αντιθέτως, φαίνεται πως εδώ και αρκετό καιρό χρησιμοποιούνται ακόμη και στην καθημερινή ζωή - πολλές φορές χωρίς να το γνωρίζουμε καν - με αποτέλεσμα να απευθύνονται σε μία τεράστια αγορά σε παγκόσμιο επίπεδο. Συσκευές όπως κινητά τηλέφωνα, μηχανές παιχνιδιών, ηχοσυστήματα και συστήματα ασφαλείας αποτελούν ένα μικρό δείγμα τέτοιων συστημάτων. Λόγω της μεγάλης τους απήχησης στην αγορά, οι σχεδιαστές προσπαθούν να επιτύχουν όσον το δυνατό μικρότερο κόστος για την μαζική παραγωγή τέτοιων συστημάτων, κρατώντας ταυτόχρονα και μικρό χρόνο στην ανάπτυξη του εν λόγω συστήματος, το γνωστό σε όλους time-to-market. Η ταχύτατη εξέλιξη της τεχνολογίας θέτει στενά χρονικά περιθώρια για την εμφάνιση τέτοιων συσκευών στην αγορά, ενώ από την άλλη πλευρά η πολυπλοκότητά τους αυξάνεται με πολύ γρήγορους ρυθμούς.

Για να επιτευχθούν τέτοιου είδους στόχοι, οι σχεδιαστές αυτών των συστη-

μάτων αναζητούν τρόπους και μεθόδους οι οποίοι θα μπορούν να ακολουθούν και να πληρούν αυτούς τους απαιτητικούς ρυθμούς ανάπτυξης, δίνοντας ταυτόχρονα και όσον το δυνατόν καλύτερα αποτελέσματα. Η κατάτμηση του υλικού και του λογισμικού εμφανίσθηκε σαν μία μέθοδος η οποία θα μειώνει δραματικά τον χρόνο ανάπτυξης πολύπλοκων ενσωματωμένων συστημάτων, μέσω αποφάσεων που λαμβάνονται από τα πολύ αρχικά στάδια του σχεδιασμού. Μέσω της κατάτμησης, είναι δυνατόν να εκτιμηθούν με μεγάλη ακρίβεια η συμπεριφορά και οι απαιτήσεις ενός συστήματος, δεδομένου ενός μοντέλου του συστήματος σε επίπεδο αφαίρεσης προδιαγραφών. Τα αποτελέσματα που λαμβάνονται από μία τέτοια κατάτμηση είναι ανεξάρτητα από την πλατφόρμα υλοποίησης μιας και λαμβάνονται από το στάδιο των προδιαγραφών και αποτελούν ανεκτίμητης αξίας πληροφορίες για τον περαιτέρω σχεδιασμό του συστήματος εφόσον αυτό γίνει στα αρχικά στάδια της ανάπτυξης.

Στο Κεφάλαιο 2, γίνεται μία αναφορά στα ενσωματωμένα συστήματα, τόσο από την πλευρά του χρήστη όσο και από την πλευρά του σχεδιασμού και της ανάπτυξής τους. Επισημαίνεται η χρησιμότητά τους στην καθημερινή ζωή για θέματα ψυχαγωγίας αλλά και για σκοπούς όπου η απόδοση και η διεκπεραίωση των λειτουργιών του είναι κρίσιμη. Επίσης, τονίζονται τα μοντέλα που χρησιμοποιούνται για την ανάλυση τους τα οποία είναι ποικίλουν ανάλογα με την εφαρμογή. Στο τρίτο Κεφάλαιο περιγράφεται αναλυτικά τι ακριβώς εννοούμε με τον όρο κατάτμηση συστήματος, δίνοντας έμφαση στην κατάτμηση μεταξύ υλικού και λογισμικού. Παρουσιάζονται επίσης μέθοδοι και αλγόριθμοι οι οποίοι είναι χρήσιμοι ανάλογα με τη φύση του συστήματος και περιγράφεται ο τρόπος επιλογής τους ανάλογα με την περίπτωση που μελετάται. Στο τέταρτο Κεφάλαιο, αναλύεται ένα κατεξοχήν πολύπλοκο τηλεπικοινωνιακό ενσωματωμένο σύστημα που αναπτύχθηκε στο Εργαστήριο Τηλεπικοινωνιών του Εθνικού Μετσοβίου Πολυτεχνείου και είναι ένας επεξεργαστής δικτύου για IP

και ATM εφαρμογές στα 2.4 Gbps. Το σύστημα αυτό ονομάστηκε PRO^3 και παρουσιάζεται πως ακριβώς έγινε η επιλογή για την κατάτμηση του συστήματος σε υλικό και λογισμικό όσον αφορά την εφαρμογή του ATM για ένα σύστημα εντός ολοκληρωμένου κυκλώματος (SoC) όπως είναι το PRO^3. Επιπλέον, το 5ο Κεφάλαιο παρουσιάζει την εφαρμογή μεθόδων κατάτμησης σε συστήματα ψηφιακής επεξεργασίας σημάτων, προβάλλοντας επίσης και τα σημαντικά οφέλη που μπορούν να αποκομιστούν από τέτοιες διεργασίες ακόμη και σε συστήματα DSP. Τέλος, προτείνονται κάποιες ιδέες για μελλοντική συνέχεια της συγκεκριμένης έρευνας, έτσι ώστε η κατάτμηση υλικού λογισμικού να αποτελεί ένα απόλυτο και έγκυρο μέτρο για τη σχεδίαση ενός τηλεπικοινωνιακού συστήματος.

Κεφάλαιο 2

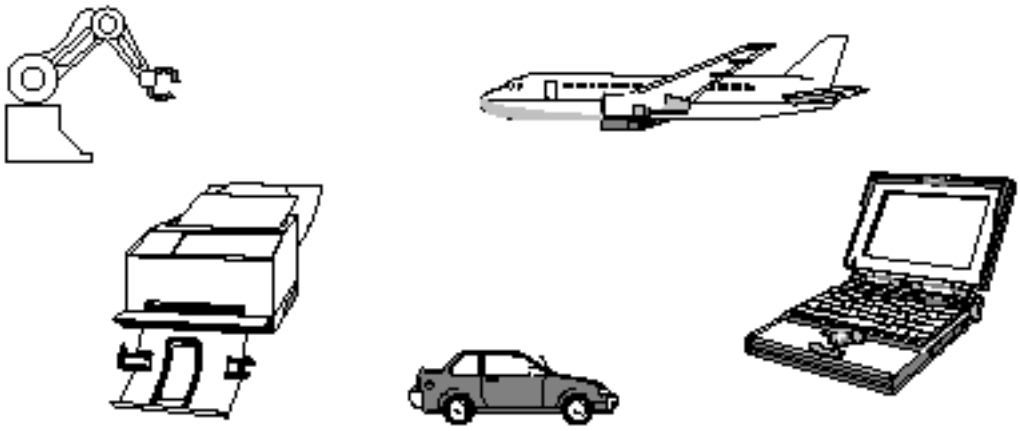
Ενσωματωμένα συστήματα

2.1 Περιγραφή ενσωματωμένων συστημάτων

Στη σημερινή εποχή τα υπολογιστικά συστήματα έχουν κατακτήσει την καθημερινότητα των περισσοτέρων ανθρώπων μέσω συσκευών, τις οποίες χρησιμοποιούν πολλές φορές χωρίς καν να γίνεται αντιληπτό ότι πρόκειται για τέτοια περίπλοκα συστήματα. Συνήθως τα ενσωματωμένα συστήματα υλοποιούνται για να εκτελούν συγκεκριμένες λειτουργίες και όχι σαν υπολογιστικές μηχανές γενικών εφαρμογών. Το Σχήμα 2.1 δείχνει παραστατικά μερικές τέτοιες συσκευές οι οποίες συναντιούνται σε καθημερινή μερικές φορές βάση και εμπεριέχουν ένα ή περισσότερα ενσωματωμένα συστήματα. Μερικά χαρακτηριστικά παραδείγματα ενσωματωμένων συστημάτων, σε συνδυασμό με την περιγραφή των λειτουργιών που εκτελούν παρουσιάζονται συνοπτικά παρακάτω:

- *Ηχοσυστήματα* - Παρέχουν την δυνατότητα επικοινωνίας με το χρήστη σχετικά με την επεξεργασία του ήχου.

- *Συστήματα κλιματισμού* - Παρέχουν την επικοινωνία για τον έλεγχο της θερμοκρασίας ανάλογα με την επιλογή του χρήστη.



Σχήμα 2.1: Συσκευές που χρησιμοποιούν ενσωματωμένα συστήματα

- *Αυτοκίνητα* - Τα σύγχρονα αυτοκίνητα περιλαμβάνουν ποικιλία ενσωματωμένων συστημάτων όπως το σύστημα αποφυγής μπλοκαρίσματος των τροχών κατά το φρενάρισμα, την ηλεκτρονική ανάφλεξη και τον κλιματισμό.

- *Κινητά τηλέφωνα* - Τα ενσωματωμένα συστήματα στην περίπτωση αυτή χρησιμοποιούνται για την απεικόνιση γραφικών, για την κωδικοποίηση των δεδομένων, για την συμπίεση των σημάτων φωνής κτλ.

- *Τηλεοράσεις* - Στον έλεγχο της επιλογής προγραμμάτων και στην επεξεργασία εικόνας.

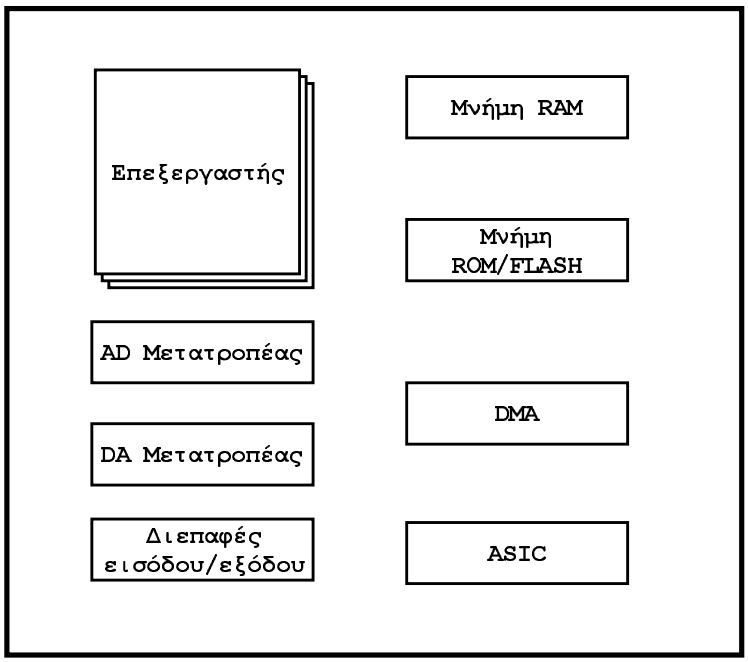
- *Αυτόματοι τηλεφωνητές* - Στη συμπίεση και στην αποθήκευση των δεδομένων φωνής.

Ένας λόγος για τον οποίο τα ενσωματωμένα συστήματα είναι τόσο διαδομένα είναι το ότι για την υλοποίηση τους χρησιμοποιείται συνδυασμός υψηλής απόδοσης υλικού με ευέλικτο λογισμικό, όπου το τελευταίο έχει και μικρό χρόνο

κατά το σχεδιασμό του. Αντιθέτως, παλιότερα τα πρώτα ενσωματωμένα συστήματα ήταν κυρίως συστήματα τραπεζικών συναλλαγών τα οποία υλοποιούνταν σε πολύ μεγάλα υπολογιστικά συστήματα όπως είναι τα γνωστά mainframes τα οποία αποτελούνταν από μεγάλο αριθμό επεξεργαστών και μονάδων αποθήκευσης δεδομένων. Επιπλέον λόγω του ότι αυτό το είδος των συστημάτων κατασκευαζόταν από πολύ υψηλού κόστους εξαρτήματα, χρησιμοποιούνταν για σχετικά λίγες αλλά σημαντικές εφαρμογές.

Αν και τα παραπάνω αποτελούν κάποια παραδείγματα ενσωματωμένων συστημάτων ο ορισμός τους είναι κάτι το περίπλοκο. Το πλήθος των ορισμών για τα ενσωματωμένα συστήματα είναι ίσως ανάλογο του αριθμού των σχεδιαστών που ασχολούνται με αυτά. Η λέξη ενσωματωμένος υποδηλώνει ότι το σύστημα δεν θα αλλάξει αφότου έχει σχεδιασθεί και ότι έχει μία πολύ συγκεκριμένη λειτουργία. Αυτό φαίνεται να ισχύει για τα παραδείγματα που παρουσιάστηκαν παραπάνω αλλά δεν φαίνεται να επαρκεί σαν ένας γενικά αποδεκτός ορισμός του τι είναι ένα ενσωματωμένο σύστημα. Βάσει των όσων αναφέρθηκαν ένας προσωπικός υπολογιστής ο οποίος εκτελεί ένα συγκεκριμένο πρόγραμμα (π.χ. μία εφαρμογή τηλεφωνικού καταλόγου) θα μπορούσε να χαρακτηριστεί ως ένα ενσωματωμένο σύστημα. Παρομοίως, ένα PDA το οποίο χρησιμοποιεί τον ίδιο επεξεργαστή και εκτελεί την ίδια εφαρμογή του τηλεφωνικού καταλόγου φαίνεται να αποτελεί και αυτό ως ένα τέτοιου είδους σύστημα. Για τον λόγο αυτό, όλα τα ενσωματωμένα συστήματα τα οποία παρουσιάζονται σε αυτήν εδώ τη διατριβή αποδίδονται ως συστήματα "υλικού και λογισμικού" τα οποία χρησιμοποιούν για τη λειτουργία τους έναν ή περισσότερους μικροεπεξεργαστές, όπως εικονίζεται και στο Σχήμα 2.2.

Ένα ενσωματωμένο σύστημα αποτελείται από διάφορα εξαρτήματα όπως φαίνεται στο Σχήμα 2.2. Ωστόσο, το σημαντικότερο εξάρτημα - συσκευή, η οποία και συνετέλεσε δραματικά στην εξέλιξη των ενσωματωμένων συστημάτων



Σχήμα 2.2: Τυπικό ενσωματωμένο σύστημα

είναι ο μικροεπεξεργαστής [Up90]. Οι μικροεπεξεργαστές αναπτύχθηκαν από τον απλό 4004 [Coc71] της εταιρείας Intel ο οποίος περιείχε μόνο λίγα τρανζίστορς, στους σημερινούς μικροεπεξεργαστές με περισσότερα από 100 εκατομμύρια τρανζίστορς σε ένα μόνο ολοκληρωμένο σύστημα όπως για παράδειγμα ο επεξεργαστής Alpha 21364 [Ev98], ο PowerPC 750 [Mot97] και ο επεξεργαστής Pentium IV [Intel97]. Η ανακάλυψη των μικροεπεξεργαστών έδωσε τη δυνατότητα χρήσης ενσωματωμένων συστημάτων σε καθημερινές συσκευές όπως αυτές που παρουσιάστηκαν προηγουμένως διατηρώντας το κόστος χαμηλό. Η πολυπλοκότητα των προαναφερθέντων επεξεργαστών, σε συνδυασμό με τη δυνατότητα των μνημών να περιλαμβάνουν 256 εκατομμύρια στοιχεία σε ένα μόνο ολοκληρωμένο κύκλωμα [Sam99] έδωσαν τη δυνατότητα στα ολοκληρωμένα κυκλώματα να περιλαμβάνουν ολόκληρα και αυτόνομα συστήματα.

Η πολυπλοκότητα των σημερινών συστημάτων έχει σαν αποτέλεσμα να εμφανίζονται πολλά προβλήματα κατά τη διάρκεια της διαδικασίας σχεδιασμού. Ο σχεδιασμός ενσωματωμένων συστημάτων εμπλέκει διαφορετικές περιοχές σχεδίασης όπως είναι ο κατανεμημένος σχεδιασμός ενός συστήματος για εφαρμογές δικτύων και επικοινωνιών και ο σχεδιασμός συστημάτων πραγματικού χρόνου που επιβάλει τη λειτουργία σε τους σε αυστηρούς χρονικούς περιορισμούς που τίθενται από τις απαιτήσεις των χρηστών. Αυτοί οι περιορισμοί μπορεί να είναι χαλαροί, όπως για παράδειγμα κατά το σχεδιασμό ενός εκτυπωτή, όπου μία πιθανή καθυστέρηση κατά την εκτύπωση μίας σελίδας δεν είναι κρίσιμη, ή μπορεί να είναι πραγματικά αυστηροί όπως για παράδειγμα σε έναν αυτόματο πιλότο όπου η απόκρισή του πρέπει να γίνεται σε πραγματικό χρόνο και χωρίς λάθη. Πολλά ενσωματωμένα συστήματα περιλαμβάνουν τουλάχιστον μερικούς αυστηρού πραγματικού χρόνου περιορισμούς. Η δουλειά του σχεδιαστή είναι να επιλέξει τον μηχανισμό ο οποίος είναι ικανός να φέρει σε πέρας την εκάστοτε εργασία με χαμηλό κόστος, ικανοποιώντας παράλληλα κάποιους άλλους περιορισμούς όπως είναι το μέγεθος και η κατανάλωση ισχύος.

Τα παραπάνω φαίνεται να είναι αρκετά για να καλύψουν σε υψηλό επίπεδο τα ενσωματωμένα συστήματα όσον αφορά τη χρήση τους στην καθημερινή ζωή. Ωστόσο, η επόμενη παράγραφος, προσεγγίζει τα συστήματα αυτά σε βάθος δίνοντας μία εικόνα των συστημάτων σε χαμηλό επίπεδο και αντιμετωπίζοντας τα με περισσότερο επιστημονικούς ορισμούς.

2.2 Χαρακτηριστικά διαφοροποίησης

Στην προηγούμενη παράγραφο αναφέρθηκε ένα παράδειγμα όπου συγκρίνεται ένα PDA με έναν υπολογιστή τύπου PC τα οποία εκτελούν την ίδια ακριβώς εφαρμογή. Ασφαλώς δεν θα ήταν λογικό να πει κανείς ότι ένα τέτοιο σύστημα

όπως το PC θα μπορούσε να αποτελεί ένα ενσωματωμένο σύστημα. Για τον λόγο αυτό παρατίθενται κάποια βασικά χαρακτηριστικά που ταυτοποιούν και διαφοροποιούν τα ενσωματωμένα συστήματα από άλλα παρόμοια υπολογιστικά συστήματα.

Τα ενσωματωμένα συστήματα εκτελούν αποκλειστικά και μόνο συγκεκριμένες λειτουργίες. Αντιθέτως οι προσωπικοί υπολογιστές αποτελούν γενικές υπολογιστικές πλατφόρμες. Ένα άλλο όνομα που χρησιμοποιείται συχνά για έναν ενσωματωμένο μικροεπεξεργαστή είναι το *αποκλειστικός (dedicated)* μικροεπεξεργαστής. Αυτό προκύπτει από το γεγονός ότι αυτοί οι μικροεπεξεργαστές εκτελούν μία ή λίγες και συγκεκριμένες λειτουργίες. Μία πιθανή αλλαγή στην λειτουργία που εκτελείται από ένα τέτοιο σύστημα, θα οδηγούσε στην αχρήστευση του όλου συστήματος και στον επανασχεδιασμό του από την αρχή. Για παράδειγμα, ο επεξεργαστής ο οποίος αποτελεί την βασική λειτουργική μονάδα μίας ιατρικής μικροσυσκευής συλλογής μετρήσεων, δεν θα μπορύσε να χρησιμοποιηθεί για εφαρμογές προγραμμάτων επεξεργασίας κειμένου.

Τα ενσωματωμένα συστήματα υποστηρίζονται από μεγάλη ποικιλία επεξεργαστών και αρχιτεκτονικών. Οι περισσότεροι προσωπικοί υπολογιστές είναι εξοπλισμένοι με ίδιου τύπου επεξεργαστές ή αν όχι ακριβώς ίδιοι, όλοι ακολουθούν μία κοινή αρχιτεκτονική. Στα ενσωματωμένα συστήματα, μπορεί να υπάρχουν δύο συστήματα που εκτελούν την ίδια λειτουργία αλλά παρόλα αυτά να αποτελούνται από τελείως διαφορετικά δομικά στοιχεία όπως π.χ. ο μικροεπεξεργαστής.

Ένας ακόμη σημαντικός παράγοντας είναι συνήθως το κόστος τους. Αν και το κόστος δεν είναι πάντοτε κριτήριο για το σχεδιασμό, ωστόσο σχετίζεται άμεσα από την ποσότητα παραγωγής και από την χρήση του τελικού συστήματος. Σε συστήματα τα οποία παράγονται μαζικά σε πολλές δεκάδες ε-

κατομμύρια, η εξοικονόμηση του κόστους ακόμη και μερικών μόνο λεπτών θα είχε φοβερή επίδραση. Ωστόσο όταν σχεδιάζονται πρωτότυπα, ή συστήματα τα οποία θα είναι μικρής παραγωγής και συχνά απαιτείται η απόλυτη ακρίβεια και σταθερότητα, τότε το κόστος δεν είναι και από τους πιο σημαντικούς παράγοντες που λαμβάνονται υπόψη κατά τη σχεδίαση.

Περιορισμοί πραγματικού χρόνου. Οι περιορισμοί πραγματικού χρόνου συνήθως ομαδοποιούνται σε δύο διαφορετικές κατηγορίες: *περιορισμοί ευαίσθητοι στο χρόνο* και *περιορισμοί κρίσιμοι στο χρόνο*. Όταν μία διεργασία είναι κρίσιμη στο χρόνο, τότε πρέπει να λάβει χώρα μέσα σε ένα περιορισμένο χρονικό παράθυρο, διαφορετικά η λειτουργία που ελέγχεται από αυτήν τη διεργασία αποτυγχάνει. Ο έλεγχος ενός συστήματος ενός αεροσκάφους αποτελεί ένα καλό παράδειγμα. Αν ο βρόγχος ανάδρασης δεν είναι αρκετά γρήγορος, ο αλγόριθμος ελέγχου καθίσταται ασταθής και το αεροσκάφος δεν θα αποκριθεί σωστά. Αντιθέτως μία διεργασία ευαίσθητη στο χρόνο μπορεί να περατωθεί κανονικά ακόμη και με παραβίαση των χρονικών περιορισμών. Για παράδειγμα, αν μία διεργασία εκτύπωσης χρειάζεται για την διεκπεραίωση της 4.5 ms αλλά τελικά διατηρηθεί για 6.3 ms, τότε το πιο πιθανόν είναι ότι ο εκτυπωτής θα τυπώσει δύο σελίδες ανά λεπτό, τη στιγμή που θα είχε τη δυνατότητα να τυπώσει τρεις σελίδες ανά λεπτό.

Το λειτουργικό σύστημα ενός ενσωματωμένου συστήματος είναι συνήθως ένα RTOS[1]. Όπως υπάρχει ποικιλία στους ενσωματωμένους επεξεργαστές, αντίστοιχα υπάρχει και ποικιλία στα λειτουργικά συστήματα των ενσωματωμένων επεξεργαστών. Τα λειτουργικά συστήματα των ενσωματωμένων συστημάτων φαίνεται πως είναι πολύ διαφορετικά από τα κλασσικά λειτουργικά συστήματα. Καταρχήν δεν είναι διαθέσιμα στο να προσφέρουν τον απαιτούμενο χρόνο σε κάθε διεργασία η οποία είναι έτοιμη προς εκτέλεση. Τα RTOS δίνουν

[1] Real Time Operating System

στην διεργασία με την υψηλότερη προτεραιότητα όλο τον χρόνο που απαιτούν για την περάτωσή της. Αν οι υπόλοιπες διεργασίες αποτύχουν στο να δεσμεύσουν επεξεργαστικό χρόνο, αυτό θα είναι πρόβλημα του προγραμματιστή και όχι του λειτουργικού συστήματος. Αν η σχεδίαση του λογισμικού έχει γίνει σωστά, τότε τα RTOS αποδεικνύονται σταθερότατα σε σύγκριση με τα κοινά λειτουργικά συστήματα.

Περιορισμοί ισχύος σε ενσωματωμένα συστήματα. Συνήθως αυτή η διεργασία αποτελεί θέμα του σχεδιαστή του υλικού του συστήματος. Γενικά ακόμη και η επιλογή του επεξεργαστή που θα χρησιμοποιηθεί στο σύστημα γίνεται χωρίς την γνώμη των σχεδιαστών του λογισμικού του συστήματος. Αν ο γενικός σχεδιασμός του συστήματος βασίζεται σε ένα στενό περιθώριο όσον αφορά την κατανάλωση ισχύος, τότε το πιο πιθανόν είναι ο σχεδιασμός του λογισμικού να γίνεται με τέτοιον τρόπο ώστε ο επεξεργαστής να είναι σε λειτουργία καταστολής (sleep mode), για τον περισσότερο χρόνο λειτουργίας του και να ενεργοποιείται μόνον μέσω συμβάντων. Με άλλα λόγια θα είναι ένα σύστημα πλήρως οδηγούμενο από αιτήσεις διακοπών (interrupt requests). Οι περιορισμοί στην ισχύ επηρεάζουν κάθε σημείο στις αποφάσεις σχεδιασμού του συστήματος, εφόσον επηρεάζουν την επιλογή του επεξεργαστή, την ταχύτητά του και την αρχιτεκτονική της μνήμης του. Οι περιορισμοί που εισάγονται από τις απαιτήσεις του συστήματος θα καθορίσουν αν το λογισμικό θα πρέπει να γραφτεί σε μία γλώσσα χαμηλού επιπέδου από ότι σε μία γλώσσα όπως η C ή η C++, επειδή η μέγιστη επιτρεπόμενη απόδοση πρέπει να επιτευχθεί με περιορισμένη ισχύ. Οι απαιτήσεις ισχύος καθορίζονται από την ταχύτητα του ρολογιού του επεξεργαστή, τον αριθμό των ενεργών στοιχείων του συστήματος όπως οι μνήμες RAM και ROM κτλ.

Ο κώδικας των ενσωματωμένων συστημάτων αποθηκεύεται εξόλοκλήρου σε μνήμη ROM. Όταν εκκινείται ένα ενσωματωμένο σύστημα

πρέπει να υπάρχει κάποιος κώδικας ο οποίος αρχικοποιεί το σύστημα ώστε έτσι ώστε να μπορεί να εκτελεσθεί ο υπόλοιπος κώδικας. Αυτό συνεπάγεται τη δημιουργία του περιβάλλοντος πραγματικού χρόνου, όπως η αρχικοποίηση και η τοποθέτηση των μεταβλητών στην μνήμη RAM, ο έλεγχος ακεραιότητας της μνήμης, ο έλεγχος ακεραιότητας της μνήμης ROM συνήθως μέσω ενός ελέγχου αθροίσματος (checksum) και άλλες διαδικασίες αρχικοποίησης. Από την πλευρά της εκσφαλμάτωσης του συστήματος (debugging), ο κώδικας της ROM έχει πολλούς περιορισμούς και δυσκολίες. Καταρχήν ο εκσφαλματωτής δεν είναι σε θέση να ορίσει ένα σημείο διακοπής διαδικασίας (breakpoint) στην μνήμη ROM. Για να γίνει αυτό θα πρέπει να είναι ικανός να αφαιρέσει την εντολή του χρήστη και να την αντικαταστήσει με μία ειδική εντολή όπως μία εντολή `TRAP` ή μία εντολή διακοπής λογισμικού. Τα `TRAPS` οδηγούν μία μεταφορά σε ένα βολικό σημείο εισαγωγής του εκσφαλματωτή. Σε ορισμένα συστήματα αυτό το πρόβλημα αποφεύγεται με το φόρτωμα της συγκεκριμένης εφαρμογής στην μνήμη RAM. Φυσικά, αυτό προϋποθέτει την ύπαρξη μεγάλου μνήμης RAM έτσι ώστε να μπορεί να φορτώσει όλες τις εφαρμογές.

2.3 Μοντέλα ενσωματωμένων συστημάτων

Το πρώτο βήμα κατά το σχεδιασμό ενός συστήματος είναι ο καθορισμός της λειτουργίας του και κατά συνέπεια, η πρώτη έκδοση των προδιαγραφών του συστήματος θα πρέπει να περιγράφει ξεκάθαρα και επακριβώς την λειτουργία του. Για την κατανόηση και την οργάνωση της καταγραφής της λειτουργίας ενός τέτοιου συστήματος, μπορούν να χρησιμοποιηθούν διάφορα θεμελιώδη μοντέλα τα οποία έχουν αναπτυχθεί κατά καιρούς. Ο σχεδιασμός ενός συστήματος αποτελεί την υλοποίηση της επιθυμητής λειτουργικότητας χρησιμοποιώντας ένα σύνολο από εξαρτήματα (components). Ωστόσο αυτό δεν είναι μία εύκολη

διαδικασία. Για παράδειγμα, ας θεωρήσουμε την διαδικασία προσδιορισμού της λειτουργίας ενός ελεγκτή ανελκυστήρα. Πως περιγράφεται η λειτουργικότητά του με τόσο αρκετή λεπτομέρεια έτσι ώστε να μπορούμε να προβλέψουμε με απόλυτη ακρίβεια ποια θα μπορεί να είναι η θέση του ανελκυστήρα μετά το πάτημα μιας σειράς κουμπιών. Το πρόβλημα με την έκφραση των προδιαγραφών σε φυσική γλώσσα είναι ότι συνήθως καταλήγουν να είναι ασαφής και ελλιπέστατες, πάσχοντας σε λεπτομέρεια η οποία απαιτείται από τέτοιου είδους διεργασίες. Κατά συνέπεια, απαιτείται μία πιο ακριβής προσέγγιση για να καθοριστεί η λειτουργία ενός συστήματος.

Ο πιο κοινός τρόπος για να επιτευχθεί αυτό το επίπεδο της ακρίβειας, είναι η σύλληψη του συστήματος ως μία συλλογή από απλούστερα υποσυστήματα ή στοιχειώδη κομμάτια. Παρακάτω παρουσιάζονται πέντε διαφορετικοί μέθοδοι ανάλυσης και αποσυνθεσιμότητας της λειτουργίας σε απλούστερα τμήματα. Βασικά αυτό που διαχωρίζει τις μεθόδους αυτές είναι οι τύποι των τμημάτων και οι κανόνες οι οποίοι χρησιμοποιούνται για τη σύνθεση των συστημάτων από τέτοιου είδους στοιχειώδη ή μικρότερα κομμάτια έτσι ώστε να δημιουργηθεί η λειτουργία του συστήματος. Κάθε τέτοια μέθοδος ονομάζεται **μοντέλο**.

Αξίζει να σημειωθεί ότι ένα μοντέλο αποτελεί ένα τεκμηριωμένο σύστημα αποτελούμενο από αντικείμενα και συνθετικούς κανόνες και χρησιμοποιείται για την περιγραφή των χαρακτηριστικών ενός ενσωματωμένου συστήματος. Τυπικά θα μπορούσε να χρησιμοποιηθεί ένα συγκεκριμένο μοντέλο για την ανάλυση ενός συστήματος στα συνθετικά του κομμάτια, και στη συνέχεια να παραχθούν οι προδιαγραφές με την περιγραφή αυτών των κομματιών σε μία συγκεκριμένη γλώσσα προγραμματισμού. Ο στόχος όμως ενός μοντέλου είναι να παρέχει μία αφηρημένη έκφραση ενός συστήματος. Οι σχεδιαστές συστημάτων επιλέγουν διαφορετικά μοντέλα σε διαφορετικές φάσεις της διαδικασίας σχεδιασμού, έτσι ώστε να τονιστούν τα σημεία του συστήματος τα οποία ενδιαφέρουν

κάθε συγκεκριμένη στιγμή. Για παράδειγμα, στο στάδιο των προδιαγραφών, ο σχεδιαστής δεν γνωρίζει τίποτα πέρα από την λειτουργία του συστήματος οπότε οφείλει να χρησιμοποιήσει ένα μοντέλο το οποίο δεν επηρεάζει καμία πληροφορία σχετική με την υλοποίηση του συστήματος. Κατά το στάδιο της υλοποίησης όπου η πληροφορία σχετικά με την υλοποίηση είναι ήδη διαθέσιμη, ο σχεδιαστής θα χρησιμοποιήσει άλλο μοντέλο το οποίο μπορεί να περιγράψει και να λάβει υπόψη του τη δομή του συστήματος.

Από τη στιγμή που ο σχεδιαστής έχει καταλήξει στο κατάλληλο μοντέλο ώστε να καθορίσει τη λειτουργία του συστήματος, είναι σε θέση να περιγράψει με λεπτομέρεια το τι ακριβώς πρόκειται να κάνει αυτό το σύστημα. Σε αυτό το σημείο ωστόσο, η διαδικασία σχεδιασμού δεν έχει ολοκληρωθεί καθώς ένα τέτοιο μοντέλο δεν έχει ακόμη περιγράψει επακριβώς το πως θα κατασκευασθεί αυτό το σύστημα. Το επόμενο βήμα για τη διαδικασία του σχεδιασμού είναι η μετατροπή αυτού του μοντέλου σε μία **αρχιτεκτονική**, η οποία εξυπηρετεί στον καθορισμό της υλοποίησης του μοντέλου αναφέροντας λεπτομερώς τον αριθμό και τον τύπο των συνθέτων εξαρτημάτων καθώς επίσης και τις διασυνδέσεις μεταξύ τους. Τα μοντέλα και οι αρχιτεκτονικές είναι σχετικά με την θεμελίωση και την υλοποίηση των συστημάτων σε υψηλού επιπέδου σύλληψη. Τα μοντέλα περιγράφουν πώς λειτουργεί ένα σύστημα ενώ η αρχιτεκτονική περιγράφει το πως ακριβώς θα κατασκευασθεί το εν λόγω σύστημα. Η **διαδικασία του σχεδιασμού** είναι ένα σύνολο σχεδιαστικών διεργασιών όπου μετατρέπει ένα μοντέλο σε μία αρχιτεκτονική.

2.4 Κατηγορίες μοντέλων

Οι σχεδιαστές συστημάτων χρησιμοποιούν πολλά διαφορετικά μοντέλα κατά τις διάφορες μεθοδολογίες σχεδιασμού, είτε πρόκειται για συστήματα υλικού

hardware είτε για συστήματα λογισμικού software. Γενικά, αυτά τα μοντέλα εντάσσονται σε πέντε διαφορετικές κατηγορίες ανάλογα με τον προσανατολισμό τους: (1) κατάστασης, (2) δραστηριότητας, (3) δομής, (4) δεδομένων και (5) ετερογένειας. Ένα μοντέλο **κατάστασης**, όπως είναι το μοντέλο μηχανής πεπερασμένων καταστάσεων (finite-state machine), είναι αυτό το οποίο αναπαριστά το σύστημα σαν ένα σύνολο καταστάσεων και ένα σύνολο μεταβολών μεταξύ τους, το οποίο δέχεται ερεθίσματα από εξωτερικά γεγονότα. Ένα μοντέλο καταστάσεων είναι περισσότερο κατάλληλο για συστήματα ελέγχου, όπως είναι τα συστήματα αντίδρασης πραγματικού χρόνου, κατά την οποία η χρονική συμπεριφορά του συστήματος είναι το σημαντικότερο στοιχείο κατά το σχεδιασμό του. Ένα μοντέλο **δραστηριότητας**, όπως είναι για παράδειγμα τα διαγράμματα ροής, είναι αυτό που περιγράφει ένα σύστημα ως ένα σύνολο από δραστηριότητες οι οποίες σχετίζονται με εξαρτήσεις δεδομένων ή εκτελέσεων εντολών. Αυτό το μοντέλο είναι περισσότερο εφαρμόσιμο και χρήσιμο σε περιπτώσεις συστημάτων μετασχηματισμών, όπως είναι τα συστήματα ψηφιακής επεξεργασίας σημάτων, στα οποία τα δεδομένα περνούν μέσα από μία σειρά μετασχηματισμών με ένα σταθερό συνήθως ρυθμό. Χρησιμοποιώντας ένα μοντέλο **δομής**, όπως είναι για παράδειγμα ένα μπλοκ διάγραμμα, θα μπορούσαμε να περιγράψουμε την διασυνδεσημότητα μεταξύ των διαφόρων υπομονάδων ενός συστήματος. Σε αντίθεση με τα μοντέλα κατάστασης και δραστηριότητας τα οποία εκφράζουν ως επί το πλείστον τη λειτουργία ενός συστήματος, το μοντέλο δομής επικεντρώνεται περισσότερο στην περιγραφή της φυσικής σύνδεσης του συστήματος. Εναλλακτικά, θα μπορούσε να χρησιμοποιηθεί ένα μοντέλο **δεδομένων**, όπως ένα διάγραμμα συσχετισημότητας οντοτήτων, όταν υπάρχει η ανάγκη για απεικόνιση του συστήματος σαν μία συλλογή δεδομένων σχετικά με τα χαρακτηριστικά του. Αυτό το μοντέλο θα ήταν περισσότερο κατάλληλο για συστήματα πληροφοριών όπως είναι για παράδειγμα οι βάσεις δεδομένων,

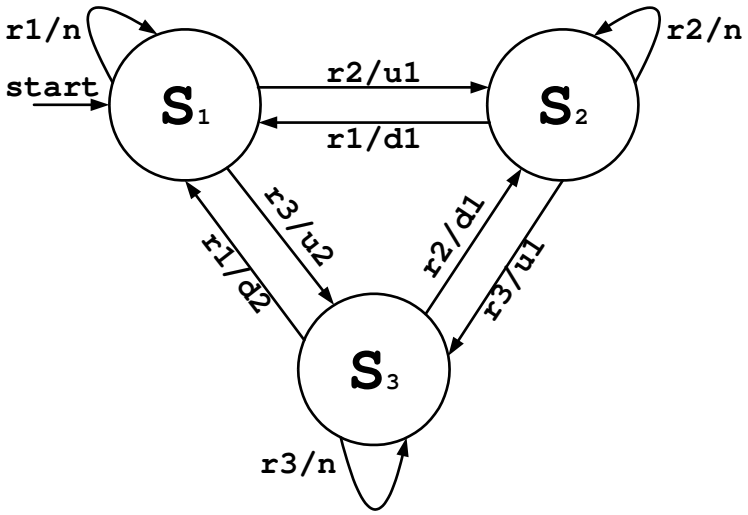
στις οποίες οι λειτουργίες του συστήματος είναι λιγότερο σημαντικές σε σχέση με την οργάνωση των δεδομένων του συστήματος. Τέλος, ένας σχεδιαστής μπορεί να χρησιμοποιήσει το μοντέλο της **ετερογένειας**, το οποίο ενοποιεί πολλά από τα χαρακτηριστικά των τεσσάρων προαναφερθέντων ως τώρα μοντέλων, κάθε φορά που χρειάζεται η αναπαράσταση ενός αρκετά πολύπλοκου συστήματος από διαφορετικές απόψεις.

Αξίζει να σημειωθεί ότι σε ορισμένες μεθοδολογίες που ακολουθούνται χρησιμοποιούνται παραπάνω από ένα διαφορετικά μοντέλα μαζί έτσι ώστε να αναλυθεί και να εποπτευτεί ένα σύστημα από διαφορετικές οπτικές γωνίες. Για παράδειγμα, το εργαλείο Statemate [Harel88] ενσωματώνει τρία ξεχωριστά μοντέλα: (1) Διαγράμματα δραστηριότητας για την ανάλυση του συστήματος σε επίπεδο λειτουργίας και ροής της πληροφορίας, (2) διαγράμματα κατάστασης για τη χρονική συμπεριφορά και τον έλεγχο του συστήματος και (3) διαγράμματα δομής για την ανάλυση της φυσικής του δομής. Αυτό το είδος του σύνθετου μοντέλου ανάλυσης με διαφορετικές εκτιμήσεις δεν αποτελεί ένα μοντέλο ετερογένειας, εφόσον οι πληροφορίες που παρουσιάζονται από τα τρία διαφορετικά μοντέλα, δεν μπορούν να συσχετισθούν μεταξύ τους μέσω μιας κοινής δομής δεδομένων. Αντιθέτως, στην περίπτωση που χρησιμοποιείται μοντέλο ετερογένειας, εφαρμόζεται ένα μοναδικό μοντέλο στο οποίο διαφορετικές απόψεις του συστήματος προκύπτουν από ένα συγκεκριμένο μοντέλο πληροφορίας.

2.5 Μοντέλα κατάστασης

2.5.1 Μηχανή πεπερασμένης κατάστασης

Μία **μηχανή πεπερασμένης κατάστασης** (FSM), αποτελεί ένα παράδειγμα του μοντέλου κατάστασης. Είναι το πιο δημοφιλές μοντέλο για την περιγραφή συστημάτων, εφόσον η χρονική συμπεριφορά ενός τέτοιου συστή-



Σχήμα 2.3: Μοντέλο FSM για ενάν ελεγκτή ανελκτυστήρα

ματος είναι περισσότερο φυσικά αναπαριστάμενη υπό την μορφή καταστάσεων και μεταβάσεων μεταξύ των καταστάσεων.

Βασικά, το μοντέλο μιας FSM αποτελείται από ένα σύνολο **καταστάσεων**, ένα σύνολο **μεταβάσεων** και από ένα σύνολο **ενεργειών** οι οποίες και σχετίζονται με αυτές τις καταστάσεις ή τις μεταβάσεις. Μαθηματικά μία FSM μπορεί να περιγραφεί ως εξης:

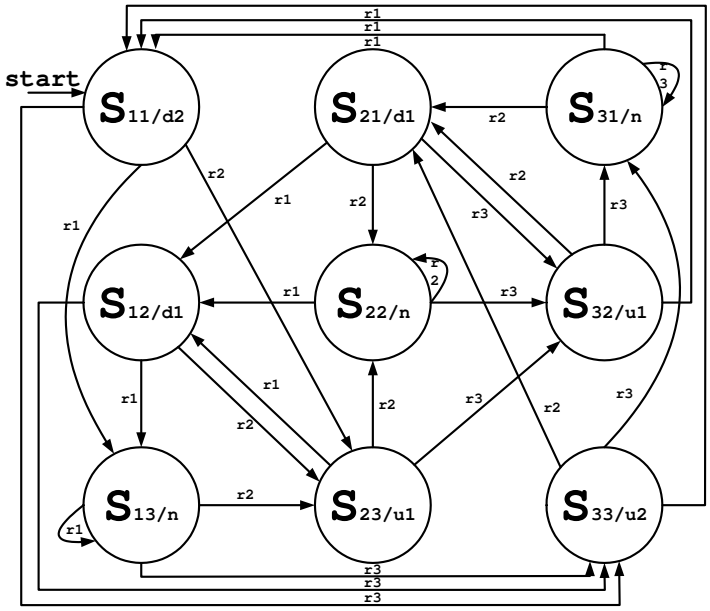
$$< S, I, O, f: S \times I \rightarrow S, h: S \times I \rightarrow O > \tag{2.1}$$

όπου το $S = \{s_1, s_2, \ldots, s_l\}$ είναι ένα σύνολο από καταστάσεις, $I = \{i_1, i_2, \ldots, i_m\}$ είναι ένα σύνολο από εισόδους και $O = \{o_1, o_2, \ldots, o_n\}$ είναι ένα σύνολο από εξόδους. Όπου f αντιστοιχεί η συνάρτηση της επόμενης κατάστασης, η οποία προσδιορίζει την επόμενη κατάσταση λαμβάνοντας υπόψη την τρέχουσα κατάσταση καθώς και την τρέχουσα είσοδο του συστήματος. Αξίζει να σημειωθεί ότι κάθε FSM έχει μία κατάσταση η οποία και διακρίνεται ως η

αρχική κατάσταση και ένα σύνολο από καταστάσεις οι οποίες και διακρίνονται ως οι τελικές καταστάσεις του συστήματος.

Στο Σχήμα 2.3, παρουσιάζεται μία FSM η οποία μοντελοποιεί έναν ελεγκτή ενός ανελκυστήρα σε ένα κτήριο το οποίο αποτελείται από τρεις ορόφους. Σε αυτό το μοντέλο, το σύνολο των εισόδων $I = \{r_1, r_2, r_3\}$ εκφράζει τον επιθυμητό όροφο. Για παράδειγμα, r_2 σημαίνει ότι ο ανελκυστήρας έχει ζητηθεί από τον δεύτερο όροφο. Το σύνολο των εξόδων $O = \{d_2, d_1, n, u_1, u_2\}$ εκφράζει την κατεύθυνση και τον αριθμό των ορόφων όπου πρέπει να μετακινηθεί ο ανελκυστήρας. Για παράδειγμα, d_2 σημαίνει ότι ο ανελκυστήρας πρέπει να κινηθεί προς τα κάτω κατά δύο ορόφους, u_2 σημαίνει ότι πρέπει να κινηθεί προς τα πάνω κατά δύο ορόφους και n σημαίνει ότι ο ανελκυστήρας πρέπει να παραμείνει ανενεργός. Στο Σχήμα 2.3, μπορεί να παρατηρηθεί ότι αν ο ανελκυστήρας βρίσκεται στον δεύτερο όροφο (π.χ. η τρέχουσα κατάσταση είναι η S_2) και ζητηθεί να μεταβεί στον πρώτο όροφο τότε η έξοδος θα είναι η d_1.

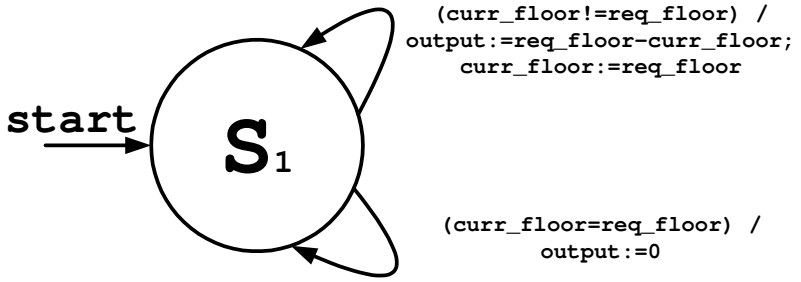
Υπάρχουν δύο τύποι FSM οι οποίοι είναι αρκετά διαδεδομένοι. Ονομαστικά αυτοί είναι οι βασιζόμενοι σε **μετάβαση** (Mealy) και οι βασιζόμενοι σε **κατάσταση** (Moore), οι οποίοι διαφέρουν κατά βάση στον ορισμό της συνάρτησης εξόδου h. Στις FSM μετάβασης η τιμή της εξόδου εξαρτάται από τις τιμές της κατάστασης και της εισόδου ($h\colon S \times I \rightarrow O$). Στις FSM κατάστασης η τιμή της εξόδου εξαρτάται μόνο από την κατάσταση της FSM ($h\colon S \rightarrow O$). Επομένως η έξοδος θα σχετίζεται με τις μεταβάσεις στις FSM μετάβασης ενώ στις FSM κατάστασης θα σχετίζεται με με τις καταστάσεις του συστήματος. Εν αντιθέσει με το Σχήμα 2.3 στο οποίο για την απεικόνιση του μοντέλου χρησιμοποιείται FSM μεταβάσεων, το μοντέλο καταστάσεων για τον ίδιο ελεγκτή που περιγράφηκε παραπάνω, παρουσιάζεται και στο Σχήμα 2.4, στο οποίο η τιμή της εξόδου υποδεικνύεται σε κάθε κατάσταση και συνεπώς αποτελεί ένα μοντέλο FSM καταστάσεων.



Σχήμα 2.4: Μοντέλο κατάστασης FSM για τον ελεγκτή ανελκυστήρα

Κοιτώντας τα μοντέλα αυτά από την πρακτική τους πλευρά, η ουσιαστική διαφορά μεταξύ τους είναι ότι η FSM κατάστασης πιθανόν να απαιτεί μερικές παραπάνω καταστάσεις σε σχέση με το μοντέλο μετάβασης. Αυτό συμβαίνει επειδή στο μοντέλο μετάβασης, μπορεί να υπάρχουν περισσότερες καμπύλες οι οποίες οδηγούν στην ίδια κατάσταση, έστω και αν έχουν διαφορετικές τιμές εξόδου. Στο μοντέλο κατάστασης ωστόσο, κάθε διαφορετική τιμή εξόδου θα απαιτούσε τη δική του κατάσταση, όπως φαίνεται και στην περίπτωση του Σχήματος 2.4.

Στις περιπτώσεις όπου μία FSM πρέπει να αναπαριστά έναν ακέραιο ή έναν κινητής υποδιαστολής αριθμό, θα οδηγούμασταν σε ένα πρόβλημα λόγω των πάρα πολλών καταστάσεων, εφόσον αν κάθε κατάσταση απαιτούσε τη δική της περιγραφή κατάστασης, τότε η FSM θα χρειαζόταν έναν τεράστιο αριθμό καταστάσεων. Για παράδειγμα, ένα αριθμός μήκους 16 bit αναπαρίσταται από 2^{16}



Σχήμα 2.5: Μοντέλο FSMD για τον ελεγκτή ανελκυστήρα

ή 65536 καταστάσεις. Υπάρχει ένας τρόπος ώστε να αποφευχθεί το παραπάνω πρόβλημα, με την επέκταση μίας τέτοιας FSM με ακέραιους και κινητής υποδιαστολής μεταβλητές, έτσι ώστε μία μεταβλητή να αντικαθιστά πολλές χιλιάδες από καταστάσεις. Η εισαγωγή μίας μεταβλητής μήκους 16 bit, θα μείωνε τον αριθμό των καταστάσεων στο μοντέλο της εν λόγω FSM κατά 65536 φορές.

Αυτό το είδος της επέκτασης της FSM ονομάζεται FSM με διαδρομή (FSM with datapath, FSMD), και ορίζεται ως ακολούθως [Ga91]: Ορίζουμε ένα σύνολο από μεταβλητές αποθήκευσης VAR, ένα σύνολο από παραστάσεις $EXP = \{f(x, y, z, \dots)|x, y, z, \dots \in VAR\}$ και ένα σύνολο από αναθέσεις αποθήκευσης $A = \{X \Leftarrow e|X \in VAR, e \in EXP\}$. Στη συνέχεια ορίζουμε ένα σύνολο από παραστάσεις κατάστασης σαν λογικές σχέσεις μεταξύ των δύο εκφράσεων από το σύνολο $EXP, STAT = \{Rel(a, b)|a, b \in EXP\}$. Δοθέντος αυτών των ορισμών, μία μηχανή FSMD μπορεί να ορισθέι ώς εξής:

$$< S, I \cup STAT, O \cup A, f, h > \tag{2.2}$$

όπου το σύνολο των τιμών εισόδου έχει επεκταθεί έτσι ώστε να συμπεριλάβει εκφράσεις καταστάσεων, το σύνολο των εξόδων έχει επεκταθεί ώστε να συμπεριλαμβάνει αναθέσεις αποθήκευσης, καθώς f και h ορίζονται ως οι

αντιστοιχίες $S \times (I \in STAT) \rightarrow S$ και $S \times (I \in STAT) \rightarrow (O \in A)$ αντιστοίχως. Χρησιμοποιώντας αυτό το είδος της FSMD το μοντέλο του ανελκυστήρα που παρουσιάζεται στο Σχήμα 2.3 μπορεί να μοντελοποιηθεί όπως φαίνεται στο Σχήμα 2.5. Η μείωση στον αριθμό των καταστάσεων είναι δυνατή επειδή έχει εισαχθεί η έννοια της μεταβλητής *curr_floor* ώστε να αποθηκεύεται κάθε φορά η τιμή του τρέχοντος ορόφου και κατά συνέπεια να ελαχιστοποιείται η ανάγκη για δέσμευση μίας κατάστασης για κάθε δυνατό όροφο που υπάρχει στο μοντέλο.

Γενικότερα, μία FSM είναι κατάλληλη για την μοντελοποίηση συστημάτων ελέγχου κατά κύριο λόγο, ενώ η FSMD μπορεί να είναι κατάλληλη τόσο για συστήματα ελέγχου όσο και για υπολογιστικά συστήματα. Ωστόσο πρέπει να τονισθεί ότι κανένα από τα δύο αυτά μοντέλα δεν είναι κατάλληλο για την μοντελοποίηση πολύπλοκων συστημάτων και αυτό συμβαίνει διότι κανένα από τα δύο δεν φαίνεται να υποστηρίζει ξεκάθαρα τις έννοιες της συμπτωσιμότητας (concurrency) και της ιεραρχίας (hierarchy). Χωρίς την ξεκάθαρη υποστήριξη για συμπτωσιμότητα, ένα πολύπλοκο σύστημα θα κατέρρεε πολύ εύκολα εξαιτίας του πολύ μεγάλου αριθμού καταστάσεων. Ας θεωρήσουμε για παράδειγμα, ένα σύστημα το οποίο αποτελείται από δύο συμπτωτικά υποσυστήματα όπου το καθένα να αποτελείται από 100 πιθανές καταστάσεις. Εάν γίνει προσπάθεια αναπαράστασης αυτού του συστήματος ως μία μόνο FSM ή FSMD θα πρέπει να αντιστοιχηθούν όλες οι πιθανές καταστάσεις του συστήματος οι οποίες θα ήταν $100 \times 100 = 10000$. Την ίδια στιγμή, η απουσία ύπαρξης της έννοιας της ιεραρχίας θα προκαλούσε μία αύξηση στον αριθμό των καμπυλών όπως αυτές παρουσιάσθηκαν στα προηγούμενα σχήματα. Για παράδειγμα, εάν υπήρχαν 100 διαφορετικές καταστάσεις, οι οποίες θα απαιτούσαν την δική τους ξεχωριστή καμπύλη η κάθε μία, έτσι ώστε να παρασταθεί η μετάβαση σε μία συγκεκριμένη κατάσταση με συγκεκριμένη τιμή εισόδου, θα απαιτούνταν 100 διαφορετικές

καμπύλες, σε αντίθεση με την μία μόνο καμπύλη που θα χρειαζόταν ένα μοντέλο το οποίο θα είχε τη δυνατότητα να ομαδοποιήσει ιεραρχικά 100 ομάδες σε μία μόνο κατάσταση.

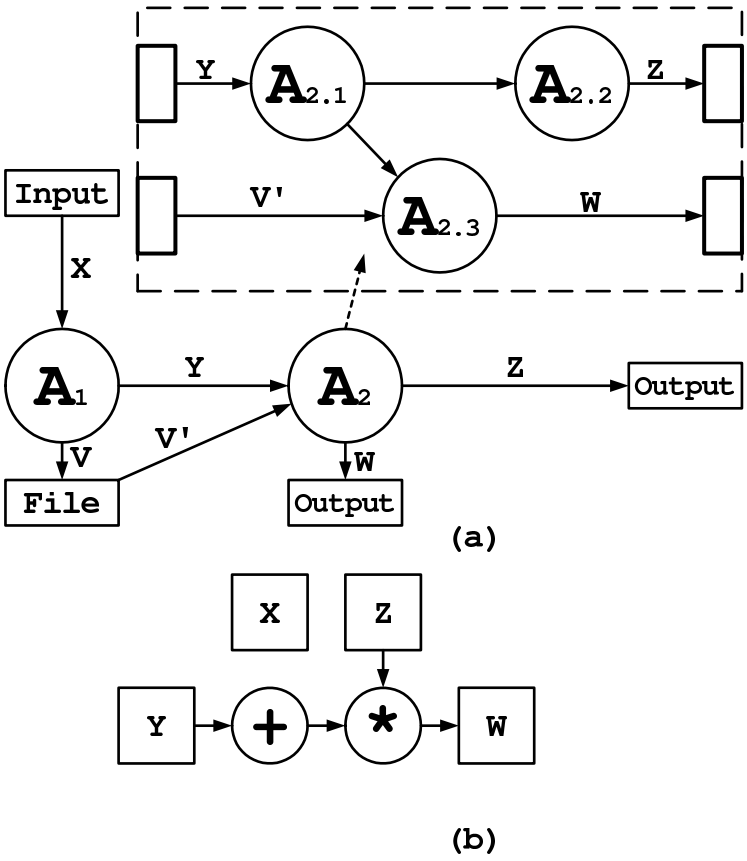
2.6 Μοντέλα δραστηριότητας

2.6.1 Γράφημα ροής δεδομένων

Τα μοντέλα καταστάσεων που παρουσιάστηκαν στην προηγούμενες παραγράφους, χρησιμοποιούνται κυρίως σε συστήματα αντίδρασης, στα οποία η κατάσταση του συστήματος αλλάζει ως απόκριση σε κάποια εξωτερικά γεγονότα. Τα **γραφήματα ροής** (DFG[2]) [Dem79], [Dav83], [Ga91] αντιθέτως, χρησιμοποιούνται κυρίως σε συστήματα μετασχηματισμών, στα οποία οι έξοδοί τους προσδιορίζονται από ένα σύνολο υπολογισμών πάνω στις εισόδους του συστήματος. Τα DFGs συνεπώς δεν έχουν καταστάσεις ούτε όμως και εξωτερικά ερεθίσματα τα οποία θα μπορούσαν να αλλάξουν μία κατάσταση. Αντιθέτως, αποτελούνται απλώς από ένα σύνολο από **δραστηριότητες** (μετασχηματισμούς) οι οποίες συσχετίζονται με ένα σύνολο από καμπύλες οι οποίες αναπαριστούν τη **ροή δεδομένων**.

Πιο συγκεκριμένα, ένα DFG αποτελείται από ένα σύνολο κόμβων και από ένα σύνολο αιχμών. Υπάρχουν διάφοροι τύποι κόμβων στα γραφήματα ροών δεδομένων. Ένας τύπος κόμβου συμπεριλαμβάνει κόμβους **εισόδου** (γνωστά και ως πηγές) και **εξόδου** (γνωστά και ως προορισμοί), οι οποίοι και αντικατοπτρίζουν την είσοδο ή την έξοδο των δεδομένων αντίστοιχα. Ένα δεύτερος τύπος είναι οι κόμβοι **δραστηριότητας** (οι οποίοι αποκαλούνται και ως διεργασίες) οι οποίοι αναπαριστούν δραστηριότητες που μετασχηματίζουν ή παραποιούν δεδομένα. Τέτοιες δραστηριότητες μπορούν να περιγραφούν με διαφόρους τρό-

[2]Dataflow graphs



Σχήμα 2.6: Γράφημα ροών δεδομένων (a) επιπέδου δραστηριότητας (b) επιπέδου λειτουργίας.

πους από προγράμματα, διαδικασίες, συναρτήσεις ή μία μόνο εντολή ή ακόμη και με μία αριθμητική λειτουργία. Ο τελικός τύπος του κόμβου είναι ο τύπος **αποθήκευσης δεδομένων** που αναπαριστά διαφορετικές μορφές των αποθηκευμένων δεδομένων όπως είναι αρχεία βάσεων δεδομένων, αρχεία σε κάποιο λειτουργικό σύστημα, ή μία μεταβλητή σε μία μνήμη ή έναν καταχωρητή.

Ένα παράδειγμα φαίνεται στο Σχήμα 2.6. Το σύστημα αυτό αποτελείται από από δύο δραστηριότητες, A_1 και A_2 όπου η δεύτερη στη συνέχεια αναλύεται στις δραστηριότητες $A_{2.1}, A_{2.2}$ και $A_{2.3}$. Σε αυτό το σύστημα τα δεδομένα Q

θα έχουν μία ροή από την είσοδο προς την A_1, καθώς τα δεδομένα V θα πρέπει να υπολογιστούν από την A_1 και να αποθηκευτούν στο $File$. Τα δεδομένα V' θα πρέπει στη συνέχεια να διαβασθούν από το $File$ και να χρησιμοποιηθούν ως είσοδος στην A_2 σε σύνδεση με τα δεδομένα U τα οποία παρήχθησαν από την A_1. Τα δεδομένα Z και W είναι οι έξοδοι οι οποίες παράγονται από την A_2.

Ένα μοντέλο γραφήματος ροής είναι πολύτιμο επειδή μπορεί να χρησιμοποιηθεί σε εφαρμογές διαφορετικών πεδίων, ή σε διαφορετικές φάσεις του σχεδιασμού του ίδιου πεδίου, απλώς με την σύνδεση διαφορετικών αντικειμένων με κόμβους και αιχμές στο γράφημα. Για παράδειγμα, στον τομέα της ψηφιακής επεξεργασίας σημάτων, οι κόμβοι σε ένα γράφημα γραφήματος ροής θα μπορούσαν να αναπαριστούν **μεταβλητές** και **αριθμητικές λειτουργίες**, όπως είναι η πρόσθεση και ο πολλαπλασιασμός, ενώ οι αιχμές του γραφήματος ροής θα μπορούσαν να δηλώνουν **εξαρτώμενα δεδομένα**, όπως φαίνεται και στο δεύτερο τμήμα του Σχήματος 2.6. Σε αυτό το παράδειγμα, η πράξη '+' εξαρτάται από τα δεδομένα Q και U και η πράξη '*' εξαρτάται από τα δεδομένα Z καθώς και από την έξοδο της πράξης '+'.

Πρέπει να σημειωθεί ότι ένα DFG δεν περιγράφει τίποτα παραπάνω από το πρόβλημα της εξάρτησης των δεδομένων μεταξύ των διαφόρων δραστηριοτήτων. Πιο συγκεκριμένα, το DFG δεν περιέχει καμία πληροφορία σχετικά με την υλοποίηση του. Για αυτούς τους λόγους, το DFG χρησιμοποιείται περισσότερο κατά τη διάρκεια της φάσης των προδιαγραφών ενός συστήματος σαν ένα μέσο επικοινωνίας μεταξύ των σχεδιαστών και των πελατών. Επίσης, λόγω της υποστήριξης ιεραρχικής ανάλυσης, αυτό το μοντέλο είναι κατάλληλο για τον καθορισμό περίπλοκων συστημάτων μετασχηματισμών. Ωστόσο, λόγω του ότι το μοντέλο αυτό δεν εκφράζει καμμία χρονική συμπεριφορά ή ενέργεια ελέγχου του συστήματος πέρα από την εξάρτηση των δεδομένων, φαίνεται πως είναι

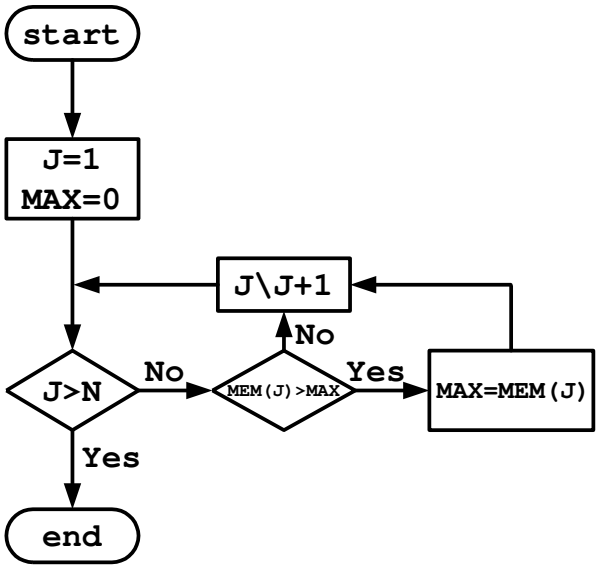
ανίκανο για την πλήρη περιγραφή ενσωματωμένων συστημάτων.

2.6.2 Γράφημα ροής flowchart

Τα γραφήματα ροής flowcharts [Dav83], [Sod90], τα οποία είναι γνωστά και ως **γραφήματα ελέγχου ροής** (CFG[3]), είναι μοντέλα σχετικά με δραστηριότητες τα οποία είναι πολύ συναφή με τα μοντέλα DFG, ωστόσο διαφέρουν στη χρήση των καμπυλών που παρουσιάστηκαν προηγουμένως. Σε ένα μοντέλο DFG, οι καμπύλες χρησιμοποιούνται για να αναπαραστήσουν τη ροή των δεδομένων, ενώ τα γραφήματα ροής flowchart χρησιμοποιούν τις καμπύλες για να δείξουν τη ροή του ελέγχου. Τα γραφήματα ροής είναι επίσης παρόμοια με τα μοντέλα FSMD με την έννοια ότι και τα δύο δίνουν έμφαση στην πλευρά του ελέγχου ενός συστήματος, ωστόσο διαφέρουν στους μηχανισμούς οι οποίοι υποκινούν τις μεταβάσεις του. Πιο συγκεκριμένα, οι μεταβάσεις σε ένα μοντέλο FSMD υποκινούνται από την ύπαρξη εξωτερικών γεγονότων, ενώ οι μεταβάσεις σε ένα γράφημα ροής προκαλείται κάθε φορά που μία συγκεκριμένη δραστηριότητα ολοκληρώνεται.

Βασικά, ένα γράφημα ροής τύπου flowchart αποτελείται από ένα σύνολο από κόμβους και από ένα σύνολο από καμπύλες. Μεταξύ των διαφόρων τύπων κόμβων είναι και οι κόμβοι **εκκίνησης** και **τέλους** οι οποίοι υποδεικνύουν τα σημεία εκκίνησης και τερματισμού ενός γραφήματος ροής. Ένας δεύτερος τύπος είναι οι κόμβοι **επεξεργασίας** οι οποίοι χρησιμοποιούνται ώστε να ορίσουν τον μετασχηματισμό των δεδομένων μέσα από μία σειρά δηλώσεων ανάθεσης. Τέλος, υπάρχουν και οι κόμβοι **απόφασης** οι οποίοι χρησιμοποιούνται για τον έλεγχο των διακλαδώσεων της ροής. Οι διάφοροι τύποι κόμβων σε ένα γράφημα ροής flowchart διασυνδέονται μεταξύ τους με κατευθυνόμενες αιχμές οι οποίες υποδεικνύουν τη σειρά με την οποία θα εκτελούνται οι διάφοροι κόμβοι.

[3]Control Flow Graphs



Σχήμα 2.7: Γράφημα ροής για τη διαδικασία εύρεσης μεγίστων.

Γραφικά, ένα γράφημα ροής θα χρησιμοποιούσε στρογγυλεμένα τετράγωνα για να υποδηλώσει τους κόμβους εκκίνησης και τερματισμού, απλά τετράγωνα για τους κόμβους υπολογισμού, και ρόμβους για κόμβους αποφάσεων. Στο Σχήμα 2.7 παρουσιάζεται ένα παράδειγμα ενός γραφήματος ροής το οποίο υπολογίζει την μέγιστη τιμή μεταξύ N αριθμών οι οποίοι είναι αποθηκευμένοι σε έναν πίνακα που ονομάζεται MEM.

Ένα γράφημα ροής flowchart είναι αρκετά χρήσιμο όταν ένα σύστημα χρειάζεται να περιγραφεί ως ένα σύνολο από διαδοχικές δραστηριότητες, το οποίο κατευθύνεται από μία ροή ελέγχου. Αυτό το μοντέλο φαίνεται να είναι κατάλληλο για αυτά τα συστήματα τα οποία έχουν σαφώς ορισμένες λειτουργίες που πρέπει να εκτελέσουν και τα οποία επίσης δεν εξαρτώνται από εξωτερικά γεγονότα. Μπορεί επίσης να χρησιμοποιηθεί για να εκμεταλλευθεί η συγκεκριμένη σειρά στην εκτέλεση των δραστηριοτήτων ενός DFG όταν απαιτείται αντικατάσταση στις εξάρτηση φυσικών δεδομένων. Λόγω του ότι μία τέτοια

προσέγγιση θα πρότεινε αυτόματα και μία συγκεκριμένη υλοποίηση του συστήματος, τα γραφήματα ροής flowchart χρησιμοποιούνται με τέτοιο τρόπο μόνο όταν η υλοποίηση ενός συστήματος είναι απόλυτα κατανοητή.

2.7 Μοντέλα δομής

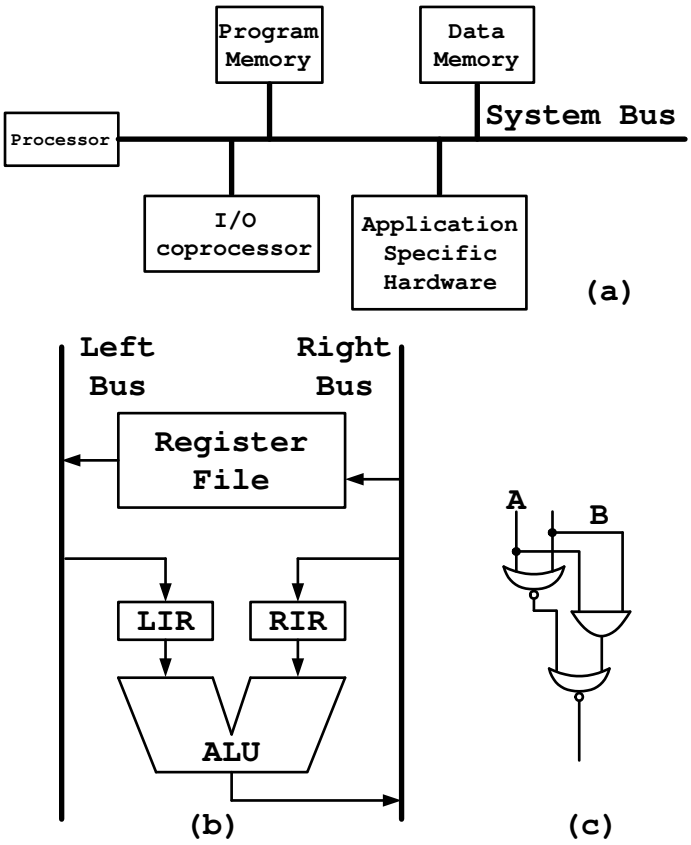
2.7.1 Γράφημα διασυνδεσημότητας στοιχείων

Τα γραφήματα διασυνδεσημότητας στοιχείων CCD[4], αποτελούν μία κατηγορία μοντέλων σχετιζόμενα με τη δομή, τα οποία χρησιμοποιούνται για να περιγράψουν τη φυσική δομή ενός συστήματος όπως προβλέπεται από την λειτουργία του. Σε αντίθεση με τα μοντέλα DFG ή τα γραφήματα ροής flowchart τα οποία αναπαριστούν ένα σύνολο από δραστηριότητες ενός συστήματος οι οποίες σχετίζονται με εξαρτήσεις δεδομένων ή ελέγχου, ένα μοντέλο CCD αναπαριστά ένα σύνολο από τα στοιχεία ενός συστήματος και τις διασυνδέσεις τους. Πιο απλά, μοντελοποιεί τη δομή ενός συστήματος.

Ένα μοντέλο CCD αποτελείται από ένα σύνολο από κόμβους και από αιχμές. Οι κόμβοι απεικονίζουν τα διάφορα **στοιχεία**, τα οποία ορίζονται ως δομημένα αντικείμενα με ένα καθορισμένο σύνολο από εισόδους και εξόδους, όπως είναι οι πύλες, οι μονάδες αριθμητικής λογικής, οι επεξεργαστές και γενικότερα κάθε υποσύστημα. Οι αιχμές αντίστοιχα απεικονίζουν τις διάφορες **διασυνδέσεις** μεταξύ αυτών των συνδέσεων, όπως είναι οι δίαυλοι και τα καλώδια δεδομένων.

Λόγω του ότι αυτό το μοντέλο επιτρέπει στον σχεδιαστή να αντιστοιχίσει διαφορετικά αντικείμενα στους κόμβους του συστήματος και τις αιχμές, το μοντέλο CCD, μπορεί να συγχωνευτεί από μία ποικιλία μοντέλων. Στο Σχήμα 2.8 το μοντέλο CCD παρουσιάζεται με τρία διαφορετικά επίπεδα αφαί-

[4]Component Connectivity Diagram

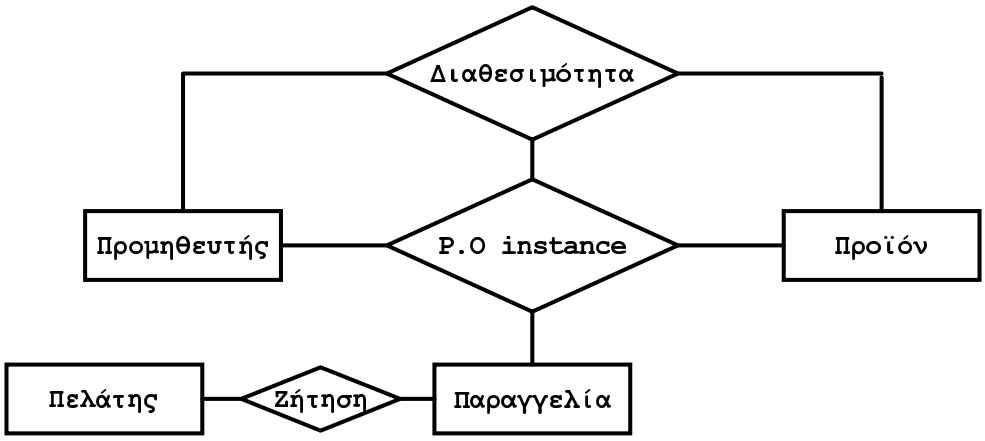


Σχήμα 2.8: Μοντελοποίηση δομής.(a) Μπλοκ διάγραμμα συστήματος, (b) Σχηματικό σε επίπεδο καταχωρητών, (c) Σχηματικό σε επίπεδο πυλών

ρεσης, παράγοντας ένα *σύστημα μπλοκ γραφήματος*, ένα *σχηματικό σε επίπεδο καταχωρητών* και ένα *σχηματικό σε επίπεδο πυλών*. Στο μπλοκ γράφημα τα στοιχεία ορίζονται ως δομικά στοιχεία σε επίπεδο συστήματος όπως είναι οι επεξεργαστές, οι μνήμες και τα ολοκληρωμένα κυκλώματα ειδικής λειτουργίας ASIC[5]. Αξίζει να σημειωθεί ότι διασυνδέσεις μεταξύ αυτών των στοιχείων είναι μόνο μερικώς προδιαγεγραμμένες, εφόσον αυτό το γράφημα δεν εμπεριέχει λεπτομερή πληροφορία για τη διασυνδεσημότητά τους, όπως είναι για παράδειγμα το πλάτος του δίαυλου δεδομένων και συγκεκριμένα σήματα ελέγχου. Στο σχηματικό σε επίπεδο καταχωρητών τα στοιχεία του συστήματος πρόκειται να αναπαραστήσουν μονάδες σε επίπεδο καταχωρητών, όπως είναι οι μονάδες αριθμητικής λογικής, οι καταχωρητές, οι επιλογείς ή οι δίαυλοι και οι διασυνδέσεις καθορίζουν τον τρόπο μεταφοράς δεδομένων μεταξύ των αριθμητικών στοιχείων και των στοιχείων αποθήκευσης. Σε αυτό τον τύπο των σχηματικών συνήθως δεν προβάλλονται τα σήματα ελέγχου. Τέλος, τα σχηματικά σε επίπεδο πυλών χρησιμοποιούν πύλες σαν στοιχεία του συστήματος και σε αυτήν την περίπτωση οι διασυνδέσεις μεταξύ αυτών των στοιχείων αναπαριστούν πραγματικά φυσικά καλώδια. Πιο απλά, οι διασυνδέσεις δεδομένων και ελέγχου είναι πλήρως καθορισμένες.

Εξαιτίας του γεγονότος ότι το μοντέλο της διασυνδεσιμότητας στοιχείων είναι πολύ κατάλληλο για την ολοκληρωμένη αναπαράσταση των συστημάτων, χρησιμοποιείται κυρίως στα τελευταία στάδια της διαδικασίας σχεδιασμού, όταν ο σχεδιαστής επιθυμεί να προσδιορίσει την υλοποίηση του συστήματος.

[5] Application Specific Integrated Circuits



Σχήμα 2.9: Παράδειγμα γραφήματος σχέσης οντοτήτων

2.8 Μοντέλα δεδομένων

2.8.1 Γράφημα σχέσης οντότητας

Ένα μοντέλο δεδομένων είναι πολύ διαφορετικό από τα μοντέλα καταστάσεων και δραστηριοτήτων που παρουσιάστηκαν ως τώρα, κυρίως διότι επικεντρώνεται στην αναπαράσταση των δεδομένων σε αντίθεση με την παρουσίαση των δραστηριοτήτων τα οποία επεμβαίνουν στα δεδομένα. Τα μοντέλα δεδομένων χρησιμοποιούνται γενικότερα στο σχεδιασμό πληροφοριακών συστημάτων λόγω του ότι σε αυτό το είδος των συστημάτων η οργάνωση των δεδομένων υπερτερεί υπέρ όλων των υπολοίπων παραμέτρων του σχεδιασμού. Ένα παράδειγμα ενός συστημάτος μοντέλου δεδομένων είναι το **γράφημα σχέσης οντοτήτων** (ERD[6]) [Che77], [Teo90], το οποίο ορίζει ένα σύστημα με την μορφή μίας συλλογής από οντότητες και τις διάφορες σχέσεις μεταξύ τους.

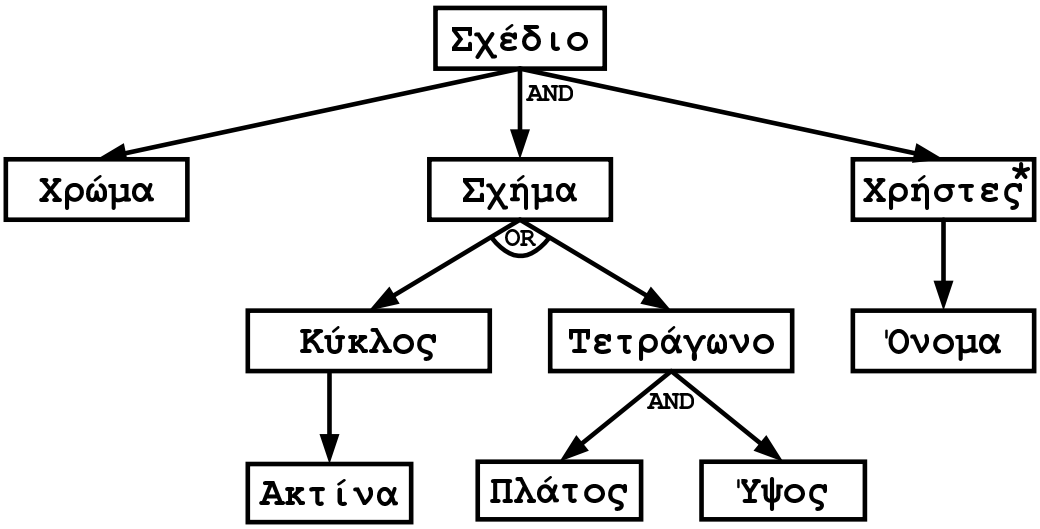
Σε ένα ERD κάθε οντότητα συνήθως αναπαριστάται γραφικά από ένα τετράγωνο πλαίσιο, ενώ μία σχέση με ένα ρόμβο. Ας υποθέσουμε για παράδειγμα

[6]Entity Relationship Diagram

ότι θέλουμε να αναπαραστήσουμε την πληροφορία κατά την οποία ένα κατάστημα παραγγέλνει διαφορετικά προϊόντα για τους πελάτες του. Σε ένα ERD μοντέλο θα απαιτούνταν τέσσερις οντότητες: ο *Πελάτης*, η *Παραγγελία*, το *Προϊόν* και ο *Προμηθευτής* όπως φαίνεται και στο Σχήμα 2.9. Στη συνέχεια θα χρειαζόταν η αναπαράσταση της σχέσης μεταξύ αυτών των οντοτήτων, ορισμένες από το γεγονός ότι οι πελάτες ζητούν συγκεκριμένα προϊόντα τη στιγμή που το κατάστημα θέτει μία παραγγελία σε έναν από τους προμηθευτές οι οποίοι κατασκευάζουν το εν λόγω προϊόν. Στο επικείμενο γράφημα αυτές οι σχέσεις δηλώνονται ως ακολούθως: Η *Διαθεσιμότητα* η οποία ορίζει όλους τους προμηθευτές και τα προϊόντα που κατασκευάζονται, οι *Απαιτήσεις* συνδέουν τους πελάτες με τα προϊόντα τα οποία χρειάζονται και η *Υπόδειξη* η οποία συνδέει ένα συγκεκριμένο προϊόν με έναν συγκεκριμένο προμηθευτή και μία παραγγελία ενός πελάτη.

Σε αυτό το είδος του γραφήματος, κάθε **οντότητα** υποδηλώνει έναν μοναδικό τύπο δεδομένων ο οποίος κατέχει έναν ή περισσότερα χαρακτηριστικά. Στην περίπτωση του καταστήματος, τα χαρακτηριστικά της οντότητας *Πελάτης* θα μπορούσε να αποτελεί το όνομα και τη διεύθυνση του εκάστοτε πελάτη, ενώ τα χαρακτηριστικά που σχετίζονται με την οντότητα *Προϊόν* θα μπορούσε να είναι το όνομα και η τιμή κάθε προϊόντος. Σαν κανόνας, κάθε σχέση απεικονίζει κάποιο γεγονός σχετικό με τις οντότητες του, όπως για παράδειγμα η σχέση της *Υπόδειξης* αναπαριστά την πληροφορία σχετικά με το ποιοι πελάτες ζητούν κάποια προϊόντα και από ποιους προμηθευτές προέρχονται αυτά.

Ενώ το ERD παρέχει μία καλή εικόνα σχετικά με τα δεδομένα του συστήματος, αυτό το μοντέλο είναι ιδιαίτερα κατάλληλο όταν απαιτείται οργάνωση πολύπλοκων σχέσεων μεταξύ διαφόρων ειδών δεδομένων. Πρέπει να σημειωθεί ωστόσο, ότι τα ERD μοντέλα δεν είναι ικανά να περιγράψουν καθόλου τη χρονική ή την λειτουργική συμπεριφορά ενός συστήματος.



Σχήμα 2.10: Γράφημα Jackson

2.8.2 Γραφήματα Jackson

Ένα άλλο είδος μοντέλου σχετικό με τα δεδομένα είναι το **γράφημα Jackson** [Sut88], το οποίο έχει τα δικά του συγκεκριμένα πλεονεκτήματα έναντι των άλλων μοντέλων. Σε αντίθεση με τα γραφήματα οντοτήτων, τα οποία τείνουν να τονίζουν τα χαρακτηριστικά και της αλληλοσυσχέτισης των δεδομένων που μοντελοποιούν, ένα γράφημα Jackson μοντελοποιεί τα δεδομένα υπό την μορφή της **δομής** του, με την ανάλυση των δεδομένων σε υποδεδομένα. Για παράδειγμα, ένα μοντέλο Jackson θα ήταν κατάλληλο για την μοντελοποίηση ενός αρχείου το οποίο θα περιείχε διάφορα υποπεδία. Στην πραγματικότητα, ένα γράφημα Jackson θα ήταν κατάλληλο για δεδομένα τα οποία θα χρειάζονταν ανάλυση με έναν τρόπο ο οποίος είναι πολύ πιο περίπλοκος από έναν άλλο ο οποίος πιθανόν να φαινόταν κατάλληλος για απλότερα αρχεία.

Τα δεδομένα σε ένα γράφημα Jackson αναλύονται υπό μία δενδροειδή δομή, στην οποία οι κόμβοι φύλλων αποτελούν τους **βασικούς** τύπους δεδομένων και οι υπόλοιποι κόμβοι αποτελούν τους **σύνθετους** τύπους δεδομένων οι οποίοι

έχουν προκύψει μέσω διαφόρων λειτουργιών και πράξεων όπως είναι η **σύνθεση** (AND), η **επιλογή** (OR)και η **επανάληψη** (*). Η σύνθεση για παράδειγμα, παράγει έναν τύπο δεδομένων ο οποίος ενσωματώνει δύο ή περισσότερους μικρότερους τύπους. Τέλος, η επανάληψη παράγει δεδομένα με το να αναπαράγει αντίγραφα συγκεκριμένων στοιχείων του κάθε μικρότερου τύπου.

Το Σχήμα 2.10 δείχνει ένα παράδειγμα ενός γραφήματος Jackson το οποίο μοντελοποιεί ένα σύνολο από αντικείμενα σχεδίασης. Σε αυτό το διάγραμμα, το *Σχέδιο* παριστάνει ένα σύνθετο τύπο δεδομένων που αποτελείται από *Χρώμα*, *Σχήμα* και από πολλαπλούς *Χρήστες* που υποδεικνύονται από την πράξη του πολλαπλασιασμού '*'. Στη συνέχεια, κάθε τύπος *Χρήστη* θα αποτελείται από ένα *Όνομα* ενώ κάθε τύπος για το *Σχήμα* θα μπορεί να είναι είτε ένας *Κύκλος* είτε ένα *Τετράγωνο*. Τέλος, ο τύπος του *Τετραγώνου* από μόνος του, αποτελείται από δύο υποδιέστερους τύπους. Αυτοί είναι το *Πλάτος* και το *Ύψος* ενώ ο *Κύκλος* έχει μόνο έναν υποδιέστερο τύπο και αυτός είναι η *Ακτίνα*.

Ένα διάγραμμα Jackson φαίνεται να είναι πιο κατάλληλο για την αναπαράσταση δεδομένων τα οποία έχουν μία περίπλοκη και σύνθετη δομή, σε αντίθεση με τα μοντέλα ERD τα οποία είναι προτιμότερο να χρησιμοποιούνται για την αναπαράσταση δεδομένων τα οποία έχουν περίπλοκη αλληλοσυσχέτιση. Πρέπει να σημειωθεί, ότι οι περιορισμοί αυτού του μοντέλου είναι παρόμοιοι με αυτούς του μοντέλου ERD κατά το ότι είναι και αυτό ανεπαρκές να περιγράψει την λειτουργικότητα ή την χρονική συμπεριφορά του συστήματος.

2.9 Μοντέλα ετερογένειας

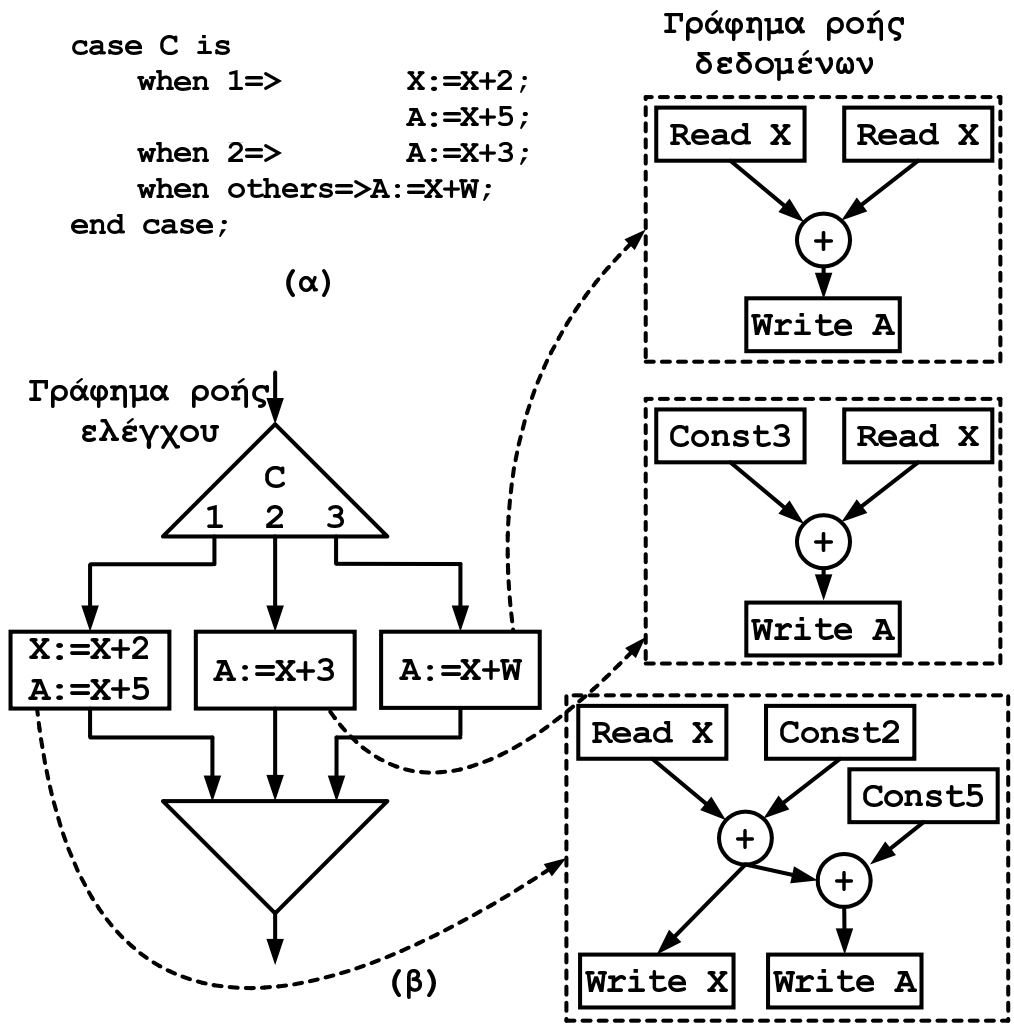
Το **γράφημα ροής ελέγχου/δεδομένων** (CDFG[7]) [OG86], [LG88], αποτελεί ένα μοντέλο ετερογένειας, το οποίο σχεδιάσθηκε ώστε να συνδυάσει

[7]Control/Data Flow Graph

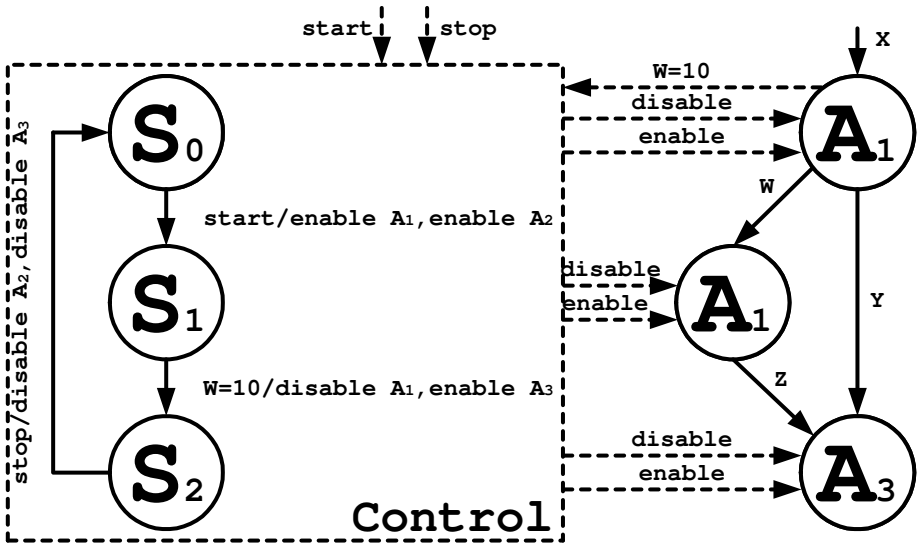
τα πλεονεκτήματα ενός γραφήματος ροής (CFG) με αυτά που παρέχουν τα μοντέλα (DFG). Πιο απλά, το CDFG ενσωματώνει τα DFG έτσι ώστε να αναπαραστήσει τις ροές των δεδομένων μεταξύ των διαφόρων δραστηριοτήτων και τα CFG τα οποία μπορούν να αναπαραστήσουν τη αλληλουχία των διαφόρων DFG. Κατά συνέπεια το μοντέλο CDFG είναι ικανό να παρουσιάσει ταυτόχρονα την εξάρτηση των δεδομένων και την αλληλουχία ελέγχων σε ένα σύστημα με μία μόνο αναπαράσταση.

Στο Σχήμα 2.11(β) βλέπουμε μία CDFG αναπαράσταση του προγράμματος το οποίο περιγράφεται στο Σχήμα 2.11(α). Ας σημειωθεί ότι τα σχήματα ελέγχου στην γλώσσα προγραμματισμού, όπως είναι για παράδειγμα οι δηλώσεις *case* έχουν αντιστοιχισθεί σε κόμβους ροής ελέγχου, ενώ οι ομάδες των δηλώσεων αναθέσεων μεταξύ αυτών των σχημάτων της ροής του ελέγχου, έχουν αναπαρασταθεί από DFG. Φαίνεται επίσης ότι τα CFG και τα DFG διασυνδέονται με διακεκομμένες γραμμές, οι οποίες υποδηλώνουν τις δραστηριότητες των DFG οι οποίες σχετίζονται με τους κόμβους στα CFG. Κάθε φορά που ο έλεγχος μπαίνει σε έναν κόμβο, θα εκτελέσει οποιαδήποτε δραστηριότητα συσχετίζεται με αυτόν τον κόμβο.

Ωστόσο, ένα CDFG μοντέλο δεν περιορίζεται στο να αναπαραστήσει τα σχήματα ελέγχου και τις δηλώσεις ανάθεσης σε μια γλώσσα προγραμματισμού. Αντιθέτως, μπορεί ακόμη να χρησιμοποιηθεί ώστε να αναπαραστήσει οποιοδήποτε πολύπλοκη δραστηριότητα και ενέργεια ελέγχου που ίσως απαιτείται από το σύστημα και αυτό έχει σαν αποτέλεσμα να χρησιμοποιείται ευρέως στο σχεδιασμό συστημάτων πραγματικού χρόνου. Για παράδειγμα, το μοντέλο που παρουσιάζεται στο [WM85] αποτελείται από ένα CDFG στο οποίο έχει προστεθεί ένα CFG το οποίο προδιαγράφεται από ένα μοντέλο μηχανής καταστάσεων. Σε αυτό το σύστημα, το CFG μπορεί να αποκρίνεται σε εξωτερικά και εσωτερικά γεγονότα και μπορεί επίσης να ελέγχει την εκτέλεση του DF-



Σχήμα 2.11: Γράφημα ροής ελέγχου/δεδομένων: (α) Κώδικας προγράμματος, (β) CDFG



Σχήμα 2.12: CDFG σε επίπεδο δραστηριότητας

G με την δημιουργία ενεργειών ελέγχου. όπως είναι το σήμα *ενεργοποίησης* για την εκκίνηση μίας δραστηριότητας και το σήμα *απενεργοποίησης* για τον τερματισμό μίας δραστηριότητας.

Στο Σχήμα 2.12 μπορούμε να δούμε ένα παράδειγμα ενός CDFG σε επίπεδο δραστηριότητας. Σύμφωνα με αυτήν την αναπαράσταση, όταν το CFG είναι στην κατάσταση S_0 και το γεγονός *εκκίνηση* λάβει χώρα, οι δραστηριότητες A_1 και A_2 θα ενεργοποιηθούν και το σύστημα θα μεταβεί στην κατάσταση S_1. Από την άλλη πλευρά, εάν βρίσκεται ήδη στην κατάσταση S_1 και ένα γεγονός $W = 10$ λάβει χώρα, η δραστηριότητα A_1 θα απενεργοποιηθεί, η δραστηριότητα A_3 θα ενεργοποιηθεί και η κατάσταση του συστήματος θα αλλάξει σε S_2. Τέλος, εάν το σύστημα βρίσκεται στην κατάσταση S_2 και προκληθεί το γεγονός *τερματισμός*, οι δραστηριότητες A_2 και A_3 θα απενεργοποιηθούν και το σύστημα θα επιστρέψει στην κατάσταση S_0. Τα W, X, Y και Z αναπαριστά δεδομένα τα οποία ρέουν μεταξύ των διαφόρων δραστηριοτήτων όπως αυτές

προδιαγράφονται στο DFG.

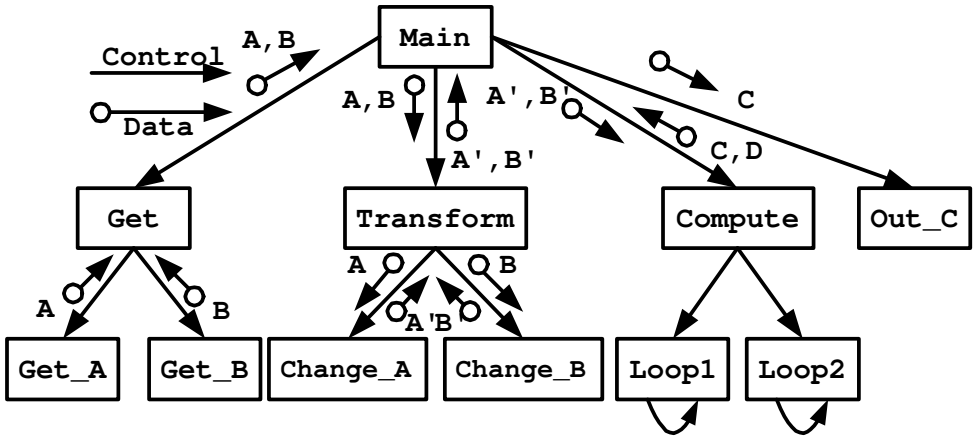
Τα βασικότερα πλεονεκτήματα των CDFG είναι ότι διορθώνει την αδυναμία ενός DFG να αναπαριστά τον έλεγχο του συστήματος, καθώς και την αδυναμία των CFG να αναπαριστά εξαρτήσεις δεδομένων. Συμπερασματικά, φαίνεται ως ένα πλήρες και αρκετά κατάλληλο μοντέλο για πολλούς και διαφόρους τομείς σχεδιασμού, όπως είναι τα συστήματα πραγματικού χρόνου και η σύνθεση συστημάτων ASIC σε επίπεδο συμπεριφοράς (behavioral).

2.9.1 Γράφημα δομής

Το μοντέλο του **γραφήματος δομής** [YC78], αποτελεί ένα ακόμη μοντέλο ετερογένειας, το οποίο σχεδιάσθηκε για να προδιαγράψει τα δεδομένα, τις δραστηριότητες και της προσβάσεις ελέγχου ενός συστήματος σε μία μόνο αναπαράσταση. Λόγω του χαρακτήρα και του μεγάλου εύρους εφαρμογής, το γράφημα δομής είναι χρήσιμο για σχεδιαστές οι οποίοι αναπτύσσουν προγράμματα.

Βασικά, το γράφημα δομής αποτελείται από ένα σύνολο κόμβων οι οποίοι αναπαριστούν **δραστηριότητες** και ένα σύνολο από αιχμές αναπαριστώντας τις διαδικασίες (procedures) ή τις κλήσεις συναρτήσεων (function calls) σε μία γλώσσα προγραμματισμού. Τα **δεδομένα** τα οποία περνάνε μεταξύ των δραστηριοτήτων παρουσιάζονται στις αιχμές. Ο έλεγχος της εκτέλεσης των δραστηριοτήτων περιγράφεται υπό την έννοια ενός συνόλου από δομές ελέγχου, συμπεριλαμβανομένου των **διακλαδώσεων**, των **επαναλήψεων** και των **κλήσεων υπορουτίνων**.

Κατά τη γραφική αναπαράσταση, όλες οι δραστηριότητες στο σύστημα, θα αντιστοιχούνται από τετράγωνα σχήματα και τα δεδομένα τα οποία ανταλλάσσονται μεταξύ αυτών των δραστηριοτήτων θα αντιστοιχούνται από βέλη με ετικέτες. Τα γραφήματα δομής προδιαγράφουν επίσης ότι οι κλήσεις των δια-



Σχήμα 2.13: Παράδειγμα γραφήματος δομής

δικασιών μεταξύ των δραστηριοτήτων θα αναπαρίστανται από καμπύλες, μία διακλάδωση θα περιγράφεται από ρόμβους και οι επαναλήψεις θα δείχνονται με μία καμπύλη βρόγχου.

Στο Σχήμα 2.13 παρουσιάζεται ένα παράδειγμα ενός γραφήματος δομής. Όσον αφορά την αλληλουχία των δραστηριοτήτων μπορούμε να παρατηρήσουμε ότι το στοιχείο *Main* καλεί πρώτα το στοιχείο *Get* για να αποκομίσει τα δεδομένα A και B. Σε αυτό το σημείο το *Get* θα καλέσει το *Get_A* και στη συνέχεια το *Get_B*. Το στοιχείο *Main* θα περάσει στη συνέχεια τα δεδομένα A και B στο στοιχείο *Transform*, το οποίο θα καλέσει το *Change_A* για να μετασχηματίσει το A σε A', ή θα καλέσει το στοιχείο *Change_B* για να μετασχηματίσει το B σε B', ανάλογα με τις τιμές των όποιων συνθηκών. Αφού εξασφαλισθεί το A' ή το B' από το *Transform*, το στοιχείο *Main* θα περάσει αυτά τα δεδομένα στο *Compute* το οποίο θα καλέσει δύο στοιχεία επανάληψης, τα *Loop1* και *Loop2*. Όταν αυτά τα στοιχεία ολοκληρώσουν τις λειτουργίες τους, το *Compute* θα επιστρέψει τα δεδομένα C και D στο στοιχείο *Main*, το οποίο με τη σειρά του θα περάσει τα δεδομένα C στο στοιχείο *Out_C*.

Όπως και στην περίπτωση με το CDFG, το γράφημα δομής μπορεί να αναπαραστήσει ταυτόχρονα πληροφορία ελέγχου και δεδομένων. Πρέπει να σημειωθεί ότι το γράφημα δομής δεν προδιαγράφει πλήρως την αλληλουχία των εκτελέσεων. Για παράδειγμα, δεν φαίνεται να γνωρίζουμε τη σειρά με την οποία τα Get_A και Get_B θα κληθούν, καθώς και η σειρά με την οποία θα εκτελεστούν περιορίζεται μόνο από τις εξαρτήσεις των δεδομένων. Από την άλλη πλευρά, το μοντέλο του γραφήματος δομής παρέχει τα απαραίτητα για την αναπαράσταση εκμεταλλεύσιμων ελέγχων για διακλαδώσεις, επαναλήψεις και κλήσεις διαδικασιών μεταξύ των στοιχείων. Λόγω του ότι μπορεί να καθορίσει τη σειρά της εκτέλεσης σε κάποιο βαθμό, τα γραφήματα δομής χρησιμοποιούνται κυρίως στα αρχικά στάδια του σχεδιασμού σειριακών προγραμμάτων.

Κεφάλαιο 3

Καταμερισμός Υλικού και Λογισμικού

Ο σχεδιασμός του υλικού σε ένα ενσωματωμένο σύστημα είναι κάτι παραπάνω από την απλή επιλογή του σωστού επεξεργαστή έτσι ώστε να διασυνδεθούν απλώς μερικά ακόμη περιφερειακά σε αυτόν. Η απόφαση για τον σωστό καταμερισμό του σχεδιασμού σε λειτουργίες οι οποίες αναπαρίστανται από το υλικό και άλλες από το λογισμικό αποτελεί το κλειδί για την σωστή δημιουργία ενός ενσωματωμένου συστήματος. Αυτή η απόφαση δεν αποτελεί μόνο μία εργασία σε ακαδημαϊκό επίπεδο καθώς επίσης δεν είναι και τόσο προφανής όσο ίσως φαίνεται. Η ορθή επιλογή του καταμερισμού έχει σημαντικό αντίκτυπο στο κόστος, στον χρόνο ανάπτυξης και στην μείωση του ρίσκου αποτυχίας ενός έργου.

3.1 Δυικότητα υλικού-λογισμικού

Ο καταμερισμός είναι δυνατός να υπάρξει και πολλές φορές απαραίτητος εξαιτίας της δυικότητας μεταξύ του υλικού και του λογισμικού. Για παράδειγμα,

πριν από την εμφάνιση του πολύ δημοφιλούς επεξεργαστή 486 της Intel ο πιο διαδεδομένος επεξεργαστής ήταν ο 386.

Ο επεξεργαστής 386 είναι ένας επεξεργαστής ο οποίος διαχειρίζεται μόνο ακέραιους αριθμούς. Για να επιτευχθούν καλύτερες αποδόσεις στις μαθηματικές εφαρμογές χρειαζόταν η ύπαρξη ενός μαθηματικού συνεπεξεργαστή ο οποίος είναι γνωστός με την ονομασία FPU 387. Τα συστήματα τα οποία δεν ήταν εφοδιασμένα με έναν τέτοιο συνεπεξεργαστή, κάθε φορά που εντόπιζαν μαθηματικές πράξεις κινητής υποδιαστολής, προσομοίωναν την ύπαρξη μίας τέτοιας μονάδας FPU σε λογισμικό. Αυτό είχε σαν αποτέλεσμα το να πραγματοποιούν διακλαδώσεις σε υπορουτίνες οι οποίες εκτελούσαν σε λογισμικό τέτοιου είδους συναρτήσεις και λειτουργίες, έστω και αν αυτές πραγματοποιούνταν πολύ πιο αργά. Ο μαθηματικός συνεπεξεργαστής 387 πραγματοποιούσε τις συναρτήσεις FPU απευθείας σε υλικό, από το να εκτελεί αργές διεργασίες για την εκτέλεση αυτών σε λογισμικό. Αυτή η διαδικασία συνήθως είχε ένα κόστος στην απόδοση το οποίο έφτανε τόσο, ώστε η εκτέλεση να γίνεται μέχρι και δέκα φορές γρηγορότερα όταν λάμβανε χώρα σε υλικό.

Το παραπάνω αποτελεί ένα παράδειγμα του προβλήματος του καταμερισμού υλικού και λογισμικού. Ο συνεπεξεργαστής 387 ήταν αρκετά πιο ακριβός από τον ίδιο τον επεξεργαστή 386 και αυτό είχε σαν αποτέλεσμα στο να μην περιλαμβάνεται σε έναν σχεδιασμό όπου δεν ήταν απαραίτητος σε ορισμένες περιπτώσεις. Ωστόσο, η απουσία του 387 δεν εμποδίζει το χρήστη από το να χρησιμοποιεί υπολογισμούς κινητής υποδιαστολής· απλά σημαίνει ότι αυτοί οι υπολογισμοί δεν θα περατωθούν τόσο γρήγορα όσο σε ένα σύστημα με μία μονάδα FPU.

Συνεπώς επειδή ο σχεδιασμός ενός ενσωματωμένου συστήματος θα εμπλέκει σχεδόν πάντοτε δομικές μονάδες τόσο σε υλικό όσο και σε λογισμικό, κάποιος πρέπει να αποφασίσει ποιο μέρος του εκάστοτε προβλήματος θα επιλυθεί

σε υλικό και ποιο σε λογισμικό. Αυτή η επιλογή ονομάζεται και ως *απόφαση καταμερισμού*. Αν η παραπάνω περίπτωση γενικευθεί στο σχεδιασμό συστημάτων και ήταν επιθυμητό να μοντελοποιηθεί η διαδικασία του καταμερισμού υλικού λογισμικού, αρχικά ο σχεδιαστής θα πρέπει να σκέφτεται έναν αλγόριθμο σαν ένα συνδυασμό από μονάδες υλικού και λογισμικού έστω και αν κάποια από αυτά είναι δυνατόν να υλοποιηθούν πλήρως σε υλικό ή σε λογισμικό.

Ωστόσο η πολυπλοκότητα του στο χώρο του προβλήματος καταμερισμού είναι πολύ μεγάλη. Για να ήταν δυνατόν να περιγραφεί το μέγεθος του προβλήματος, θα έπρεπε να λάβουμε υπόψη μας πολλαπλές αρχιτεκτονικές, διαφορετικές τεχνολογίες, σχεδιαστικά εργαλεία κτλ. Σήμερα, πολλά συστήματα είναι τόσο πολύπλοκα ώστε τα εργαλεία που εκτελούνται σε υπολογιστές για τον καταμερισμό του υλικού και του λογισμικού είναι πλέον απαραίτητα. Παρόλα αυτά ο Charles H. Small περιγράφει την απόφαση καταμερισμού σε υλικό και λογισμικό ως εξής: "*Στην πράξη, η ανάλυση και η ανταλλαγές για τον καταμερισμό ενός συστήματος αποτελεί τις περισσότερες φορές μία ανεπίσημη διαδικασία η οποία γίνεται με χαρτί και μολύβι.*" [Char98]

Ιδανικά, η απόφαση καταμερισμού δεν θα μπορούσε να γίνει αν δεν είχε γίνει κατανοητό ότι υπάρχουν και άλλοι εναλλακτικοί τρόποι για την επίλυση ενός προβλήματος. Όσο περισσότερο χρόνο καθυστερεί η λήψη για την απόφαση του καταμερισμού, τόσο καλύτερα θα γίνεται γνωστό ποια τμήματα ενός αλγορίθμου χρειάζεται να εκτελεσθούν σε υλικό και ποια από αυτά μπορούν να εκτελούνται σε λογισμικό με ικανοποιητική απόδοση. Η προσθήκη υλικού οδηγεί σε ταχύτερα συστήματα αλλά με αυξημένο κόστος. Αυτό βέβαια δεν σημαίνει πως κάτι τέτοιο είναι απόλυτο γιατί περισσότερη λειτουργικότητα σε λογισμικό οδηγεί σε περισσότερο λογισμικό, επομένως μετά θα πρέπει να επανεξετασθεί και η επιλογή του εν λόγω επεξεργαστή που πρόκειται να εκτελεί αυτό το λογισμικό. Επίσης, η προσθήκη υλικού σημαίνει ακόμη ότι η

διαδικασία σχεδιασμού αποκτά μεγαλύτερες πιθανότητες αποτυχίας διότι ο επανασχεδιασμός του συστήματος σε υλικό είναι κάτι το περίπλοκο σε σύγκριση με τη διόρθωση μίας πιθανής αδυναμίας σε ένα λογισμικό.

Το σημαντικότερο πρόβλημα ωστόσο στον σχεδιασμό συστημάτων με υλικό και λογισμικό, είναι ότι δεν καθίσταται δυνατός ο έλεγχος του λογισμικού μέχρις ότου το υλικό είναι διαθέσιμο για να λειτουργήσει. Επιπλέον, μεγάλες καθυστερήσεις της απόφασης καταμερισμού, αφήνει την πιθανή ομάδα ανάπτυξης λογισμικού ανενεργή περιμένοντας την ομάδα ανάπτυξης του υλικού να ολοκληρώσει την διαδικασία της ανάπτυξης.

3.2 Δομική και λειτουργική κατάτμηση

Η λειτουργία ενός συστήματος υλοποιείται ως ένα σύνολο από διασυνδεόμενα στοιχεία. Παραδείγματα τέτοιων στοιχείων συστήματος αποτελούν τυπικοί επεξεργαστές ή μικροελεγκτές, ASIC ολοκληρωμένα κυκλώματα, μνήμες και δίαυλοι. Για την επίτευξη της υλοποίησης ενός τέτοιου συστήματος ο σχεδιαστής πρέπει να επιλύσει δύο βασικά προβλήματα: Να διαλέξει ένα σύνολο των παραπάνω στοιχείων του συστήματος και να κατατμήσει την λειτουργικότητα του συστήματος μεταξύ αυτών των στοιχείων. Η επιλογή του συνόλου των στοιχείων ονομάζεται **κατανομή** και η διαδικασία του διαχωρισμού των λειτουργιών ονομάζεται **κατάτμηση**. Η κατανομή και η κατάτμηση πρέπει να επιλέγεται με τέτοιον τρόπο ώστε να οδηγεί σε μία υλοποίηση η οποία ικανοποιεί τις δεσμεύσεις και τους περιορισμούς του συστήματος, όπως για παράδειγμα το κόστος, την απόδοση, την επιφάνεια που θα καταλαμβάνει καθώς και την κατανάλωση ισχύος.

Υπάρχουν δύο πολύ διαφορετικές προσεγγίσεις στον καταμερισμό ενός συστήματος, η δομική και η λειτουργική κατάτμηση. Στην προσέγγιση της δο-

μικής κατάτμησης, το σύστημα υλοποιείται πρώτα δομικά και στη συνέχεια κατατμήζεται. Στην προσέγγιση της λειτουργικής κατάτμησης το σύστημα πρώτα κατατμήζεται και μετά υλοποιείται. Στη συνέχεια παρουσιάζονται τα χαρακτηριστικά και τα πλεονεκτήματα της κάθε μίας προσέγγισης κατάτμησης με περισσότερη λεπτομέρεια. Υποθέτουμε ότι τα στοιχεία του συστήματος κατανέμονται χειροκίνητα από τον σχεδιαστή πριν εφαρμόσουμε την κατάτμηση.

3.2.1 Δομική κατάτμηση

Σε αυτή την προσέγγιση πρώτα υλοποιούμε το σύστημα δομικά. Η δομή είναι μία αλληλοσύνδεση αντικειμένων υλικού, όπου κάθε αντικείμενο μπορεί να είναι μία πύλη τύπου boolean ή ένα flip-flop, μία μονάδα καταχωρητών όπως ένας μετρητής ή ένας απλός καταχωρητής, ή ακόμη και μία πολύπλοκη υπολογιστική μονάδα όπως ένας πολλαπλασιαστής αριθμών κινητής υποδιαστολής ή ένας μετασχηματιστής Fourier. Το επόμενο βήμα είναι η κατάτμηση αυτής της δομής. Η κατάτμηση διαχωρίζει τα αντικείμενα σε ομάδες όπου κάθε ομάδα αναπαριστά ένα στοιχείο του συστήματος.

Η προσέγγιση της δομικής κατάτμησης είναι πολύ διαδεδομένη, και αυτό οφείλεται κατά ένα μέρος στην ευθεία μέθοδο για την εξασφάλιση της εκτίμησης του μεγέθους και των ακίδων του συστήματος. Το μέγεθος μίας ομάδας εκτιμάται ως το άθροισμα των μεγεθών του κάθε αντικειμένου της ομάδας και ο αριθμός των ακίδων υπολογίζεται ως ο αριθμός των καλωδίων τα οποία διασταυρώνουν την μία ομάδα με την άλλη. Ένα άλλο σημείο το οποίο συμβάλλει στην διάδοση αυτής της μεθόδου είναι η ευκολία με την οποία επιτρέπεται το πρόβλημα της δομικής κατάτμησης να αντιστοιχισθεί σε ένα πρόβλημα γραφήματος κατάτμησης, για το οποίο μία μεγάλη σειρά από επίσημες θεωρίες, σοφιστικούς αλγορίθμους και εργαλεία βοηθούν σε αυτήν την κατάτμηση. Για να επιδείξουμε αυτήν την κατάτμηση, ορίζουμε ένα γράφημα ως ένα σύνολο

από κορυφές και αιχμές όπου κάθε αιχμή συνδέει ένα υποσύνολο των κορυφών. Κάθε κορυφή και κάθε αιχμή συσχετίζεται με μία τιμή η οποία ονομάζεται βάρος. Ένα σύμπλεγμα p αποτελεί ένα υποσύνολο από κορυφές το οποίο έχει ένα συνολικό βάρος ίσο με το άθροισμα των βαρών των αιχμών οι οποίες συνδέουν κάθε μία από τις κορυφές του p με τουλάχιστον μία άλλη κορυφή η οποία βρίσκεται έξω από το p. Στη συνέχεια αντιστοιχούμε κάθε αντικείμενο υλικού που ανήκει σε μία δομή, σε μία κορυφή με ένα βάρος ίσο με το μέγεθος του αντικειμένου και επίσης αντιστοιχούμε κάθε διασύνδεση που ανήκει σε μία δομή, σε μία αιχμή με ένα βάρος ισότιμο με τον αριθμό των διασυνδεόμενων καλωδίων. Στη συνέχεια είναι δυνατό να εκτιμηθεί το μέγεθος μίας ομάδας δομής σαν το βάρος του αντίστοιχου γραφήματος συμπλέγματος και το εξωτερικό άθροισμα των ακίδων μίας δομικής ομάδας ως το σημείο αποκοπής του αντίστοιχου γραφήματος συμπλέγματος.

Ένας άλλος λόγος για τον οποίο η δομική κατάτμηση είναι διαδεδομένη είναι ότι στατιστικά φαίνεται να παράγει καλά αποτελέσματα για πολλά και διαφορετικά συστήματα. Αυτά τα αποτελέσματα ήταν ανεξάρτητα με την έννοια ότι στο παρελθόν, οι χωρητικότητες των στοιχείων ενός συστήματος δεν ήταν τρομερά μεγαλύτερες συγκρινόμενες με τα μεγέθη των αντικειμένων υλικού τα οποία επρόκειτο να κατατμηθούν καθώς επίσης επειδή ο αριθμός των αντικειμένων υλικού δεν ήταν και αυτός αρκετά μεγάλος.

Ας σημειωθεί ωστόσο, ότι η δυνατότητα να ληφθούν τέτοια αποτελέσματα, που ίσως να επηρεαζόταν στη χωρητικότητα των στοιχείων του συστήματος, συνέχισε να αυξάνεται καθώς αυξανόταν επίσης και ο αριθμός των αντικειμένων υλικού που υλοποιούν το σύστημα. Παρόλα αυτά, υπάρχουν τρία βασικά μειονεκτήματα στην προσέγγιση της δομικής κατάστασης:

1. **Η ανταλλαγή μεταξύ μεγέθους και απόδοσης είναι αρκετά δύσκολη** - Κατά την υλοποίηση της δομής, είναι αρκετά δύσκολο να λη-

φθούν αποφάσεις οι οποίες έχουν να κάνουν με ανταλλαγές στο μέγεθος και στην απόδοση, εφόσον διαδοχικά βήματα κατάτμησης μπορεί να οδηγήσουν στην ακύρωση αυτών των αποφάσεων. Το αποτέλεσμα θα μπορούσε να οδηγήσει σε εκτεταμένες καθυστερήσεις στην επικοινωνία μεταξύ ολοκληρωμένων κυκλωμάτων ή μονάδων υλικού. Για παράδειγμα ας θεωρήσουμε το παρακάτω κοινό πρόβλημα: ένας σχεδιαστής επιθυμεί να δημιουργήσει το μικρότερο σχεδιασμό ώστε να επιτυγχάνει ταυτόχρονα και τους σχετικούς περιορισμούς απόδοσης. Θα μπορούσε να επιλέξει να χρησιμοποιήσει από κοινού έναν προσθετή μεταξύ δύο συμπεριφορών, γνωρίζοντας ότι ένας τέτοιος καταμερισμός θα μείωνε το μέγεθος χωρίς να επηρεάζει καθόλου την απόδοση. Ωστόσο, η κατάτμηση αυτή μπορεί να μην τοποθετήσει όλα τα αντικείμενα που υλοποιούν αυτές τις δύο συμπεριφορές στο ίδιο ολοκληρωμένο σύστημα. Για να λάβει δεδομένα στις εισόδους του προσθετή, πρέπει να πραγματοποιηθούν μεταφορές μεταξύ δύο διαφορετικών ολοκληρωμένων κυκλωμάτων, το οποίο έχει σαν άμεσο αποτέλεσμα την μείωση της απόδοσης του συστήματος. Συνεπώς ο κοινόχρηστος προσθετής φαίνεται ότι ήταν μία κακή επιλογή. Από την άλλη πλευρά, υποθέτοντας ότι ο σχεδιαστής χρησιμοποιούσε έναν προσθετή για κάθε συμπεριφορά, αλλά η κατάτμηση τοποθετούσε όλα τα αντικείμενα και για τις δύο συμπεριφορές στο ίδιο ολοκληρωμένο σύστημα. Ένας προσθετής θα ήταν τότε αρκετός. Τώρα, θεωρώντας ότι οι ενέργειες σχεδιασμού δομών απαιτούσε εκατοντάδες, ίσως και χιλιάδες, αποφάσεις παρόμοιες με τις παραπάνω αποφάσεις κόστους και απόδοσης που έγιναν για έναν μόνο προσθετή. Τα παραδείγματα που περιγράφηκαν παραπάνω δείχνουν ότι η απόπειρα τέτοιων αποφάσεων πριν την λήψη κατάτμησης μπορεί να οδηγήσουν σε όχι και τόσο καλά αποτελέσματα.

Μία πιθανή λύση θα ήταν η τροποποίηση της δομής μετά την κατάτμη-

ση. Δυστυχώς, μεγάλες τροποποιήσεις στη δομή είναι δύσκολο να γίνουν επειδή μία συμπεριφορά έχει αντιστοιχισθεί σε διάφορα αντικείμενα και συμπεριφορές οι οποίες διαμοιράζονται ένα μόνο αντικείμενο. Συνεπώς, αλλάζοντας ένα αντικείμενο μπορεί να επηρεάσει πολλά τμήματα διαφόρων συμπεριφορών, οι διακλαδώσεις των οποίων είναι δύσκολο να εντοπισθούν. Οι αλλαγές που γίνονται στη δομή ενός συστήματος είναι συνήθως μικρές, όπως η εισαγωγή εφεδρικών πυλών ή μετακινήσεις πυλών από ένα σημείο σε ένα άλλο. Ένα ανάλογο παράδειγμα σε λογισμικό είναι το παρακάτω: από τη στιγμή που έχει παραχθεί ο assembly κώδικας, μόνο μικρές βελτιώσεις σχετικά με την απόδοση μπορούν να γίνουν, όπως για παράδειγμα η εξάλειψη κάποιας πιθανής πρόσβασης στη μνήμη χρησιμοποιώντας καταχωρητές. Μεγάλες αλλαγές όπως για παράδειγμα μετατροπές στον κώδικα ώστε το πρόγραμμα να μπορεί να εκτελείται από συστήματα πολλαπλών επεξεργαστών είναι δύσκολες. Στο σχεδιασμό του υλικού όπως και στον σχεδιασμό του λογισμικού, οι ουσιαστικές αλλαγές απαιτούν γνώσεις σε υψηλό επίπεδο της λειτουργικότητας και του χρονισμού του συστήματος, οι οποίες δεν είναι εύκολο να διακριθούν από τη δομή και μόνο π.χ. του assembly κώδικα.

2. **Μεγάλος αριθμός αντικειμένων** - Τα συστήματα μεγαλώνουν διαρκώς σε μέγεθος και συνεπώς υλοποιούνται με όλο και μεγαλύτερο αριθμό αντικειμένων. Μεγαλύτερος αριθμός αντικειμένων οδηγεί σε όχι και τόσο καλά αποτελέσματα όσον αφορά τους αλγόριθμους κατάτμησης. Τέτοιοι μεγάλοι αριθμοί αντικειμένων επίσης κάνουν πιο δύσκολο το έργο του σχεδιαστή στο να αλληλεπιδράσει κατά τη διάρκεια της διαδικασίας κατάτμησης και να αντιληφθεί το αποτέλεσμα της κατάτμησης.

3. **Λύσεις χρησιμοποιώντας μόνο υλικό** - Ένα τρίτο μειονέκτημα της

προσέγγισης δομικής κατάτμησης είναι ο περιορισμός στο σχεδιασμό του υλικού. Όταν κάποιοι επεξεργαστές ή μικροελεγκτές κατανέμονται ως στοιχεία του συστήματος, κάποιο μέρος της λειτουργικότητας υλοποιείται σε λογισμικό. Λόγω του ότι στην δομική κατάτμηση όλη η λειτουργικότητα μετατρέπεται πρώτα σε δομές, τέτοιες λύσεις με λογισμικό λαμβάνουν πολύ μικρή σημασία. Καθώς τα συστήματα γίνονται ολοένα και περισσότερο πολύπλοκα και τα στοιχεία του συστήματος γίνονται μεγαλύτερα, τα μειονεκτήματα του να βασίζεται κανείς αποκλειστικά σε δομικές κατατμήσεις γίνονται περισσότερο εμφανή.

3.2.2 Λειτουργική κατάτμηση

Στην προσέγγιση της λειτουργικής κατάτμησης, πρώτα αναλύεται η λειτουργικότητα του συστήματος σε μη διαιρούμενα τμήματα τα οποία και ονομάζονται **αντικείμενα λειτουργίας**. Στη συνέχεια διαχωρίζονται αυτά τα αντικείμενα μεταξύ στοιχείων του συστήματος, τα οποία στη συνέχεια υλοποιούν κάθε δομικού στοιχείου την λειτουργία είτε σε επίπεδο υλικού είτε σε επίπεδο λογισμικού. Υπάρχουν διάφορα πλεονεκτήματα της λειτουργικής κατάτμησης σε σύγκριση με αυτήν της δομικής κατάτμησης.

1. **Ανταλλαγή μεταξύ μεγέθους και απόδοσης** - Ένα σημαντικό πλεονέκτημα της λειτουργικής κατάτμησης είναι ότι, κατά τη διάρκεια των διαδοχικών βημάτων που ακολουθούνται στην δομική κατάτμηση, οι ανταλλαγές μεταξύ του μεγέθους και της απόδοσης μπορούν να γίνουν με πλήρη γνώση της κατάτμησης. Αυτού του είδους η γνώση επιτρέπει στα στοιχεία του συστήματος να είναι σε μεγάλο βαθμό κοινόχρηστα μέσα σε ένα στοιχείο του συστήματος πλέον. Επίσης, οι εκτιμήσεις στην απόδοση που γίνονται κατά τη διάρκεια της δομικής υλοποίησης είναι ακριβείας, εφόσον όλες οι μεταφορές δεδομένων μεταξύ των διαφόρων στοιχείων του

συστήματος είναι γνωστές.

2. **Μικρός αριθμός αντικειμένων** - Ένα δεύτερο πλεονέκτημα είναι η μείωση του αριθμού των αντικειμένων που θα υποστούν την κατάτμηση, εφόσον υπάρχουν λίγα αντικείμενα λειτουργιών από ότι τα δομικά αντικείμενα σε επίπεδο καταχωρητών. Με λιγότερα αντικείμενα, η απόδοση του αλγορίθμου είναι καλύτερη καθώς επίσης είναι ευκολότερη η επέμβαση του σχεδιαστή.

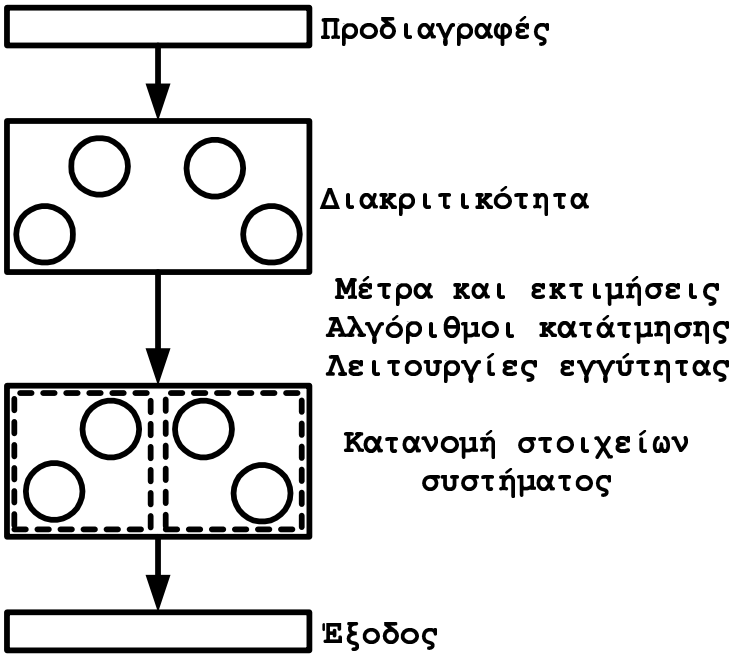
3. **Λύσεις υλικού/λογισμικού** - Το πιο σημαντικό πλεονέκτημα της λειτουργικής κατάτμησης είναι ότι επιτρέπει την κατάτμηση υλικού και λογισμικού, επειδή τα υπό κατάτμηση αντικείμενα είναι λειτουργικά. Τα λειτουργικά αντικείμενα τα οποία αντιστοιχίζονται σε έναν επεξεργαστή μπορούν να μεταγλωττιστούν σε ένα σύνολο εντολών, ενώ άλλα αντικείμενα που αντιστοιχίζονται σε υλικό μπορούν να υλοποιηθούν με δομές. Μιας και πολλά συστήματα περιλαμβάνουν τμήματα τόσο υλικού όσο και λογισμικού, η ικανότητα της κατάτμησης μεταξύ υλικού και λογισμικού είναι μία κρίσιμη απαίτηση για κάθε προσέγγιση κατάτμησης συστήματος.

Η κλασσική προσέγγιση στην λειτουργική κατάτμηση, η οποία ξεκινά από μία ανεπίσημη περιγραφή σε φυσική γλώσσα δεν φαίνεται να είναι αρκετή για διαφόρους λόγους. Πρώτα από όλα, εφόσον οι περιγραφές σε φυσική γλώσσα δεν είναι αναγνώσιμες από τις μηχανές, δεν είναι δυνατή η αυτοματοποίηση της εκτίμησης και της κατάτμησης. Η ποιότητα μίας κατάτμησης είναι συνεπώς κατά πολύ επηρεαζόμενη από την εμπειρία του σχεδιαστή. Κατά δεύτερον, δεν είναι δυνατή η εφαρμογή της επαλήθευσης σε αρχικό στάδιο όταν χρησιμοποιούμε την φυσική γλώσσα, επομένως λειτουργικά λάθη δεν είναι δυνατόν να εντοπισθούν παρά μονό στα τελευταία στάδια του κύκλου σχεδίασης. Τρίτον, από τη στιγμή που οι περιγραφές σε φυσική γλώσσα είναι συνήθως ασαφής και

διφορούμενες, είναι δύσκολο να ολοκληρωθούν και να υλοποιηθούν τα στοιχεία του συστήματος. Το πρόβλημα της ολοκλήρωσης ξεκινά όταν διαφορετικοί σχεδιαστές υλοποιούν διαφορετικά στοιχεία και μονάδες του συστήματος. Κάθε σχεδιαστής πιθανόν να ερμηνεύσει την εκάστοτε περιγραφή με διαφορετικό τρόπο. Τέτοιες διαφορετικές ερμηνείες εμφανίζονται σε αυτούς ως λάθη όταν υλοποιούν και δοκιμάζουν τα στοιχεία αυτά.

Η **κατάτμηση εκτελέσιμων προδιαγραφών** αναπτύχθηκε για να αντιμετωπίσει τους παραπάνω περιορισμούς. Σε αυτήν την συγκεκριμένη προσέγγιση στην λειτουργική κατάτμηση, πρώτα γίνεται η σύλληψη της λειτουργικότητας του συστήματος σε μία γλώσσα εκτελέσιμων προδιαγραφών. Από αυτές τις προδιαγραφές προκύπτουν τα λειτουργικά αντικείμενα και στη συνέχεια πραγματοποιείται η κατάτμησή τους. Αυτή η προσέγγιση περιλαμβάνει πολλά πλεονεκτήματα τα οποία είναι κοινά στις περισσότερες προσεγγίσεις σχεδιασμού εκτελέσιμων γλωσσών. Πρώτα, επειδή οι προδιαγραφές είναι αναγνωρίσιμες από μηχανές, είναι δυνατή η ανάπτυξη εργαλείων τα οποία μπορούν να αυτοματοποιήσουν την εκτίμηση και την κατάτμηση του συστήματος, έτσι ώστε να μειωθεί η εξάρτηση από τους πολύ πεπειραμένους σχεδιαστές. Δεύτερον, η επαλήθευση των προδιαγραφών μέσω προσομοίωσης ελαχιστοποιεί τα λάθη από τα αρχικά στάδια του σχεδιασμού, προλαβαίνοντας αλλαγές στο κόστος όταν θα είναι πλέον αργά για το σχεδιασμό. Τρίτον, επειδή τα λειτουργικά αντικείμενα που αντιστοιχίζονται σε κάθε στοιχείο του συστήματος είναι καθορισμένα με μεθοδικό τρόπο, οι προδιαγραφές για κάθε στοιχείο του συστήματος είναι ακριβής. Συνεπώς, πολύ λιγότερα προβλήματα εμφανίζονται κατά τη διάρκεια της ολοκλήρωσης των υπό υλοποίηση στοιχείων, εφόσον δεν υπάρχει περιθώριο για διαφορετικές ερμηνείες της λειτουργικότητας του συστήματος.

Πολλές τεχνικές έχουν προταθεί για την κατάτμηση των εκτελέσιμων προδιαγραφών. Επειδή όμως αυτή η περιοχή είναι σχετικά καινούρια, οι περισ-



Σχήμα 3.1: Βασικά θέματα κατάτμησης ενός συστήματος

σότερες τεχνικές έχουν επικεντρωθεί σε ένα μικρό υποσύνολο του γενικού προβλήματος της λειτουργικής κατάτμησης.

3.3 Θέματα κατάτμησης

Οι διάφορες τεχνικές κατάτμησης, μπορούν να γίνουν καλύτερα κατανοητές και συγκρίσιμες αν διαχωρίσουμε το συνολικό πρόβλημα της κατάτμησης σε οχτώ βασικά θέματα: στο επίπεδο αφαίρεσης των προδιαγραφών, στη διακριτικότητα, στην κατανομή των στοιχείων του συστήματος, σε μέτρα και εκτιμήσεις, σε συναρτήσεις εγγύτητας, στους αλγορίθμους κατάτμησης, στις εξόδους, στον έλεγχο της ροής και στην αλληλεπίδραση με τον σχεδιαστή.

Το Σχήμα 3.1 παρουσιάζει τον διαχωρισμό του προβλήματος σε αυτά τα θέματα. Ας σημειωθεί ότι αυτό το σύνολο των θεμάτων αυτών δεν σκοπεύει

να προβληθεί σαν μία συμβατική επιστημονική ταξινόμηση.

3.3.1 Αφαίρεση σε επίπεδο προδιαγραφών

Οι διάφορες τεχνικές κατάτμησης συνήθως καθορίζουν την είσοδό τους από την γλώσσα που χρησιμοποιείται για τις προδιαγραφές του συστήματος. Ωστόσο, η γλώσσα από μόνη της αποτελεί ανεπαρκή είσοδο για τον καθορισμό του συστήματος, εφόσον η ίδια γλώσσα μπορεί να χρησιμοποιηθεί για την αναπαράσταση πολλών διαφορετικών θεμελιωδών μοντέλων, το καθένα από τα οποία απαιτεί διαφορετικές προσεγγίσεις κατάτμησης. Για παράδειγμα η γλώσσα VHDL μπορεί να χρησιμοποιηθεί για να περιγράψει έναν μετασχηματισμό Fourier ως έναν αλγόριθμο, ως ένα σύνολο από καταχωρητές ή ακόμη σαν μία λίστα διασυνδέσεων σε επίπεδο πυλών.

Είναι επομένως πιο κατάλληλο να καθορισθεί η είσοδος στο επίπεδο της αφαίρεσης του θεμελιώδους μοντέλου. Με αυτόν τον τρόπο ορίζεται το επίπεδο αφαίρεσης ως ένα μέτρο του αριθμού των μικρής πολυπλοκότητας αντικειμένων δομής στις προδιαγραφές. Η χαμηλού επιπέδου αφαίρεση υποδεικνύει ένα μεγάλο αριθμό από χαμηλής πολυπλοκότητας αντικείμενα. Για παράδειγμα, μία προδιαγραφή που αποτελείται από μία διασύνδεση πυλών έχει μία πολύ χαμηλού επιπέδου αφαίρεση.

Υπάρχουν διάφορα επίπεδα αφαίρεσης προδιαγραφών τα οποία λαμβάνονται υπόψη από διάφορες τεχνικές κατάτμησης. Παρακάτω παρουσιάζονται τα διάφορα επίπεδα αφαίρεσης ξεκινώντας από το υψηλότερο:

1. **Γράφημα ροής δεδομένων σε επίπεδο διεργασίας** - Η είσοδος αναπαριστά ένα γράφημα ροής δεδομένων, όπου κάθε λειτουργία αναπαριστά μία διεργασία. Μία διεργασία είναι κάθε ένας υπολογισμός ο οποίος είναι πιο περίπλοκος από κάθε βασική αριθμητική λειτουργία. Ένα τέτοιο μοντέλο δεν ορίζει τους πραγματικούς υπολογισμούς από κάθε διεργασία,

αλλά αντιθέτως καθορίζει τα δεδομένα τα οποία μεταφέρονται μεταξύ των διεργασιών και διάφορα χαρακτηριστικά από κάθε διεργασία όπως είναι η καθυστέρηση ή το μέγεθος.

2. **Διεργασίες** - Η είσοδος αντικατοπτρίζει ένα σύνολο από διεργασίες όπου κάθε διεργασία περιγράφεται από ένα σειριακό πρόγραμμα. Αυτό το επίπεδο της αφαίρεσης ονομάζεται επίσης και ως επιπέδου συμπεριφοράς.

3. **Γράφημα ροής δεδομένων σε αριθμητικό επίπεδο** - Η είσοδος είναι ένα γράφημα ροής δεδομένων από αριθμητικές λειτουργίες όπως είναι η πρόσθεση ή η αφαίρεση, πιθανόν μαζί με κάποιες λειτουργίες ελέγχου. Αυτό είναι το πιο κοινό μοντέλο που χρησιμοποιείται στις τεχνικές κατάτμησης, επειδή χρησιμοποιείται από τα περισσότερα εργαλεία σύνθεσης και έρευνας υψηλού επιπέδου. Ονομάζεται επίσης και ως γράφημα ελέγχου/ροής δεδομένων και είναι γνωστό και ως CDFG.

4. **FSM με διαδρομές δεδομένων** - Η είσοδος αναπαριστά μία μηχανή πεπερασμένων καταστάσεων με πολύπλοκες δηλώσεις να υπολογίζονται σε μία κατάσταση ή κατά τη διάρκεια μίας μετάβασης, όπως για παράδειγμα η έκφραση $a = b + c \times d$. Φυσικά, μπορούν να χρησιμοποιηθούν και οι κλασσικές FSMs στις οποίες επιτρέπεται η χρήση μόνο εκφράσεων τύπου boolean.

5. **Μεταφορές καταχωρητών** - Η είσοδος αναπαριστά ένα σύνολο από μεταφορές καταχωρητών, όπου για κάθε μηχανή καταστάσεων, παρέχονται οι μεταφορές μεταξύ των καταχωρητών.

6. **Δομή** - Η είσοδος αναπαριστά μία δομική διασύνδεση από φυσικά στοιχεία, η οποία ονομάζεται και λίστα διασυνδέσεων (netlist). Όπως αναφέρθηκε προηγουμένως, τα δομικά στοιχεία από μόνα τους, μπορεί να

βρίσκονται σε οποιοδήποτε από τα διάφορα επίπεδα της αφαίρεσης.

Φυσικά, μία απλή είσοδος προδιαγραφών μπορεί να αποτελείται από πολλαπλά τμήματα σε διάφορα επίπεδα αφαίρεσης. Για παράδειγμα, ένα τμήμα της εισόδου μπορεί να περιγράφεται ως διεργασίες, ενώ ένα άλλο ήδη σχεδιασμένο τμήμα περιγράφεται ως μία διασύνδεση από πύλες.

Τα διάφορα επίπεδα αφαίρεσης αντικατοπτρίζουν τις διάφορες ενδιάμεσες υλοποιήσεις κατά τη διάρκεια του σχεδιασμού. Ο σχεδιασμός συχνά ξεκινά με υψηλότερα επίπεδα αφαίρεσης εφόσον οι σχεδιαστές είναι αυτοί οι οποίοι αρχικά θεμελιώνουν αυτά τα επίπεδα. Κατά συνέπεια τα διάφορα επίπεδα εισόδων στις τεχνικές κατάτμησης αναπαριστούν τα ποικίλα μέρη του σχεδιασμού που έχουν ήδη πραγματοποιηθεί πριν να εφαρμοσθεί η διαδικασία της κατάτμησης.

3.3.2 Διακριτικότητα

Το δεύτερο θέμα κατάτμησης είναι η διακριτικότητα των λειτουργικών αντικειμένων στα οποία αναλύεται η δεδομένη είσοδος. Όπως αναφέρθηκε προηγουμένως, οι προδιαγραφές αναλύονται σε λειτουργικά αντικείμενα τα οποία στη συνέχεια κατατμούνται μεταξύ των στοιχείων του συστήματος. Η διακριτικότητα της ανάλυσης αποτελεί ένα μέτρο του μεγέθους των προδιαγραφών σε κάθε αντικείμενο. Μικρή διακριτικότητα σημαίνει ότι το κάθε αντικείμενο περιέχει ένα μεγάλο ποσό προδιαγραφών. Αντιθέτως, μεγάλη διακριτικότητα σημαίνει ότι κάθε αντικείμενο περιέχει μόνο ένα μικρό ποσό από προδιαγραφές, οπότε θα υπάρχουν και πολύ περισσότερα αντικείμενα.

Για συγκεκριμένα επίπεδα αφαίρεσης εισόδου, υπάρχει μόνο μία λογική διακριτικότητα της ανάλυσης. Για παράδειγμα, το γράφημα της ροής δεδομένων σε επίπεδο διεργασιών αναλύεται συνήθως στο επίπεδο των διεργασιών, ενώ το γράφημα ροής δεδομένων σε αριθμητικό επίπεδο αναλύεται στο επίπεδο των αριθμητικών πράξεων.

Η είσοδος του επιπέδου αφαίρεσης των διεργασιών έχει την μεγαλύτερη παραμετροποιησιμότητα για την ανάλυση. Κάποιος μπορεί να θεωρήσει μόνο τις διεργασίες σαν ένα σύνολο, ή στη συνέχεια να αναλύσει τις διεργασίες σε διαδικασίες, ή επιπλέον να αναλύσει τις διεργασίες και τις διαδικασίες σε μπλοκ δηλώσεων, όπως είναι τα τμήματα βρόγχων (loops) και αποφάσεων (if-then-else). Ωστόσο κάποιος άλλος μπορεί να θεωρήσει κάθε δήλωση του συστήματος ανεξάρτητα.

Είναι ξεκάθαρο ότι όσο περισσότερο λεπτομερώς αναλύει κάποιος την είσοδο, τόσο μεγαλύτερος είναι ο αριθμός των πιθανών κατατμήσεων που μπορεί να ληφθούν και επομένως τόσο καλύτερη η βελτιστοποίηση που μπορεί να επιτευχθεί. Ωστόσο, η υψηλή διακριτικότητα έχει διάφορα μειονεκτήματα. Πρώτα απ΄ όλα, εφόσον υπάρχουν πολλά αντικείμενα για κατάτμηση, η κατάτμηση πρέπει να χρησιμοποιεί περισσότερο υπολογιστικό χρόνο γιατί διαφορετικά τα αποτελέσματα δεν θα είναι ικανοποιητικά. Κατά δεύτερον, η χειροκίνητη επέμβαση είναι πολύ πιο δύσκολη διαδικασία για να υποστηριχθεί επειδή οι σχεδιαστές δεν μπορούν να αναγνωρίσουν τα αντικείμενα υψηλής διακριτικότητας. Τρίτον, η έξοδος της κατάτμησης είναι λιγότερο κατανοητή στους ανθρώπους και κατά συνέπεια περισσότερο δύσκολο να σχεδιασθεί ή να τροποποιηθεί χειροκίνητα. Τέταρτον, η εκτίμηση είναι μία εξαιρετικά δύσκολη διαδικασία.

Για να γίνει κατανοητό το γιατί η εκτίμηση είναι κάτι το πραγματικά δύσκολο να γίνει για χαμηλότερα επίπεδα διακριτικότητας, ας θεωρήσουμε την εκτίμηση του μεγέθους του υλικού κατά την κατάτμηση. Υποθέτοντας ότι είναι διαθέσιμος ένας εκτιμητής μεγέθους με μεγάλη ακρίβεια, με υπολογιστική πολυπλοκότητα της τάξης του $O(n^2)$, όπου n είναι το πλήθος των αριθμητικών πράξεων που λαμβάνουν χώρα. Αν η κατάτμηση πραγματοποιηθεί στο χαμηλό επίπεδο των λειτουργιών και υπάρχουν 10,000 πράξεις, ο αριθμός των υπολογισμών που απαιτούνται για την εκτίμηση του μεγέθους θα είναι 100,000,000

υπολογισμοί ανά κατάτμηση, το οποίο είναι ξεκάθαρα μη πρακτικό αν πρόκειται να εξετασθεί ένας μεγάλος αριθμός από κατατμήσεις. Αντιθέτως, αν θεωρηθεί κατάτμηση στα υψηλότερα επίπεδα των διεργασιών, τότε μπορεί να χρησιμοποιηθεί ο εκτιμητής του μεγέθους ώστε να ληφθεί μία εκτίμηση για κάθε διεργασία πριν την κατάτμηση. Ενώ αυτή η εκτίμηση είναι πολύπλοκη από την υπολογιστική πλευρά, εντούτοις πραγματοποιείται μόνο μία φορά. Το αποτέλεσμα είναι ότι κάθε διεργασία έχει ήδη μία εκτίμηση για το μέγεθος της. Το εκτιμώμενο μέγεθος για μία δεδομένη κατάτμηση σε επίπεδο διεργασίας απαιτεί μόνο τον συνδυασμό των εκτιμήσεων των διεργασιών. Συνδυάζοντας τέτοιου είδους εκτιμήσεις για κάθε κατάτμηση είναι κάτι πολύ πιο απλό από την πραγματοποίηση μίας εκτίμησης σε επίπεδο λειτουργίας για κάθε κατάτμηση.

3.3.3 Κατανομή στοιχείων συστήματος

Μία τεχνική κατάτμησης πρέπει να καθορίζει του τύπους των στοιχείων του συστήματος στους οποίους πρέπει να αντιστοιχισθούν τα αντικείμενα λειτουργίας. Ορισμένοι κοινοί τύποι στοιχείων είναι οι επεξεργαστές, τα ASICs, οι μνήμες, τα αρχεία καταχωρητών, οι διαδρομές δεδομένων και οι δίαυλοι.

Η κατανομή των στοιχείων του συστήματος είναι η διαδικασία της επιλογής τύπων στοιχείων του συστήματος μεταξύ των επιτρεπόμενων, καθώς και η επιλογή του αριθμού από τον καθένα ώστε να χρησιμοποιηθούν σε ένα δεδομένο σχεδιασμό. Το σύνολο των επιλεγμένων στοιχείων ονομάζεται **κατανομή**. Για την υλοποίηση δεδομένων προδιαγραφών, μπορούν να χρησιμοποιηθούν διάφορες κατανομές όπου η κάθε μία διαφέρει από την άλλη στην απόδοση και στο κόστος. Άλλα θέματα που πρέπει να λαμβάνονται υπόψη κατά την επιλογή της κατανομής είναι η κατανάλωση ισχύος, ο χρόνος σχεδίασης και η αξιοπιστία του εκάστοτε προμηθευτή. Η κατανομή μπορεί να γίνει είτε με την άμεση επέμβαση του σχεδιαστή είτε αυτοματοποιημένα σε συνδυασμό με τον αλγόριθμο

κατάτμησης.

3.3.4 Μέτρα και εκτιμήσεις

Μία τεχνική πρέπει να καθορίζει τα χαρακτηριστικά μίας κατάτμησης η οποία προσδιορίζει την ποιότητα της κατάτμησης. Τέτοια χαρακτηριστικά ονομάζονται **μέτρα**. Τυπικές παράμετροι των μέτρων είναι το κόστος, ο χρόνος εκτέλεσης, οι ρυθμοί επικοινωνίας, η κατανάλωση ισχύος, η επιφάνεια, οι ακροδέκτες, η δυνατότητα του για υποβολή σε δοκιμές, η αξιοπιστία, το μέγεθος του προγράμματος, το μέγεθος των δεδομένων και το μέγεθος της μνήμης.

Μερικές τεχνικές ομαδοποιούν αντικείμενα πριν τη στιγμή που θα εφαρμοσθεί η κατάτμηση. Εφόσον τα παραπάνω μέτρα ορίζονται μόνο για μία περατωμένη κατάτμηση, απαιτείται ένας νέος τύπος μέτρων που προβλέπει το πλεονέκτημα της ομαδοποίησης δύο νέων αντικειμένων. Τέτοια μέτρα ονομάζονται μέτρα **εγγύτητας**. Για παράδειγμα ένα αντικείμενο δομής είναι κοντά σε μία συγκεκριμένη ομάδα δομικών αντικειμένων αν υπάρχουν πολλά καλώδια που τα διασύνδεουν. Μία διεργασία είναι κοντά σε μία άλλη ομάδα διεργασιών αν πολλές μεταβλητές διαμοιράζονται μεταξύ τους.

Για τον λόγο αυτό πρέπει να υπάρχουν μέθοδοι για τον υπολογισμό των τιμών τέτοιου είδους μέτρων. Η δυσκολία εμφανίζεται επειδή όλα τα μέτρα ορίζονται με την έννοια της δομής (ή του λογισμικού), το οποίο υλοποιεί τα αντικείμενα λειτουργίας τη στιγμή που κατά τη κατάτμηση, δεν υπάρχει τέτοιου είδους υλοποίηση. Υπάρχουν δύο επιλογές για τον υπολογισμό των μέτρων. Η πρώτη είναι η δημιουργία μίας υλοποίησης. Παρόλο που αυτή η προσέγγιση παράγει ακριβή αποτελέσματα για τον προσδιορισμό των τιμών των μέτρων, δεν είναι τόσο πρακτική γιατί απαιτεί πάρα πολύ χρόνο για να γίνει. Μία υλοποίηση μπορεί να απαιτεί μερικές μέρες ή μερικούς μήνες για να γίνει. Ακόμα και αν υπήρχε η δυνατότητα σε λίγο σχετικά χρόνο να μπορεί να γίνει μία

υλοποίηση, τότε αυτή η προσέγγιση θα απαιτούσε πάλι αρκετό χρόνο, εφόσον τέτοιοι αλγόριθμοι μπορούν να παράγουν χιλιάδες κατατμήσεις για τις οποίες πρέπει κάθε φορά να υπολογισθούν τα μέτρα.

Η δεύτερη επιλογή είναι η δημιουργία μίας γρήγορης υλοποίησης μέσω της οποίας να υπολογιστούν τα μέτρα. Μία τέτοιου είδους υλοποίηση εμπεριέχει τα μεγαλύτερα στοιχεία RT του σχεδιασμού, αλλά δεν περιλαμβάνει πολλές λεπτομέρειες, όπως η ακριβής δρομολόγηση ή βελτιστοποιημένη λογική η οποία απαιτεί μεγάλο χρόνο σχεδίασης. Ο προσδιορισμός των τιμών των μέτρων από μία γρήγορη υλοποίηση ονομάζεται **εκτίμηση**.

Η ταχύτητα και η ακρίβεια είναι δύο συναγωνιστικοί στόχοι στην εκτίμηση. Η ταχύτητα είναι αποτέλεσμα της γρήγορης υλοποίησης λόγω έλλειψης λεπτομέρειας, ενώ η ακρίβεια απορρέει από το γεγονός ότι συμπεριλαμβάνονται οι λεπτομέρειες. Συνεπώς, υπάρχουν διάφορα μοντέλα για κάθε μέτρο που συναλλάζονται μεταξύ ταχύτητας και ακρίβειας.

Η έλλειψη ακρίβειας μπορεί ορισμένες φορές να είναι παραδεκτή όσο η σχετική ποιότητα από την κάθε μία κατάτμηση προσδιορίζεται ως ικανοποιητική από την εκτίμηση. Οι εκτιμήσεις οι οποίες επιτρέπουν την πρόβλεψη της σχετικής ποιότητας ανάμεσα σε δύο κατατμήσεις ονομάζεται και **πιστότητα**.

Ας σημειωθεί ότι όσο χαμηλότερο είναι το επίπεδο αφαίρεσης της εισόδου στην διεργασία της κατάτμησης, τόσο πιο ακριβές θα είναι οι εκτιμήσεις, λόγω του ότι η εκτίμηση επιχειρεί να προβλέψει τα αποτελέσματα από συνακόλουθα βήματα του σχεδιασμού. Ένα χαμηλότερο επίπεδο αφαίρεσης με λιγότερα βήματα σχεδιασμού που χρειάζεται να προβλεφθεί θα απέδιδε προφανώς λιγότερο λανθασμένες προβλέψεις.

3.3.5 Συναρτήσεις αντικειμενικότητας και εγγύτητας

Οι κατατμήσεις δεν αξιολογούνται κατά βάση από ένα μόνο μέτρο. Αντιθέτως, πολλαπλά μέτρα όπως είναι το κόστος, η απόδοση και η ισχύς, υπολογίζονται μεταφέροντας την βαρύτητα από το ένα στο άλλο. Μία έκφραση του συνδυασμού πολλαπλών τιμών μέτρων σε μία μόνο τιμή η οποία καθορίζει την ποιότητα της κατάτμησης, ονομάζεται **αντικειμενική συνάρτηση**. Η τιμή η οποία επιστρέφεται από μία τέτοια συνάρτηση ονομάζεται **κόστος**.

Εφόσον μπορεί να υπάρχουν πολλά μέτρα μεταβλητής σημασίας, ο συνδυασμός αυτών των μέτρων σε μία μόνο τιμή δεν είναι μία απλή διαδικασία. Οι περισσότερες προσεγγίσεις χρησιμοποιούν μία αντικειμενική συνάρτηση αθροίσματος βαρών η οποία προσθέτει όλα τα γινόμενα των μέτρων με τα αντίστοιχα βάρη, όπου τα βάρη υποδεικνύουν κάθε φορά τη σχετική σημασία του κάθε μέτρου. Για παράδειγμα, η παρακάτω συνάρτηση αντικειμενικότητας είναι μία συνάρτηση αθροισμάτων με βάρη στην οποία τρεις τιμές μέτρων $area$ (επιφάνεια), $delay$ (καθυστέρηση) και $power$ (ισχύς), δέχονται ως συντελεστές βαρών τις σταθερές k_1, k_2, και k_3 αντίστοιχα και στη συνέχεια προστίθενται.

$$Objfct = k_1 \cdot area + k_2 \cdot delay + k_3 \cdot power \tag{3.1}$$

Η ανάθεση στη σταθερά k_1, μεγαλύτερης τιμής από ότι στις k_2 και k_3 κάνει την επιφάνεια να είναι περισσότερο σημαντικό μέτρο.

Εφόσον οι περισσότερες αποφάσεις σχεδιασμού προέρχονται από περιορισμούς, απλές συναρτήσεις όπως αυτή της Εξίσωσης 3.1 δεν χρησιμοποιούνται συχνά. Οι περιορισμοί πρέπει να υπεισέρχονται μέσα στις αντίστοιχες συναρτήσεις έτσι ώστε οι κατατμήσεις που ικανοποιούν τους περιορισμούς να μελετούνται καλύτερα σε σύγκριση με αυτές που δεν τους ικανοποιούν. Για παράδειγμα,

η παραπάνω σχέση μπορεί να αναπτυχθεί ως εξής:

$$\begin{aligned} Objfct &= k_1 \cdot F(area, area\_constr) \\ &+ k_2 \cdot F(delay, delay\_constr) \\ &+ k_3 \cdot F(power, power\_constr) \end{aligned} \tag{3.2}$$

όπου F είναι μία συνάρτηση που υποδεικνύει πόσο κοντά είναι η κάθε εκτίμηση του μέτρου σε έναν δεδομένο περιορισμό. Μία κοινή μορφή της συνάρτησης F επιστρέφει το ποσό το οποίο η εκτίμηση του μέτρου παραβιάζει τον περιορισμό και επιστρέφει μηδέν όταν δεν υπάρχει καμία παραβίαση. Συνεπώς, όταν δύο κατατμήσεις ικανοποιούν τον περιορισμό της ισχύος, η ισχύς δεν παίζει κανένα ρόλο στη σύγκρισή τους, ακόμη και αν η ισχύς που καταναλώνεται από μία κατάτμηση είναι μεγαλύτερη από την άλλη. Αυτή η μορφή της συνάρτησης F κάνει την αντικειμενική συνάρτηση να επιστρέφει μηδέν όταν μία κατάτμηση ικανοποιεί όλους τους περιορισμούς, επιτυγχάνοντας το στόχο της κατάτμησης να λαμβάνει μηδενικό κόστος.

Ένα άλλο θέμα στο συνδυασμό διαφόρων μέτρων σε μία μόνο τιμή, είναι η κανονικοποίηση των μονάδων μέτρων. Για παράδειγμα, η εκτίμηση της επιφάνειας μπορεί να είναι 10,000 μονάδες επιφάνειας έχοντας ως περιορισμό τις 9,000 μονάδες επιφάνειας, όπου επίσης η καθυστέρηση μπορεί να είναι 10 μονάδες χρόνου έχοντας ως περιορισμό την μία μονάδα χρόνου. Υποθέτοντας ότι όλοι συντελεστές k_1, k_2 και k_3 της αντικειμενικής συνάρτησης είναι ίσοι μεταξύ τους, τότε θα πρέπει να δοθεί μεγαλύτερη προσοχή στην μείωση της καθυστέρησης από ότι την επιφάνεια, εφόσον η καθυστέρηση είναι δέκα φορές μεγαλύτερη από τον περισορισμό της. Συγκεκριμένα, η συνάρτηση πρέπει να προτιμήσει μία μείωση πέντε μονάδων καθυστέρησης αντί για μείωση πέντε μονάδων επιφάνειας, εφόσον η πρώτη αναπαριστά μία μείωση της τάξης του 50%, ενώ η δεύτερη αντιπροσωπεύει μία μείωση της επιφάνειας της τάξης του

1%. Μία προσέγγιση για να προτιμήσει την προαναφερθείσα μείωση είναι η κανονικοποίηση της κάθε τιμής της F μέσω της διαίρεσής της με το μέτρο του περιορισμού.

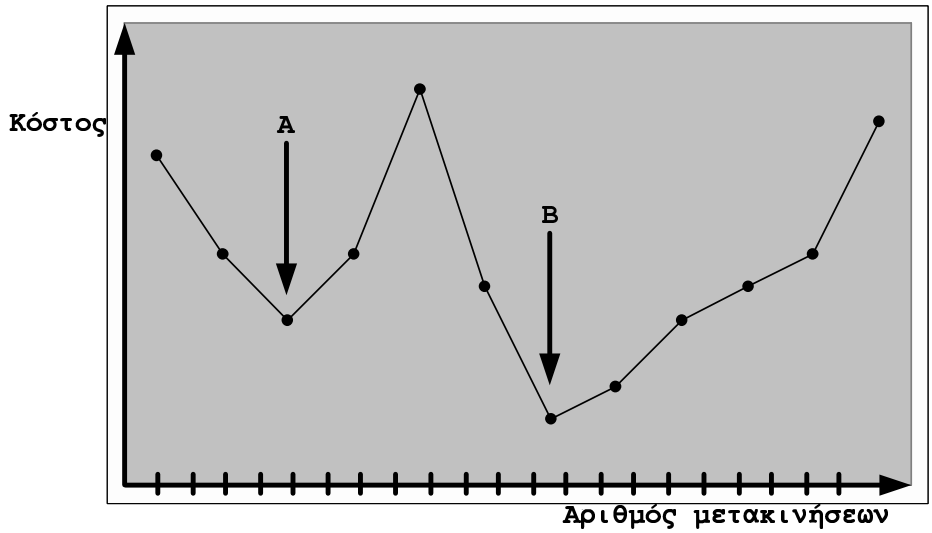
Ενώ μία αντικειμενική συνάρτηση συνδυάζει μέτρα ώστε να εκτιμήσει μία κατάτμηση, μία **συνάρτηση εγγύτητας** συνδυάζει μέτρα εγγύτητας ώστε να υποδείξει την επιθυμία για ομαδοποίηση αντικειμένων, πρώτου περατωθεί μία πλήρης κατάτμηση. Η διαδικασία που περιγράφηκε παραπάνω σχετικά με τον υπολογισμό των βαρών και την κανονικοποίηση των αντικειμενικών συναρτήσεων ισχύει και εφαρμόζεται επίσης και στις συναρτήσεις εγγύτητας.

3.3.6 Αλγόριθμοι κατάτμησης

Δεδομένου ενός συνόλου από λειτουργικά αντικείμενα και ενός συνόλου από στοιχεία του συστήματος, ένας αλγόριθμος κατάτμησης ψάχνει για την βέλτιστη κατάτμηση. Η βέλτιστη κατάτμηση είναι αυτή με το χαμηλότερο κόστος, όπως αυτό υπολογίζεται από μία αντικειμενική συνάρτηση.

Η βέλτιστη κατάτμηση μπορεί φυσικά να βρεθεί μέσω μίας εξαντλητικής αναζήτησης η οποία παράγει όλες τις πιθανές κατατμήσεις, αξιολογώντας την κάθε μία και επιλέγοντας την καλύτερη. Δυστυχώς, ένας τέτοιος αλγόριθμος θα ήταν μη πρακτικός εξαιτίας του ποσού της επεξεργασίας που απαιτείται. Αν υπάρχουν n αντικείμενα και m στοιχεία συστήματος, τότε υπάρχουν m^n πιθανές αντιστοιχίσεις. Ένα σχετικά μικρό πρόβλημα για παράδειγμα, είναι αυτό στο οποίο $n = 20$ και $m = 4$, το οποίο θα οδηγούσε σε πάνω από ένα τρισεκατομμύριο πιθανές αντιστοιχίσεις. Κατά συνέπεια η ουσία ενός αλγορίθμου κατάτμησης είναι ο τρόπος με τον οποίο επιλέγεται ένα υποσύνολο από όλες τις πιθανές κατατμήσεις για να εξετασθεί για την καταλληλότητά του.

Γενικά υπάρχουν δύο διαφορετικοί τρόποι για τους αλγορίθμους κατάτμησης: οι κατασκευαστικοί και οι επαναληπτικοί. Οι κατασκευαστικοί αλγόριθμοι



Σχήμα 3.2: Διαφυγή τοπικών ελαχίστων στην επαναληπτική κατάτμηση

ομαδοποιούν αντικείμενα σε μία πλήρης κατάτμηση. Τέτοιοι αλγόριθμοι χρησιμοποιούν μέτρα εγγύτητας για να ομαδοποιήσουν αντικείμενα με την ελπίδα ότι τέτοια έμμεσα μέτρα θα οδηγήσουν σε μία καλή κατάτμηση. Ο χρόνος υπολογισμού σε έναν κατασκευαστικό αλγόριθμο καταναλώνεται στην κατασκευή ενός μικρού αριθμού κατατμήσεων.

Οι επαναληπτικοί αλγόριθμοι τροποποιούν μία πλήρης κατάτμηση, ελπίζοντας ότι κάποιες από τις τροποποιήσεις θα βελτιώσουν την κατάτμηση. Τέτοιοι αλγόριθμοι χρησιμοποιούν μία αντικειμενική συνάρτηση για να αξιολογήσουν την εκάστοτε κατάτμηση, η οποία παρέχει πιο ακριβείς αξιολογήσεις από ότι οι συναρτήσεις εγγύτητας οι οποίες χρησιμοποιούνται από τους κατασκευαστικούς αλγορίθμους. Ο υπολογιστικός χρόνος σε έναν κατασκευαστικό αλγόριθμο καταναλώνεται στην αξιολόγηση μεγάλων αριθμών από κατατμήσεις. Οι επαναληπτικοί αλγόριθμοι διαφέρουν ο ένας από τον άλλο κυρίως στους τρόπους με τους οποίους τροποποιούν την κατάτμηση και στο πως αποδέχονται ή απορρίπτουν κακές τροποποιήσεις. Ο στόχος είναι η διαφυγή από τα

τοπικά ελάχιστα ενώ πραγματοποιείται όσον το δυνατόν λιγότερη επεξεργασία. Η σύλληψη του τοπικού ελαχίστου παρουσιάζεται στο Σχήμα 3.2, όπου παρουσιάζεται μία σειρά από μετακινήσεις σε συνάρτηση με το κόστος μετά από κάθε μετακίνηση. Μία κίνηση είναι μία τροποποίηση κατάτμησης η οποία επαναντιστοιχίζει ένα αντικείμενο από ένα σύνολο σε ένα άλλο. Το αριστερότερο σημείο παριστάνει το κόστος της αρχικής κατάτμησης. Οι δύο πρώτες μετακινήσεις μειώνουν το κόστος στην τιμή A. Οι επόμενες δύο κινήσεις αυξάνουν το κόστος, αλλά διαδοχικές μετακινήσεις μειώνουν το κόστος στο B, το οποίο είναι μικρότερο από το A. Η τιμή του A αναφέρεται και ως **τοπικό ελάχιστο** ενώ το B αποτελεί ένα ολικό ελάχιστο. Ένας αλγόριθμος ο οποίος επιτρέπει και αποδέχεται μετακινήσεις οι οποίες οδηγούν μόνο σε μείωση του κόστους ονομάζεται **πλεονεκτικός αλγόριθμος** (greedy algorithm). Τέτοιοι αλγόριθμοι δεν έχουν τη δυνατότητα να διαφύγουν από τοπικά ελάχιστα. Ένας αλγόριθμος ο οποίος μπορεί να διαφύγει από ένα τοπικό ελάχιστο ονομάζεται και ως **αλγόριθμος αναρρίχησης λόφου** (fill-climbing algorithm). Στην πράξη, χρησιμοποιούνται αλγόριθμοι οι οποίοι είναι συνδυασμός από κατασκευαστικούς και επαναληπτικούς αλγορίθμους.

3.3.7 Αποτέλεσμα αλγορίθμου

Κάθε τεχνική κατάτμησης πρέπει να καθορίσει την μορφή αναπαράστασής της και την ενδεχόμενη χρήση της εξόδου της, δηλαδή των αποτελεσμάτων της. Η μορφή αυτή για παράδειγμα, μπορεί να είναι μία λίστα η οποία υποδεικνύει ποια λειτουργικά αντικείμενα αντιστοιχίζονται σε ποιο στοιχείο του συστήματος. Μπορεί επίσης να υπάρξει μία νεότερη έκδοση των προδιαγραφών εισόδου η οποία περιέχει δομικά αντικείμενα για τα στοιχεία του συστήματος και καθορίζει την λειτουργία ενός στοιχείου χρησιμοποιώντας τα αντιστοιχιζόμενα λειτουργικά αντικείμενα σε αυτό. Η ενδεχόμενη χρήση των εξόδου περιγρά-

φει το ρόλο που παίζει η έξοδος σε διαδοχικές διεργασίες σχεδιασμού. Για παράδειγμα, η έξοδος μπορεί να παρασχεθεί σαν λειτουργική προδιαγραφή για ανθρώπους που πρέπει να υλοποιήσουν το κάθε στοιχείο του συστήματος. Επίσης μπορεί να χρησιμοποιηθεί σαν είσοδος σε ένα εργαλείο σύνθεσης. Μπορεί επίσης να παρέχει οδηγίες και συμβουλές σε ένα εργαλείο σύνθεσης το οποίο μπορεί να επιλέξει να κάνει χρήση μερικών ή όλων των εξόδων της κατάτμησης έτσι ώστε να οδηγηθεί στις δικές του αποφάσεις.

3.3.8 Ροή ελέγχου και αλληλεπίδραση σχεδιαστή

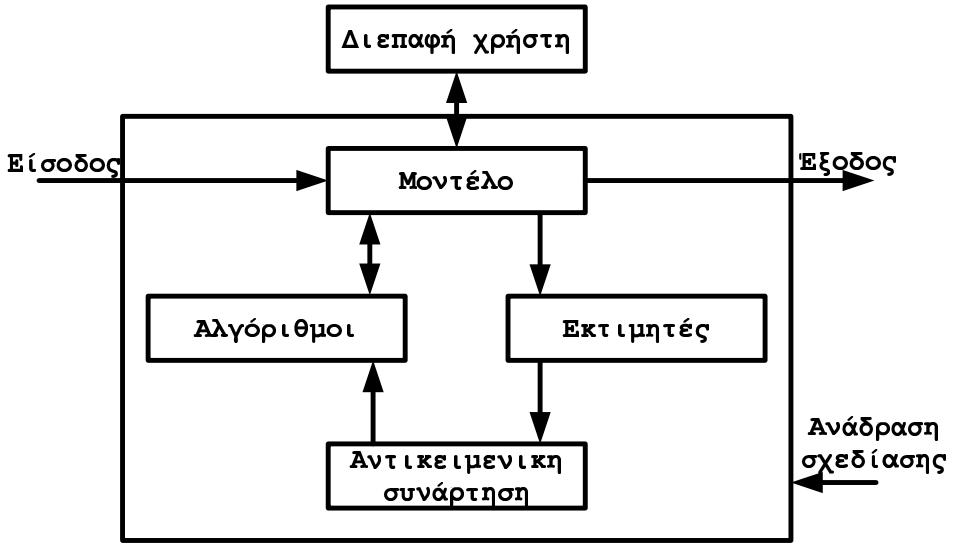
Τα παραπάνω παρουσιάζουν ότι η ενέργεια της κατάτμησης μίας προδιαγραφής μεταξύ διαφόρων στοιχείων του συστήματος απαιτεί πολλές αποφάσεις όπως η επιλογή της διακριτικότητας των αντικειμένων, η κατανομή των στοιχείων του συστήματος, η επιλογή των μέτρων ποιότητας, η επιλογή των αντικειμενικών ή των συναρτήσεων εγγύτητας καθώς και την επιλογή του αλγορίθμου κατάτμησης. Αυτές οι αποφάσεις μπορούν να πραγματοποιηθούν με την παραπάνω σειρά ή και τυχαία, όμως κάποιες από τις αποφάσεις αυτές ίσως χρειάζεται να αναθεωρηθούν και να επαναληφθεί η εφαρμογή τους πάρα πολλές φορές. Για παράδειγμα, μπορεί να γίνει η επιλογή της διακριτικότητας, των μέτρων εγγύτητας, της συνάρτησης εγγύτητας, η κατανομή των στοιχείων, να εφαρμοσθεί ένας κατασκευαστικός αλγόριθμος κατάτμησης, να γίνει επανεπιλογή μίας άλλης συνάρτησης εγγύτητας, να ξαναεφαρμοσθεί ο κατασκευαστικός αλγόριθμος, να γίνει η επιλογή μίας αντικειμενικής συνάρτησης, να εφαρμοσθεί ένας επαναληπτικός αλγόριθμος κατάτμησης, να ανακατανεμηθούν τα στοιχεία του συστήματος και να ξαναεφαρμοσθεί ο επαναληπτικός αλγόριθμος κατάτμησης. Συνεπώς η ροή του ελέγχου κατά την όλη διεργασία μπορεί να ποικίλλει. Μία τεχνική κατάτμησης πρέπει μεταξύ άλλων να καθορίζει τις πιθανές διαδοχές των αποφάσεων. Εφόσον διαφορετικές ακολουθίες

οδηγούν σε διαφορετικά αποτελέσματα, η τεχνική μπορεί να ορίζει ακολουθίες οι οποίες παρέχουν καλά αποτελέσματα για έναν συγκεκριμένο σχεδιαστικό στόχο, όπως είναι για παράδειγμα η μεγιστοποίηση της απόδοσης.

Από την πρακτική πλευρά, ένα σύστημα κατάτμησης πρέπει να επιτρέπει την αλληλεπίδραση του σχεδιαστή. Υπάρχουν δύο τύποι πιθανών αλληλεπιδράσεων. Ο πρώτος τύπος, που ονομάζεται κατευθυντήριος, περιγράφει τις πιθανές ενέργειες που μπορεί να κάνει ο σχεδιαστής, όπως είναι η κατανομή, η μετακίνηση συγκεκριμένων αντικειμένων σε συγκεκριμένα στοιχεία, υπερισχύοντας της προηγούμενης εκτίμησης όταν φαίνεται ότι τα αποτελέσματα είναι καλύτερα. Ο δεύτερος τύπος, που ονομάζεται ανάδραση, περιγράφει την τρέχουσα σχεδιαστική πληροφορία που είναι διαθέσιμη στον σχεδιαστή. Για παράδειγμα, ένα γράφημα μπορεί να αναπαριστά τον αριθμό των καλωδίων μεταξύ των αντικειμένων, ή ένα ιστόγραμμα μπορεί να αναπαριστά το βαθμό στον οποίο παραβιάζονται οι περιορισμοί.

3.3.9 Τυπικός σχηματισμός συστήματος

Το Σχήμα 3.3 περιγράφει έναν τυπικό σχηματισμό μίας εκτελέσιμης προδιαγραφής ενός συστήματος κατάτμησης. Η προδιαγραφή *Είσοδος* μετατρέπεται σε ένα εσωτερικό και ισοδύναμο *Μοντέλο* στο οποίο εφαρμόζονται διάφοροι *Αλγόριθμοι* κατάτμησης. Οι αλγόριθμοι χρειάζονται *Εκτιμητές* οι οποίοι λειτουργούν από το μοντέλο και μία *Αντικειμενική συνάρτηση* η οποία χρησιμοποιεί τις εκτιμήσεις. Το κατατμημένο μοντέλο μετατρέπεται σε μορφή *Εξόδου* η οποία είναι κατάλληλη για περαιτέρω σχεδιασμό ή ανάλυση. Τα μέτρα τα οποία λαμβάνονται από μία ακολουθιακή υλοποίηση, που ονομάζεται *Ανάδραση σχεδίασης*, μπορεί να χρησιμοποιηθεί για την μετέπειτα βελτίωση της κατάτμησης. Ο σχεδιαστής μπορεί συνήθως να αλληλεπιδρά με τα διάφορα τμήματα του συστήματος μέσω της *Διεπαφής χρήστη*.



Σχήμα 3.3: Τυπικό σχηματισμός ενός συστήματος κατάτμησης

3.4 Βασικοί αλγόριθμοι κατάτμησης

Ένας αλγόριθμος κατάτμησης αντιστοιχεί κάθε λειτουργικό αντικείμενο σε μία μόνο ομάδα, όπου η κάθε ομάδα αναπαριστά ένα στοιχείο του συστήματος. Υπό ιδανικές συνθήκες, η κατάτμηση η οποία προκύπτει από έναν αλγόριθμο είναι αυτή η οποία οδηγεί στο μικρότερο κόστος, όπως αυτό υπολογίζεται από την αντικειμενική συνάρτηση. Το πρόβλημα της κατάτμησης μπορεί να οριστεί μαθηματικά από την παρακάτω διατύπωση:

Δεδομένου ενός συνόλου αντικειμένων $O = \{o_1, o_2, \ldots, o_n\}$, να καθοριστεί μία **κατάτμηση** $P = \{p_1, p_2, \ldots, p_m\}$ έτσι ώστε $p_1 \cup p_2 \cup \ldots p_m = O$, $p_i \cap p_j = 0$ για όλα τα $i, j, i \neq j$ και το κόστος της οποίας υπολογίζεται από μία αντικειμενική συνάρτηση $Objfct(P)$ να είναι ελάχιστη.

Στη συνέχεια περιγράφονται διάφοροι αλγόριθμοι κατάτμησης. Αυτοί είναι οι τυχαίας αντιστοίχησης, ιεραρχικής ομαδοποίησης, πολλαπλών επιπέδων σύμπλεξης, εικονικής ισχυροποίησης και της γενετικής εξέλιξης.

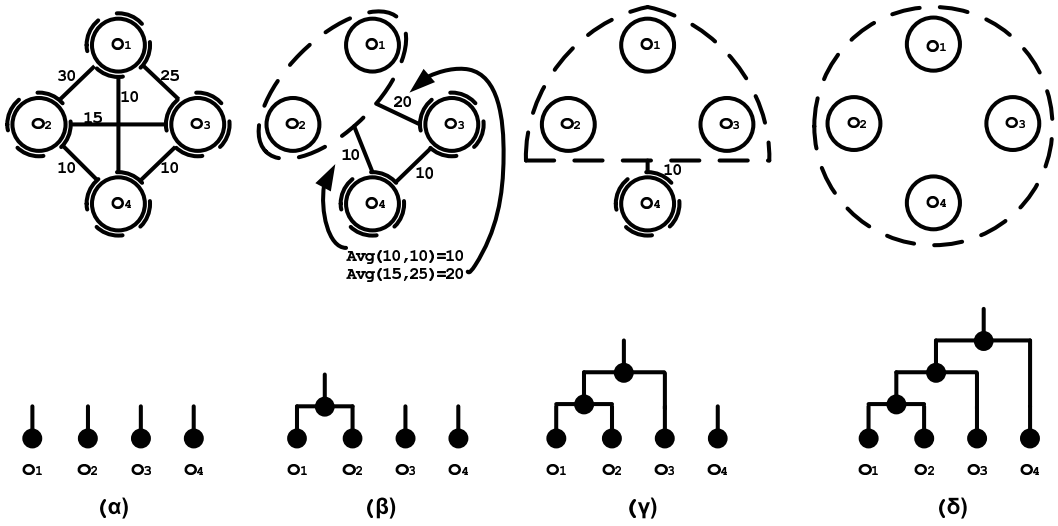
3.4.1 Τυχαίας αντιστοίχησης

Ένας συχνά χρησιμοποιούμενος κατασκευαστικός αλγόριθμος είναι αυτός της τυχαίας αντιστοίχησης στον οποίο κάθε αντικείμενο αντιστοιχίζεται με τυχαίο τρόπο σε ένα από τα δεδομένα στοιχεία του συστήματος. Ο αλγόριθμος χρησιμοποιείται συνήθως για τη δημιουργία μίας αρχικής κατάτμησης η οποία θα τροφοδοτήσει έναν επαναληπτικό αλγόριθμο κατάτμησης. Η επεξεργαστική πολυπλοκότητα του εν λόγω αλγορίθμου είναι της τάξης $O(n)$, όπου n είναι ο αριθμός των αντικειμένων.

3.4.2 Ιεραρχικής ομαδοποίησης

Η ιεραρχική ομαδοποίηση αποτελεί μία κοινά χρησιμοποιούμενη ομάδα κατασκευαστικών αλγορίθμων κατάτμησης [Joh94], [LT91], [MK90], [CB87], [Ga91]. Χρησιμοποιεί μέτρα εγγύτητας ώστε να ομαδοποιεί αντικείμενα εφόσον δεν μπορούν να προσδιορισθούν άλλα μέτρα χωρίς να υπάρχει μία πλήρης κατάτμηση. Τα μέτρα εγγύτητας σκοπεύουν να αποδώσουν μία κατάτμηση με καλές τιμές γενικών μέτρων. Αυτή η προσέγγιση ομαδοποιεί τα κοντινά αντικείμενα μεταξύ τους, επαναϋπολογίζει την εγγύτητα μετά την ομαδοποίηση αυτή και επαναλαμβάνεται μέχρις ότου πραγματοποιηθεί κάποια από τις συνθήκες τερματισμού του αλγορίθμου.

Στην εν λόγω απεικόνιση, επεξήγεται ένας τέτοιος αλγόριθμος χρησιμοποιώντας τον παραπάνω ορισμό. Ο αλγόριθμος χρησιμοποιεί μία διαδικασία, την $ComputeCloseness$ η οποία υπολογίζει την τιμή της εγγύτητας μεταξύ δύο αντικειμένων o_i και o_j, έχοντας την προκύπτουσα εγγύτητα αποθηκευμένη στην μεταβλητή $c_{i,j}$ η οποία ανήκει σε ένα σύνολο C. Για τα αρχικά αντικείμενα, η $ComputeCloseness$ χρησιμοποιεί μία συνάρτηση εγγύτητας για να προσδιορισθεί η εγγύτητα, όπως αυτή περιγράφηκε στην παράγραφο 3.3.4.



Σχήμα 3.4: Ιεραρχική ομαδοποίηση

Ωστόσο, για μία μορφή ιεραρχικού αντικειμένου το οποίο έχει συγχωνευθεί, υπάρχουν διάφοροι τρόποι για να υπολογισθεί η εγγύτητα. Ένας τρόπος μπορεί μεν να χρησιμοποιεί την συνάρτηση εγγύτητας, αλλά η επαναλαμβανόμενη χρήση αυτής της συνάρτησης απαιτεί μεγάλο υπολογιστικό χρόνο. Ένας άλλος τρόπος είναι η προσέγγιση της εγγύτητας μεταξύ ενός ιεραρχικού αντικειμένου o_{ij} και ενός αντικειμένου o_k ως το ελάχιστο, μέγιστο, μέσο, ή το άθροισμα των τιμών εγγύτητας $c_{i,k}$ και $c_{j,k}$.

Μία διαδικασία $FindClosestObjects$ βρίσκει το ζευγάρι των αντικειμένων με την μεγαλύτερη τιμή εγγύτητας. Μία άλλη διεργασία, η $Terminate$, επιστρέφει αληθής όταν ο αλγόριθμος θα πρέπει να τερματιστεί. Μία τυπική κατάσταση στην οποία η συνάρτηση $Terminate$ επιστρέφει αληθής συμβαίνει όταν τα αντικείμενα που μας ενδιαφέρουν έχουν συγχωνευθεί σε ένα συγκεκριμένο αριθμό ομάδων. Μία άλλη περίπτωση που μπορεί να επιστρέψει αληθής η συνάρτηση αυτή, μπορεί να συμβεί όταν όλες οι τιμές εγγύτητας είναι μικρότερες από ένα συγκεκριμένο αριθμό. Αυτός ο αριθμός ονομάζεται **κατώφλι**

εγγύτητας.

Αλγόριθμος ιεραρχικής ομαδοποίησης

```
 /* Initialize each object as a group */
 for each o(i) loop
   p(i) = o(i)
   P = P cap p(i)
 end loop

/* Compute closeness between objects */
for each p(i) loop
   for each p(j) loop
      c(i,j) = ComputeCloseness(p(i),p(j))
      C = C cap c(i,j)
   end loop
end loop

/* Merge closest objects and recompute closeness */
while not Terminate(P) loop
   p(i),p(j) = FindClosestObjects(P,C)
   P = P - p(i) - p(j) cap p(ij)
   for each p(k) loop
     c(ij,k) = ComputeCloseness(p(ij),p(k))
   end loop
end loop

return P
```

Ο αλγόριθμος ξεκινά με την αρχικοποίηση του κάθε αντικειμένου ως μία

ομάδα-σύνολο και στη συνέχεια με τον υπολογισμό της συνάρτησης εγγύτητας μεταξύ κάθε ζεύγους αντικειμένων. Το βασικό τμήμα το αλγορίθμου συγχωνεύει τα κοντινά αντικείμενα σε ένα νέο αντικείμενο (στην πραγματικότητα σε μία ομάδα) και στη συνέχεια επαναϋπολογίζει την εγγύτητα μεταξύ αυτού του νέου αντικειμένου και κάθε άλλου αντικειμένου. Το βασικό τμήμα επαναλαμβάνεται μέχρις ότου πληρωθούν οι συνθήκες τερματισμού του αλγορίθμου. Η πολυπλοκότητα του αλγορίθμου επηρεάζεται από τον υπολογισμό της εγγύτητας μεταξύ όλων των ζευγών των αντικειμένων, ο οποίος είναι της τάξης $O(n^2)$.

Για παράδειγμα, το Σχήμα 3.4(α) δείχνει τέσσερα αντικείμενα και τις τιμές της εγγύτητάς τους. Τα δύο πιο κοντινά αντικείμενα είναι τα o_1 και o_2 με τιμή εγγύτητας 30. Στο Σχήμα 3.4(β), συγχωνεύονται τα o_1 και o_2 και εκτιμώνται οι τιμές της εγγύτητας μεταξύ του νέου αντικειμένου και των o_3 και o_4 ως η μέση τιμή των προηγουμένων τιμών εγγύτητας. Στο Σχήμα 3.4(γ), επαναπραγματοποιείται η συγχώνευση των κοντινών αντικειμένων και υπολογίζεται μία νέα τιμή εγγύτητας. Υποθέτοντας ότι ο αλγόριθμος τερματίζεται όταν καμία τιμή εγγύτητας δεν υπερβαίνει το κατώφλι της τιμής 15, η τελική κατάτμηση είναι τα o_1, o_2, o_3 να ανήκουν σε μία ομάδα και το o_4 να ανήκει σε ένα άλλο.

Ο αλγόριθμος μπορεί να τροποποιηθεί έτσι ώστε να δημιουργήσει ένα **δέντρο ομαδοποίησης** για να διατηρεί ένα ιστορικό της σειράς με την οποία συγχωνεύονται τα διάφορα αντικείμενα και να συνεχίζει τις συγχωνεύσεις μέχρις ότου να υπάρξει μία μόνο ομάδα. Μπορούμε να χρησιμοποιήσουμε ένα τέτοιο ιστορικό για να δημιουργήσουμε μία ποικιλία διαφόρων κατατμήσεων. Η δημιουργία και η χρήση ενός τέτοιου δέντρου μπορεί να περιγραφεί ακόμη καλύτερα μέσω ενός παραδείγματος. Στο Σχήμα 3.4(α) τα αντικείμενα o_1, o_2, o_3 και o_4 παρουσιάζονται ώς φύλλα ενός δέντρου. Όταν πραγματοποιείται η συγχώνευση των o_1 και o_2 σε ένα νέο αντικείμενο στο Σχήμα 3.4(β), προστίθεται ένας

νέος κόμβος στο δέντρο ως γόνος των δύο συγχωνευμένων κόμβων. Όταν γίνει η συγχώνευση του νέου αντικειμένου με το o_3 στο Σχήμα 3.4(γ) προσθέτουμε έναν ακόμη κόμβο στο δέντρο. Όταν γίνεται και η τελική συγχώνευση με το αντικείμενο o_4, ο κόμβος που προστίθεται είναι ουσιαστικά η ρίζα του δέντρου. Το Σχήμα 3.4(δ) παρουσιάζει το δέντρο της ομαδοποίησης που παράγεται από μία τέτοια διαδικασία. Μία οριζόντια γραμμή η οποία σχηματίζεται κατά μήκος του δέντρου παράγει μία κατάτμηση. Για παράδειγμα, ο σχηματισμός μίας γραμμής πάνω από το δεύτερο επίπεδο των κόμβων, όπως στο Σχήμα 3.4(α), καθορίζει μία κατάτμηση η οποία αποτελείται από τα o_1, o_2 σε μία ομάδα, έχοντας τα o_3 και o_4 σε δύο άλλες ξεχωριστές ομάδες. Μία τέτοιου είδους γραμμή ονομάζεται ως **γραμμή αποκοπής**. Η γραμμή μπορεί να σχεδιασθεί σε κάθε επίπεδο του δέντρου. Ο σχεδιασμός τέτοιων γραμμών μπορεί να παράγει πολλές διαφορετικές κατατμήσεις, όπου κάθε κατάτμηση αποτελείται από ομάδες των οποίων τα αντικείμενα είναι κοντά σε κάποια άλλα.

3.4.3 Εξομοιωμένη ισχυροποίηση

Πολλοί αλγόριθμοι κατάτμησης παρουσιάζουν αδυναμία στο να διαφύγουν από κάποιο τοπικό ακρότατο, το οποίο συνήθως είναι τοπικό ελάχιστο. Ωστόσο, αυτή η αδυναμία μπορεί να λυθεί με την αποδοχή αυξητικών κινήσεων στον αλγόριθμο όταν αυτές μπορεί να οδηγήσουν σε κατάτμηση μικρότερου κόστους. Η υπολογιστική πολυπλοκότητα μπορεί να περιορισθεί με την μετακίνηση ενός αντικειμένου κάθε φορά σε μία σειρά μετακινήσεων. Ο αλγόριθμος της εξομοιωμένης ισχυροποίησης [KGV83] επιδέχεται κινήσεις που αυξάνουν το κόστος. Σε αντίθεση με άλλους αλγορίθμους, η εξομοιωμένη ισχυροποίηση μπορεί να μετακινήσει ένα αντικείμενο παραπάνω από μία φορά, περιορίζοντας την πολυπλοκότητα μειώνοντας κατά την πάροδο του χρόνου την ανοχή για την αποδοχή κινήσεων που προκαλούν αύξηση του κόστους. Ο αλγόριθμος

σκοπεύει στο να μοντελοποιήσει τη διεργασία ισχυροποίησης στη φυσική, όπου ένα υλικό λιώνει και η ελάχιστη ενέργεια κατάστασης επιτυγχάνεται με την μείωση της θερμοκρασίας του τόσο αργά ώστε σε κάθε θερμοκρασία το υλικό να επανέρχεται σε κατάσταση ισορροπίας.

Ο αλγόριθμος ξεκινά με μία πρώτη κατάτμηση και με μία εξομοιωμένη θερμοκρασία ενώ στη συνέχεια η θερμοκρασία μειώνεται με πολύ αργό ρυθμό. Για κάθε μία τιμή της θερμοκρασίας δημιουργούνται τυχαίες μετακινήσεις. Ο αλγόριθμος επιδέχεται κάθε κίνηση η οποία βελτιώνει το κόστος. Διαφορετικά, μπορεί πάλι να επιδεχθεί την μετακίνηση, αλλά αυτές οι αποδοχές γίνονται λιγότερο πιθανές στις χαμηλές θερμοκρασίες.

Ο αλγόριθμος που παρουσιάζεται παρακάτω περιγράφει λεπτομερώς τον αλγόριθμο της εξομοιωμένης ισχυροποίησης. Μία μεταβλητή $temp$ καθορίζει την τρέχουσα εξομοιωμένη θερμοκρασία. Μία διαδικασία $RandomMove$ δημιουργεί μία νέα κατάτμηση $P\_tentative$ επιλέγοντας με τυχαίο τρόπο ένα αντικείμενο της τρέχουσας κατάτμησης P και μετακινώντας αυτό το αντικείμενο από την μία ομάδα σε μία άλλη. Οι μεταβλητές $cost$ και $cost\_tentative$ αποθηκεύουν τα κόστη των τρεχόντων και το δοκιμαστικών κατατμήσεων αντιστοίχως. Μία μεταβλητή $\Delta cost$ αποθηκεύει την (πιθανά αρνητική) βελτίωση του κόστους μεατξύ της δοκιμαστικής κατάτμησης και της τρέχουσας κατάτμησης. Η διαδικασία $Accept$ προσδιορίζει αν θα γίνει αποδεκτή μία μετακίνηση η οποία βασίζεται στην βελτίωση του κόστους και την τρέχουσα θερμοκρασία και καθορίζεται στο [KGV83] ώς:

$$Accept(\Delta cost, temp) = min(1, e^{-\frac{\Delta cost}{temp}}) \tag{3.3}$$

Όταν το $\Delta cost$ είναι αρνητικό, αυτό σημαίνει ότι η δοκιμαστική κατάτμηση είναι καλύτερη από την τρέχουσα και η συνάρτηση $Accept$ επιστρέφει την τιμή 1. Διαφορετικά, επιστρέφει μία τιμή η οποία διακυμαίνεται στο [0,1]. Η διαδικασία $Random(0, 1)$ επιστρέφει μία τυχαία τιμή κυμαινόμενη μεταξύ του

[0,1]. Η διαδικασία *Equilibrium* προσδιορίζει αν η διαδικασία κατάτμησης για την τρέχουσα τιμή της *temp* έχει φτάσει στο σημείο ισορροπίας, η οποία μπορεί να προσεγγιστεί ως μη βελτίωση για έναν αριθμό από επαναλήψεις. Η διαδικασία *DecreaseTemp* μειώνει τη θερμοκρασία αφού έχει φτάσει στο σημείο ισορροπίας και συνήθως ορίζεται ως ακολούθως, όπου $0 < \alpha < 1$: $temp\_new = \alpha \times temp\_old$. Η διαδικασία *Frozen* προσδιορίζει αν έχει επιτευχθεί η μικρότερη τιμή της *temp*, η οποία είναι συνήθως μηδέν, το οποίο και σημαίνει ότι ο αλγόριθμος θα τερματισθεί.

Αλγόριθμος εξομοιωμένης ισχυροποίησης

```
temp = initial temperature
cost = Objfct(P)
while not Frozen loop
   while not Equilibrium loop
      P_tentative = RandomMove(P)
      cost_tentative = Objfct(P_tentative)
      DeltaCost = cost_tentative - cost
      if (Accept(DeltaCost,temp) > Random(0,1)) then
         P = P_tentative
         cost = cost_tentative
      end if
   end loop
   temp = DecreaseTemp(temp)
end loop
```

Για μία συγκεκριμένη θερμοκρασία, προσπαθούμε να βελτιώσουμε την κατάτμηση με τη δημιουργία και πιθανόν την αποδοχή τυχαίων μετακινήσεων, μέχρις ότου φτάσουμε στο σημείο ισορροπίας. Μία μετακίνηση γίνεται αποδεκτή αν η τιμή της *Accept* είναι μεγαλύτερη από μία τυχαία τιμή μεταξύ του

0 και του 1. Εφόσον η *Accept* επιστρέφει 1 αν μία κίνηση βελτιώνει το κόστος, τέτοιες μετακινήσεις είναι πάντα αποδεκτές. Αφού επιτευχθεί το σημείο ισορροπίας, η θερμοκρασία μειώνεται και γίνεται νέα προσπάθεια για βελτίωση μέχρις ότου επιτευχθεί η χαμηλότερη δυνατή θερμοκρασία.

Θεωρητικές μελέτες [RSV85], [Len83] έχουν αποδείξει ότι ο αλγόριθμος της εξομοιωμένης ισχυροποίησης μπορεί να διαφύγει από ένα τοπικό ελάχιστο και να βρει την γενικά βέλτιστη λύση αν η διεργασία φτάνει στην κατάσταση ισορροπίας σε κάθε θερμοκρασία και αν η θερμοκρασία ελαττώνεται απείρως αργά. Οι παραπάνω συνθήκες απαιτούν ένα άπειρο αριθμό επαναλήψεων σε ένα άπειρο αριθμό θερμοκρασιών, το οποίο είναι φυσικά μη πρακτικό και συνεπώς πολλές λύσεις σε αυτό το θέμα έχουν προταθεί [OvG84], [HRSV83] για να ελέγξουν την διαδικασία της προσομοιωμένης ισχυροποίησης. Η πολυπλοκότητα αυτών των αλγορίθμων εξαρτάται από την μορφή των διαδικασιών $Accept, Equilibrium, DecreaseTemp$ και $Frozen$ που χρησιμοποιούνται στον αλγόριθμο. Συνεπώς, η πολυπλοκότητα μπορεί να είναι μεταβαλλόμενη. Γενικότερα, η εξομοιωμένη ισχυροποίηση παράγει συνήθως καλά αποτελέσματα αλλά υποφέρει από τους πολύ μεγάλους χρόνους που απαιτούνται κατά την εκτέλεσή τους.

3.4.4 Γενετικής εξέλιξης

Ο αλγόριθμος της εξομοιωμένης ισχυροποίησης βελτιώνει μία κατάτμηση με την μετακίνηση ενός αριθμού αντικειμένων, αποθηκεύοντας την καλύτερη κατάτμηση που συναντήθηκε και επαναλαμβάνοντας τη διεργασία με την καλύτερη δυνατή κατάτμηση. Ωστόσο, είναι ξεκάθαρο ότι δεν πρέπει να υπάρχει η δέσμευση για αποθήκευση μίας μόνο κατάτμησης από την μία επανάληψη στην επόμενη.

Μία ομάδα αλγορίθμων η οποία αποθηκεύει ένα σύνολο κατατμήσεων με-

ταξύ των επαναλήψεων, μοντελοποιήθηκε μετά τη διεργασία της γενετικής εξέλιξης. Σε τέτοιου είδους αλγορίθμους, ένα σύνολο από κατατμήσεις αναφέρεται και ως μία γενιά. Οι αλγόριθμοι γενετικής εξέλιξης δημιουργούν μία νέα γενιά από την τρέχουσα, απομιμούμενοι τρεις εξελικτικές μεθόδους οι οποίες συναντιούνται στη φύση. Μία μέθοδος είναι η **επιλογή** η οποία επιλέγει τυχαία μία χαμηλού κόστους κατάτμηση P και την αντιγράφει στην επόμενη γενιά. Με πιο απλά λόγια κάποια "ισχυρά" μέλη της γενιάς επιβιώνουν και στην επόμενη γενιά. Μία δεύτερη μέθοδος είναι η **γενετική διασταύρωση** η οποία επιλέγει τυχαία δύο ισχυρές κατατμήσεις P_a και P_b και αντιγράφει πιστά τα κληρονομημένα ιδιαίτερα χαρακτηριστικά του ενός στο άλλο, όπως η ομάδα p_i στο P_a. Με άλλα λόγια, κάποια ισχυρά στοιχεία θα αναμιχθούν μεταξύ τους και θα επιβιώσουν και στην επόμενη γενιά. Μία τρίτη μέθοδος είναι η **μετάλλαξη**, η οποία επιλέγει τυχαία μία κατάτμηση και την μετατρέπει μετακινώντας κάποια τυχαίως επιλεγμένα αντικείμενα μεταξύ ομάδων.

Ο παρακάτω αλγόριθμος, περιγράφει λεπτομερώς έναν γενετικό αλγόριθμο κατάτμησης ο οποίος αποτελεί μία παραλλαγή του αλγορίθμου που παρουσιάζεται στο [KV93]. Μία διαδικασία $CreateRandomPart(O)$ επιστρέφει μία τυχαία κατάτμηση από ένα δεδομένο σύνολο αντικειμένων O. Η διαδικασία $Select(G, num\_sel)$ επιστρέφει num_sel κατατμήσεις οι οποίες δημιουργήθηκαν από την εφαρμογή επιλογής στην γενιά G. Η διαδικασία $Cross(G, num\_cross)$ επιστρέφει κατατμήσεις οι οποίες δημιουργούνται από την εφαρμογή γενετικών διασταυρώσεων στη γενιά G. Η διαδικασία $Mutate(G, num\_mutate)$ μεταλλάσει κατατμήσεις στη γενιά G. Τα num_sel, num_cross, num_mutate αποτελούν εισόδους στον αλγόριθμο. Η διαδικασία $BestPart(G)$ επιστρέφει την χαμηλότερου κόστους κατάτμηση στη γενιά G. Η διαδικασία $Terminate$ επιστρέφει αληθής όταν ικανοποιηθεί μία συνθήκη τερματισμού, που σημαίνει ότι ο αλγόριθμος πρέπει να σταματήσει, μία κοινή συνθήκη είναι ότι η καλύτε-

ρη κατάτμηση έχει επιβιώσει από έναν δεδομένο αριθμό από κατατμήσεις. Η μεταβλητή P_best αναπαριστά την κατάτμηση χαμηλότερου κόστους που συναντήθηκε κατά την όλη διαδικασία. Η είσοδος gen_size καθορίζει τον αριθμό των κατατμήσεων που θα εμφανιστούν στην πρώτη γενιά.

```
/* Create first generation with gen_size random partitions */
G = NULL
for i in 1 to gen_size loop
   G = G cap CreateRandomPart(O)
end loop
P_best = BestPart(G)

/* Evolve generation */
while not Terminate loop
   G = Select(G,num_sel) cap Cross(G,num_cross)
   Mutate(G,num_mutate)
   if Objfct(BestPart(G)) < Objfct(P_best) then
      P_best = BestPart(G)
   end if
end loop

/* Return best partition in final generation */
return P_best
```

Πρώτα δημιουργούμε gen_size τυχαίες κατατμήσεις για να σχηματίσουμε την πρώτη γενιά. Στη συνέχεια δημιουργούμε μία νέα γενιά χρησιμοποιώντας την επιλογή, την διασταύρωση και την μετάλλαξη και επαναλαμβάνεται μέχρις ότου ικανοποιηθεί μία από τις συνθήκες τερματισμού. Επιστρέφουμε την καλύτερη κατάτμηση που παρουσιάστηκε καθ'όλη την διαδικασία.

Η πολυπλοκότητα του γενετικού αλγορίθμου επηρεάζεται άμεσα από την μορφή της διαδικασίας τερματισμού $Terminate$. Όπως και στην εξομοιωμένη ισχυροποίηση, οι γενετικοί αλγόριθμοι συνήθως παράγουν καλά αποτελέσματα αλλά υποφέρουν από τον πολύ μεγάλο χρόνο επεξεργασίας. Επίσης, εφόσον οι γενετικοί αλγόριθμοι συγκρατούν περισσότερες από μία κατατμήσεις, απαιτούν και μεγαλύτερα ποσά μνήμης.

3.5 Αλγόριθμοι κατάτμησης υλικού/λογισμικού

Οι υλοποιήσεις συνδυασμού υλικού/λογισμικού είναι πολύ κοινές στα ενσωματωμένα συστήματα. Το λογισμικό που εκτελείται σε έναν εξωτερικό επεξεργαστή είναι λιγότερο δαπανηρό, πολύ πιο εύκολα τροποποιήσιμο και πιο γρήγορο στη σχεδίασή του από ότι μία ισοδύναμη υλοποίηση σε ένα υλικό συγκεκριμένης λειτουργίας. Ωστόσο η υλοποίηση σε υλικό μπορεί να παρέχει καλύτερη απόδοση. Ο στόχος ενός σχεδιαστή ενός συστήματος είναι να υλοποιήσει ένα σύστημα χρησιμοποιώντας το ελάχιστο ποσό υλικού για συγκεκριμένες εφαρμογές έτσι ώστε να ικανοποιεί την απαιτούμενη απόδοση. Πιο απλά ο σχεδιαστής πρέπει να προσπαθεί να υλοποιήσει όσον το δυνατόν περισσότερο μέρος της λειτουργικότητας του συστήματος σε λογισμικό.

Το πρόβλημα της κατάτμησης υλικού/λογισμικού αποτελεί ένα μέρος από το γενικό πρόβλημα της κατάτμησης. Ένα από τα ειδικά χαρακτηριστικά της είναι ότι αποτελεί μία κατάτμηση διπλής κατεύθυνσης εφόσον εμπλέκεται το υλικό και το λογισμικό. Πιο συγκεκριμένα υπάρχουν δύο είδη μέτρων εκ των οποίων το ένα (η απόδοση), μπορεί να βελτιωθεί μετακινώντας αντικείμενα σε ένα συγκεκριμένο σύνολο (υλικό), ενώ το δεύτερο (μέγεθος υλικού) μπορεί να βελτιωθεί με την μετακίνηση των αντικειμένων εκτός του ιδίου συνόλου. Αυτά τα συγκεκριμένα χαρακτηριστικά έχουν οδηγήσει στη δημιουργία ειδικευμένων

αλγορίθμων για την κατάτμηση υλικού/λογισμικού. Σε αυτό το κεφάλαιο περιγράφονται οι τρεις πιο σημαντικοί αλγόριθμοι οι οποίοι προτείνονται ειδικά για την επίλυση του προβλήματος κατάτμησης υλικού/λογισμικού.

3.5.1 Πλεονεκτικοί αλγόριθμοι

Ένας απλός και γρήγορος αλγόριθμος ξεκινά με μία αρχική κατάτμηση και μετακινεί αντικείμενα στο αντίθετο σύνολο από αυτό που βρίσκονταν αρχικά μέχρις ότου επιτευχθεί κάποια βελτίωση. Ένας τέτοιος αλγόριθμος παρουσιάζεται παρακάτω. Χρησιμοποιεί μία διαδικασία η οποία ονομάζεται $Move(P, o_i)$ η οποία επιστρέφει μία νέα κατάτμηση P', η οποία επιτυγχάνεται από την μετακίνηση του o_i σε λογισμικό αν ήδη βρίσκεται σε υλικό και το αντίστροφο.

Πλεονεκτικός αλγόριθμος μετακίνησης

```
repeat
    P_orig = P
    for i in 1 to n loop
        if Objfct(Move(P,o(i)) < Objfct(P) then
            P = Move(P,o(i))
        end if
    end loop
until P = P_orig
```

Ο παρακάτω αλγόριθμος παρουσιάζει τον πλεονεκτικό αλγόριθμο του [GD92] ο οποίος είναι παρόμοιος με τον παραπάνω με μία επέκταση ώστε να επιβεβαιώνει ότι τηρούνται οι περιορισμοί που τίθενται στην απόδοση ενός συστήματος. Ο αλγόριθμος χρησιμοποιεί μία διαδικασία που ονομάζεται $Successors(o_i)$ η οποία επιστρέφει ένα σύνολο από αντικείμενα τα οποία έχουν επιτύχει o_i στο εσωτερικό μοντέλο της λειτουργικότητας του συστήματος. Η διαδικασία

$StatisfiesPerformance(P)$ επιστρέφει ως αληθής αν η κατάτμηση P ικανοποιεί όλους τους περιορισμούς της απόδοσης.

Αλγόριθμος Vulcan II

```
P = {0} /* all-hardware initial partition */
repeat
    P_orig = P
    for each o(i) belongs to hardware loop
        AttemptMove(P,o(i))
            P = Move(P,o(i))
        end loop
until P = P_orig

procedure AttemptMove(P,o(i))
    if SatisfiesPerformance(Move(P,o(i)))
        and Objfct(Move(P,o(i))) < Objfct(P) then
      P = Move(P,o(i))
      for each o(j) belongs to Successors(o(i)) loop
        AttemptMove(P,o(ji))
      end loop
    end if
```

Ο αλγόριθμος ξεκινά δημιουργώντας μία κατάτμηση αποκλειστικά σε υλικό έτσι ώστε να εγγυάται ότι μία κατάτμηση η οποία είναι ικανοποιητική ως προς την απόδοση, έχει βρεθεί αν φυσικά μπορεί να υπάρξει. Στην πραγματικότητα βέβαια, ορισμένες συμπεριφορές οι οποίες δεν επηρεάζουν την απόδοση του συστήματος μεταφέρονται και υλοποιούνται σε λογισμικό. Μία ικανοποιητικής απόδοσης κατάτμηση είναι αυτή κατά την οποία ικανοποιούνται όλοι οι περιορισμοί κατάτμησης. Για να μετακινηθεί ένα αντικείμενο, ο αλγόριθμος αυτός

απαιτεί όχι μόνο βελτίωση του κόστους αλλά ταυτόχρονα και την ικανοποίηση όλων των περιορισμών που έχουν τεθεί. Αν μετακινηθεί μία συμπεριφορά, ο αλγόριθμος προσπαθεί να μετακινήσει εγγύτερα τα συσχετιζόμενα αντικείμενα. Ενώ οι πλεονεκτικοί αλγόριθμοι είναι αρκετά γρήγοροι, το μεγάλο τους μειονέκτημα είναι ότι δεν μπορούν να διαφύγουν από ένα τοπικό ελάχιστο στο οποίο μπορεί να βρεθεί η συνάρτηση.

3.5.2 Αλγόριθμοι hill-climbing

Για να ξεπεραστούν οι περιορισμοί των πλεονεκτικών αλγορίθμων, έχουν προταθεί διάφοροι άλλοι αλγόριθμοι βασιζόμενοι σε έναν ήδη υπάρχον hill-climbing αλγόριθμο όπως είναι και αυτός της πλαστής ισχυροποίησης (simulated annealing). Ένας τέτοιος αλγόριθμος δέχεται και μερικές αρνητικές κινήσεις έτσι ώστε να μπορεί να ξεπεράσει πολλά τοπικά ελάχιστα. Για την εφαρμογή του απαιτείται μία αρχική κατάτμηση.

Στο [EHB94] περιγράφεται μία προσέγγιση η οποία χρησιμοποιεί μία υλοποίηση εξ΄ ολοκλήρου σε λογισμικό για την αρχική κατάτμηση. Στη συνέχεια χρησιμοποιείται ένας αλγόριθμος κατάτμησης hill-climbing για την εξαγωγή των αντικειμένων από το λογισμικό και την μετακίνησή τους σε υλικό ώστε να ικανοποιηθούν οι περιορισμοί της απόδοσης. Μία τέτοια εξαγωγή μπορεί να οδηγήσει σε λιγότερο υλικό από ότι η προηγούμενη προσέγγιση, όπου τα αντικείμενα εξάγονται προς την άλλη κατεύθυνση, από το υλικό σε λογισμικό.

Συνάρτηση κόστους

Ας θεωρήσουμε τώρα την επινόηση μίας συνάρτησης κόστους η οποία χρησιμοποιείται από τον αλγόριθμο κατάτμησης hill-climbing. Το πρόβλημα μεταφέρεται στην προσπάθεια εξισορρόπησης σε μία συνάρτηση τους στόχους της ικανοποίησης των περιορισμών της απόδοσης καθώς και της ελαχιστοποίησης του υλικού. Ο αλγόριθμος αυτός αποφεύγει αυτό το πρόβλημα με την αφαίρε-

ση των περιορισμών της απόδοσης από την συνάρτηση κόστους· ο αλγόριθμος απλά απορρίπτει όλους τους αλγορίθμους οι οποίοι δεν ικανοποιούν την απόδοση του συστήματος. Ωστόσο παρατηρήθηκε ότι αυτή η προσέγγιση μπορεί να παγιδευτεί πολύ εύκολα σε ένα τοπικό ελάχιστο. Μερικές προσεγγίσεις hill-climbing αποφεύγουν το πρόβλημα με την αφαίρεση του μεγέθους του υλικού. Αυτή η προσέγγιση έχουν τον περιορισμό της απαίτησης από τον σχεδιαστή να δοκιμάσει πολλαπλά μεγέθη υλικού χειροκίνητα, με το να επανεφαρμόζουν την κατάτμηση κάθε φορά, έτσι ώστε να δοκιμάζουν να βρουν το μικρότερο μέγεθος υλικού ώστε να αποδοθεί μία κατάτμηση ικανοποίησης της απόδοσης.

Μία τρίτη λύση είναι η χρήση μίας συνάρτησης κόστους με δύο όρους, ώστε η μία να υποδεικνύει το άθροισμα όλων των παραβιάσεων της απόδοσης και η άλλη το μέγεθος του υλικού. Ο όρος της απόδοσης έχει μεγάλη βαρύτητα ώστε να επιβεβαιώσει ότι μία λύση ικανοποιητική ως προς την απόδοση έχει επιτευχθεί. Η συνάρτηση κόστους είναι η παρακάτω:

$$Objfct(P) = k_{perf} \times \sum_{i=1}^{m} Violation(C_i) + k_{area} \times Size(hardware) \quad (3.4)$$

$Violation(C_i) = Performance(G_i) - V_i$ αν η διαφορά είναι μεγαλύτερη του μηδενός· διαφορετικά $Violation(C_i) = 0$. Επίσης, $k_{perf} >> k_{area}$, αλλά το k_{perf} $\delta\epsilon\nu$ θα πρέπει να είναι άπειρο, εφόσον τότε ο αλγόριθμος δεν θα μπορούσε να διακρίνει μία κατάτμηση η οποία ήδη θα ικανοποιούσε τους περιορισμούς από μία άλλη η οποία παραβιάζει τους εκάστοτε περιορισμούς. Αναφερόμαστε σε αυτήν την λύση ως αλγόριθμος PWHC (performance-weighted hill-climbing), ο οποίος παράγει εξαιρετικά αποτελέσματα όταν συγκρίνεται με τον κλασσικό αλγόριθμο hill-climbing.

3.5.3 Δυαδικοί αλγόριθμοι δυαδικής αναζήτησης

Εφόσον ενσωματώνονται οι μελέτες για την απόδοση και το μέγεθος του υλικού στην ίδια συνάρτηση κόστους, όπως γίνεται στον PWHC, και τείνουμε να οδηγηθούμε σε πολύ καλύτερα αποτελέσματα από ότι οι προηγούμενες προσεγγίσεις, μία τρίτη προσέγγιση μπορεί να οδηγήσει σε ακόμη λιγότερο υλικό. Αυτή η προσέγγιση εμπλέκει την αποσύζευξη σε ένα βαθμό, του προβλήματος της ικανοποίησης της απόδοσης από το πρόβλημα της ελαχιστοποίησης του υλικού. Το πρόβλημα της κατάτμησης υλικού/λογισμικού παρατηρείται ως μία διεργασία του προσδιορισμού του περιορισμού για μικρότερο μέγεθος υλικού για το οποίο ένας δεδομένος αλγόριθμος κατάτμησης να μπορεί να βρει μία λύση μηδενικού κόστους. Με άλλα λόγια, πρέπει να γίνει αναζήτηση για την πρώτη μηδενικού κόστους λύση στη διαδοχή των περιορισμών του μεγέθους του υλικού μεταξύ του μηδενός και του μεγέθους του υλικού για μία υλοποίηση αποκλειστικά σε υλικό ($AllHwSize$). Η πρώτη μηδενικού κόστους λύση που θα βρεθεί κατά την αναζήτηση θα είναι κοντά στο ελάχιστο μέγεθος υλικού. Είναι γνωστό ότι το πρόβλημα της αναζήτησης μίας σειράς από στοιχεία μπορεί να λυθεί πολύ αποδοτικά μέσω της χρήσης της δυαδικής αναζήτησης.

Στη συνέχεια περιγράφεται ο αλγόριθμος κατάτμησης υλικού λογισμικού ο οποίος βασίζεται στη δυαδική αναζήτηση της σειράς από κόστη για το εύρος των πιθανών περιορισμών υλικό, ο οποίος αναφέρεται και ως αλγόριθμος BCS (binary constraint-search). Ο αλγόριθμος χρησιμοποιεί τις μεταβλητές low και $high$ για να υποδεικνύεται το τρέχον παράθυρο των πιθανών περιορισμών στο οποίο βρίσκεται ένας περιορισμός μηδενικού κόστους και η μεταβλητή mid για να αναπαριστά το μέσον του παραθύρου. Μία άλλη μεταβλητή P_zero, αποθηκεύει την κατάτμηση μηδενικού κόστους η οποία έχει το μικρότερο περιορισμό υλικού που έχει συναντηθεί ως εκείνη τη στιγμή. Η διαδικασία $PartAlg$ αναπαριστά έναν επαναληπτικό αλγόριθμο κατάτμησης όπως είναι η πλαστή

ισχυροποίηση.

Αλγόριθμος δυαδικής αναζήτησης BCS για κατάτμηση

```
low = 0, high = AllHwSize
while low < high loop
  mid = (low+high+1)/2
  P' = PartAlg(P,C,mid,Cost())
  if Cost(P',C,mid) = 0 then
    high = mid - 1
    P_zero = P'
  else
    low = mid
  end if
end loop
return P_zero
```

Ο αλγόριθμος πραγματοποιεί μία δυαδική αναζήτηση μέσω του εύρους των πιθανών περιορισμών, εφαρμόζοντας την κατάτμηση και στη συνέχεια επικαλείται η συνάρτηση κόστους του κάθε περιορισμού. Ο αλγόριθμος αυτός μοιάζει πάρα πολύ με τον κλασσικό αλγόριθμο δυαδικής αναζήτησης με δύο όμως τροποποιήσεις. Πρώτον, η μεταβλητή mid χρησιμοποιείται ως περιορισμός του υλικού για κατάτμηση της οποίας το αποτέλεσμα χρησιμοποιείται στη συνέχεια για τον προσδιορισμό του κόστους, σε αντίθεση με τη χρήση της μεταβλητής mid ως ένδειξη σε ένα στοιχείο ενός πίνακα. Δεύτερον, το κόστος συγκρίνεται με το μηδέν σε αντίθεση με ένα στοιχείο πίνακα το οποίο συγκρίνεται με ένα κλειδί.

Κεφάλαιο 4

Ο επεξεργαστής δικτύων PRO^3

Ένα περιβάλλον κεντρικής αποθήκευσης με πολύ διαφορετικές απαιτήσεις από αυτά που παρουσιάστηκαν στα προηγούμενα κεφάλαια περιγράφεται και αναλύεται σε αυτό το κεφάλαιο. Πρόκειται για ένα σύγχρονο, πρότυπο σύστημα ενός επεξεργαστή πρωτοκόλλων που υλοποιείται σε ένα μόνο στοιχείο υψηλής ολοκλήρωσης (System-On-Chip) με στόχο την παροχή υψηλής διεκπεραιωτικής ικανότητας και την επιτάχυνση της εκτέλεσης τηλεπικοινωνιακών πρωτοκόλλων διαφορετικών στοιβών σε έναν τηλεπικοινωνιακό κόσμο που οι χωρητικότητες και οι ταχύτητες μετάδοσης αυξάνονται ραγδαία μεταθέτοντας το πρόβλημα στην επεξεργασία των δεδομένων. Το σύστημα αυτό αποτελείται από μονάδες επεξεργασίας σχεδιασμένες με ειδικές τεχνικές προκειμένου να παρέχουν την απαιτούμενη υπολογιστική ισχύ ενώ το ενδιαφέρον μας εστιάζεται στην προσπέλαση σε αυτές, η οποία πρέπει να λαμβάνει υπόψη τις σχεδιαστικές και λειτουργικές τους παραμέτρους, να μεγιστοποιεί τη χρησιμοποίησή τους παρέχοντας ταυτόχρονα διαφορετικά επίπεδα εξυπηρέτησης. Ένα εξίσου ενδιαφέρον ζήτημα που αντιμετωπίζουμε είναι η προσπέλαση στη ζεύξη εξόδου, η οποία διέπεται από κανόνες μορφοποίησης της κίνησης και όχι μεγιστοποίησης της χρησιμοποίησης. Κριτήριο για την επιλογή των αλγορίθμων προσπέλασης

ή ορθότερα του προγραμματισμού (scheduling), όπως συχνά θα αναφέρεται, δεν είναι μόνο η ικανοποίηση των λειτουργικών απαιτήσεων αλλά και η μείωση του κόστους υλοποίησης. Στο κεφάλαιο λοιπόν αυτό θα προσδιορισθούν οι λειτουργικές παράμετροι και θα διερευνηθούν οι απαιτήσεις διαφορετικών δομών δεδομένων, οι οποίες θα λειάνουν το έδαφος για τη διατύπωση των αλγορίθμων και την περιγραφή της αρχιτεκτονικής υλοποίησης που θα παρατεθούν στη συνέχεια της παρούσας διατριβής.

4.1 Περιγραφή του ολοκληρωμένου κυκλώματος PRO^3

Το σύστημα του PRO^3 αποτελεί μία προσέγγιση υβριδικής αρχιτεκτονικής στις απαιτήσεις των επεξεργαστών πρωτοκόλλων. Το PRO^3 στοχεύει στην στενή σχέση που μπορεί να υπάρξει μεταξύ του λογισμικού και του υλικού για την αποδοτική εκτέλεση των τηλεπικοινωνιακών πρωτοκόλλων σε ενσωματωμένες και προγραμματιζόμενες αρχιτεκτονικές.

Το σύστημα ολοκληρωμένου κυκλώματος PRO^3 αποτελεί έναν επεξεργαστή πολλαπλών πρωτοκόλλων ο οποίος επιταχύνει την εκτέλεση τηλεπικοινωνιακών και μεταφοράς δεδομένων πρωτοκόλλων κάνοντας χρήση ενός υψηλής απόδοσης πυρήνα τύπου RISC και επαναπρογραμματιζόμενο υλικό μέσω τεχνικών pipeline. Οι απαιτητικές συναρτήσεις πραγματικού χρόνου των πρωτοκόλλων σε επεξεργαστική ισχύ, χειρίζονται από το επαναπρογραμματιζόμενο υλικό, ενώ οι υπόλοιπες συναρτήσεις καθώς επίσης και τα υψηλότερα επίπεδα του πρωτοκόλλου διαχειρίζονται από έναν RISC επεξεργαστή που βρίσκεται ενσωματωμένος στο σύστημα. Οι εφαρμογές όπου το PRO^3 μπορεί να χρησιμοποιηθεί ποικίλλουν από κάρτες δρομολογητών υψηλών ταχυτήτων μέχρι και διεπαφές μεγάλων εξυπηρετητών.

4.1.1 Βασικά χαρακτηριστικά

Η παρακάτω λίστα συνοψίζει τα βασικά χαρακτηριστικά του ολοκληρωμένου κυκλώματος PRO^3:

- Παρέχει μία διεπαφή εκπομπής και λήψης τύπου OC-48/UTOPIA-III έτσι ώστε να μπορεί να επεξεργάζεται δεδομένα ρυθμού 2.5Gbps.
- Υποστηρίζει δεδομένα σε επίπεδο cell και πακέτου, με αποτέλεσμα να μπορεί να χρησιμοποιηθεί τόσο για εφαρμογές τύπου IP όσο και για εφαρμογές ATM.
- Υποστηρίζει μέχρι και 500K ροές ή ομάδες ροών. Η κατηγοριοποίηση σε πραγματικό χρόνο πραγματοποιείται μέσω μίας εξωτερικής μονάδας μνήμης CAM.
- Εκτελεί υπηρεσίες ουρών ανά ροή για όλες τις ροές. Κάθε ροή μπορεί να οδηγηθεί με τις κατάλληλες ρυθμίσεις στην έξοδο προς το δίκτυο, στον επεξεργαστή ελέγχου ή στις εσωτερικές μονάδες επεξεργασίας.
- Υλοποιεί δύο επαναπρογραμματιζόμενους πυρήνες για επεξεργασία των μηνυμάτων του πρωτοκόλλου. Κάθε ένας από τους πυρήνες αυτούς αποτελείται από:
  - Έναν εξαγωγέα πεδίων για την επεξεργασία των επικεφαλίδων των πακέτων στην ταχύτητα άφιξης των πακέτων.
  - Έναν υψηλής απόδοσης 32-bit Hyperstone RISC επεξεργαστή ο οποίος ενισχύει την υποστήριξη για υψηλής ταχύτητας δεδομένα εισόδου και εξόδου.
  - Έναν τροποποιητή πεδίων για σύνθεση και μετατροπή των πακέτων στον ρυθμό με τον οποίο εισέρχονται από το δίκτυο.

- Για εφαρμογές βασιζόμενες σε cells (ATM) το ολοκληρωμένο σύστημα υποστηρίζει:
  - UTOPIA επιπέδου III.
  - Τα ATM SAR 5 και CPCS στρώματα σε υλικό τόσο για τον εκπομπό όσο και για τον δέκτη. Η αναγνώριση των καναλιών VP,VC μπορεί να γίνει για μέχρι 500K ροές.
  - Το CRC32 υπολογίζεται σε υλικό.
- Για εφαρμογές βασιζόμενες σε πακέτα το ολοκληρωμένο σύστημα υποστηρίζει:
  - Διεπαφή πακέτων OC-48 & POS-PHY.
  - Κατηγοριοποίηση για IP μέχρι 500K κανόνες βασιζόμενοι σε ένα προγραμματιζόμενο κλειδί μήκους 144 bit. Συγκεκριμένοι κανόνες ροών, γενικοί κανόνες αποδοχής ή απόρριψης μπορούν να υπάρχουν ταυτόχρονα στην μνήμη CAM.
  - Επαλήθευση πακέτων σε υλικό σε ταχύτητες ίσες με αυτές της γραμμής δεδομένων.
- Προγραμματιζόμενη χρονοδρομολόγηση των αποθηκευμένων πακέτων:
  - Εσωτερική χρονοδρομολόγηση των πακέτων (ή τμημάτων αυτών) προς τα στοιχεία της επεξεργασίας πρωτοκόλλων ή προς την εξωτερική επεξεργαστική μονάδα.
  - Χρονοδρομολόγηση κίνησης των δεδομένων προς την έξοδο του δικτύου βασιζόμενη σε κανόνες που καθορίζονται κάθε φορά από τον χρήστη.
- Διεπαφή εισαγωγής και εξαγωγής δεδομένων προς:

  - Έναν ενσωματωμένο 32-bit επεξεργαστή τύπου Hyperstone RISC/DSP CPU.
  - Έναν εξωτερικό επεξεργαστή.

- Μία διεπαφή για δοκιμές και απομάκρυνση λαθών των υπό ανάπτυξη εφαρμογών.

4.1.2 Εφαρμογές του συστήματος PRO^3

Οι αντικειμενικές λειτουργίες πρωτοκόλλων που μπορεί να διεκπεραιώσει το PRO^3 φτάνουν τόσο χαμηλά ώστε να πραγματοποιούν λειτουργίες σε επίπεδο bit και byte, όπως άλλωστε συμβαίνει και με τους κλασσικές αρχιτεκτονικές επεξεργασίας πρωτοκόλλων. Οι αρχιτεκτονικές αυτές χειρίζονται πακέτα μέσω της προσωρινής αποθήκευσης των πακέτων καθώς και μέσω πολλαπλών λειτουργιών αντιγραφής και ολίσθησης των δεδομένων.

Επιπλέον η μεικτή υποδομή των δικτύων φέρνει και νέες έννοιες εφόσον διαφορετικά πρωτόκολλα απαιτούν τον επαναπρογραμματισμό του συστήματος ώστε να μπορούν να αντιμετωπίζουν περιπτώσεις όπως:

- Πρωτόκολλα με μεταβλητού μήκους επικεφαλίδες.
- Έλεγχο αθροίσματος (checksum) σε τυχαίες θέσεις μέσα στα δεδομένα.
- Πεδία δεδομένων σε τυχαίες σειρές και συστοιχίες.

Οι απαιτήσεις για υποστήριξη πολλαπλών πρωτοκόλλων έχουν ήδη αρχίσει να γίνονται πραγματικότητα λόγω της τάσης των υψηλών ταχυτήτων των δικτύων τα οποία συμπεριλαμβάνουν λειτουργίες όπως:

- Προώθηση πακέτων χρησιμοποιώντας επεξεργασία σε όλα τα επίπεδα.

- Προχωρημένες δυνατότητες για ποιότητα υπηρεσιών, μέχρι και για επεξεργασία στο έβδομο επίπεδο, όταν πρόκειται για διαχείριση κίνησης και επιβολή πολιτικής στους χρήστες του δικτύου.

- Περίπλοκα πρωτόκολλα για έλεγχο, διαχείριση και ασφάλεια για όσον το δυνατόν μεγαλύτερο αριθμό συνδέσεων.

Η επεξεργασία δεδομένων στα υψηλά επίπεδα βασίζεται στην έννοια των *συνδέσεων* και απαιτείται αποθήκευση και διαχείριση της κατάστασης της εν λόγω πληροφορίας. Τα πρωτόκολλα των υψηλότερων επιπέδων όπως το TCP και η σηματοδοσία του ATM απαιτούν επίσης την υλοποίηση χρονομετρητών οι οποίοι μπορεί να αποδειχθούν ως ένα σημαντικό σημείο περιορισμού στην ταχύτητα επεξεργασίας. Η αντίστοιχη υλοποίηση απαιτεί προσεκτική βελτιστοποίηση είτε στην οργάνωση του λειτουργικού συστήματος είτε στην υλοποίηση των τμημάτων που υλοποιούνται αποκλειστικά σε υλικό. Μία άλλη συνέπεια στην επεξεργασία των υψηλότερων επιπέδων και του τερματισμού των εκάστοτε συνδέσεων είναι η απαίτηση για εκτέλεση πρωτοκόλλων πολλαπλών επιπέδων, τα οποία μπορεί να οδηγήσουν σε επαναλήψεις των συναρτήσεων και των λειτουργιών καθώς επίσης και πολλαπλές μετακινήσεις δεδομένων μεταξύ των διαφόρων λειτουργικών οντοτήτων. Το κόστος που εισέρχεται από αυτές τις αλληλεπιδράσεις μπορεί εύκολα να ξεπεράσει τις επεξεργαστικές απαιτήσεις για πραγματική επεξεργασία πρωτοκόλλων.

Ο γενικός περιορισμός των πόρων στην απόδοση των συστημάτων επικοινωνιών τα οποία εμπλέκονται στην επεξεργασία πρωτοκόλλων μπορεί γενικά να κατηγοριοποιηθεί σε:

- **Επεξεργαστική ισχύ και αρχιτεκτονική**. Η πραγματική απόδοση της κεντρικής μονάδας επεξεργασίας ή των ανεξάρτητων στοιχείων επεξεργασίας τα οποία χρησιμοποιούνται για την υλοποίηση των πρωτοκόλλων και

την αρχιτεκτονική τους όσον αφορά την εσωτερική τους δομή για γρήγορη εναλλαγή περιεχομένου, βελτιστοποιημένα στοιχεία για συγκεκριμένες λειτουργίες όπως είναι η αριθμητικές μονάδες κινητής υποδιαστολής ή τα DSPs, διεπαφές, ελεγκτές μνημών κτλ.

- **Δίαυλοι συστημάτων**. Το συνολικό throughput και οι μηχανισμοί επικοινωνιών, όπως οι καθυστερήσεις σε γέφυρες, οι δυνατότητες για συναλλαγές συνεχούς διοχέτευσης, κτλ.

- **Μνήμες**. Οι χρόνοι πρόσβασης, οι οποίοι επίσης εξαρτώνται από την εσωτερική δομή, την καθυστέρηση, την δυνατότητα για προσβάσεις συνεχούς διοχέτευσης, κτλ.

Για να είναι δυνατή η αντιμετώπιση των παραπάνω περιορισμών, η αρχιτεκτονική του PRO^3 παρέχει τις απαραίτητες μονάδες υλικού για την εκτέλεση των λειτουργιών για τα πρωτόκολλα που υποστηρίζονται, καθώς επίσης και τους μηχανισμούς διαχείρισης μνήμης και των σχετικών διεπαφών ώστε να επιτύχει την απαιτούμενη απόδοση για πακέτα δεδομένων και για τον έλεγχο της πληροφορίας της σύνδεσης. Στην περίπτωση των εφαρμογών οι οποίες περιλαμβάνουν παρακολούθηση της προόδου μίας σύνδεσης και εξαγωγή της πληροφορίας κατάστασης και υπολογισμών σε αυτήν, το μεγαλύτερο κόστος δεν σχετίζεται με την πραγματική επεξεργασία για την εκτέλεση των σχετικών αλγορίθμων, αλλά εξαρτάται κυρίως σε λειτουργίες όπως είναι η κατηγοριοποίηση των πακέτων, η αποθήκευση τους, και η λήψη των πακέτων. Εάν η επεξεργασία συνεχούς διοχέτευσης μπορεί να παρέχει την απαιτούμενη απόδοση, τότε οι περιορισμοί του επεξεργαστή RISC μπορεί να ξεπεραστεί και ο ο διαθέσιμος αριθμός εντολών ανά δευτερόλεπτο[1], θα είναι αρκετός για να υποστηρίξει 500K ροές, αφήνοντας περιθώριο για διάφορες εφαρμογές υψηλού επιπέδου να εκτελούνται

[1] Εκτιμάται σε 180 MIPS για την τεχνολογία των 0.18.

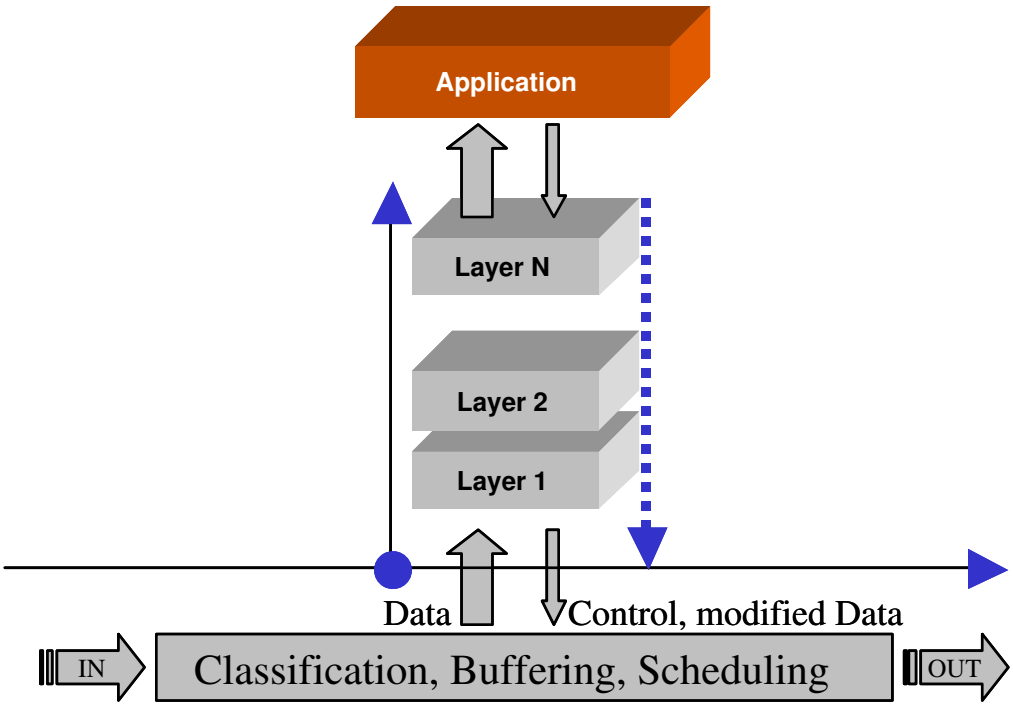
στο σύστημα του PRO^3.

Στην περίπτωση όπου εφαρμογές απαιτούν τον τερματισμό των συνδέσεων και την εκτέλεση πολλαπλών πρωτοκόλλων μέσω της στοίβας, όπως γίνεται στις εφαρμογές σηματοδοσίας του ATM, οι απαιτούμενες εντολές καθώς επίσης και η επικοινωνία μεταξύ των λειτουργικών οντοτήτων και στρωμάτων είναι πολύ λιγότερο απαιτητικές αφήνοντας έτσι μεγάλα περιθώρια για εκτέλεση και άλλων εφαρμογών στο σύστημα όπως είναι η διαχείριση των συνδέσεων, η παρακολούθηση, κτλ. Με γρήγορες υλοποιήσεις των χαμηλότερων επιπέδων των πρωτοκόλλων σε υλικό, είναι πλέον εύκολο να επιτευχθούν οι αποδόσεις οι οποίες έχουν τεθεί σαν απαιτήσεις του συστήματος.

Ένας σημαντικός πόρος επίσης θα είναι και το σημαντικό ποσό της διαθέσιμης μνήμης εφόσον η αποθήκευση της κατάστασης των συνδέσεων μπορεί να φτάσει να απαιτεί πολλά Megabytes από την μνήμη. Η δυνατότητα εισόδου και εξόδου των αντιστοίχων μονάδων διαχείρισης μνήμης αποτελεί το στοιχείο κλειδί για την επίτευξη των στόχων για την απόδοση του συστήματος. Επίσης, η επικοινωνιακές διεπαφές μεταξύ των διαφόρων στρωμάτων του πρωτοκόλλου και η διασυνδεσιμότητα με τον εξωτερικό επεξεργαστή, είναι τα στοιχεία που προσδιορίζουν την επιτευξιμότητα της απόδοσης για κατανεμημένες εφαρμογές οι οποίες απαιτούν υλοποιήσεις λογισμικού σε εξωτερικές επεξεργαστικές μονάδες.

Επεξεργασία σε ταχύτητα δικτύου για μη ταυτόχρονη αποστολή και λήψη

Σε αυτήν την λειτουργία του συστήματος υποτίθεται ότι το PRO^3 λειτουργεί σε έναν κόμβο του δικτύου στον οποίο λαμβάνεται κίνηση και επεξεργάζεται για να αποφασισθεί το πως θα διαχειρισθούν τα δεδομένα στη συνέχεια. Τέτοιου είδους εφαρμογές περιλαμβάνουν συστήματα ασφαλείας δικτύων ή εργαλεία

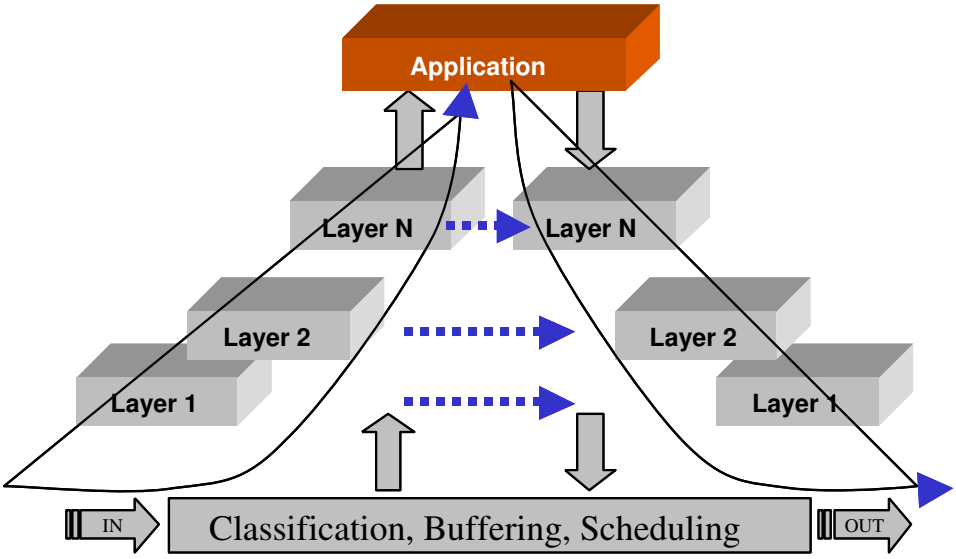


Σχήμα 4.1: Επεξεργασία πρωτοκόλλων σε μη ταυτόχρονη εκπομπή και λήψη

παρακολούθησης και διαχείρισης αυτών. Το Σχήμα 4.1 παρουσιάζει το μοντέλο αυτής της εφαρμογής.

Επεξεργασία σε ταχύτητα δικτύου για ταυτόχρονη αποστολή και λήψη

Σε αυτήν την λειτουργία του συστήματος θεωρείται ότι το PRO^3 λειτουργεί σε έναν κόμβο του δικτύου στον οποίο λαμβάνεται κίνηση και αφού επεξεργαστεί κατάλληλα προωθείται στον επόμενο κόμβο του δικτύου. Η επεξεργασία των πρωτοκόλλων που παρουσιάζεται πραγματοποιείται σε ένα πρωτόκολλο κάθε φορά. Κλασσικά παραδείγματα τέτοιων εφαρμογών είναι τα συστήματα διασυνδεσημότητας, οι δρομολογητές καθώς και τα τερματικά υψηλών ταχυ-



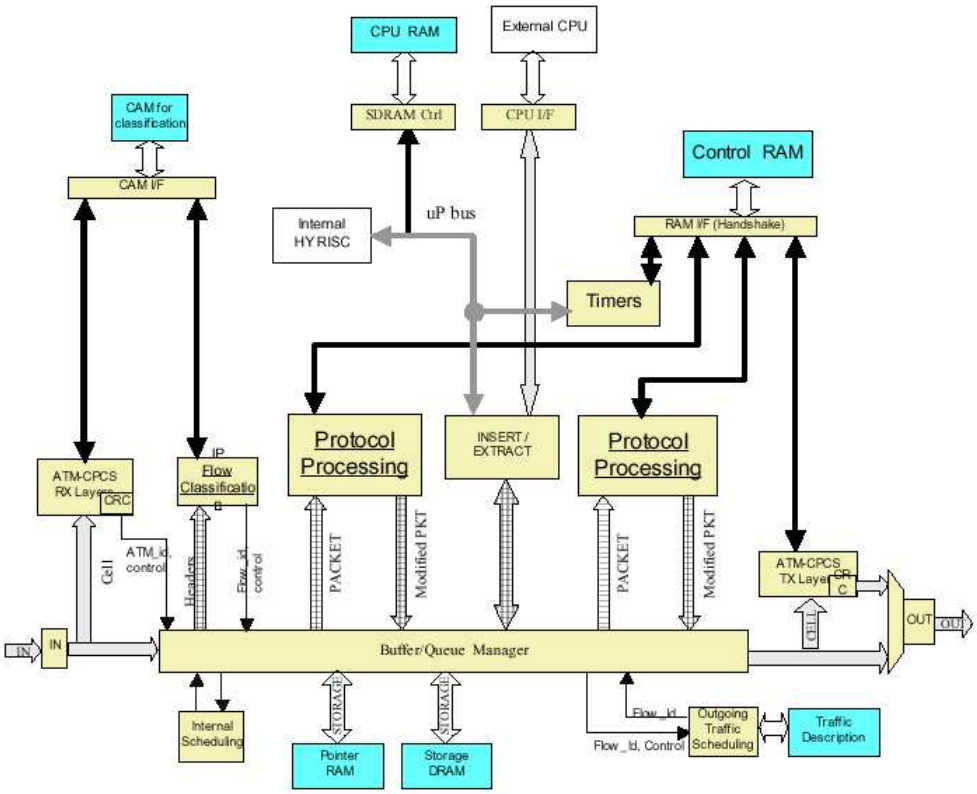
Σχήμα 4.2: Επεξεργασία πρωτοκόλλων σε ταυτόχρονη εκπομπή και λήψη

τήτων. Τα μοντέλα εφαρμογών αυτών των περιπτώσεων παρουσιάζονται στο Σχήμα 4.2.

4.1.3 Μπλοκ διάγραμμα του PRO^3

Όπως παρουσιάζεται στο Σχήμα 4.3, τα δεδομένα του δικτύου αποθηκεύονται προσωρινά στην μνήμη δεδομένων DRAM και στη συνέχεια προωθούνται είτε στο ένα από τα δύο μπλοκ των RPM είτε στην εσωτερική on-chip επεξεργαστική μονάδα RISC. Οι διαδρομές που ακολουθούν τα δεδομένα καθώς και η σειρά εκτέλεσης των λειτουργιών περιγράφονται στο κείμενο που ακολουθεί.

Συνοπτικά το PRO^3 αποτελείται από τα εξής στοιχεία του Σχήματος 4.3:
IN: Το μπλοκ αυτό λαμβάνει τα δεδομένα εισόδου και μπορεί να λειτουργεί είτε σε επίπεδο cell (UTOPIA III) είτε σε επίπεδο πακέτου (Packet over SONET). Στην περίπτωση του ATM η μονάδα του IN απλώς προωθεί τα cells ενώ στην άλλη λειτουργία πραγματοποιεί κατακερματισμός σε τμήματα. Το λαμβα-



Σχήμα 4.3: PRO^3 μπλοκ διάγραμμα

νόμενο πακέτο κατακερματίζεται σε τμήματα σταθερού μεγέθους έτσι ώστε να επιβοηθήσει την διαχειριστή της μνήμης δεδομένων.

ATM CPCS Rx (ACRX): Αυτό το μπλοκ πραγματοποιεί τις μηχανές περασμένης κατάστασης του ATM, του SAR5 και του CPCS πρωτοκόλλου. Εξάγει τα απαραίτητα πεδία για την κατηγοριοποίηση των ATM πακέτων και αποστέλλει στον διαχειριστή της μνήμης δεδομένων τον έλεγχο για την αποθήκευση και τον διαχωρισμό τους.

Διεπαφή CAM: Αυτό το μπλοκ υλοποιεί την διαδικασία της κατηγοριοποίησης των πακέτων χρησιμοποιώντας μία τριαδική μνήμη CAM. Είναι δυνατόν να υποστηρίζει έως και 500K ροές. Για την περίπτωση του ATM αποθηκεύει κάποιες πληροφορίες που σχετίζονται με τον έλεγχο των πρωτοκόλλων που χρησιμοποιούνται από το μπλοκ ACRX.

Διαχειριστής Μνήμης Δεδομένων (DMM): Αυτό αποτελεί ένα από τα σημαντικότερα μπλοκ του συστήματος. Εκτελεί την αποθήκευση ανά ροή των δεδομένων για 500K διαφορετικές ροές. Λειτουργεί σε επίπεδο cell και σε επίπεδο πακέτου έτσι ώστε να αποτελεί μία αποδοτική λύση και για τις δύο εφαρμογές που υποστηρίζει το σύστημα. Χρησιμοποιεί μνήμες DRAM για την αποθήκευση των δεδομένων και μνήμες SRAM για την αποθήκευση των δεικτών των πακέτων και των τμημάτων. Το μπλοκ του DMM εξυπηρετεί τέσσερις ξεχωριστές πόρτες παράλληλα εκ των οποίων μία για την λήψη των δεδομένων από το δίκτυο, μία για την έξοδο των δεδομένων προς το δίκτυο, μία για την λήψη δεδομένων από τον εσωτερικό δίαυλο και μία για την αποστολή δεδομένων προς τον εσωτερικό δίαυλο. Υποστηρίζει ταχύτητες έως και 10Gbps κατανέμοντάς τες δυναμικά στις τέσσερις διαφορετικές πόρτες. Τέλος διαθέτει και μία απευθείας διεπαφή με τους χρονοδρομολγητές του συστήματος από τους οποίους λαμβάνει πληροφορία ελέγχου για τα πακέτα τα οποία πρέπει να αποσταλλούν για επεξεργασία ή προς το δίκτυο.

Επαναπρογραμματιζόμενες μονάδες συνεχούς διοχέτευσης (RPM): Το RPM αποτελεί την κεντρική μονάδα επεξεργασίας του ολοκληρωμένου κυκλώματος. Επιτρέπει την επεξεργασία σε πραγματικό χρόνο των πακέτων και των μηνυμάτων των πρωτοκόλλων όταν παρουσιάζονται λειτουργίες απαλλαγμένες από λάθη. Το PRO^3 διαθέτει δύο τέτοιες μονάδες έτσι ώστε να είναι δυνατόν να επεξεργάζεται ταυτόχρονα τα πρωτόκολλα τόσο για την πλευρά του δέκτη όσο και για την πλευρά του πομπού. Επίσης ενσωματώνει έναν προγραμματιζόμενο εξαγωγέα πεδίων και έναν προγραμματιζόμενο τροποποιητή πακέτων για την κωδικοποίηση και την αποκωδικοποίηση των μηνυμάτων και των πακέτων. Επίσης διαθέτει μία απευθείας διεπαφή με την μνήμη ελέγχου του συστήματος έτσι ώστε να μπορεί να διαβάζει και να ανανεώνει το περιεχόμενο της κάθε μίας διεργασίας των μηχανών καταστάσεων. Τέλος διαθέτει επίσης μία διεπαφή μικροεπεξεργαστή για την αρχικοποίησή του και για το φόρτωμα του εκάστοτε προγράμματος σε αυτό.

Εισαγωγή/Εξαγωγή (INEX): Αυτό το μπλοκ παρέχει μία διεπαφή στην εσωτερική και στην εξωτερική επεξεργαστική μονάδα. Επίσης πραγματοποιεί κατακερματισμό και επανασύνθεση κατά την ανταλλαγή των πακέτων.

Χρονομετρητές (TIM): Μέσω αυτού του μπλοκ είναι διαθέσιμος ένας μεγάλος αριθμός ομάδων χρονομετρητών οι οποίοι χρειάζονται για την λειτουργία των χιλιάδων οντοτήτων των πρωτοκόλλων που υποστηρίζονται από το σύστημα. Γενικότερα, υποθέτουμε ότι οι χρονομετρητές λήγουν όταν συμβούν λάθη στο δίκτυο, οπότε και ο εσωτερικός ή ο εξωτερικός επεξεργαστής θα λάβει τα περισσότερα από αυτά τα γεγονότα λαθών.

Διαιτησία διαύλου (ARB): Αυτή η μονάδα υλοποιεί την διαιτησία μεταξύ των υπολοίπων μονάδων για την λειτουργικότητα και την πρόσβαση στον εσωτερικό δίαυλο.

Χρονοδρομολογητής διεργασιών (TSC): Αποτελεί ένα από τα σημαντι-

κότερα μπλοκ του συστήματος εφόσον ελέγχει και συντηρεί την λειτουργία ολόκληρου του ολοκληρωμένου κυκλώματος. Καθορίζει τη σειρά με την οποία τα πακέτα και τα μηνύματα πρόκειται να αποστέλλονται στις επεξεργαστικές μονάδες RPM για περαιτέρω επεξεργασία. Επίσης υποστηρίζει 32 διαφορετικές προτεραιότητες έτσι ώστε να διαφοροποιεί την εισερχόμενη κίνηση και πραγματοποιεί εσωτερική επεξεργασία σύμφωνα με τα χαρακτηριστικά της ποιότητας υπηρεσίας συγκεκριμένων ροών. Λόγω της εξαιρετικά στενής σχέσης αυτών των δύο στοιχείων σε πολλά σημεία του κειμένου και στα διάφορα σχήματα μπορεί να αναφέρεται και ως ένα μπλοκ με το όνομα "Ελεγκτής διαύλου, Εσωτερικός χρονοδρομολογητής".

Χρονοδρομολογητές κίνσης (TRS): Αυτό το μπλοκ χρονοδρομολογεί την εξερχόμενη κίνηση. Υποστηρίζει 32 διαφορετικές κλάσεις κίνησης και μπορεί να χρονοδρομολογήσει έως και 64K διαφορετικές ροές εξόδου οι οποίες συναθροίζονται στις προηγουμένως αναφερόμενες 32 προτεραιότητες. Τόσο ο χρονοδρομολγητής διεργασιών όσο και αυτός της κίνησης χρησιμοποιούν από κοινού μία μονάδα μνήμης για την αποθήκευση των καταστάσεων όλων των εσωτερικών ουρών του PRO^3. Πρέπει να τονισθεί ότι οι ροές οι οποίες ανήκουν σε μία προτεραιότητα εξυπηρετούνται κατά έναν κυκλικό τρόπο.

Διεπαφή μνήμης ελέγχου (CRIM): Μέσω αυτής της διεπαφής πραγματοποιείται ο έλεγχος της λειτουργίας της μνήμης ελέγχου η οποία απαιτείται για την αποθήκευση της πληροφορίας καταστάσεων των πακέτων που επεξεργάζονται από το PRO^3. Η μνήμη ελέγχου κρατάει το περιεχόμενο της πληροφορίας που σχετίζεται με κάθε μηχανή πεπερασμένης κατάστασης. Διαθέτει τέσσερις πόρτες μία προς το RPM1, μία προς το RPM2, μία προς το ATM-CPCS Tx και μία προς την μονάδα των χρονομετρητών. Για την υποστήριξη των δυνατοτήτων λειτουργίας ταυτόχρονης επεξεργασίας σε δεδομένα λήψης και εκπομπής, τότε το εν λόγω μπλοκ μπορεί να επικοινωνεί με ακόμη ένα ολοκληρωμένο σύστημα

PRO^3 έτσι ώστε να μοιράζονται το ίδιο περιεχόμενο.

Διεπαφή εξόδου (OUT): Το μπλοκ αυτό αποστέλλει την πληροφορία προς το δίκτυο. Μπορεί να λειτουργεί όπως και το IN τόσο σε επίπεδο cell (UTOPIA III) όσο και σε επίπεδο πακέτων (Packet over SONET).

Εσωτερικός επεξεργαστής ΗΥ (HY): Αποτελεί έναν ενσωματωμένο επεξεργαστή τύπου RISC ο οποίος χρησιμοποιείται από το σύστημα για την εκτέλεση γενικού λογισμικού. Το μπλοκ αυτό διαθέτει μία απευθείας διεπαφή προς μία εξωτερική μνήμη για την αποθήκευση του λογισμικού του και των δεδομένων.

Διεπαφή εξωτερικού επεξεργαστή (CIM): Μέσω αυτής της διεπαφής μπορεί να συνδεθεί ένας εξωτερικός επεξεργαστής με το υπόλοιπο σύστημα. Αυτός ο επεξεργαστής μπορεί να χρησιμοποιηθεί για την εκτέλεση λογισμικού όπως είναι ο έλεγχος των κλήσεων, επεξεργασία υψηλότερων στρωμάτων του πρωτοκόλλου και λειτουργίες αρχικοποίησης και ρύθμισης του PRO^3.

Μονάδα παρατήρησης και παρακολούθησης(DEB): Αυτό το μπλοκ παρέχει μία μέθοδο παρατηρισιμότητας και μέσα στο ολοκληρωμένο σύστημα και συγκεκριμένα στα δεδομένα τα οποία διακινούνται μέσω του εσωτερικού του διαύλου. Έχει τη δυνατότητα να προγραμματιστεί έτσι ώστε να χρησιμοποιεί κριτήρια σχετικά με φιλτράρισμα των δεδομένων σύμφωνα με διάφορες παραμέτρους που χρησιμοποιούνται από το εν λόγω σύστημα. Ο τελικός του σκοπός είναι να παρέχει στη διαδικασία της ανάπτυξης των εφαρμογών ευκολίες μέσω της παρατήρησης των εντολών που πραγματοποιούνται εσωτερικά στο ολοκληρωμένο κύκλωμα.

Εσωτερικός δίαυλος μικροεπεξεργαστή (IMB): Γενικά όλα τα στοιχεία και οι μονάδες του συστήματος είναι εφοδιασμένα με μία διεπαφή μικροεπεξεργαστή για την ρύθμιση και την αρχικοποίησή τους καθώς και για το φόρτωμα λογισμικού σε αυτά.

4.1.4 Ενσωματωμένοι IP πυρήνες

Εσωτερικός επεξεργαστής Hyperstone

Ο πυρήνας Hyperstone E1-32XSR χρησιμοποιείται για την εκτέλεση λογισμικού υψηλού επιπέδου καθώς επίσης για τον έλεγχο του ολοκληρωμένου κυκλώματος και την διαχείριση των εξαιρέσεων του πρωτοκόλλου. Τα βασικά χαρακτηριστικά αυτού του επεξεργαστή είναι τα παρακάτω:

- Πλήρης υλοποίηση της αρχιτεκτονικής του επεξεργαστή Hyperstone E1-32XSR RISC/DSP.
- 32 γενικούς και 64 τοπικούς καταχωρητές μήκους 32 bit, 16 γενικούς και 16 τοπικούς καταχωρητές απευθείας διευθηνσιοδοτούμενους.
- 16 KB πλήρης εσωτερική στατική μνήμη (IRAM).
- Mem0 (SDRAM) και Mem3 (EPROM/Flash) περιοχές μνήμης διαθέσιμες για χρήση εκτός του ολοκληρωμένου κυκλώματος και οι περιοχές μνήμης Mem1 και Mem2 διαθέσιμες για προσβάσεις αντιστοίχησης μνήμης εσωτερικά στο ολοκληρωμένο κύκλωμα.

Ο μικροεπεξεργαστής E1-32XSR RISC/DSP αποτελεί το κεντρικό στοιχείο ελέγχου για το σύστημα του PRO^3. Διασυνδέεται σε ένα εσωτερικό δίαυλο μικροεπεξεργαστή για την ρύθμιση και την αρχικοποίηση των άλλων μονάδων του συστήματος του PRO^3 καθώς και με την μονάδα εισαγωγής/εξαγωγής για να έχει πρόσβαση στα δεδομένα των πακέτων. Ο βασικός δίαυλος της εξωτερικής μνήμης του επεξεργαστή Hyperstone είναι διαθέσιμος εξωτερικά του PRO^3 για τη διασύνδεση μνήμης DRAM και FLASH ή ακόμη και για τη διασύνδεση του εξωτερικού εργαλείου επαληθευσιμότητας. Συνοψίζοντας, ο μικροεπεξεργαστής Hyperstone E1-32XSR είναι υπεύθυνος για τις παρακάτω εργασίες στο σύστημα του PRO^3:

- Κατά την εκκίνηση του συστήματος, ο μικροεπεξεργαστής Hyperstone E1-32XSR ξεκινά να φορτώνει τον κώδικα εκκίνησης από την περιοχή μνήμης Mem3 μέσω μίας εξωτερικής μνήμης τύπου FLASH, και εκτελεί το λειτουργικό σύστημα hyRTK (real-time kernel). Στη συνέχεια είτε ένα πρόγραμμα του χρήστη φορτώνεται από την FLASH μνήμη ή μπορεί να χρησιμοποιηθεί η λειτουργία παρακολούθησης του συστήματος μέσω ενός προσωπικού υπολογιστή ο οποίος εκτελεί το λογισμικό hydebug.

- Ο επεξεργαστής E1-32XSR αρχικοποιεί όλα τα στοιχεία του PRO^3 μέσω της διασύνδεσής του στο εσωτερικό δίαυλο του μικροεπεξεργαστή. Αυτό περιλαμβάνει την εγκατάσταση των ρυθμίσεων όλων των προγραμματιζόμενων μονάδων, όπως είναι ο εξαγωγές πεδίων ή ο μορφοποιητής πεδίων, καθώς και η αρχικοποίηση των τροποποιημένων Hyperstone RISC επεξεργαστών που βρίσκονται στα RPMs μέσω του φορτώματος εκτελέσιμου κώδικα στην μνήμη IRAM του MHY και της απελευθέρωσης των μπλοκ από την αρχική τους κατάσταση (reset).

- Κατά την φυσιολογική λειτουργία του συστήματος του PRO^3 ο επεξεργαστής E1-32XSR ειδοποιείται από τα RPMs ή από άλλα πιθανά μπλοκ για καταστάσεις εξαιρέσεων που μπορούν να συμβούν στο πρωτόκολλο, όπως για παράδειγμα καταστάσεις όπου κάποιοι χρονιστές μπορεί να λήξουν κατά τη λειτουργία του πρωτοκόλλου. Αυτές οι περιπτώσεις στη συνέχεια χειρίζονται από υψηλού επιπέδου λογισμικό επεξεργασία πρωτοκόλλων το οποίο εκτελείται στον E1-32XSR. Τέλος, ο επεξεργαστής μπορεί να έχει πρόσβαση στην μνήμη των πακέτων μέσω της μονάδας εισαγωγής και εξαγωγής.

Όλες οι λειτουργίες που περιγράφηκαν για τον E1-32XSR μπορούν επίσης να χειριστούν και από έναν εξωτερικό μικροεπεξεργαστή ο οποίος διασυνδέεται

στην διεπαφή εξωτερικού επεξεργαστή του PRO^3 συστήματος.

RISC CPU συνεχούς διοχέτευσης για επεξεργασία πρωτοκόλλων

Το στοιχείο επεξεργασίας για τις λειτουργίες της επεξεργασίας πρωτοκόλλων είναι βασικά ένας τροποιοποιημένος επεξεργαστής E1-32XSR RISC. Συγκρινόμενος με τον βασικό επεξεργαστή E1-32XSR έχουν πραγματοποιηθεί οι παρακάτω αλλαγές:

- Στο αρχείο καταχωρητών έχει προστεθεί μία επιπλέον θύρα για ανάγνωση και μία για εγγραφή. Μέσω της επιπλέον θύρας εγγραφής, η πληροφορία κατάστασης και πακέτων τοποθετείται στο αρχείο καταχωρητών από τον εξαγωγέα πεδίων, και μέσω της επιπλέον θύρας ανάγνωσης, τα ανανεωμένα πεδία της κατάστασης και των πακέτων διαβάζονται από την μονάδα του τροποιποιητή πεδίων.

- Το τοπικό τμήμα του αρχείου καταχωρητών έχει κατατμηθεί από 64 καταχωρητές σε 2x32 καταχωρητές, το γενικό τμήμα έχει διπλασιαστεί έτσι ώστε διαφορετική να μπορεί να επιτευχθεί διαφορετική κατάτμηση του γενικού τμήματος καταχωρητών. Η υποστηριζόμενη κατάτμηση του γενικού τμήματος των 16 καταχωρητών θα είναι 4 εναλλασσόμενοι συν 12 σταθεροί, 8 εναλλασσόμενοι συν 8 σταθεροί και 12 εναλλασσόμενοι συν 4 σταθεροί καταχωρητές.

- Έχει προστεθεί ένας μηχανισμός εναλλαγής των καταχωρητών ώστε να μπορεί να γίνει ανεξάρτητη επιλογή μεταξύ των τμημάτων των 32 τοπικών καταχωρητών και μεταξύ των τμημάτων των γενικών εναλλασσόμενων καταχωρητών.

- Έχει προστεθεί λογική συγχρονισμού για την έναρξη της επεξεργασίας όταν ένα νέο σύνολο από πληροφορία κατάστασης και πακέτων είναι δια-

θέσιμο στο αρχείο των καταχωρητών για επεξεργασία καθώς και για την σηματοδοσία της περάτωσης μίας τέτοιας επεξεργασίας.

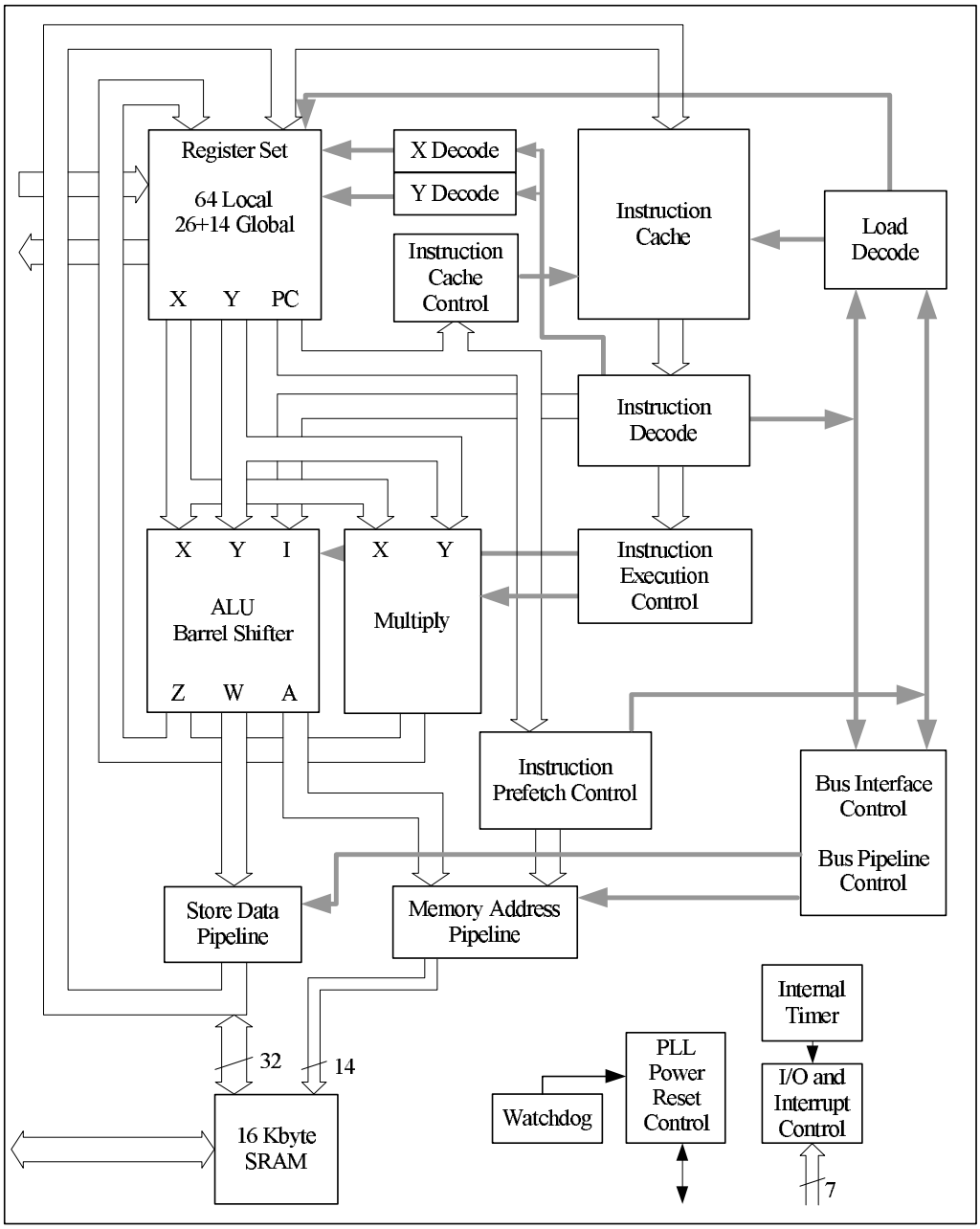
- Αφαιρέθηκε το τμήμα εντολών του επεξεργαστή ψηφιακών σημάτων (D-SP).

- Οι εντολές που ανανεώνουν τον καταχωρητή κατάστασης SR έχουν αλλάξει έτσι ώστε το πεδίο FP του καταχωρητή κατάστασης να μπορεί να ανανεώνεται απευθείας ώστε να υποστηρίζει γρήγορη μετεγκατάσταση των 16 ορατών στον χρήστη τοπικών επεξεργαστών.

- Σχεδόν όλη η λογική της διεπαφής του δίαυλου έχει αφαιρεθεί.

- Για το αρχικό φόρτωμα του κώδικα, έχει προστεθεί μία ξεχωριστή διαδρομή δεδομένων προς την εσωτερική μνήμη του τροποποιημένου RISC τη στιγμή που ο τροποποιημένος RISC βρίσκεται στην αρχική του κατάσταση και είναι ανενεργός. Μετά την αρχικοποίηση της εσωτερικής μνήμης, ο τροποποιημένος RISC ξεκινά την εκτέλεση του κώδικα από το διάνυσμα αρχικοποίησής του το όποιο έχει αλλάξει και δείχνει στην εσωτερική μνήμη.

Εκτός από τις παραπάνω αλλαγές, ο τροποποιημένος RISC υποστηρίζει το ίδιο σύνολο εντολών με τον κλασσικό Hyperstone E1-32XSR και προγραμματίζεται με τον βασικό assembler της Hyperstone. Μετά την αρχικοποίησή του, ο τροποποιημένος RISC θα περιμένει για την εισαγωγή δεδομένων κατάστασης και πακέτων στην μνήμη του από την λογική του RPM. Όταν λάβει το σήμα εκκίνησης από την λογική του RPM θα ξεκινήσει να εκτελεί ένα μικρό κομμάτι κώδικα το οποίο εξαρτάται από τα δεδομένα των πακέτων, την κατάσταση της σύνδεσης και φυσικά από την εφαρμογή την οποία θα εκτελέσει το σύστημα του PRO^3.



Σχήμα 4.4: Μπλοκ διάγραμμα του E1-32XSR

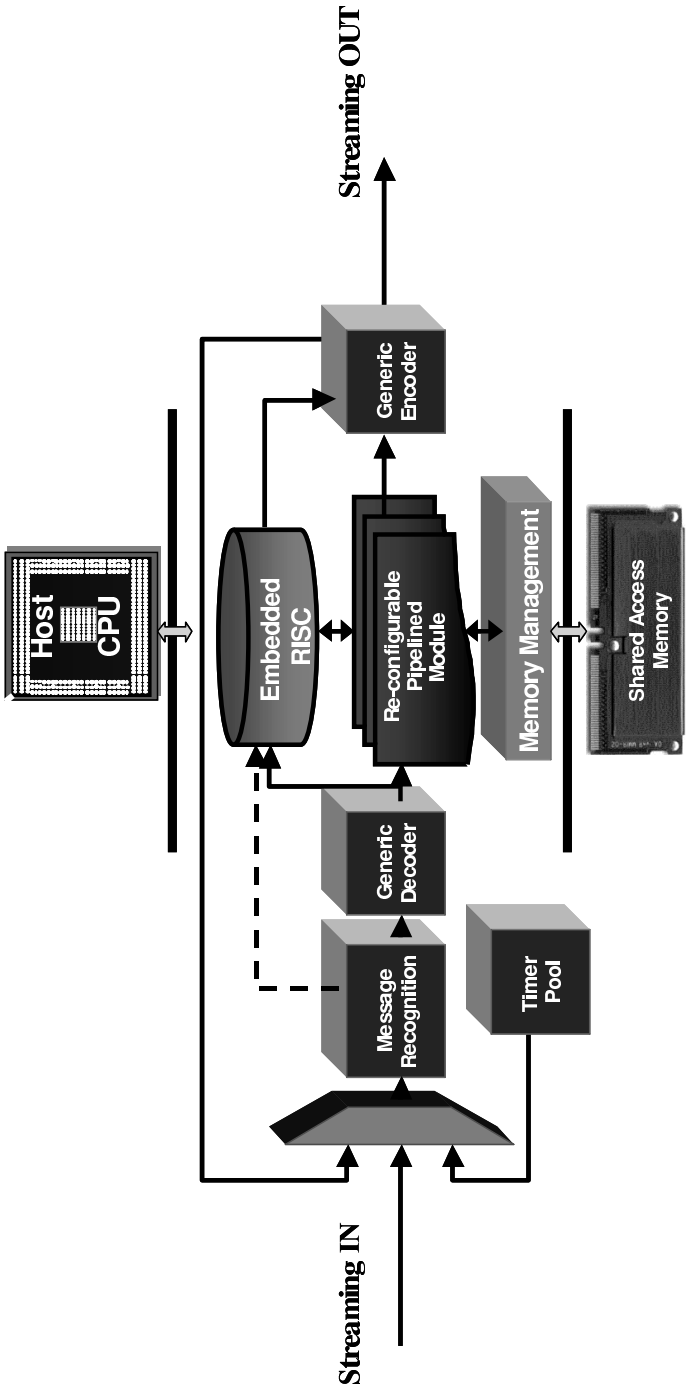
4.2 Η αρχιτεκτονική του PRO^3

4.2.1 Λειτουργική αρχιτεκτονική του PRO^3

Η λειτουργική αρχιτεκτονική του συστήματος παρουσιάζεται στο Σχήμα 4.5. Το ολοκληρωμένο κύκλωμα του PRO^3 αποτελείται από μία κεντρική μονάδα επεξεργασίας (τον πυρήνα RISC του RPM) καθώς και από ένα σύνολο περιφερειακών μέσα στο ολοκληρωμένο σύστημα, τα οποία είναι κοινά στα υπό λειτουργία πρωτόκολλα. Κατά συνέπεια, οι ίδιες μονάδες, με διαφορετικές ρυθμίσεις και αρχικοποιήσεις θα είναι ικανές να πραγματοποιούν τις μηχανές πεπερασμένων καταστάσεων (FSMs) πολλών διαφορετικών πρωτοκόλλων, τα οποία απαιτούν εκτέλεση υψηλής απόδοσης και διαχείριση των μηνυμάτων με μικρή καθυστέρηση διάδοσης/επεξεργασίας. Ο δίαυλος ανάδρασης επιστρέφει μηνύματα στην είσοδο του στοιχείου σε περίπτωση που έχει υλοποιηθεί μια στοίβα πολλαπλών πρωτοκόλλων. Σε αυτήν την λογική εφαρμόζεται επιπλέον και κάποιος μηχανισμός με κανόνες προτεραιότητας ώστε να καθορίζεται η πρόσβαση προς και από το σύστημα.

Το στοιχείο που παρουσιάζεται στο Σχήμα 4.5, είναι εφοδιασμένο με ένα κανάλι δεδομένων συνεχούς διοχέτευσης στο οποίο επεξεργάζονται τα μηνύματα μεγάλου εύρους ζώνης και με μία διεπαφή ελέγχου για την συναλλαγή λογισμικού. Σε αυτήν την λειτουργική αρχιτεκτονική, προβλέπονται να χρησιμοποιηθούν οι παρακάτω ροές μηνυμάτων και δεδομένων. Σε αυτό το μοντέλο λειτουργίας τα μπλοκ έχουν διαχωριστεί σε μονάδες μνήμης και μονάδες επεξεργασίας. Τα επι μέρους μπλοκ είναι τα παρακάτω:

- **Απεικόνιση**: Τα δεδομένα εισόδου τοποθετούνται σε έναν απομονωτή μέχρι να ληφθεί ένα ολόκληρο πακέτο. Προαιρετικά, αυτό το μπλοκ μπορεί να προσθέσει ορισμένα δεδομένα τα οποία δεν παρίστανται μέσα στο πακέτο, όπως για παράδειγμα ο αριθμός της θύρας από την οποία ελή-



Σχήμα 4.5: Λειτουργική αρχιτεκτονική του PRO^3

IN → OUT	Μία ροή δεδομένων φτάνει στην διεπαφή 'IN', επεξεργάζεται και το μήνυμα με το αποτέλεσμα αποχωρεί από τη διεπαφή του 'OUT'. Σε αυτήν την περίπτωση ο επεξεργαστής διασυνδέεται με το επόμενο στοιχείο του συστήματος το οποίο επεξεργάζεται το αποτέλεσμα.
IN → CONTROL	Μία ροή δεδομένων φτάνει στην διεπαφή 'IN', επεξεργάζεται και το μήνυμα με το αποτέλεσμα αποχωρεί από τη διεπαφή του ελέγχου 'CONTROL'.
CONTROL → OUT	Η λειτουργία είναι παρόμοια με την παραπάνω αλλά αφορά το κανάλι της αποστολής δεδομένων.
Μήνυμα βρόγχου επιστροφής	Η μονάδα πραγματοποιεί τη στοίβα για τουλάχιστον δύο στρώματα του πρωτοκόλλου. Το αποτέλεσμα του επιπέδου πρωτοκόλλου Α, τροφοδοτείται μέσω βρόγχου και επεξεργάζεται από την μηχανή αυτή για το επόμενο στρώμα του πρωτοκόλλου που είναι το Β.

Πίνακας 4.1: Ροές δεδομένων και μηνυμάτων

φθησαν τα δεδομένα. Αυτό επιτρέπει στην εφαρμογή να χρησιμοποιήσει αυτήν την τιμή σε αποφάσεις κατηγοριοποίησης.

- **Διαχείριση απομονωτή πακέτων**: Τα έγκυρα πακέτα αποθηκεύονται σε μία δομή δεδομένων αφού πρώτα έχουν σημαδευτεί με έναν κωδικό ταυτοποίησης. Αυτός ο κωδικός επιτρέπει την μετέπειτα ανάκτηση του πακέτου. Οι πραγματικοί μηχανισμοί μεταφοράς και πολιτικής υλοποιούνται σε ένα εξωτερικό μπλοκ για την διαχείριση των πακέτων, το οποίο υποστηρίζει λειτουργίες όπως είναι η αποθήκευση, η διαγραφή και η ανάκτηση.

- **Γενικός αποκωδικοποιητής (εξαγωγής πεδίων)**: Πολλά πρωτόκολλα έχουν πεδία επικεφαλίδων τα οποία δεν είναι ιδανικά τοποθετημένα για επεξεργασία σε ένα γενικού σκοπού μικροεπεξεργαστή. Είναι απαραίτητο να παρέχεται ένας προγραμματιζόμενος εξαγωγέας πεδίων ώστε να εξάγει αυτά τα πεδία και να είναι διαθέσιμα για περαιτέρω επεξεργασία. Πρέπει επίσης να είναι τόσο παραμετροποιήσιμο ώστε να μπορεί να λειτουργεί με επικεφαλίδες μεταβλητού μήκους όπως αυτοί που βρίσκονται στα πακέτα του IP πρωτοκόλλου.

- **Επαλήθευση**: Το πακέτο ελέγχεται για πιθανά λάθη. Αυτό μπορεί να γίνει χρησιμοποιώντας μία ποικιλία σχετικών με πρωτόκολλα τεχνικών όπως είναι ο έλεγχος αθροίσματος ως προς δύο, ο έλεγχος αθροίσματος ως προς ένα, και ο έλεγχος λαθών ή η κωδικοποίηση για τη διόρθωση λαθών. Αν ένα πακέτο βρεθεί να περιέχει λάθη ώστε δεν είναι δυνατή η επεξεργασία του σε επίπεδο πρωτοκόλλου, απορρίπτεται από αυτό το μπλοκ. Επίσης, από αυτό το μπλοκ μπορούν να ληφθούν στατιστικά σχετικά με τον ρυθμό λαθών που μπορεί να εμφανιστεί σε μία γραμμή.

- **Κατηγοριοποίηση πακέτων**: Τα προαναφερθέντα εξαγόμενα πεδία

συνδυάζονται σε ένα κλειδί και έτσι πραγματοποιείται μία αναζήτηση σε μία βάση δεδομένων η οποία περιέχει έναν αριθμό από κανόνες και φίλτρα. Οι τύποι των απαντήσεων των αναζητήσεων που μπορεί να οδηγήσουν σε απόφαση για επεξεργασία των πακέτων είναι παραμετροποιήσιμη και μπορεί να είναι ακριβές ταίριασμα του κλειδιού, μερική ή ενός εύρους. Ανάλογα με το αποτέλεσμα της κατηγοριοποίησης, μπορεί να ληφθεί μία συγκεκριμένη απόφαση. Αυτή μπορεί να είναι τόσο απλή όσο η προώθηση του πακέτου με λίγα πιθανόν τροποποιημένα πεδία (εφαρμογές μεταγωγής και δρομολόγησης), ή μπορεί να εμπλέκει πιο περίπλοκη επεξεργασία δεδομένων πρωτοκόλλου. Στην δεύτερη περίπτωση, εισάγεται το μπλοκ του εξαγωγέα πεδίων και το πακέτο των δεδομένων προωθείται στην μονάδα επεξεργασίας πρωτοκόλλων για περισσότερες περίπλοκες λειτουργίες.

- **Επεξεργασία πρωτοκόλλου ανά εφαρμογή**: Αυτό το τμήμα του μοντέλου είναι υπεύθυνο για την επεξεργασία σε σχέση με την εκάστοτε εφαρμογή και την επεξεργασία που πρέπει να γίνει για το τρέχον πακέτο. Στην περίπτωση της εφαρμογής της σηματοδοσίας του ATM, θα αποστέλλει και θα λαμβάνει μηνύματα Q.2931 όταν επεξεργάζεται τα υψηλότερα στρώματα του πρωτοκόλλου. Για το TCP/IP ο κώδικας της επιθεώρηση κατάστασης[2], θα εξετάζει την ροή δεδομένων που αποστέλλονται για παράδειγμα σε μία εφαρμογή FTP, ώστε να ανανεώνει τα χαμηλότερα επίπεδα για την πολιτική ασφαλείας για να επιτρέπονται οι συνδέσεις από μία συγκεκριμένη διεύθυνση ή και από μία συγκεκριμένη θύρα του πρωτοκόλλου.

- **Μετατροπή επικεφαλίδας**: Αυτό το μπλοκ αποτελεί την ακριβώς α-

[2]State-full Inspection

ντίστροφη διαδικασία από αυτήν του εξαγωγέα πεδίων. Επιπλέον, η εισαγωγή νέων πεδίων μπορεί να απαιτεί τον υπολογισμό ενός νέου αθροίσματος ελέγχου.

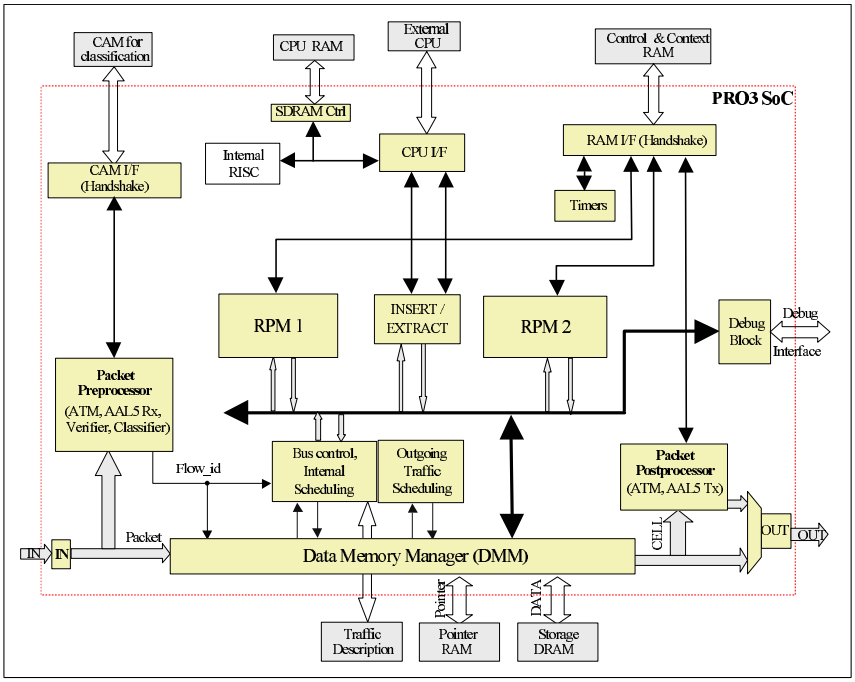
- **Διαχειριστής ουρών δεδομένων**: Ο διαχειριστής ουρών αποτελεί την διεπαφή του συστήματος με τις μνήμες που αποθηκεύονται τα πακέτα δεδομένων. Αποθηκεύει, ανακτά και διαγράφει πακέτα από την μνήμη καθώς επίσης μπορεί να κρατήσει πληροφορίες σχετικά με την κατάσταση των ουρών που ανήκει κάποιο πακέτο.

- **Χρονοδρομολόγηση**: Επιτρέπει στον επεξεργαστή να χειριστεί δεδομένα τα οποία επηρεάζονται με τον χρόνο (χρησιμοποιώντας αλγορίθμους όπως είναι ο Weight Fair Queueing βασιζόμενος σε ετικέτες χρόνου), ισορροπίας φορτίου (για παράδειγμα ίσου καταμερισμού του εύρους ζώνης μεταξύ του TCP το οποίο είναι εφοδιασμένο με έλεγχο ροής και του UDP που δεν είναι), και ευρείας γνωστοποίησης (με την αντιγραφή συγκεκριμένων πακέτων πολλές φορές αλλά προς διαφορετικές κατευθύνσεις).

- **Έλεγχος χρονομετρητών**: Οι στοίβες πρωτοκόλλων έχουν γενικότερα έναν μεγάλο αριθμό από ενεργούς χρονομετρητές, ώστε να διαχειρίζονται καταστάσεις όπως είναι η λήξη των χρονομετρητών και οι επαναμεταδόσεις σε περιπτώσεις λαθών. Το PRO^3 παρέχει ένα αποκλειστικό μπλοκ ώστε να επιταχύνει τον χειρισμό αυτών των χρονομετρητών.

4.3 Αρχιτεκτονική

Το Σχήμα 4.6 παρουσιάζει την αρχιτεκτονική του δικτυακού επεξεργαστή. Στις αρχικές προδιαγραφές έχει προστεθεί ένα μπλοκ ελέγχου και εποπτείας του συστήματος. Επίσης, στο Σχήμα 4.6 δίνονται και οι συνδέσεις του συστή-



Σχήμα 4.6: Αρχιτεκτονική του PRO^3

ματος με την εξωτερική CPU καθώς επίσης και με τον ενσωματωμένο RISC επεξεργαστή.

Στο Σχήμα 4.6 παρουσιάζεται επίσης και ο εσωτερικός κοινός δίαυλος ο οποίος χρησιμοποιείται για την μεταφορά τον δεδομένων προς τα RPMs από τον DMM. Εκτός από αυτή την επικοινωνία σημείου προς σημείο υφίσταται και επικοινωνία μεταξύ των δύο RPMs ή μεταξύ των ενός εκ των δύο RPMs και των δύο CPUs.

Εκτός από τον κοινό δίαυλο επικοινωνίας, υπάρχει ένα κλασικός δίαυλος microporcessor ο οποίος χρησιμεύει για τον προγραμματισμό και τον έλεγχο των εσωτερικών μπλοκ. Επίσης είναι εφικτός ο επαναπρογραμματισμός των υπο-μονάδων αυτών οι οποίες εκτελούν μικροκώδικα. Στην πραγματικότητα αυτός ο δίαυλος ανήκει στον εσωτερικό Hyperstone. Στην παραπάνω αρχι-

τεκτονική έχουν ληφθεί υπόψη οι παρακάτω θεωρήσεις για την απόδοση του συστήματος:

1. **CAM interface** για την εφαρμογή του IP πρέπει να υποστηρίζεται ρυθμός μέχρι 7,5Mpackets/sec ενώ για την ATM εφαρμογή πρέπει να είναι εφικτοί ρυθμοί μέχρι 5,6Mpackets/sec.

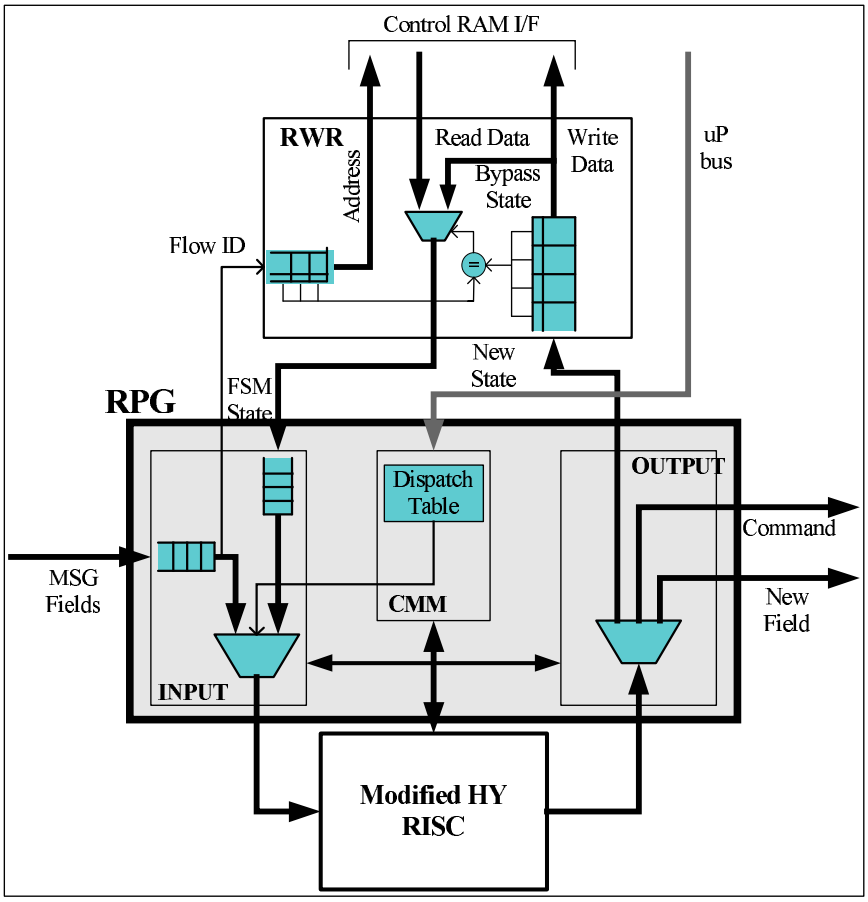
2. **DMM interface** για την αποθήκευση των λαμβανομένων πακέτων σε εξωτερική μνήμες έχει επιλεχθεί η χρήση μίας υπομονάδας DDR-SDRAM η οποία είναι ικανή να προσφέρει εύρος ζώνης μέχρι 17Gbps. Ο δίαυλος δεδομένων έχει πλάτος 64bits και λειτουργεί στα 133MHz.

3. **DMM Interface** για την μνήμη ελέγχου. Αν και η μνήμη αυτή αποθηκεύει σαφώς μικρότερο αριθμό δεδομένων η απόδοση της θα πρέπει να αγγίζει ρυθμούς της τάξης των Gbps. Ο δίαυλος δεδομένων έχει πλάτος 32bits.

4. **Ο εσωτερικός δίαυλος δεδομένων** έχει επιλεχθεί ώστε να μπορεί να αντεπεξέλθει στις χείριστες συνθήκες φόρτου λειτουργίας. Ο δίαυλος χρησιμοποιεί πλάτος 64bits και λειτουργεί στα 200MHz. Ό δίαυλος είναι δι-κατευθυντικός και μοιράζεται μεταξύ όλων των υπομονάδων. Επίσης, μοιράζονται σήματα ελέγχου και διευθυνσιοδότησης. Μία ένωση σημείου-προσ-σημείο για την αστυνόμευση της κίνησης πάνω από τον κοινό δίαυλο ενώνει το κάθε μπλοκ ξεχωριστά με τον "διατητή"(arbiter) του συστήματος.

Το σύστημα PRO^3 βασίζεται σε δύο μηχανές επεξεργασίας πρωτοκόλλων (RPM) οι οποίες μπορούν να προγραμματιστούν για να εκτελέσουν λειτουργίες χαμηλού επιπέδου. Το μπλοκ διάγραμμα τους δίνεται στο Σχήμα 4.7.

Οι δύο υπομονάδες RPM αποτελούν την καρδιά του συστήματος. Αυτές με τη σειρά τους αποτελούνται από τρεις υπομονάδες οι οποίες εκτελούν συγκε-



Σχήμα 4.7: Μπλοκ διάγραμμα του RPM

κριμένες διεργασίες απαραίτητες για την επεξεργασία του πρωτοκόλλου. Οι υπομονάδες αυτές που διακρίνονται στο Σχήμα 4.7 ονομάζονται Εξαγωγέας Πεδίων (Field Extractor), Μηχανή Επεξεργασίας Πρωτοκόλλων (Protocol Processing Engine) και Τροποποιητής Πεδίων (Field Modifier). Για κάθε πακέτο, τα δεδομένα εισόδου στο RPM είναι:

1. Η ετικέτα ταξινόμησης FLow_ID καθώς και ο τύπος του πακέτου όπως έχει προσδιορισθεί από τον ταξινομητή.

2. Οι πληροφορίες κατάστασης του συγκεκριμένου πακέτου που είναι αποθηκευμένες στην μνήμη ελέγχου.

3. Ολόκληρη η κεφαλίδα του πακέτου.

Τα εξαγόμενα δεδομένα περνούν διαμέσου του σωλήνα συνεχούς διοχέτευσης (pipeline) για να εκτελεστεί ο κώδικας επεξεργασίας του πακέτου. Μεταξύ των υπομονάδων χρησιμοποιούνται ουρές για την αποσύζευξη και την καθυστέρηση των υπομονάδων.

4.4 Εσωτερικές διαδρομές δεδομένων

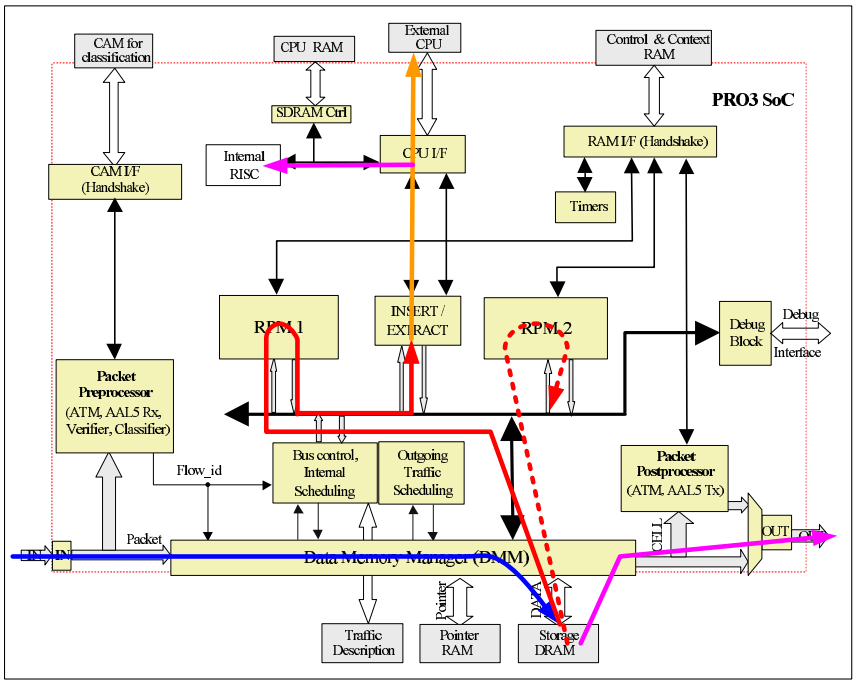
Οι διαδρομές δεδομένων καθώς και η αλληλουχία των διεργασιών για την IP εφαρμογή αλλά και για την ATM εφαρμογή περιγράφονται παρακάτω.

4.4.1 IP εφαρμογή

Σε αυτή την παράγραφο αναλύονται οι διαδρομές των δεδομένων σε εφαρμογές που βασίζονται στο IP πρωτόκολλο. Βασιζόμενοι στο μπλοκ διάγραμμα μπορούμε να διακρίνουμε ένα μεγάλο πλήθος από διαδρομές δεδομένων που διέρχονται μέσα από το σύστημα, ανάλογα με τον απαιτούμενο τύπο επεξεργασίας.

Τα λαμβανόμενα πακέτα προωθούνται άμεσα στο Διαχειριστή Μνήμης Δεδομένων (Data Memory Manager, DMM). Η κατανομή της μνήμης βασίζεται σε καθορισμένου μεγέθους τμήματα, συνεπώς απαιτείται μια λειτουργία κατακερματισμού και επανασύνδεσης η οποία ενσωματώνεται στη μονάδα αυτή. Καθώς το πακέτο στέλνεται στο DMM, αντιγράφεται και στον ταξινομητή IP για να του αποδοθεί αριθμός ροής. Τα πεδία που απαιτούνται για την ταξινόμηση του πακέτου, εξάγονται και δίνονται στην CAM για να παρθούν αποφάσεις για την συνέχεια του πακέτου. Εφόσον το πακέτο έχει ταξινομηθεί σωστά, ο αριθμός ροής (Flow_ID) λαμβάνεται και στέλνεται στον DMM ώστε να αποθηκευτεί το προσωρινά αποθηκευμένο πακέτο στην κατάλληλη ουρά. Τέλος, ο χρονοδρομολογητής διεργασιών ενημερώνεται για την ύπαρξη του νέου πακέτου. Ακολουθεί μία εντολή από τον χρονοδρομολογητή διεργασιών όπου το πακέτο προωθείται σε μία από τις δύο μονάδες επεξεργασίας ή σε μία από τις δύο CPUs είτε ολόκληρο είτε μερικώς. Μετά το τέλος της επεξεργασίας, τα ενημερωμένα δεδομένα επιστρέφονται στο DMM και το πακέτο προγραμματίζεται για μετάδοση.

Όταν πρέπει να εξυπηρετηθούν συνθήκες χείριστης περίπτωσης, απαιτείται επεξεργασία κατά την διέλευση του πακέτου. Αυτό προϋποθέτει ότι πακέτα ελάχιστου μεγέθους (π.χ. 40 bytes) θα ληφθούν και θα μεταδοθούν με ρυθμό 2.5Gbps. Μεταφράζοντας αυτά τα δεδομένα σε χρόνους επεξεργασίας, προκύπτει ότι 128ns είναι διαθέσιμα για την επεξεργασία του κάθε πακέτου. Από την άλλη πλευρά, όταν απαιτείται περισσότερο περίπλοκη επεξεργασία, μπορεί να θεωρηθεί ασφαλώς ότι οι συνθήκες χείριστης κατάστασης δε θα παρατηρηθούν σε κανονικές συνθήκες συνεπώς για τους υπολογισμούς μπορεί να ληφθεί υπόψη η μέση τιμή. Η διαδρομή δεδομένων δεν αλλάζει αλλά τα δεδομένα μπορεί να χρειαστεί να παραμείνουν αποθηκευμένα για μεγαλύτερα χρονικά διαστήματα. Για τέτοιες περιπτώσεις απαιτείται αποδοτική αποθήκευση από



Σχήμα 4.8: Ροές δεδομένων για εφαρμογές IP πάνω από SONET

το σύστημα. Στα παρακάτω αναλύουμε τις ενέργειες που υπάρχουν στο IP πρωτόκολλο υποθέτοντας ότι η κίνηση μεταφέρεται διαμέσου της τεχνολογίας διεπαφής SONET.

Κατά την μεταφορά του IP πρωτοκόλλου δια μέσου της SONET διεπαφής οι ροές δεδομένων και η αλληλουχία των ενεργειών δίνεται στο Σχήμα 4.8. Σε αυτού του τύπου της εφαρμογές (επεξεργασία και προώθηση) υπάρχουν δύο εναλλακτικές υπομονάδες για την επεξεργασία των πρωτοκόλλων, η υπομονάδα RPM1 καθώς επίσης και η RPM2. Γενικά, υπάρχουν τρεις διαφορετικές αλληλουχίες λειτουργιών οι οποίες λαμβάνουν μέρος παράλληλα και έχουν τον DMM ως σημείο αναφοράς. Αυτές είναι *λήψη πακέτου, επεξεργασία πακέτου και μετάδοση πακέτου*. Επιπλέον, υπάρχει μία επιπρόσθετη λειτουργία και μία επιπλέον διαδρομή δεδομένων μεταξύ των δύο CPUs η οποία δεν ελέγχεται από τον χρονοδρομολογητή.

Οι λειτουργίες ξεκινούν με την εμφάνιση συγκεκριμένων γεγονότων και συμπληρώνονται αμέσως μόλις το πακέτο παραληφθεί από τον προορισμό ή αποσταλεί στο δίκτυο. Επιπλέον οι λειτουργίες ξεκινούν από το μπλοκ IN (κατά την λήψη του πακέτου), από τον χρονοδρομολογητή (αποστολή πακέτου για επεξεργασία ή επιστροφή του πακέτου πίσω στον DMM) και από τον δρομολογητή κυκλοφορίας (αποστολή πακέτων στο δίκτυο). Ουσιαστικά, η υπομονάδα IN ξεκινάει και ελέγχει την κυκλοφορία πάνω από την διεπιφάνεια της εισόδου (διαδικασία λήψης), ο χρονοδρομολογητής ξεκινάει και ελέγχει την κίνηση στον εσωτερικό δίαυλο (επεξεργασία πακέτου) και ο δρομολογητής κυκλοφορίας ξεκινάει και ελέγχει την κίνηση προς την διεπιφάνεια εξόδου προς το δίκτυο (μετάδοση). Οι λειτουργίες αυτές αντιστοιχούν σε εύρος ζώνης 2.5Gbps για την διαδικασία λήψης και αποστολής πακέτων και 5Gbps για την επεξεργασία πακέτων (2.5Gbps αποστέλλονται για επεξεργασία και 2.5Gbps λαμβάνονται μετά την επεξεργασία).

Α. Λήψη πακέτων

1. **IN**: Λαμβάνει ένα πακέτο και το προωθεί στον καταχωρητή IP και στον DMM.

2. (α΄) **Ο ταξινομητής IP**: Λαμβάνει πακέτα, εκτελεί ένα προκαταρκτικό έλεγχο και πραγματοποιεί την εξαγωγή των κατάλληλων πεδίων για την ταξινόμηση του πακέτου. Τα εξαγόμενα πεδία μαζί με επιπλέον πληροφορία ελέγχου αποστέλλονται στον DMM.

   (β΄) **DMM**: Λαμβάνει πακέτα, τα κατακερματίζει και τα αποθηκεύει σε προσωρινές ουρές, στο τέλος εφόσον υπολογιστεί το μέγεθος του πακέτου αποθηκεύεται μόνιμα στην ανάλογη ουρά.

3. **DMM**: Λαμβάνει την συνοδευτική ετικέτα του πακέτου (Flow_ID) μαζί με πληροφορία ελέγχου από τον ταξινομητή και εκτελεί τα παρακάτω:

   (α΄) Επισυνάπτει το πακέτο στην ουρά που αντιστοιχεί στο πεδίο Flow_ID και ενημερώνει τους σχετικούς δείκτες.

   (β΄) Ανανεώνει στατιστικές πληροφορίες και πληροφορίες ελέγχου.

   (γ΄) Αποστείλει στον χρονοδρομολογητή πληροφορίες βασισμένες στο Flow_ID. Εάν το πακέτο πρέπει να επεξεργαστεί τότε ανανεώνεται η πληροφορία στον χρονοδρομολογητή του συστήματος, αλλιώς εάν πρέπει το πακέτο να αποσταλεί στο δίκτυο ενημερώνεται ο χρονοδρομολογητής της κίνησης.

Β. Επεξεργασία πακέτων

Ο χρονοδρομολογητής του συστήματος αρχικοποιεί και ελέγχει την λειτουργία του εσωτερικού διαύλου μεταξύ των αυτόνομων υπομονάδων. Από τις πληροφορίες που λαμβάνει αναγνωρίζει και το μπλοκ που θα ενεργήσει ως

αποστολέας αλλά και η υπομονάδα που θα δεχθεί τα δεδομένα. Η λειτουργία του συνοψίζεται στα εξής:

- Λαμβάνει αιτήσεις από τα RPMs και τις κεντρικές μονάδες επεξεργασίας και στέλνει τα πακέτα μέσω του εσωτερικού διαύλου επικοινωνίας (η υπομονάδα που ενεργεί ως αποστολέας προσδιορίζει τον παραλήπτη).

- Ανακτά την πληροφορία που είναι αποθηκευμένη στην εξωτερική του μνήμη και λαμβάνει απόφαση για το ποίο θα είναι το επόμενο πακέτο προς επεξεργασία.

- Λαμβάνει σήματα ελέγχου από τις μονάδες επεξεργασίας για να γνωρίζει την διαθεσιμότητα τους (π.χ. σήματα busy).

- Επεξεργάζεται όλα τα εισερχόμενα σήματα και με το προκαθορισμένο σύστημα προτεραιότητας που υλοποιεί αναθέτει τον δίαυλο εκδίδοντας τις κατάλληλες εντολές στην υπομονάδα που επιθυμεί να αποστείλει πληροφορίες.

Τα πέντε πιθανά ζευγάρια αποστολέα παραλήπτη δίνονται παρακάτω. Ως "οποιοδήποτε υπομονάδα" θεωρούμε ένα από τα: RPM, την εσωτερική CPU ή την εξωτερική CPU.

1. DMM προς "οποιοδήποτε υπομονάδα":

   (α′) Ο DMM βασισμένος στο Flow_ID εντοπίζει το πρώτο τεμάχιο του πακέτου που θα αποσταλεί για επεξεργασία.

   (β′) Ο DMM προετοιμάζει το κατάλληλο πρόθεμα και το πολυπλέκει με τα δεδομένα του πακέτου (τα πρώτα 64-bit του πακέτου προς αποστολή αποτελούν τον πρόθεμα ελέγχου).

(γ′) Ο DMM διαβάζει το πακέτο ή μέρος του πακέτου (εξαρτάται από την εντολή που εκδίδεται από τον χρονοδρομολογητή) και το στέλνει στον εσωτερικό δίαυλο.

(δ′) Η υπομονάδα παραλήπτης λαμβάνει το σύνολο των δεδομένων.

(ε′) Μαζί με τα δεδομένα η υπομονάδα παραλήπτης αποκωδικοποιεί την εντολή που συνοδεύει τα δεδομένα και διαβάζει τους τελεστέους της εντολής για να αποφασίσει πως θα επεξεργαστεί το πακέτο.

(ϛ′) Αρχίζει την επεξεργασία του πακέτου.

2. Από "οποιοδήποτε υπομονάδα" προς τον DMM:

Η διεργασία αυτή εκτελείται όταν ένα εισαγόμενο πακέτο ή εντολή ελέγχου στέλνεται από την CPU ή από το RPM στον DMM.

(α′) Η υπομονάδα αποστολέας: Προετοιμάζει το κατάλληλο πρόθεμα και το πολυπλέκει με τα δεδομένα.

(β′) Η υπομονάδα αποστολέας: Στέλνει το πακέτο στον DMM μέσω του εσωτερικού διαύλου.

(γ′) Ο DMM λαμβάνει το σύνολο των δεδομένων.

(δ′) Ο DMM αποκωδικοποιεί την εντολή και διαβάζει τους τελεστέους της εντολής ελέγχου.

(ε′) Ο DMM βασισμένος στο Flow_ID εντοπίζει την κατάλληλη ουρά για να γράψει τα δεδομένα.

(ϛ′) Ο DMM κατακερματίζει και αποθηκεύει τα δεδομένα.

(ζ′) Τέλος ο DMM ανανεώνει τις καταχωρήσεις της συγκεκριμένης ουράς στην μνήμη του χρονοδρομολογητή. Σε περίπτωση που το πακέτο είναι έτοιμο να αποσταλεί στο δίκτυο τοποθετείται στην ουρά

εξόδου. Σε περίπτωση που το πακέτο χρήζει περαιτέρω επεξεργασίας τοποθετείται στην κατάλληλη ουρά για τα υψηλότερα επίπεδα

Σε περίπτωση που ο DMM λάβει μόνο εντολή ελέγχου χωρίς δεδομένα το τελευταίο βήμα δεν εκτελείται.

3. RPM1 προς RPM2

   Σε αυτή την περίπτωση, το RPM1 πρέπει να ενημερώσει το RPM2 να πράξει κατάλληλα όταν του δοθεί άδεια από τον χρονοδρομολογητή διεργασιών

   (α′) Το RPM1 προετοιμάζει την κατάλληλη εντολή και την πολυπλέκει με τα δεδομένα του πακέτου (η εντολή αποτελεί πάντα την πρώτη ψηφιακή λέξη στο πακέτο).

   (β′) Το RPM1 Κάνει αίτηση για τη χρήση του κοινού διαύλου.

   (γ′) Ο Χρονοδεομολογητής διεργασιών αναθέτει δίνει άδεια στο RPM1 για τη χρήση του διαύλου.

   (δ′) Το RPM2 λαμβάνει την εντολή και εξάγει από το πακέτο διάφορα πεδία όπως: Flow_ID, δεδομένα και λοιπές πληροφορίες που σχετίζονται με τον τρόπο επεξεργασίας του πακέτου.

   (ε′) Τέλος, το RPM2 αρχίζει την επεξεργασία του πακέτου.

4. RPM προς το INEX

   Σε αυτή την περίπτωση, το RPM πρέπει να στείλει ένα πακέτο σε μία CPU.

   (α′) Το RPM προετοιμάζει την απαραίτητη κεφαλίδα ελέγχου και την πολυπλέκει με τα δεδομένα.

   (β′) Το RPM κάνει αίτηση για τη χρήση του κοινού διαύλου.

(γ′) Ο Χρονοδεομολογητής διεργασιών αναθέτει δίνει άδεια στο RPM για τη χρήση του διαύλου.

(δ′) Το Insert/Extract λαμβάνει την εντολή, εξάγει την πληροφορία ελέγχου και επιλέγει την CPU στην οποία θα αποσταλεί το πακέτο.

(ε′) Τέλος, το Insert/Extract προωθεί το πακέτο στην επιλεγμένη CPU.

Γ. Αποστολή πακέτων Η διαδικασία της αποστολής βασίζεται είτε σε ένα μηχανισμό μορφοποίησης της κίνησης, είτε στην αρχή της προώθησης σύμφωνα με την διαθεσιμότητα. Για την πρώτη περίπτωση, ένας περιορισμένος αριθμός κατηγοριών υπηρεσίας (π.χ. 64) υποστηρίζεται χρησιμοποιώντας την ενσωματωμένη μνήμη. Στην δεύτερη περίπτωση, υποθέτουμε ότι η κίνηση θα μεταδοθεί στο δίκτυο διαιτητεύοντας το προφίλ της κίνησης των εισερχομένων πακέτων. Για να διατηρηθεί το προφίλ, ο χρονοδρομολογητής διεργασιών επεξεργάζεται τα πακέτα διατηρώντας την σειρά άφιξης και υποστηρίζοντας διαφορετικές προτεραιότητες για την εσωτερική επεξεργασία.

5. Ο χρονοδρομολογητής διεργασιών:

   (α′) ελέγχει την κατάσταση των κατηγοριών κίνησης και επιλέγει το Flow_ID του επόμενου πακέτου που θα μεταδοθεί.

   (β′) στέλνει εντολή στον DMM για το επόμενο πακέτο προς αποστολή

   (γ′) ανανεώνει τις ουρές των δεδομένων του.

6. Ο DMM:

   (α′) Αποκωδικοποιεί την εντολή και διαβάζει το πακέτο/cell προς αποστολή.

   (β′) Ανανεώνει τους δείκτες του και τις πληροφορίες ελέγχου που διατηρεί.

Δ. Διαδρομή δεδομένων από την CPU1 προς CPU2 Μία επιπλέον διαδρομή δεδομένων προβλέπεται μεταξύ των δύο CPUs. Η επικοινωνία αυτή καθίσταται μεταξύ των δύο μονάδων χωρίς την παρέμβαση και τον συντονισμό του χρονοδρομολογητή διεργασιών.

4.4.2 ATM εφαρμογή

Σε αυτήν την παράγραφο αναλύουμε τις διαδρομές δεδομένων του συστήματος για εφαρμογές βασισμένες στο ATM. Οι εφαρμογές αυτές συνήθως χρησιμοποιούνται σε τερματικούς σταθμούς όπου διενεργούνται συχνά δημιουργίες και τερματισμοί πρωτοκόλλων. Τα εισερχόμενα cells προωθούνται απευθείας στον DMM για αποθήκευση και αντιγράφονται παράλληλα στο μπλοκ ATM/CPCS για ταξινόμηση. Τα cells αναμένουν σε μία ουρά μέχρι να είναι διαθέσιμο το αποτέλεσμα της διαδικασίας ταξινόμησης, έπειτα τοποθετούνται σε μία από τις ουρές.

Εφόσον, τα ATM cells, τα οποία ανήκουν σε ένα μήνυμα ATM δεν φθάνουν με σειρά πρέπει να αναμείνουν σε ένα προσωρινό καταχωρητή μέσα στο ολοκληρωμένο κύκλωμα μέχρι να προσδιοριστεί το Flow_ID καθώς και ο τύπος του αφιχθέντος cell (τελευταίο ή ενδιάμεσο). Υποθέτουμε ότι η διαδικασία της ταξινόμησης και η εύρεση του τύπου του πακέτου μπορεί να ολοκληρωθεί μέσα σε ένα ή δύο σχισμές του χρόνου. Επιπλέον, για να μπορέσουμε να ακολουθήσουμε μία γενική αρχιτεκτονική για τις δύο υποστηριζόμενες εφαρμογές από το σύστημα, ATM και IP, θεωρούμε ότι τα δεδομένα (πακέτα ή cells) δεν καταχωρούνται στην ουρά μέχρι την αναγνώριση του τύπου τους. Γενικά, όλες οι ροές των δεδομένων που λαμβάνουν χώρα στην εφαρμογή ATM είναι παρόμοιες με αυτές που περιγράφηκαν λεπτομερώς στην προηγούμενη παράγραφο. Οι βασικές διαφορές είναι ότι κατά την λειτουργία του ολοκληρωμένου σε εφαρμογές ATM εμπλέκονται και διαφορετικά μπλοκ λογικής και ο DMM

αποθηκεύει ουρές από cells.

Η ροή των δεδομένων μέσα στο ολοκληρωμένο κύκλωμα για την εφαρμογή ATM δίνεται στο Σχήμα 4.9.

Α. Λήψη cells

1. Το μπλοκ **IN** λαμβάνει τα cells και τα προωθεί στο ATM/CPCS Rx μπλοκ για ταξινόμηση ενώ ταυτόχρονα αντιγράφονται και στον DMM

2. (α′) Το μπλοκ **ATM/CPCS Rx** λαμβάνει ένα cell και πραγματοποιεί έλεγχο, εξαγωγή πεδίων και ταξινόμηση. Το μπλοκ αυτό υλοποιεί τις λειτουργίες κατακερματισμού και επαναδόμησης του πρωτοκόλλου AAL-5. Στο τέλος της διαδικασίας, λαμβάνονται τα πεδία Flow_ID, η προτεραιότητες καθώς επίσης και λοιπά πεδία ελέγχου και αποστέλλονται στον DMM.

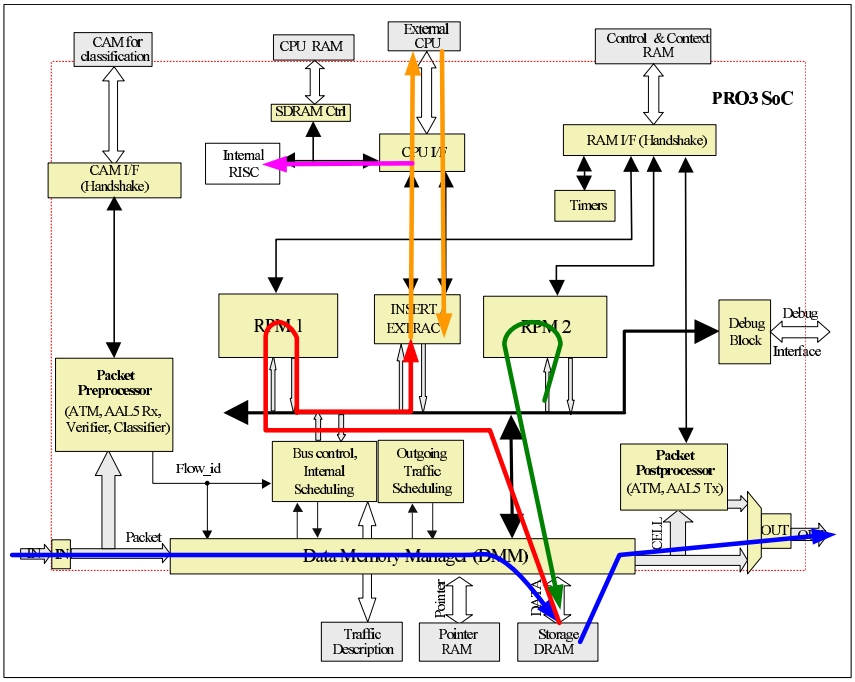
   (β′) Ο **DMM** λαμβάνει ένα cell και το αποθηκεύει στους εσωτερικούς προσωρινούς καταχωρητές περιμένοντας από το μπλοκ ATM/CPCS Rx τις απαραίτητες πληροφορίες για την αποθήκευση του.

   (γ′) Ο **DMM** λαμβάνει το Flow_ID και την προτεραιότητα από τον ATM ταξινομητή και εκτελεί τα παρακάτω:

      i. Τοποθετεί το cell σύμφωνα με την ετικέτα Flow_ID στην ανάλογη ουρά και ανανεώνει τους σχετικούς με την ουρά δείκτες.

      ii. Ανανεώνει στατιστικές πληροφορίες καθώς επίσης και πληροφορίες ελέγχου.

      iii. Αποστέλλει στον χρονοδρομολογητή διεργασιών τις σχετικές πληροφορίες κυρίως την προτεραιότητα και το τξλΦλοω_ΙΔ, αν το πακέτο πρέπει να επεξεργαστεί τότε αναμένει την συμπλήρωση του πακέτου πριν από την αποστολή του προς τις επεξεργαστικές μονάδες, σε αντίθετη περίπτωση, αν το πακέτο πρέπει



Σχήμα 4.9: Ροές δεδομένων για εφαρμογές ATM

να προωθηθεί στο δίκτυο ενημερώνει το χρονοδρομολογητή κίνησης.

Β. Επεξεργασία του πακέτου Όλες οι διεργασίες είναι παρόμοιες με αυτές που περιγράφηκαν στην προηγούμενη παράγραφο. Οι πέντε πρώτοι συνδυασμοί αποστολής από τον αποστολέα προς τον παραλήπτη είναι σχεδόν οι ίδιες. Η μόνη διαφορά που εντοπίζεται είναι ότι πλέον στον DMM γίνεται και αποθήκευση

Γ. Αποστολή του cell Η λειτουργία αυτή είναι παρόμοια με αυτή που περιγράφηκε σε προηγούμενη παράγραφο. Σε αυτή όμως την περίπτωση, τα δεδομένα διέρχονται διάμεσο του ATM/CPCS μπλοκ για να υλοποιηθούν οι διεργασίες κατακερματισμού και υπολογισμού του CRC-32 όπως προβλέπει το πρωτόκολλο AAL-5.

4.5 Ανάλυση και απόδοση του πρωτοκόλλου SSCOP

Το πρωτόκολλο Σχετικών Υπηρεσιών με Σύνδεση SSCOP[3] το οποίο καθορίζεται από την προδιαγραφή ITU Q.2110, παρέχει μηχανισμούς για την δημιουργία, τον τερματισμό και την εποπτεία αξιόπιστων συνδέσεων σηματοδοσίας μεταξύ ισότιμων οντοτήτων. Τέτοιες λειτουργίες βρίσκουν εφαρμογές σε πληθώρα πρωτοκόλλων επικοινωνίας για διαφορετικά περιβάλλοντα όπως για παράδειγμα τα X.25, LAPB, ISD, LAPD, TCP [Doer90], [Black93]. Το SSCOP αποτελεί ένα ολόκληρο τμήμα του πρωτοκόλλου της σηματοδοσίας του ATM και υλοποιείται για κάθε σύνδεση σηματοδοσίας στα δίκτυα ATM. Το πρωτόκολλο SSCOP πραγματοποιεί επίσης έλεγχο ροής δεδομένων, αναφορά λαθών στο επίπεδο εποπτείας και συναρτήσεις διατήρησης σύνδεσης. Στην μελέτη που έγινε σχετικά με την απόδοση του πρωτοκόλλου αυτού, αποδεικνύεται [Ntua02] ότι ένα μέρος μόνο από τις συναρτήσεις του SSCOP εκτελούνται πολύ συχνά στο πεδίο του χρόνου κατά τη διάρκεια της δημιουργίας ATM συνδέσεων. Η σημερινή τάση του σχεδιασμού συστημάτων σε ολοκληρωμένο κύκλωμα SoC[4], τείνει στην υλοποίηση συστημάτων τα οποία θα είναι επαρκής στο να εκτελέσουν διάφορα και πολύπλοκα πρωτόκολλα όπως είναι η σηματοδοσία των δικτύων ATM, το πρωτόκολλο TCP/IP κτλ. Ωστόσο για να επιτευχθούν οι περιορισμοί οι οποίοι σχετίζονται με την απόδοση των συστημάτων, συχνά οδηγούμαστε στην υλοποίηση πολλών πρωτοκόλλων σε υλικό. Η κατάτμηση που προτείνεται σε αυτήν την παράγραφο οδηγεί σε μία ανταλλαγή μεταξύ υλικού και λογισμικού, χωρίζοντας το εν λόγω πρωτόκολλο σε συναρτήσεις που εκτελούνται συχνά και σε αυτές που εκτελούνται πιο σπάνια.

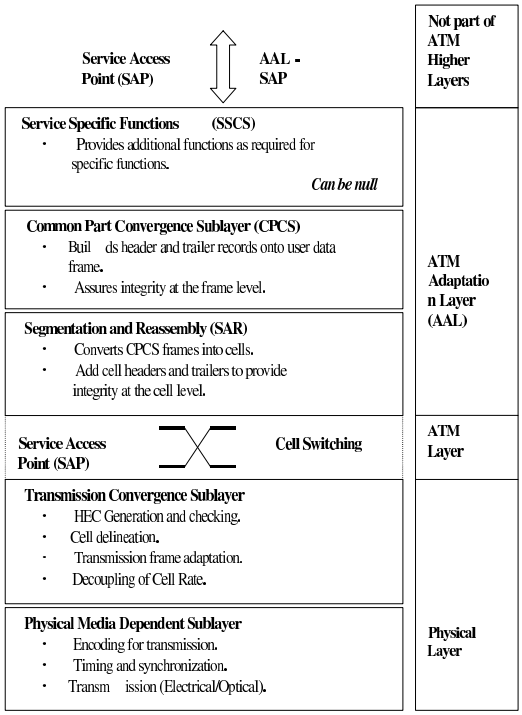
[3]Service Specific Connection Protocol

[4]System on a Chip

Οι συναρτήσεις που εκτελούνται συχνά και οι οποίες επίσης είναι περισσότερο απαιτητικές όσον αφορά την επεξεργαστική ισχύ, μπορούν να είναι υποψήφιες ώστε να υλοποιηθούν σε υλικό και ειδικότερα σε επεξεργαστές δικτύων.

Τα χαμηλότερα στρώματα των πρωτοκόλλων στις επικοινωνίες ευρείας ζώνης παραδοσιακά υλοποιούνται σε υλικό [Doum93]. Η εμφάνιση ισχυρών επεξεργαστών δικτύων έχει μετακινήσει το παραπάνω παράδειγμα σε υλοποιήσεις αποκλειστικά σε λογισμικό. Ωστόσο υπάρχουν μειονεκτήματα σε αυτήν την προσέγγιση εφόσον πολύπλοκα πρωτόκολλα σε επίπεδο bit παρουσιάζουν αυξημένη πολυπλοκότητα όταν υλοποιούνται σε λογισμικό. Οι παρακάτω παράγραφοι προτείνουν μία εναλλακτική προσέγγιση κατά την οποία τα πρωτόκολλα αναλύονται σε συναρτήσεις υλικού και λογισμικού κερδίζοντας έτσι σε ταχύτητα και πολυπλοκότητα.

Οι προκλήσεις για τους επεξεργαστές δικτύων σήμερα, είναι να φέρουν σε πέρας την ταχύτατη εκτέλεση πρωτοκόλλων (συμπεριλαμβανομένων και των υψηλών επιπέδων αυτών) καθώς και η υποστήριξη πολλαπλών ταυτόχρονων συνδέσεων. Το πρώτο βήμα προς αυτήν την κατεύθυνση, είναι ο προσδιορισμός ενός μοντέλου πρωτοκόλλου γενικών λειτουργιών. Συνήθως χρησιμοποιείται μία επίσημη γλώσσα περιγραφής, όπως π.χ. η SDL, για τον προσδιορισμό της λειτουργίας των πρωτοκόλλων. Το επόμενο βήμα είναι η ανάλυση των απαιτήσεων του συστήματος για κάθε μία εφαρμογή η οποία πρόκειται να υποστηριχθεί και η προσαρμογή του μοντέλου βάσει αυτής της πληροφορίας. Το αποτέλεσμα αυτού του βήματος είναι η βελτίωση του μοντέλου από την πλευρά της λειτουργικότητας, της απόδοσης και της δομής του. Το κέρδος της ανάλυσης αυτής θα παρέχει πληροφορία στην διαδικασία του λεπτομερή καθορισμού της αρχιτεκτονικής. Για το σκοπό αυτό, τα χαρακτηριστικά του κάθε μπλοκ του μοντέλου πρέπει να καθορισθεί. Αυτό που παρουσιάζεται παρακάτω είναι μία προσέγγιση για την υλοποίηση διαφόρων στοιχείων του πρωτοκόλλου τόσο σε



Σχήμα 4.10: Η θέση του SSCOP στη στοίβα σηματοδοσίας του ATM

υλικό όσο και σε λογισμικό, βρίσκοντας την χρυσή τομή μεταξύ αυτών των δύο, τηρώντας βέβαια πάντα τους περιορισμούς που έχει το εκάστοτε σύστημα.

4.5.1 Περιγραφή του πρωτοκόλλου SSCOP

Το SSCOP τοποθετείται μέσα στη στοίβα σηματοδοσίας του ATM όπως παρουσιάζεται στο Σχήμα 4.10.

Το SSCOP είναι ένα πρωτόκολλο στο επίπεδο μεταφοράς δεδομένων το οποίο παρέχει εγγυημένη, σε σειρά παράδοση μηνυμάτων στα υψηλότερα επίπεδα της σηματοδοσίας του ATM όπως είναι το Q.2931 [ITU95]. Το SSCOP ανήκει στην κλάση των σύγχρονων επιπέδου bit πρωτοκόλλων. Πραγματοποιεί την δημιουργία, τη συντήρηση και τον τερματισμό συνδέσεων σημείου

προς σημείου. Επίσης χρησιμοποιεί την μη εγγυημένη παράδοση υπηρεσία του κοινού τμήματος του SAAL ώστε να παρέχει αξιόπιστες υπηρεσίες μεταφοράς μεταβλητού μήκους δεδομένων, το οποίο φτάνει τα 65527 bytes, μεταξύ των χρηστών του δικτύου. Επιπλέον, είναι υπεύθυνο για τον εντοπισμό λαθών σε πακέτα τύπου AAL5 και προσθέτει λειτουργίες ανάκτησης και ελέγχου λαθών πάνω από το κοινό τμήμα του AAL5. Συγκεκριμένα οι λειτουργίες του είναι οι παρακάτω:

- *Ακεραιότητα ακολουθίας:* Αυτή η λειτουργία διατηρεί της σειρά των οντοτήτων SSCOP SDU τα οποία παραδίδονται για μεταφορά στο SSCOP.

- *Διόρθωση λαθών μέσω επιλεκτικής αναμετάδοσης:* Μέσω ενός σειριακού μηχανισμού, η οντότητα λήψης του SSCOP μπορεί να εντοπίσει χαμένα SSCOP SDU. Αυτή η λειτουργία διορθώνει λάθη διαδοχικότητας μέσω αναμετάδοσης.

- *Έλεγχος ροής:* Αυτή η λειτουργία επιτρέπει έναν δέκτη SSCOPνα ελέγχει τον ρυθμό με τον οποίο ο ομότιμος SSCOP εκπομπός στέλνει την πληροφορία.

- *Αναφορά λαθών στο επίπεδο διαχείρισης:* Η λειτουργία αυτή αναφέρει πιθανά λάθη στο επίπεδο της διαχείρισης του πρωτοκόλλου.

- *Διατήρηση σύνδεσης (keepalive):* Η λειτουργία αυτή επαληθεύει ότι οι δύο ομότιμες οντότητες του SSCOP που συμμετέχουν σε μία σύνδεση παραμένουν σε κατάσταση σύνδεσης ακόμη και στην περίπτωση της παρατεταμένης απουσίας μεταφοράς δεδομένων.

- *Ανάκτηση τοπικών δεδομένων:* Με την λειτουργία αυτή επιτρέπεται στον τοπικό χρήστη να ανακτήσει εν σειρά SDUs τα οποία δεν έχουν ακόμη αποδεσμευτεί από την μονάδα του SSCOP.

- *Έλεγχος συνδέσεων:* Πραγματοποιεί την δημιουργία, τον τερματισμό και τον επανασυγχρονισμό μίας σύνδεσης. Επίσης επιτρέπει την μετάδοση μεταβλητού μήκους πληροφοριών από χρήστη σε χρήστη χωρίς όμως να υπάρχει εγγύηση για την παράδοσή τους.

- *Μεταφορά δεδομένων χρήστη:* Χρησιμοποιείται ως μέσο μεταφοράς δεδομένων χρήστη μεταξύ χρηστών. Το SSCOP υποστηρίζει τόσο εγγυημένη όσο και μη εγγυημένη μεταφορά.

- *Ανίχνευση λαθών πρωτοκόλλου και ανάκτηση:* Με αυτήν την λειτουργία πραγματοποιείται η ανίχνευση και η ανάκτηση από λάθη τα οποία συνέβησαν κατά την λειτουργία του πρωτοκόλλου.

- *Αναφορά κατάστασης:* Επιτρέπει στις ομότιμες οντότητες των δεκτών και των εκπομπών να ανταλλάσσουν πληροφορίες κατάστασης.

4.5.2 Ορισμοί και μεθοδολογία

Αυτή η παράγραφος επικεντρώνεται στην ανάλυση της απόδοσης του πρωτοκόλλου SSCOP και προτείνει επίσης και μία μέθοδο υλοποίησης για την επιτάχυνση της εκτέλεσης του εν λόγω πρωτοκόλλου. Τα ληφθέντα αποτελέσματα από την ανάλυση αυτή μπορεί να οδηγήσουν τον σχεδιαστή σε μία αρχική ανάλυση σε δομικές μονάδες των συναρτήσεων που χρησιμοποιούνται, οι οποίες είναι κατάλληλες για υλοποίηση σε υλικό και συγκεκριμένα σε επεξεργαστές δικτύων. Για την ανάλυση αυτή χρησιμοποιήθηκε ένας κώδικας που περιγράφει την υλοποίηση του πρωτοκόλλου και το περιβάλλον λειτουργίας του είναι το λειτουργικό σύστημα VxWorks. Ωστόσο, για να γίνουν οι απαραίτητες μετρήσεις, ο κώδικας αυτόν τροποποιήθηκε ελάχιστα ώστε η μέτρηση της απόδοσης του και της πολυπλοκότητάς του, να μπορεί να γίνει σε έναν τυπικό επεξεργαστή Pentium και στο λειτουργικό σύστημα Linux. Ο κώδικας είναι

δομημένος σε διαφορετικά επίπεδα σύμφωνα πάντα με τα πρότυπα της ITU-T. Κάθε διαστρωμάτωση ανταλλάσσει μηνύματα με ένα από τα παραπάνω ή παρακάτω στρώματα μέσω των κλασσικών διεπαφών συνδέσεων sockets και συγκεκριμένα χρησιμοποιούν το πρωτόκολλο UDP για το σκοπό αυτό. Τα εργαλεία λογισμικού που χρησιμοποιήθηκαν για την ανάλυση των επιδόσεων του κώδικα ανήκουν στην οικογένεια των GNU Tools και ήταν ο μεταγλωττιστής *gcc* και για την ανάλυση το πρόγραμμα *gprof*. Για όλες τις μετρήσεις έγιναν οι παρακάτω υποθέσεις:

- Επιβεβαιώθηκαν και ελέχθησαν μόνο επιτυχημένες συνδέσεις σημείου προς σημείο. Εξομοιώσεις με διάφορες καταστάσεις λαθών δεν αξιολογήθηκαν, ακόμα και αν φαίνεται ότι τέτοιες περιπτώσεις εμπεριέχουν σημαντικές ανεπάρκειες του πρωτοκόλλου [Nug98].

- Τα μηνύματα σηματοδοσίας SETUP ήταν μηνύματα με σταθερά στοιχεία πληροφορίας IE[5].

- Η γεννήτρια κλήσεων παρήγαγε μηνύματα με σταθερό ρυθμό. Η δημιουργία μηνυμάτων με κατανομή Poisson δεν ήταν δυνατή λόγω της μορφής του κώδικα. Επιπλέον, όλες οι κλήσεις γίνονταν αποδεκτές και καμία πολιτική για ποιότητα υπηρεσιών δεν εφαρμόστηκε.

Μία επίσης μεθοδολογία για δοκιμές στην απόδοση της σηματοδοσίας ενός συστήματος παρουσιάζεται στο [Atm99].

Ως ορισμό για την έννοια της δοκιμασίας επιδόσεων (benchmarking) χρησιμοποιείται αυτός που παρουσιάζεται στο [Nieh97], στο οποίο παρουσιάζεται ως ένα σύνολο από πειράματα τα οποία μπορεί να χρησιμοποιηθούν για να αξιολογήσουν την απόδοση ενός συγκεκριμένου εξοπλισμού έναντι των προτύπων και των προδιαγραφών. Ο όρος εξοπλισμός στον παραπάνω ορισμό μπορεί να

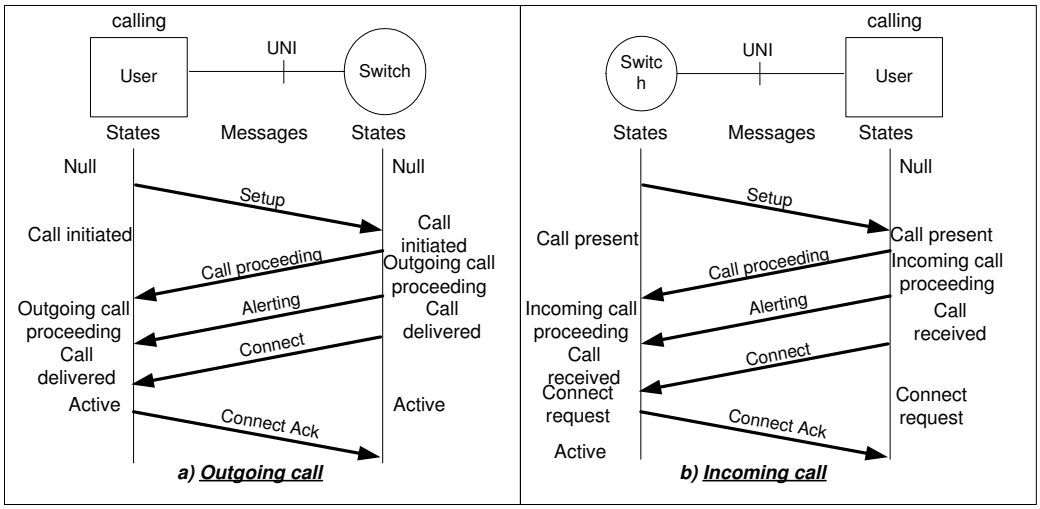
[5]Information Elements

αντικατασταθεί στη γενικότητά του από τον όρο σύστημα, εφόσον τα μέτρα που χρησιμοποιούνται για την ανάλυση σε μία μεθοδολογία ανάλυσης επιδόσεων, μπορεί να εμπλέκει διάφορα υποσυστήματα για το τελικό αποτέλεσμα. Για παράδειγμα αυτό συμβαίνει όταν μετρούνται αποδόσεις από άκρο σε άκρο μίας συγκεκριμένης εφαρμογής πάνω από ένα συγκεκριμένα σχηματισμένο δίκτυο. Μία μεθοδολογία ανάλυσης επιδόσεων ορίζεται από δύο σημαντικά στοιχεία:

- Από ένα σύνολο μέτρων αξιολόγησης και αποτίμησης.
- Από ένα σύνολο δοκιμών.

Το σύνολο των μέτρων τα οποία είναι σχετικά με την αξιολόγηση της απόδοσης θα πρέπει να επιλέγεται με τέτοιον τρόπο ώστε να επιβεβαιώνονται οι αρχικές προδιαγραφές και οι στόχοι στην απόδοση του συστήματος. Αυτά τα μέτρα επίσης αναφέρονται και στα αποτελέσματα της απόδοσης όπως αναφέρονται για άλλα συστήματα τα οποία είναι παρόμοια με το υπό δοκιμή σύστημα έτσι ώστε να χρησιμοποιηθούν για άμεση σύγκριση μεταξύ τους. Σχετικός είναι επίσης και ο ορισμός με αυτόν του συνόλου των μέτρων για την αξιολόγηση, με την μεθοδολογία μετρήσεων που ακολουθήθηκε. Η μεθοδολογία μετρήσεων πρέπει να επιλέγεται με τέτοιο τρόπο ώστε να έχει ελάχιστο αντίκτυπο στην απόδοση του συστήματος το οποίο βρίσκεται υπό δοκιμή και μελέτη. Το επιλεγμένο σύνολο θα πρέπει να καλύπτει ένα ευρύ φάσμα λειτουργικών και αντιπροσωπευτικών σεναρίων, έτσι ώστε να λαμβάνονται αντικειμενικά αποτελέσματα από την ανάλυση επιδόσεων του SSCOP για την περίπτωση που αναφέρεται σε αυτό το κείμενο. Ο σκοπός της δραστηριότητας της ανάλυσης επιδόσεων είναι:

- Να προσδιορίσει τα σημεία στα οποία δημιουργείται μποτιλιάρισμα (bottleneck), τα οποία δεν προκύπτουν άμεσα από τις προδιαγραφές του πρωτοκόλλου.



Σχήμα 4.11: Μοντέλο λειτουργάς της στοίβας σηματοδοσίας του ATM

- Να παρέχει όλα τα απαιτούμενα δυναμικά σύνολα.

- Να μετρήσει τους χρόνους εκτέλεσης του κώδικα.

4.5.3 Μετρήσεις επιδόσεων πρωτοκόλλου

Το Σχήμα 4.11 παρουσιάζει έναν τμήμα του διαγράμματος ροής της δημιουργίας μιας τυπικής σύνδεσης ATM. Για την λήψη των κατάλληλων μετρήσεων προσομοιώθηκε η λειτουργικότητα της πλευράς του δικτύου, αναπτύσσοντας έναν οδηγό ο οποίος αποκρίνεται στα μηνύματα εκκίνησης των κλήσεων.

Αυτή η διάρθρωση παρέχει ένα ελαφρύ σε επίπεδο επεξεργαστικής ισχύος περιβάλλον εξομοίωσης εφόσον δεν υπάρχει ανάγκη ύπαρξης και εκτέλεσης δύο διαφορετικών οντοτήτων που να εκτελούν τη σηματοδοσία του ATM. Για την λήψη αποτελεσμάτων επιδόσεων, ο κώδικας του επιπέδου του SSCOP μεταγλωττίστηκε με κατάλληλο τρόπο ώστε να υποστηρίζει ανάλυση πορείας και

κατάστασης[6]. Το πρόγραμμα που χρησιμοποιείται για την ανάλυση των επιδόσεων [Gprof00] παρέχει αποτελέσματα για τον αριθμό που καλείται κάθε φορά η εκάστοτε συνάρτηση στον κώδικα, καθώς επίσης και μία ανάλυση για το ποιες γραμμές του κώδικα χρησιμοποιήθηκαν κατά την εκτέλεσή του. Τα αποτελέσματα ελήφθησαν για έξι συνδέσεις σηματοδοσίας. Ο αναλυτής επιδόσεων της GNU μπορεί επίσης να παρέχει πληροφορία σχετική με τον χρόνο εκτέλεσης ανά συνάρτηση του κώδικα, αλλά η διακριτικότητα που μπορεί να επιτύχει είναι της τάξης των χιλιοστών του δευτερολέπτων, ενώ στην περίπτωση που μελετάμε, η διακριτικότητα πρέπει να είναι της τάξης των εκατομμυριοστών του δευτερολέπτου.

Ο Πίνακας 4.2 παρουσιάζει τα αποτελέσματα από την διαδικασία της ανάλυσης απόδοσης. Η πρώτη στήλη δείχνει το όνομα της συνάρτησης, η δεύτερη τον αριθμό των κλήσεων της συνάρτησης και η τρίτη παρουσιάζει το αν η εν λόγω συνάρτηση είναι απαιτητική ή όχι. Ο όρος "απαιτητική συνάρτηση" σημαίνει ότι η πολυπλοκότητα της συνάρτησης, υπό την έννοια των κύκλων σε έναν επεξεργαστή, είναι σημαντική.

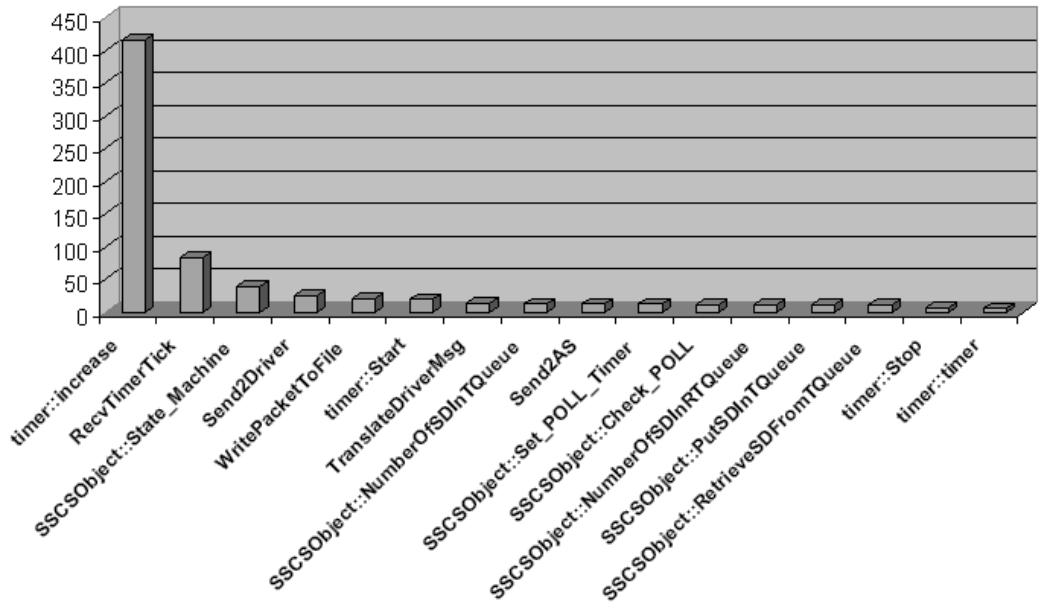
Συνεπώς αυτές οι συναρτήσεις είναι χρονοβόρες και η βελτιστοποίησή τους με την υλοποίησή τους σε υλικό, θα είχε σαν αποτέλεσμα σημαντικό κέρδος στην απόδοση του συστήματος. Το Σχήμα 4.12 παρουσιάζει τα ληφθέντα αποτελέσματα σε γραφική μορφή. Τα αποτελέσματα από την ανάλυση πορείας και κατάστασης επιδεικνύουν τις χρονοβόρες συναρτήσεις.

Βασιζόμενοι σε αυτές τις μετρήσεις είναι ξεκάθαρο ποιες συναρτήσεις θα πρέπει να εκτελούνται σε υλικό έτσι ώστε να επιτυγχάνεται η επιθυμητή επιτάχυνση στην εκτέλεση της σηματοδοσίας της στοίβας του ATM. Κυριότερα, οι προσβάσεις στους χρονομετρητές όπως επίσης και η λειτουργία των FSM φαίνεται να είναι τα σημεία που πιθανόν να εμφανίζονται προβλήματα απόδοσης σε

[6]Path-State Analysis

Συνάρτηση	**Κλήσεις**	**Απαιτητική**
timer::increase	415	NO
RecvTimerTick	83	NO
SSCSObject::State_Machine	38	YES
Send2Driver	24	YES
WritePacketToFile	21	NO
timer::Start	19	NO
TranslateDriverMsg	14	YES
SSCSObject::NumberOfSDInTQueue	12	NO
Send2AS	12	YES
SSCSObject::Set_POLL_Timer	12	NO
SSCSObject::Check_POLL	10	NO
SSCSObject::NumberOfSDInRTQueue	10	NO
SSCSObject::PutSDInTQueue	10	NO
SSCSObject::RetrieveSDFromTQueue	10	NO
timer::Stop	7	NO
timer::timer	5	NO

Πίνακας 4.2: Αριθμός κλήσεων ανά συνάρτηση

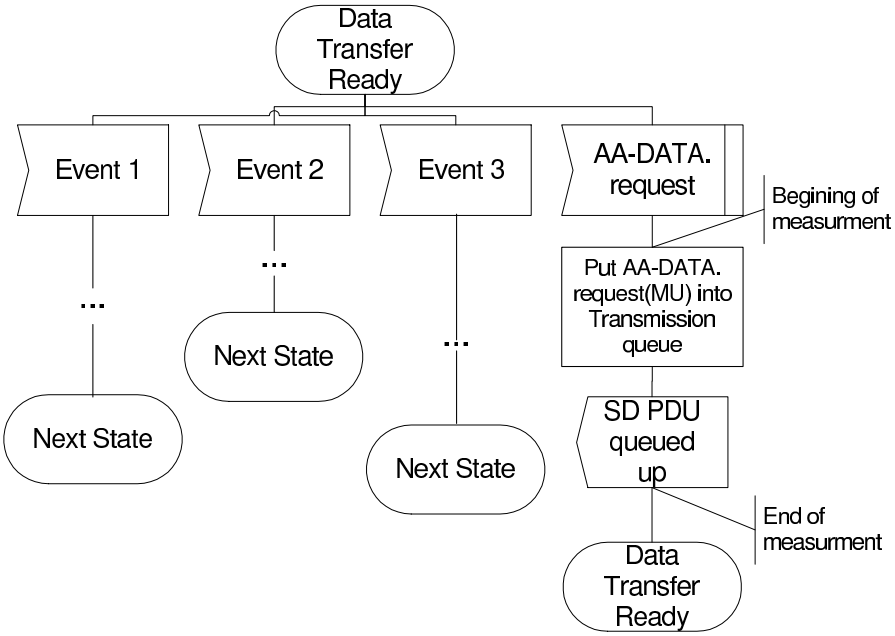


Σχήμα 4.12: Συχνότητα κλήσεων των συναρτήσεων του SSCOP

μία υλοποίηση σε λογισμικό. Επιπλέον, η αναγνώριση και ο προσδιορισμός των πακέτων στα χαμηλότερα επίπεδα απαιτεί ένα σημαντικό μέρος από τον επεξεργαστικό χρόνο. Λειτουργίες σχετικά με την διαχείριση των ουρών φαίνεται να είναι επίσης ένα πρόβλημα στις υλοποιήσεις σε λογισμικό.

4.5.4 Ανάλυση πρωτοκόλλου

Έχοντας ως δεδομένα τον κώδικα της στοίβας της σηματοδοσίας του πρωτοκόλλου ATM, καθώς επίσης και τον εικονικό οδηγό για την γεννήτρια κλήσεων τύπου ATM, η ανάλυση επιδόσεων παρέχεται με ακριβή αποτελέσματα όσον αφορά το SSCOP. Αυτά τα αποτελέσματα της ανάλυσης επιδόσεων μπορούν να μελετηθούν στη συνέχεια έτσι ώστε να προσδιορισθεί ποιες συναρτήσεις είναι κρίσιμες για την εκτέλεσή τους και κατά συνέπεια θα έπρεπε να αντιστοιχισθούν σε υλοποίηση σε υλικό. Ταυτόχρονα, προκύπτουν και οι συναρτήσεις οι



Σχήμα 4.13: Κλάδος σε μορφή SDL του SSCOP

οποίες οποίες θα μπορούσαν να εκτελούνται σε λογισμικό χωρίς να δημιουργούν προβλήματα απόδοσης. Αυτή η απόφαση βασίζεται επίσης στον διαχωρισμό μεταξύ των κανονικών καταστάσεων και αυτών που αποτελούν εξαιρέσεις του κανόνα ή αυτές που εκτελούνται κατά την αρχικοποίηση ενός συστήματος. Η παρακάτω μελέτη επικεντρώνεται σε συναρτήσεις οι οποίες εκτελούνται κατά τη διάρκεια της κανονικής, επομένως και συχνότερης, λειτουργίας. Αυτό το τμήμα αποτελεί το 90% των περιπτώσεων που συμβαίνουν σε ένα δίκτυο.

Το Σχήμα 4.13 παρουσιάζει ένα SDL κλάδο όπως παρουσιάζονται στα πρότυπα της ITU-T. Σε αυτήν την ανάλυση οι μετρήσεις χρόνου πραγματοποιήθηκαν μόνο για την λειτουργία του επιλεγμένου κλάδου, ο οποίος ξεκινά από την κατάσταση Data Transfer Ready και καταλήγει πάλι στην κατάσταση Data Transfer Ready. Αυτή άλλωστε είναι και η πιο κοινή περίπτωση που μπορεί να συμβεί στο SSCOP πρωτόκολλο εφόσον το δίκτυο συμπεριφέρεται φυσιο-

λογικά. Οι διαδρομές του δέκτη και του πομπού διαχωρίστηκαν έτσι ώστε να μετρηθούν κάθε φορά συγκεκριμένες διαδρομές είτε αυτό είναι του δέκτη είτε του πομπού. Συνοπτικά, αναφέρεται μόνο ότι κάποιες παράμετροι είναι μη υπολογίσιμοι και σαφώς επιβραδύνουν αρκετά την εκτέλεση του πρωτοκόλλου. Αυτές οι παράμετροι είναι οι εξής:

- Φόρτος από το λειτουργικό σύστημα.
- Πεπερασμένο μήκος των μη κυκλικών απομονωτών.
- Χρονοβόρες λειτουργίες για την διαχείριση των ιδεατών ουρών.

4.5.5 Επιδόσεις πρωτοκόλλου

Τα αποτελέσματα τα οποία προέκυψαν από την ανάλυση που παρουσιάστηκε στο Σχήμα 4.12 δεν διευκρινίζει συγκεκριμένα θέματα σχετικά με την απόδοση του συστήματος. Εφόσον αυτά τα αποτελέσματα είναι εξαρτώμενα από το σύστημα και δεν είναι βελτιστοποιημένα ανάλογα με την προτεινόμενη αρχιτεκτονική, χρησιμοποιήθηκε μία εκτενέστατη ανάλυση απόδοσης. Οι κρίσιμες συναρτήσεις και λειτουργίες που μετρήθηκαν επιλέχθηκαν βασιζόμενοι στην αρχική ανάλυση του SSCOP πρωτοκόλλου. Τα βασικά χαρακτηριστικά αυτής της ανάλυσης είναι:

- Προσαρμογή των απαιτήσεων της ανάλυσης επιδόσεων στο εκάστοτε λειτουργικό περιβάλλον.
- Ενσωμάτωση θεμάτων κατάτμησης υλικού και λογισμικού όπως τα παρακάτω:
  - Προϋποθέτουμε ότι η αρχικοποίηση και ο τερματισμός της διασύνδεσης αποτελούν σπάνια εμφανιζόμενες συναρτήσεις συγκρινόμενες με αυτές της μεταφοράς δεδομένων.

Τύπος πακέτου	**Μέσο μήκος σε bytes**
SETUP	60
CONNECT_ACK	9
CALL_PROCEEDING	18
CONNECT	9

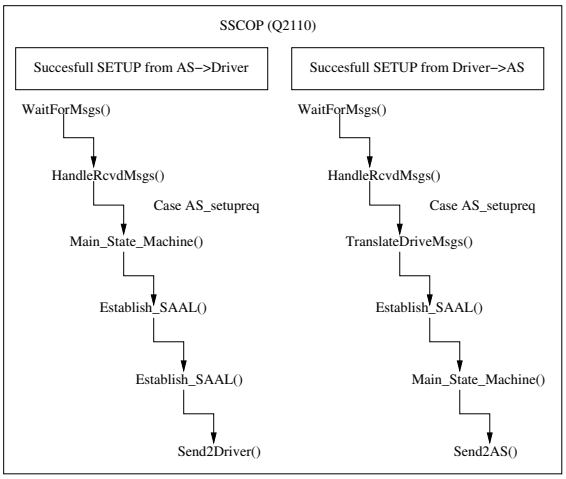
Πίνακας 4.3: Τυπικά μήκη των ανταλλασσόμενων πακέτων

- Οι κανονικές λειτουργίες (ανταλλαγή δεδομένων και παρακολούθηση διασύνδεσης) θα πρέπει να υλοποιηθούν σε υλικό.
- Οι εξαιρέσεις (όπως λάθη στη γραμμή, συντήρηση της διασύνδεσης) θα πρέπει να υλοποιηθούν σε υψηλού επιπέδου λογισμικό.

Συνοψίζοντας καταλήγουμε στα εξής:

- Οι μετρήσεις για την ανάλυση των επιδόσεων πραγματοποιήθηκαν μόνο για τη διαστρωμάτωση του SSCOP.
- Οι εν λόγω μετρήσεις ελήφθησαν μόνο για την πλευρά της διεπαφής του χρήστη UNI[7].
- Οι αποκρίσεις του ιδεατού οδηγού της γεννήτριας κλήσεων ATM ξεκινούσαν χωρίς καμία καθυστέρηση.
- Τα αποτελέσματα τα σχετικά με το χρόνο ερευνήθηκαν μόνο για το εσωτερικό συγκεκριμένων συναρτήσεων.
- Επιπρόσθετες καθυστερήσεις (λόγω της υλοποίησης του πρωτοκόλλου σε λογισμικό), ελήφθησαν υπόψη.

[7]User Network Interface



Σχήμα 4.14: Τυπική ροή για τον πομπό και τον δέκτη του SSCOP

- Η στοίβα πρωτοκόλλου δεν ήταν βελτιστοποιημένη κατά το μέγιστο όσον αφορά την απόδοση.

Κατά την ανάλυση επιδόσεων του κώδικα χρησιμοποιήθηκαν τέσσερις διαφορετικοί τύποι πακέτων για την μεταφορά δεδομένων. Αυτοί οι τύποι όπως επίσης και το μέσο μήκος τους φαίνονται στον Πίνακα 4.3. Στο Σχήμα 4.14 παρουσιάζεται ένα τυπικό διάγραμμα ροής των συναρτήσεων που εμπλέκονται στην λειτουργία του πρωτοκόλλου τόσο για τον δέκτη όσο και για τον πομπό.

Κατά τη διάρκεια μίας τυπικής διαδικασίας σύνδεσης, κατά τη λήψη ενός πακέτου τύπου SETUP, η οντότητα της σηματοδοσίας παράγει ένα πακέτο τύπου CALL_PROCEEDING έτσι ώστε να πληροφορήσει την ισότιμη οντότητα ότι το πακέτο παραλήφθηκε, επεξεργάστηκε κατάλληλα και προωθήθηκε στο επόμενο κόμβο του δικτύου. Ο χρόνος που μεσολαβεί από την λήψη του πακέτου καθώς επίσης και ο χρόνος που απαιτείται για την δημιουργία του SETUP πακέτου, έχουν υπολογισθεί βασιζόμενοι στα αποτελέσματα χρόνου που αναφέρονται στην προηγούμενη παράγραφο.

Συνάρτηση	PUTSDINTQUEUE		SDMSGFROMAS		SEND2DRIVER	
Τύπος μηνύματος	SETUP	CONNECT_ACK	SETUP	CONNECT_ACK	SETUP	CONNECT_ACK
1	81	20	78	19	4	2
2	20	22	21	19	3	2
3	22	21	21	19	2	2
4	57	28	52	30	3	2
5	27	28	28	27	2	2
6	26	26	26	26	2	2
Μέση τιμή	**38.83**	**24.17**	**37.67**	**23.33**	**2.67**	**2.00**

Πίνακας 4.4: Ληφθέντες χρόνοι για την ροή μετάδοσης

Συνάρτηση	RETREIVESDFROMRQUEUE		SDMSGFROMNW		SEND2AS	
Τύπος μηνύματος	CALL_PROC.	CONNECT	CALL_PROC.	CONNECT	CALL_PROC.	CONNECT
1	7	3	135	89	170	88
2	2	2	127	89	127	89
3	3	2	126	88	127	87
4	7	3	179	88	171	88
5	3	3	125	86	124	87
6	3	3	123	87	123	87
Μέση τιμή	**4.17**	**2.67**	**135.83**	**87.83**	**140.33**	**87.67**

Πίνακας 4.5: Ληφθέντες χρόνοι για την ροή λήψης

Στους Πίνακες 4.4 και 4.5, παρουσιάζονται οι μέσες τιμές που μετρήθηκαν των έξι διαφορετικών σεναρίων που χρησιμοποιήθηκαν. Ο Πίνακας 4.4 αντιστοιχεί για τη ροή δεδομένων του πομπού ενώ ο Πίνακας 4.5 στη ροή δεδομένων του δέκτη. Ο διαχωρισμός μεταξύ των ροών του δέκτη και του πομπού γίνεται λόγω του ότι σε επίπεδο υλικού ο δέκτης και ο πομπός είναι ανεξάρτητα στοιχεία και μπορούν να αλληλεπιδρούν και να λειτουργούν ταυτόχρονα την ίδια χρονική στιγμή.

Βασιζόμενοι σε αυτές τις μετρήσεις, οι παρερχόμενοι χρόνοι για τη δημιουργία και τη διεργασία ενός πακέτου τύπου SETUP και CALL_PROCEEDING παρουσιάζεται στον Πίνακα 4.6.

Εκτελέσιμη ενέργεια	**Παρερχόμενος χρόνος (σε μsec)**
Επεξεργασία του εισερχόμενου πακέτου SETUP	724.16
Αποστολή ενός πακέτου CALL_PROCEEDING	358.82
Αποστολή ενός πακέτου SETUP	389.99
Συνολικός χρόνος εκτέλεσης	**1472.97**

Πίνακας 4.6: Παρερχόμενος χρόνος για τα μηνύματα σηματοδοσίας

4.6 Ανάλυση του ATM στο PRO^3

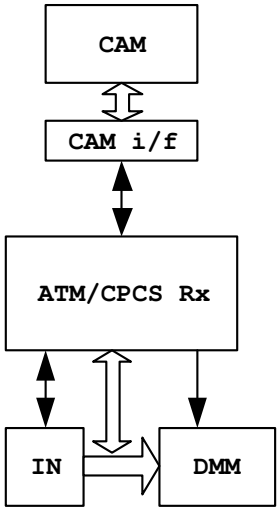
Όπως παρουσιάστηκε στην παράγραφο 4.5, τα υψηλότερα επίπεδα του πρωτοκόλλου του ATM υλοποιούνται σε έναν από τους πυρήνες (επεξεργαστές) που έχουν ενσωματωθεί στο σύστημα του PRO^3. Στη συνέχεια παρουσιάζουμε μία ανάλυση των χαμηλότερων επιπέδων του πρωτοκόλλου όπως αυτά του πομπού και του δέκτη του ATM μαζί με τα αποτελέσματα στην απόδοσή τους.

4.6.1 Δέκτης και πομπός ATM/CPCS

Τα μπλοκ του ATM/CPCS είναι υπεύθυνα για την πραγματοποίηση των λειτουργιών των επιπέδων του ATM και του AAL5 του πρωτοκόλλου του ATM. Όπως φαίνεται στα Σχήματα 4.15 και 4.16 τα κανάλια διαδρομών του δέκτη και του πομπού είναι διαχωρισμένα. Για κάθε ένα από τα δύο μπλοκ περιγράφονται τόσο η *λειτουργικότητα* όσο και θέματα *απόδοσης*.

Λειτουργία του ATM-Rx-Tx

Όταν το σύστημα του PRO^3 λειτουργεί σε ένα δίκτυο ATM, τα cells που λαμβάνονται από το IN μπλοκ προωθούνται στον διαχειριστή μνήμης δεδομένων (DMM) και στη συνέχεια αντιγράφονται στον δέκτη του ATM/CPCS. Το μπλοκ του DMM αποθηκεύει το εισερχόμενο cell σε έναν προσωρινό, επί του



Σχήμα 4.15: Μπλοκ λήψης ATM/CPCS

ολοκληρωμένου κυκλώματος, απομωνοτή (buffer). Το μπλοκ του ATM/CPCS επεξεργάζεται το cell και στέλνει την κατάλληλη εντολή στον DMM έτσι ώστε να εισαχθεί και να αποθηκευθεί στην κατάλληλη ουρά. Σε περίπτωση που δεν ληφθεί καμία εντολή, το μπλοκ του DMM γράφει πάνω από το τελευταίως λαμβανόμενο cell.

Η βασική λειτουργία του εν λόγω μπλοκ διαιρείται σε τέσσερις λειτουργίες· *εξαγωγή των πεδίων του επικεφαλίδας του cell, κατηγοριοποίηση της ροής του ATM, επιβεβαίωση του CRC και του μήκους, και η επανασυναρμολόγηση των cells.* Οι λειτουργίες παρουσιάζονται στη συνέχεια με μεγαλύτερη λεπτομέρεια.

- **Εξαγωγή των πεδίων του επικεφαλίδας του cell**: Το μπλοκ εισόδου IN στέλνει το cell στο ATM/CPCS μαζί με την ταυτότητα της πόρτας από όπου εισήχθη (PID[8]). Το PID είναι στην πραγματικότητα ένας αριθμός μήκους 10 bit. Το μπλοκ εξάγει στη συνέχεια την πληροφορία του VP, VC, την ταυτότητα του τύπου του ωφέλιμου μέρους του

[8]Port IDentification

Κλειδί	Απάντηση			
VP, VC, PTI, PID	CID	Ένδειξη	CRC	Μήκος
41 bits	16 bits	2 bits	32 bits	16 bits

Πίνακας 4.7: Λέξεις επικοινωνίας με την CAM

cell (PTI[9]) από την επικεφαλίδα και αυτή η πληροφορία αποστέλλεται κατάλληλα προς την μονάδα της μνήμης CAM για κατηγοριοποίηση.

- **Κατηγοριοποίηση της ροής του ATM**: Η απάντηση της μνήμης CAM σε περίπτωση έγκυρων τιμών, είναι η ταυτότητα της σύνδεσης (CID[10]) η οποία έχει μήκος 16 bit και μία ένδειξη 2 bit για το αν το εν λόγω cell είναι cell σηματοδοσίας ή όχι. Από την πλευρά του DMM το CID αντιστοιχεί σε μία από τις χίλιες υποστηριζόμενες ουρές του συστήματος για συνδέσεις σηματοδοσίας. Το μπλοκ εξετάζει την απάντηση και αν το cell είναι έγκυρο, τότε εμφανίζεται μία εντολή προς τον DMM ώστε να αποθηκεύσει το cell στην αντίστοιχη ουρά.

- **Επιβεβαίωση του CRC και του μήκους**: Οι προδιαγραφές που έχουν οριστεί για το CPCS περιλαμβάνουν την επιβεβαίωση του μήκους και του CRC για κάθε εισερχόμενο cell. Για την επιτάχυνση της συγκεκριμένης λειτουργίας η υλοποίησης πραγματοποιείται παράλληλα. Τα ενδιάμεσα αποτελέσματα, το σύνδρομο του CRC και η τιμή του μήκους αποθηκεύονται στην μνήμη CAM μέχρις ότου ληφθεί και το τελευταίο cell του πακέτου. Κατά την λήψη του τελευταίου cell του πακέτου, το ATM/CPCS πραγματοποιεί την επιβεβαίωση του CRC και του μήκους του πακέτου. Σε περίπτωση όπου μία από τις δύο παραπάνω παραμέτρους δεν επιβε-

[9]Payload Type Identification
[10]Connection Identifier

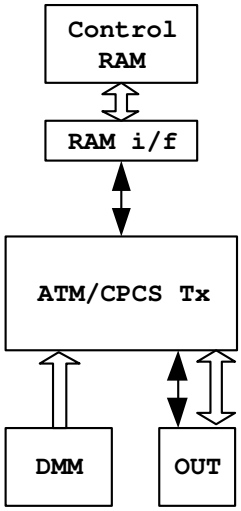
βαιωθεί και επιστρέψει λάθος, τότε ο δέκτης του ATM αποστέλλει μία εντολή προς τον DMM όπου διατάζει την διαγραφή του πακέτου από τις μνήμες.

- **Επανασυναρμολόγηση των cells**: Ο DMM αποθηκεύει τα cells του κάθε πακέτου στην κατάλληλη ουρά σύμφωνα με την ταυτότητα σύνδεσης. Όταν ληφθεί το τελευταίο cell και επίσης δεν υπάρχει και κανένα λάθος όσον αφορά το CRC και το μήκος, το μπλοκ στέλνει μία εντολή ώστε να αποθηκεύσει το cell και να δημιουργήσει το πακέτο. Μαζί με την εντολή, το μπλοκ στέλνει επίσης το μήκος του SSCOP πακέτου και το μήκος των έγκυρων δεδομένων στο τελευταίο cell. Η πληροφορία του μήκος απαιτείται από τα υπόλοιπα μπλοκ του συστήματος για να γνωρίζουν την έγκυρη πληροφορία σε επίπεδο SSCOP.

Όσον αφορά τον πομπό του ATM, το επίπεδο του SSCOP χρησιμοποιεί το μπλοκ του DMM για να αποθηκεύει τα πακέτα τα οποία κατευθύνονται προς το δίκτυο. Το μπλοκ του ATM δέχεται από τον DMM τα αντίστοιχα τμήματα των δεδομένων και το αντίστοιχο CID. Στην περίπτωση όπου το τμήμα αυτό είναι το τελευταίο του πακέτου, το μπλοκ λαμβάνει την κατάλληλη ειδοποίηση και το μήκος της έγκυρης πληροφορίας. Ο πομπός του ATM/CPCS είναι πλέον υπεύθυνος να δημιουργήσει το cell και να το αποστείλει στο δίκτυο. Η πληροφορία η οποία σχετίζεται με το κάθε πακέτο λαμβάνεται από την μνήμη ελέγχου Control RAM.

Η βασική λειτουργία του μπλοκ αυτού διαιρείται σε τέσσερις λειτουργίες· *δημιουργία της επικεφαλίδας του cell, δημιουργία του cell και μετάδοσή του, υπολογισμός του CRC και του μήκους του και η κατασκευή του trailer.* Η λεπτομερής περιγραφή των παραπάνω παρουσιάζεται στη συνέχεια:

- **Δημιουργία των πεδίων του επικεφαλίδας του cell**: Κατά την



Σχήμα 4.16: Μπλοκ εκπομπής ATM/CPCS

λήψη του CID το ATM/CPCS προσπελάζει την μνήμη ελέγχου στην διεύθυνση που αντιστοιχεί βάσει του CID, ώστε να βρει τις αντίστοιχες τιμές των VP, VC, PTI και PID. Αυτές οι τιμές είναι αρκετές για να κατασκευασθεί η επικεφαλίδα.

- **Δημιουργία του cell και μετάδοση**: Το μπλοκ δημιουργεί την επικεφαλίδα και αποστέλλει το cell στο δίκτυο. Η πληροφορία του PID αποστέλλεται στο πεδίο του HEC στην επικεφαλίδα του cell.

- **Υπολογισμός του CRC και του μήκους**: Το πρωτόκολλο του CPCS απαιτεί τον υπολογισμό του CRC και του μήκους ολόκληρου του πακέτου CPCS. Αυτοί οι υπολογισμοί πραγματοποιείται ανά τμήμα του πακέτου. Μαζί με την πληροφορία της επικεφαλίδας, το σύνδρομο του CRC και η τιμή του μήκους του πακέτου αποθηκεύονται στην μνήμη ελέγχου σύμφωνα πάντα με το CID. Όταν το μπλοκ λάβει το τελευταίο τμήμα του πακέτου, υπολογίζει το CRC και το μήκος του πακέτου και στη συνέχεια

αρχικοποιεί τις τιμές στην μνήμη ελέγχου για το συγκεκριμένο CID.

- **Δημιουργία του CPCS trailer**: Το μπλοκ έχει την πληροφορία για το μήκος του πακέτου και έτσι μπορεί και κατασκευάζει τον αριθμό των τμημάτων που το αποτελούν. Αν το μήκος του προσθέτοντας 8 bytes από το trailer του CPCS είναι πολλαπλάσιο του 48, τότε το μπλοκ προσθέτει το trailer στο τελευταίο τμήμα του πακέτου. Στην αντίθετη περίπτωση το μπλοκ στέλνει προς την μνήμη του DMM ένα ακόμη cell το οποίο όμως θα αποτελείται μόνο από μηδενικά και από το trailer απλά και μόνο για να δεσμεύσει τον απαραίτητο χώρο.

Όσον αφορά την απόδοση και των δύο αυτών μπλοκ, για την υποστήριξη ρυθμού δεδομένου στα 2.4 Gbps ο χρόνος του κάθε cell ισοδυναμεί με 176.4 nsec. Για ένα ρολόι του συστήματος στα 200 MHz υπάρχουν 35 κύκλοι στη διάθεση τους για την όλη επεξεργασία του κάθε cell.

4.7 Κατάτμηση υλικού λογισμικού στο PRO^3

Βασιζόμενοι στην ανάλυση απόδοσης που έχει γίνει στην παράγραφο 4.5, προκύπτει μία κατάτμηση σε υλικό και λογισμικό για την εφαρμογή του ATM στο σύστημα του PRO^3. Η κατάτμηση σε υλικό και σε λογισμικό αναφέρεται στο βήμα εκείνο της ροής του παράλληλου σχεδιασμού σε υλικό και σε λογισμικό όπου λαμβάνονται σημαντικές αποφάσεις για την υλοποίηση λειτουργικών ομάδων σε αντίστοιχα μπλοκ. Τα άλλα βήματα σε έναν παράλληλο σχεδιασμό υλικού και λογισμικού περιλαμβάνουν την παράλληλη προσομοίωση του συστήματος και την σύνθεση των διεπαφών μεταξύ υλικού και λογισμικού. Υπάρχουν διάφορα εργαλεία τα οποία χρησιμοποιούνται για τέτοιους σκοπούς όπως για παράδειγμα το CoWare [Co97] το οποίο αναπτύχθηκε από την IMEC. Άλλα τέτοια εργαλεία είναι το Lycos [Lyc96] και το [Cos98]. Ορισμένα από αυτά

τα εργαλεία αποτελούν εμπορικές εφαρμογές και άλλα αποτελέσματα ακαδημαϊκών ινστιτούτων. Ωστόσο, όσον αφορά τα ακαδημαϊκά δεν ήταν διαθέσιμα για να μελετηθεί η απόδοση τους και ο τρόπος λειτουργίας τους. Αντιθέτως υπάρχει ένα ακαδημαϊκό εργαλείο που ονομάζεται SCE και χρησιμοποιήθηκε για το σκοπό και την ανάλυση των αποτελεσμάτων που παρουσιάζονται σε αυτό το κεφάλαιο. Εφόσον η ανάλυση του πρωτοκόλλου SSCOP έχει γίνει με τρόπο ώστε να είναι ξεκάθαρη η αντιστοίχιση του σε μπλοκ λογισμικού (εξαιρώντας τους χρονομετρητές λόγω υψηλών απαιτήσεων), στη συνέχεια γίνεται μία ανάλυση του χαμηλότερου επιπέδου του ATM, δηλαδή του ATM/CPCS, για την εκτίμηση των απαιτήσεων των συναρτήσεων που υλοποιεί το πρωτόκολλο.

Η λειτουργία του συστήματος του PRO^3 υλοποιείται υπό την έννοια ενός αριθμού διασυνδεόμενων μπλοκ, τα οποία αντιστοιχίζονται στο να εκτελούν ένα επιλεγμένο σύνολο λειτουργιών. Τα μπλοκ τα οποία πρόκειται να φιλοξενήσουν τις λειτουργίες των διαφόρων λειτουργιών είναι ένας προγραμματιζόμενος επεξεργαστής τύπου RISC της εταιρείας Hyperstone [Hy03] για την εκτέλεση λειτουργιών σε λογισμικό, το RPM για την εκτέλεση μικροκώδικα και συγκεκριμένα μπλοκ υλικού ειδικά για την εκτέλεση και μόνο λειτουργικότητας του πρωτοκόλλου απευθείας σε υλικό. Αυτό σημαίνει ότι για τα μπλοκ τα οποία είναι υπό μελέτη θα πρέπει να ληφθεί μία απόφαση κατάτμησης για το που τελικά θα εκτελούνται οι συγκεκριμένες λειτουργίες του πρωτοκόλλου. Εξετάζεται δηλαδή το αν είναι αποδοτικότερο να εκτελεσθεί ένα τμήμα σε υλικό, στο RPM ή στον RISC επεξεργαστή. Ο στόχος είναι η κατάτμηση της πλήρους εφαρμογής μεταξύ αυτών των τριών οντοτήτων επεξεργασίας. Η έρευνα αυτή ξεκινά με την αναζήτηση των τμημάτων της εφαρμογής τα οποία μπορούν να μετακινηθούν και να εκτελούνται σε μπλοκ λογισμικού, έτσι ώστε να αυξηθεί η προσαρμοσιμότητα του συστήματος του PRO^3. Η διαδικασία της κατάτμησης οδηγείται συνήθως από έναν αριθμό μέτρων έτσι ώστε να αποδείξουν και

ποσοτικά την ποιότητα της επιλογής μίας κατάτμησης χρησιμοποιώντας μία συνάρτηση βαρών.

Τα μέτρα που χρησιμοποιούνται για την επιλογή της κατάτμησης είναι τα παρακάτω:

- **Καθυστέρηση**: Γενικότερα, μία υλοποίηση σε λογισμικό έχει μικρότερη απόδοση από ότι μία ισοδύναμη υλοποίηση σε υλικό. Κατά συνέπεια, αναζητούνται συνήθως οι συναρτήσεις οι οποίες είναι απαιτητικές από άποψη επεξεργαστικής ισχύος έτσι ώστε να αντιστοιχισθεί η εκτέλεση τους σε υλικό, καθώς οι λειτουργίες οι οποίες δεν είναι τόσο απαιτητικές και επίσης ίσως είναι και αρκετά περίπλοκες συναρτήσεις να εκτελούνται σε λογισμικό. Παρατηρώντας το σύστημα από την οπτική γωνία του επεξεργαστή δικτύου, η ύπαρξη διαφόρων μπλοκ υλικού συνεπάγεται την μείωση του φόρτου του. Ωστόσο, ειδικά για το σύστημα του PRO^3 επικεντρωνόμαστε στους περιορισμούς της καθυστέρησης και όχι τόσο στους άλλους παράγοντες όπως είναι η κατανάλωση ισχύος ή η επιφάνεια.

- **Επιφάνεια**: Εφόσον ένας μικροεπεξεργαστής είναι μία συσκευή γενικής χρήσεως και περιλαμβάνει αρκετή λειτουργικότητα η οποία μπορεί να χρησιμοποιείται ή όχι από την εφαρμογή που θέλουμε να εκτελέσουμε (π.χ. οι εντολές για την επεξεργασία ψηφιακών σημάτων του επεξεργαστή της Hyperstone), η επιφάνεια που θα καταλαμβάνεται δεν είναι πολύ αποδοτική. Επίσης απαιτείται ένα σημαντικό ποσό μνήμης για να μπορεί να εγκατασταθεί η πληροφορία του προγράμματος προς εκτέλεση καθώς και τα δεδομένα που πρόκειται να επεξεργαστούν. Η επιφάνεια που καταλαμβάνεται από τις μνήμες σχετίζεται άμεσα με τους χρονικούς περιορισμούς των μπλοκ του λογισμικού. Οι χρόνοι πρόσβασης για την λήψη των εντολών του επεξεργαστή στην πραγματικότητα επηρεάζουν και καθορίζουν σημαντικά την τελική ταχύτητα του επεξεργαστή εκτός

και αν χρησιμοποιούνται ιεραρχικά επίπεδα μνήμης. Έτσι, ακόμη και αν το μέγεθος του μικροεπεξεργαστή μπορεί να είναι αρκετά μικρό, πρέπει να λαμβάνεται σοβαρά υπόψη και η μνήμη για το πρόγραμμα και για τα δεδομένα (στην περίπτωση που η μνήμη αυτή βρίσκεται εντός του ολοκληρωμένου κυκλώματος). Μία επιλογή που μπορεί να γίνει είναι αυτή ανάμεσα σε μνήμες τύπου SRAM και DRAM, ανάλογα με το επιθυμητό σύνολο εντολών και τη συχνότητα δειγματοληψίας των δεδομένων.

- **Ισχύς**: Όσον αφορά την ισχύ τέτοιων συστημάτων εντός ολοκληρωμένου κυκλώματος, τα στοιχεία που καταναλώνουν μεγάλες ποσότητες ενέργειας είναι η δομή του δίαυλου (λόγω της μεγάλης του επιφάνειας και των απαιτήσεων για οδήγηση), οι μνήμες αποθήκευσης και τα κυκλώματα χρονισμού.

4.8 Η γλώσσα SpecC

Για το κοντινό μέλλον, οι πρόσφατες προβλέψεις της τεχνολογίας των ημιαγωγών πυριτίου, συμφωνούν στο ότι ο αριθμός των τρανζίστορ σε ένα ολοκληρωμένο κύκλωμα θα αυξάνεται με εκθετικό τρόπο σύμφωνα με το νόμο του Moore, οδηγώντας την τεχνολογία ολοένα και περισσότερο προς την εποχή του συστήματος εντός ολοκληρωμένου κυκλώματος ή System on a chip (SoC). Ωστόσο, η εμπειρία δείχνει ένα ολοένα και αυξανόμενο χάσμα μεταξύ της αύξησης της πολυπλοκότητας του συστήματος και της αύξησης των διαθέσιμων τεχνικών στη συγκεκριμένη τεχνολογία. Σε συνδυασμό με την ανάγκη για γρήγορη παράδοση των προϊόντων στην αγορά, οδηγούμαστε στην ανάγκη για καινοτόμα μέτρα έτσι ώστε να αυξηθεί η παραγωγικότητα στον σχεδιασμό κατά κάποιες τάξεις μεγέθους.

Είναι κοινά παραδεκτό ότι για να επιτευχθεί μία τόσο μεγάλη ξαφνική μεταλ-

λαγή στην παραγωγικότητα του σχεδιασμού, είναι απαραίτητο να επικεντρωθούμε στη διαδικασία του σχεδιασμού σε *υψηλότερα επίπεδα αφαίρεσης* από την μία και από την άλλη στην *επαναχρησιμοποίηση* των πολύπλοκων στοιχείων ενός συστήματος (intelectual property, IP). Για να είναι επιτυχημένα τα αποτελέσματα ενός τέτοιου σχεδιασμού πρέπει και οι δύο συλλήψεις ιδεών να υιοθετούν νέες γλώσσες και μεθοδολογίες για το σχεδιασμό των συστημάτων, υποβοηθούμενες από τα αντίστοιχα υψηλού επιπέδου εργαλεία αυτοματοποίησης συστημάτων.

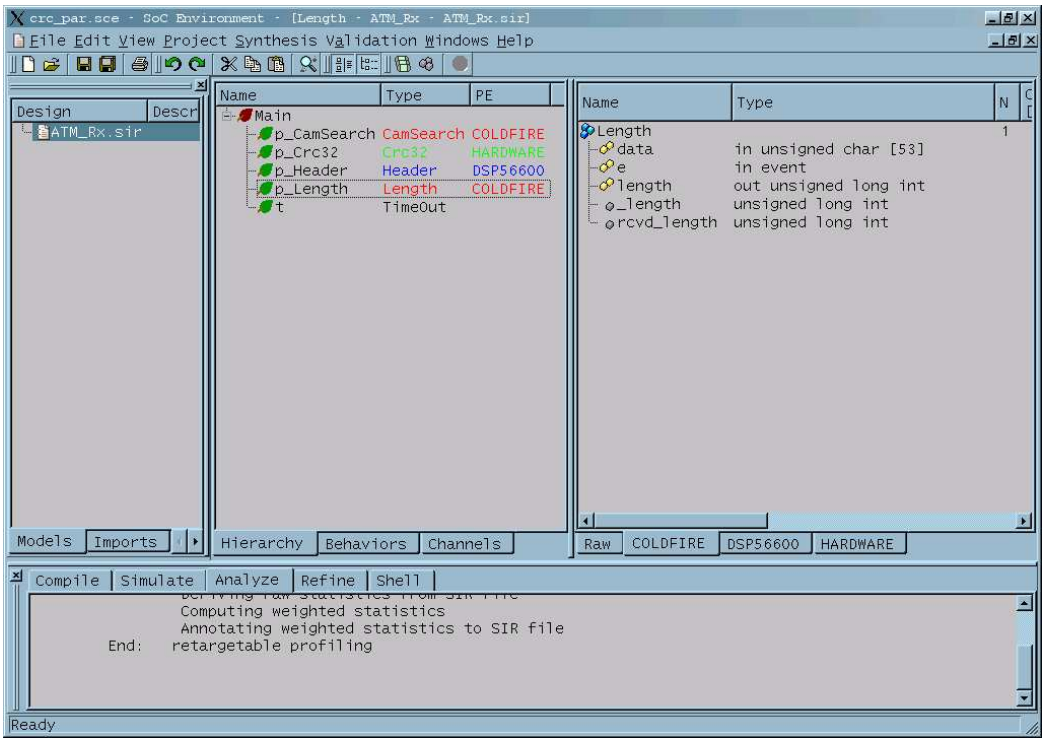
Η γλώσσα SpecC [DGGP01] στοχεύει στις προδιαγραφές και τον σχεδιασμό ενός SoC ή ενός ενσωματωμένου συστήματος συμπεριλαμβανομένου του υλικού και του λογισμικού, είτε χρησιμοποιώντας σταθερές πλατφόρμες, ολοκληρώνοντας συστήματα από διαφορετικά IPs, ή ακόμη και συνθέτοντας τα μπλοκ του συστήματος από γλώσσες περιγραφής προγραμματισμού ή υλικού. Η μεθοδολογία της SpecC οδηγεί του σχεδιαστές απο τις εκτελέσιμες προδιαγραφές του συστήματος σε μία υλοποίηση RTL μέσω μία καλά ορισμένης αλληλουχίας βημάτων. Κάθε μοντέλο περιγράφεται και οι οδηγίες σχεδιασμού του δίδονται για την παραγωγή αυτών των μοντέλων από εκτελέσιμες προδιαγραφές. Η "κοινότητα" των σχεδιαστών εισάγεται στην εποχή του σχεδιασμού σε επίπεδο αφαίρεσης και η SpecC αποτελεί το στοιχείο για την ενεργοποίηση ώστε να επιτευχθεί ένα παράδειγμα στο σχεδιασμό που απαιτείται για συστήματα ή/και προϊόντα που σχεδιάζονται. Η γλώσσα προδιαγραφών και η μεθοδολογία της SpecC αποτελεί αντικείμενο ενδιαφέροντος σε ερευνητές, σχεδιαστές όσον αφορά την μοντελοποίηση και το σχεδιασμό συστημάτων.

4.9 Το περιβάλλον SCE

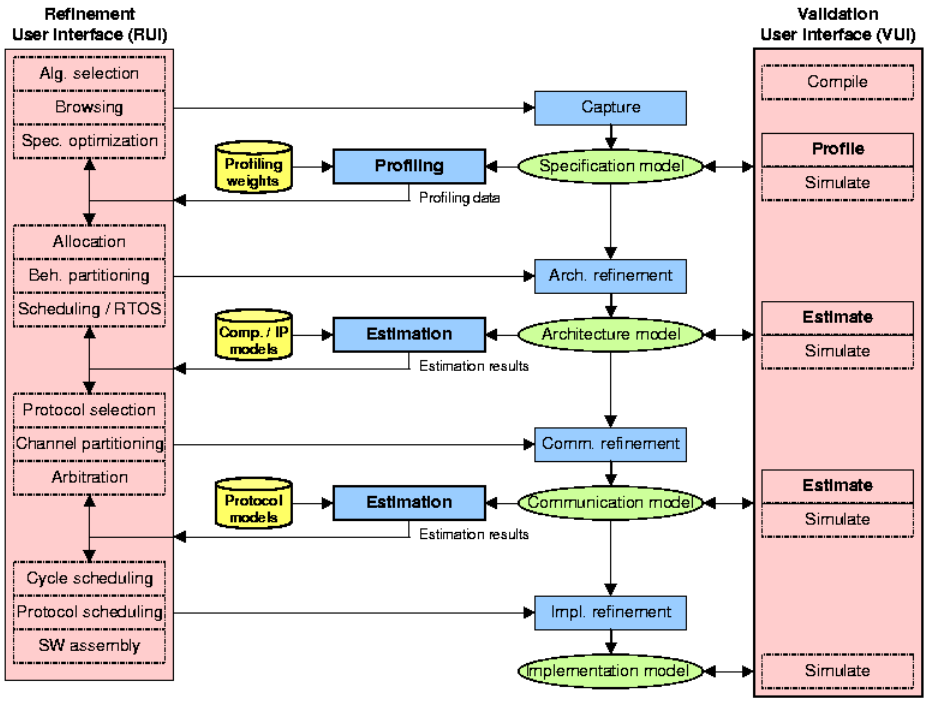
Η δυνατότητα ενσωμάτωσης ενός συστήματος σε ένα ολοκληρωμένο σύστημα, εισάγει νέες προκλήσεις στη διαδικασία της σχεδίασης. Πρώτον, ο ταυτόχρονος σχεδιασμός φαίνεται να αποτελεί ένα κρίσιμο πλέον θέμα. Το υλικό και το λογισμικό πρέπει να αναπτύσσονται παράλληλα. Ωστόσο, τόσο οι σχεδιαστές υλικού όσο και αυτοί του λογισμικού έχουν διαφορετικές απόψεις του συστήματος και χρησιμοποιούν διαφορετικές τεχνικές μοντελοποίησης και σχεδιασμού. Κατά δεύτερον, η διαδικασία του σχεδιασμού του συστήματος από τις προδιαγραφές έως τις μάσκες είναι μακρά και αρκετά περίπλοκη. Για αυτόν τον λόγο η διαδικασία πρέπει να χωρισθεί σε διάφορα βήματα. Σε κάθε βήμα σχεδιασμού πρέπει να γράφονται μοντέλα του συστήματος και οι σχετικές ιδιότητές τους να επαληθεύονται. Τρίτον, οι σχεδιαστές των συστημάτων δεν είναι πολλές φορές δυνατόν να γνωρίζουν πολλές γλώσσες. Ακόμη περισσότερο, το να γράφονται διαφορετικά μοντέλα για κάθε βήμα του σχεδιασμού και να γίνεται η επαλήθευσή τους είναι αρκετά επίπονη διαδικασία. Για τους λόγους αυτούς πρέπει να αποφεύγεται η επαναληπτική χειροκίνητη δουλειά για την δημιουργία κάθε φορά επιτυχημένων μοντέλων· αντιθέτως η μοντελοποίηση αυτή θα ήταν πολύ πιο βολική αν μπορούσε να γίνει με κάποιον αυτόματο κατά μία έννοια τρόπο. Στο Σχήμα 4.17 παρουσιάζεται η μορφή του περιβάλλοντος του λογισμικού SCE.

Το περιβάλλον SCE[11] αντιπροσωπεύει μία νέα τεχνολογία η οποία επιτρέπει την καταγραφή των προδιαγραφών ενός συστήματος ως μία σύνθεση από συναρτήσεις και ρουτίνες της γλώσσας C. Αυτές οι συναρτήσεις επέρχονται βελτίωση μέσω αυτού του εργαλείου [SCE03], αναπτύσσοντας διαφορετικά μοντέλα τα οποία απαιτούνται σε κάθε βήμα της διαδικασίας σχεδιασμού. Κατά συνέπεια οι σχεδιαστές μπορούν να επικεντρωθούν στη δημιουργία του σχε-

[11] System on a Chip Environment



Σχήμα 4.17: Το περιβάλλον του SCE



Σχήμα 4.18: Μηχανισμός λειτουργία του SCE

διασμού και τα εργαλεία είναι αυτά τα οποία θα δημιουργήσουν τα απαραίτητα μοντέλα για την επαλήθευση και την σύνθεσή τους. Το κέρδος από το περιβάλλον αυτό είναι ότι οι σχεδιαστές δεν χρειάζεται να μάθουν μία νέα γλώσσα προγραμματισμού για το σχεδιασμό, ή γλώσσες περιγραφής υλικού όπως π.χ. την VHDL. Ως αποτέλεσμα, οι σχεδιαστές οφείλουν να εισάγουν τις προδιαγραφές του συστήματος και να λάβουν τις απαραίτητες αποφάσεις σε αμφίδρομη συνεργασία με το περιβάλλον SCE. Τα μοντέλα για την προσομοίωση, τη σύνθεση και την επαλήθευση παράγονται με αυτοματοποιημένες διαδικασίες.

Η μεθοδολογία του σχεδιασμού ενός SoC παρουσιάζεται στο Σχήμα 4.18. Αποτελείται από τέσσερα επίπεδα αφαίρεσης του μοντέλου, τα οποία είναι των προδιαγραφών, της αρχιτεκτονικής, της επικοινωνίας και της υλοποίησης. Συνεπώς υπάρχουν και τρία βήματα βελτιστοποίησης από το εργαλείο τα οποία

είναι τα αντίστοιχα της αρχιτεκτονικής, της επικοινωνίας και της υλοποίησης. Αυτά τα βήματα βελτιστοποίησης πραγματοποιούνται με τη σειρά την οποία και παρουσιάστηκαν. Όπως φαίνεται και στο Σχήμα 4.18 στην αρχή ξεκινάμε με ένα μοντέλο αφαίρεσης των προδιαγραφών. Το μοντέλο αυτό δεν περιέχει καμία πληροφορία σχετική με τον χρόνο και περιέχει μόνο την λειτουργική συμπεριφορά του σχεδιασμού.

Η βελτιστοποίηση της αρχιτεκτονικής μετατρέπει τις προδιαγραφές αυτές σε ένα αρχιτεκτονικό μοντέλο. Εμπλέκει την κατάτμηση του σχεδιασμού καθώς και την αντιστοίχηση των κατατμήσεων σε καθορισμένα στοιχεία. Το αρχιτεκτονικό μοντέλο κατά συνέπεια βελτιώνει την αρχική σύλληψη των προδιαγραφών. Το επόμενο βήμα για την βελτιστοποίηση της επικοινωνίας προσθέτει δίαυλους στο σύστημα και αντιστοιχεί τις αφαιρετικές διασυνδέσεις σε στοιχεία που αντιστοιχίζονται σε διαύλους. Το αποτέλεσμα είναι η χρονική συμπεριφορά του συστήματος ακόμη και σε επίπεδο επικοινωνίας των υποσυστημάτων μεταξύ τους. Το τελικό βήμα είναι η βελτιστοποίηση της υλοποίησης η οποία παράγει μοντέλα RTL με ακρίβεια ρολογιού για τα στοιχεία του υλικού καθώς και κώδικα σετ εντολών για τους επεξεργαστές που χρησιμοποιούνται. Όλα τα μοντέλα είναι εκτελέσιμα και μπορούν να επαληθευθούν μέσω προσομοίωσης.

4.10 Ανάλυση του ATM μέσω του SCE

Στο σημείο αυτό, ερευνάται η καλύτερη λύση κατάτμησης για το πρωτόκολλο του ATM σε έναν επεξεργαστή δικτύων όπως είναι και το PRO^3. Γενικότερα, όταν επιδιώκεται να πραγματοποιηθεί κατάτμηση σε μία λειτουργία ή σε ένα πρωτόκολλο, όταν στόχος είναι η υλοποίηση του σε ένα ενσωματωμένο σύστημα, τότε η τάση είναι να υλοποιηθούν όσον το δυνατόν περισσότερες λειτουργίες σε λογισμικό. Η υλοποίηση σε λογισμικό είναι πάντοτε πιο εύ-

κολη από πλευράς σχεδιασμού και χρόνου ανάπτυξης. Επιπλέον, οι πιθανές απαιτήσεις για τροποποίηση της λογικής του προγράμματος μπορεί να γίνει σε οποιαδήποτε στιγμή χωρίς μεγάλο κόστος. Αντιθέτως, όταν σχεδιάζεται ένα τμήμα μίας λειτουργίας σε υλικό (π.χ. σε ASIC), εκτός του ότι η διαδικασία είναι αρκετά πιο περίπλοκη [Ntua03], η αλλαγή ενός πιθανού λάθους ή μίας μικρής επανασχεδίασης, θα απαιτούσε την όλη διαδικασία να ξεκινήσει σχεδόν από την αρχή. Για τον λόγο αυτό οι περισσότεροι σχεδιαστές προσπαθούν να μετακινήσουν το μεγαλύτερο δυνατό μέρος της λειτουργίας σε λογισμικό ώστε να αποφεύγονται τέτοιου είδους προβλήματα και δυσκολίες.

Όταν όμως μία λειτουργία πρέπει να φιλοξενηθεί, (hosted) σε έναν επεξεργαστή ή DSP που πιθανόν ενσωματώνεται σε ένα σύστημα εντός ολοκληρωμένου κυκλώματος, υπάρχουν αρκετοί περιορισμοί. Οι επεξεργαστές έχουν βασικά περιορισμένη υπολογιστική ισχύ με αποτέλεσμα πολλές φορές να μην είναι ικανοί στο να καλύψουν τα κριτήρια και τα μέτρα που θέτει ένας σχεδιαστής. Για παράδειγμα, όταν μία τηλεπικοινωνιακή εφαρμογή απαιτεί να εκτελέσει έναν μεγάλο ή μικρό αριθμό λειτουργιών σε ένα δεδομένο χρονικό διάστημα, τότε αυτό μπορεί να μην είναι δυνατό από τον επεξεργαστή που χρησιμοποιείται από το σύστημα. Μία τέτοια κατάσταση θα οδηγούσε αμέσως στον αποκλεισμό της λειτουργίας από αυτόν τον επεξεργαστή. Συνήθως, μία υλοποίηση σε υλικό με την σημερινή τεχνολογία μπορεί να λύσει τέτοιου είδους προβλήματα απόδοσης. Ωστόσο, επειδή είναι αρκετά περίπλοκο να υλοποιηθούν μεγάλα κομμάτια ενός πρωτοκόλλου για παράδειγμα σε υλικό, γίνεται προσπάθεια να βρεθεί η τομή μεταξύ των λειτουργιών που υλοποιούνται σε υλικό και αυτών που υλοποιούνται σε λογισμικό. Οι περιορισμοί βέβαια δεν σταματούν στην υπολογιστική ισχύ αλλά επεκτείνονται στην περιορισμένη μνήμη που διαθέτουν αυτοί οι επεξεργαστές, στην επιφάνεια που καταλαμβάνουν καθώς και στην ισχύ την οποία καταναλώνουν. Σε μία τηλεπικοινωνιακή εφαρμογή όπως είναι το PRO^3 αυτό

που ενδιαφέρει κατά κύριο λόγο δεν είναι η κατανάλωση της ισχύος ούτε η επιφάνεια· το επίκεντρο του ενδιαφέροντος δίδεται στην απόδοση του συστήματος εφόσον η σχεδίαση πρέπει να υποστηρίζει ρυθμό δεδομένων στα 2.4 Gbps.

Πολλοί αλγόριθμοι για την γενική φιλοσοφία της κατάτμησης υποστηρίζουν πως μία τέτοια κατάτμηση θα πρέπει να ξεκινάει από υλοποίηση που θα γίνεται πλήρως σε λογισμικό. Αυτές οι προσεγγίσεις αποτελούν μία οικογένεια αλγορίθμων οι οποίοι στηρίζονται σε αυτό που λέμε all software implementation, ή εξόλοκλήρου υλοποίηση σε λογισμικό. Μέσω μίας τέτοιας προσέγγισης μπορεί να εκτιμηθεί με ακρίβεια ο χρόνος ή ακόμη καλύτερα οι κύκλοι που απαιτούνται για την υλοποίηση της λειτουργίας, με αποτέλεσμα την κρίση του σχεδιαστή για το κατά πόσο αυτό είναι εφικτό με τα συγκεκριμένα στοιχεία που είναι διαθέσιμα. Για τον λόγο αυτό πρώτα από όλα πρέπει η λειτουργία αυτή να περιγραφεί σε μία γλώσσα προδιαγραφών, όπως είναι η SpecC, για να γίνει στη συνέχεια αυτή η διερεύνηση.

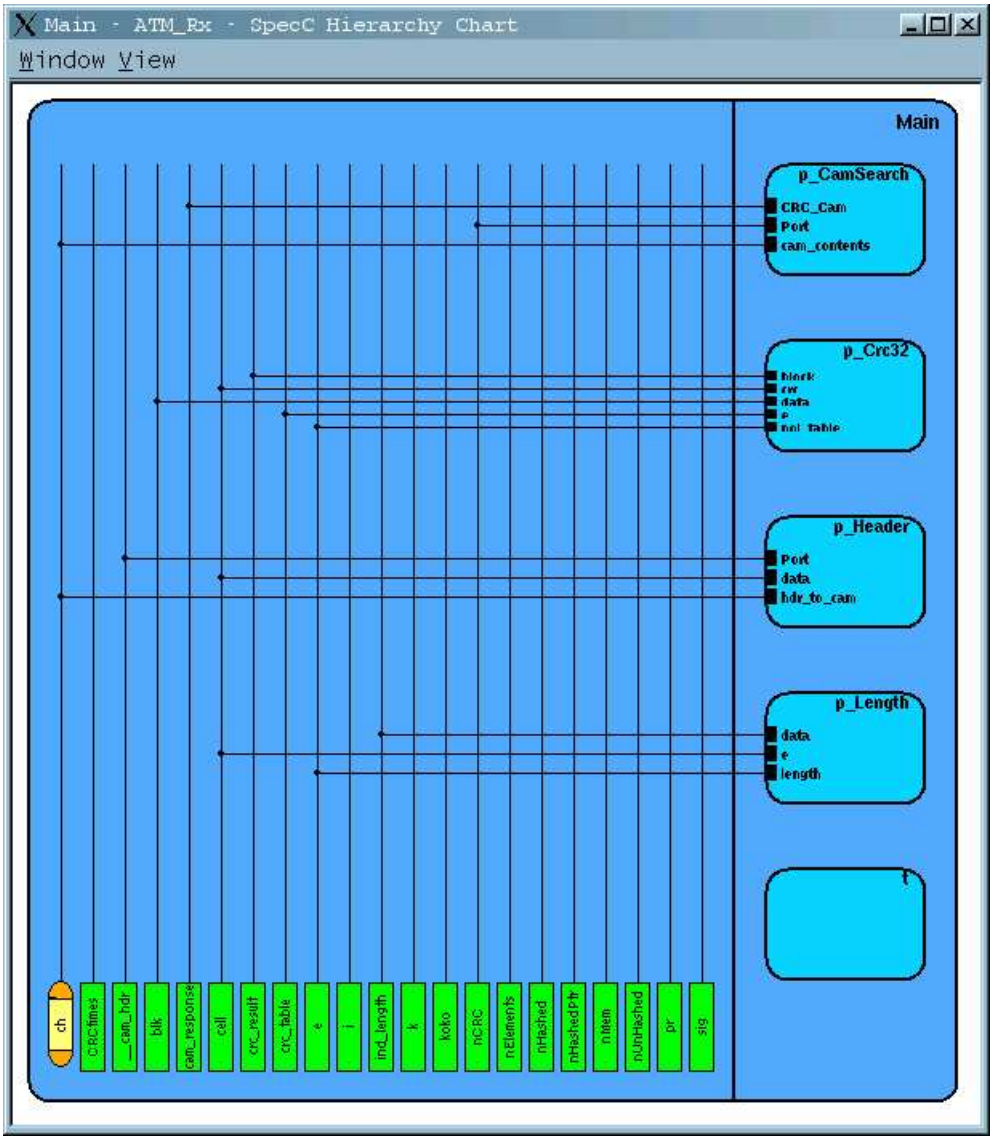
Δεδομένης της ανάλυσης απόδοσης του πρωτοκόλλου SSCOP όπως έγινε στην παράγραφο 4.5, το μεγαλύτερο μέρος του φαίνεται να εκτελείται σε επίπεδο μικροκώδικα σε επεξεργαστή. Η αναζητούμε μίας λύσης όσον αφορά την κατάτμηση για το ATM, αποτελεί μία διαδικασία στην οποία ερευνάται να βρεθεί αν οι απαιτήσεις του μπορούν να εξυπηρετηθούν από τον επεξεργαστή του συστήματος. Η λειτουργίες του ATM τόσο για τον δέκτη όσο και για τον πομπό έχουν τις ίδιες απαιτήσεις, οπότε παρουσιάζεται η μία μόνο διαδρομή και αυτή είναι του δέκτη. Οι απαιτήσεις για απόδοση του ATM σε έναν επεξεργαστή δικτύων όπως είναι το PRO^3 είναι αρκετά μεγάλες μιας και ο χρόνος που είναι διαθέσιμος για επεξεργασία είναι πολύ μικρός. Μία απλή ανάλυση του ρυθμού δεδομένων στο σύστημα, μας δίνει ως αποτέλεσμα, ότι το διαθέσιμο χρονικό διάστημα για επεξεργασία είναι μόνο 176 ns περίπου. Αυτό σχετίζεται άμεσα με τον χρόνο άφιξης μεταξύ των ATM cells. Ένα ATM cell μαζί με

την επικεφαλίδα έχουν μήκος 53 bytes. Αυτό μεταφράζεται σε 424 bits εκ των οποίων τα 284 είναι τα δεδομένα του cell (payload) και 40 είναι η επικεφαλίδα. Για τα 2.4 Gbps που πρέπει να υποστηρίζει το PRO^3 ο χρόνος άφιξης των cells είναι κάθε 176 ns. Εφόσον το ρολόι που χρονίζει το όλο σύστημα του PRO^3 είναι της τάξης των 200 MHz, αυτό σημαίνει πως μία επεξεργαστική μονάδα που λειτουργεί σε αυτή την συχνότητα υπάρχουν διαθέσιμοι 35 περίπου κύκλοι για την διεκπεραίωση της επεξεργασίας. Οτιδήποτε περισσότερο από πλευράς κύκλων ή χρόνου θα μείωνε την απόδοση του συστήματος δημιουργώντας σήματα καθυστέρησης, τα φερόμενα και ως back-pressure σήματα. Φυσικά αν η υλοποίηση κάποιας λειτουργίας πραγματοποιηθεί σε υλικό, τότε υπάρχει η δυνατότητα της χρήσης τεχνικών συνεχούς διοχέτευσης (pipeline), μέσω της οποίας ο διαθέσιμος χρόνος μπορεί να αξιοποιηθεί πιο αποδοτικά. Η λειτουργία του δέκτη του ATM στο επίπεδο που μελετάμε (ATM/CPCS), αποτελείται από τις λειτουργίες της *κατηγοριοποίησης* μέσω της μνήμης CAM, της *επαλήθευσης του μήκους* και της εκτέλεσης του *CRC32*. Ειδικά για την μοντελοποίηση της κατηγοριοποίησης, έχει χρησιμοποιηθεί μία καινούρια μέθοδος [Cam03] η οποία χρησιμοποιεί αλγορίθμους hashing για την αφαίρεση του στοιχείου της CAM.

Ξεκινώντας λοιπόν από μία πλήρη υλοποίηση σε λογισμικό, γίνεται μία ανάλυση σχετικά με την απόδοση της λειτουργίας σε έναν επεξεργαστή. Στο συγκεκριμένο περιβάλλον το αρχικό μοντέλο των προδιαγραφών σε SpecC, παρουσιάζει μία εκτίμηση των κύκλων ρολογιού που απαιτούνται από την κάθε μία λειτουργία. Στη συνέχεια εφαρμόζουμε το παρακάτω μοντέλο σε έναν τυπικό επεξεργαστή για να δούμε μία πιο ρεαλιστική συμπεριφορά του μοντέλου. Προσπαθούμε να χρησιμοποιήσουμε έναν επεξεργαστή όσον το δυνατόν πιο κοντά, από πλευράς απόδοσης και συμπεριφοράς, με τον E1-32XS της Hyperstone ώστε να έχουμε πιο ρεαλιστικά αποτελέσματα. Όπως φαίνεται από το

Σχήμα 4.19, το μοντέλο του ATM δέκτη αποτελείται από τις λειτουργίες των *CamSearch* (p_CamSearch), *Length* (p_Length), *Header* (p_Header), *CRC32* (p_Crc32) και *TimeOut* (t). Στο συγκεκριμένο Σχήμα φαίνονται επίσης και οι διασυνδέσεις των στοιχείων μεταξύ τους καθώς και τα μεταξύ τους σήματα ελέγχου events.

Η πρώτη συνάρτηση (*CamSearch*) υλοποιεί μία νέα μέθοδο στην αναζήτηση κλειδιών πολύ μεγάλου μήκους σε εφαρμογές δικτυακών επεξεργαστών. Συνήθως οι επεξεργαστές δικτύων χρησιμοποιούν μία ειδική μονάδα για την κατηγοριοποίηση των λαμβανόμενων πακέτων σύμφωνα με τα χαρακτηριστικά του. Η μέθοδος που παρουσιάζεται στο [Cam03] προτείνει την αντικατάσταση της μνήμης τύπου CAM που χρησιμοποιείται σε τέτοιες εφαρμογές, από συμβατικές μνήμες τύπου RAM διατηρώντας έτσι το κόστος πάρα πολύ χαμηλό. Ωστόσο, η ανάλυση της συνάρτησης μέσω του εργαλείου που μελετάμε, θα δείξει το κατά πόσο αυτή η συνάρτηση θα είναι αρκετά γρήγορη ώστε να μπορεί να εξυπηρετήσει τον απαιτούμενο ρυθμό δεδομένων. Οι λειτουργίες της εξαγωγής της επικεφαλίδας καθώς και η επαλήθευση του λάθους αποτελούν τις λιγότερο χρονοβόρες συναρτήσεις του συστήματος που πιθανόν δεν θα δημιουργήσουν προβλήματα στην απόδοσή τους. Στη συνέχεια υπάρχει μία συνάρτηση η οποία εκτελεί το CRC32. Αυτή είναι από τις πιο κρίσιμες συναρτήσεις όσον αφορά την πολυπλοκότητά της και τις απαιτήσεις που πιθανόν να εμφανίσει για τόσο μεγάλους ρυθμούς όπως αυτός των 2.4 Gbps. Τέλος, για την καλύτερη εποπτεία του συστήματος έχει υλοποιηθεί μία συνάρτηση η οποία ομομάζεται *TimeOut* και λειτουργεί ως μέσο για να ειδοποιείται το συνολικό σύστημα ότι ο διαθέσιμος χρόνος για επεξεργασία έχει λήξει. Στην μελέτη της πολυπλοκότητας του ATM ώστε να πραγματοποιηθεί η κατάτμηση δεν λαμβάνεται φυσικά υπόψην αυτή η λειτουργία. Η λειτουργία του *TimeOut* λειτουργεί ως ένα thread λογισμικού· δηλαδή εκτελείται ανεξάρτητα από οποιαδήποτε άλλη λειτουργία του



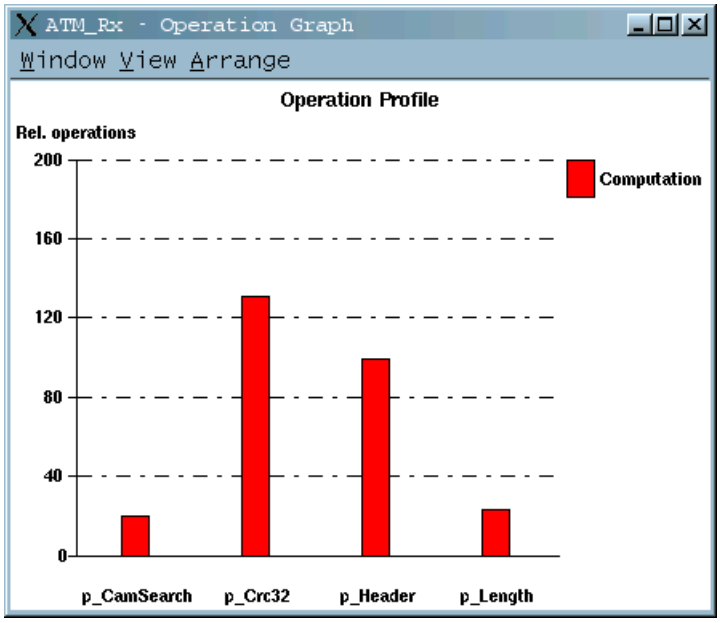
Σχήμα 4.19: Δομή του ATM στο SCE

Λειτουργίες	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Επικεφαλίδα	⋆													
Φορτίο cell			⋆											
Κατηγοριοποίηση			⋆											
CRC32				⋆										
Επαλ. Μήκους														⋆

Πίνακας 4.8: Χρονική αλληλουχία των λειτουργιών του ATM

συστήματος. Τα υπόλοιπα στοιχεία του συστήματος διασυνδέονται μεταξύ τους μέσω καναλιών τα οποία μεταφέρουν εκτός από την πληροφορία που θέλουν να ανταλλάξουν τα υποσυστήματα (συναρτήσεις) και τα απαραίτητα σήματα συγχρονισμού. Για παράδειγμα, η κατηγοριοποίηση μπορεί να ξεκινήσει μόλις ληφθεί η επικεφαλίδα ενώ το CRC32 μπορεί να ξεκινήσει μόλις ληφθεί η πρώτη 32 bit λέξη του cell. Η χρονική αλληλουχία των λειτουργιών φαίνεται πιο ξεκάθαρα στον Πίνακα 4.8. Τα αστεράκια παρουσιάζουν τη χρονική στιγμή μέσα στο δεδομένο παράθυρο χρόνου που ξεκινά η εν λόγω λειτουργία. Έτσι με μία καλή χρονοδρομολόγηση όσον αφορά την έναρξη της κάθε λειτουργίας σε συνδυασμό με κάποιο μηχανισμό συνεχούς διοχέτευσης το σύστημα δεν χρειάζεται να εκτελέσει υποχρεωτικά σειριακά τις λειτουργίες αυτές.

Το σύστημα για την πρώτη προσπάθεια εκτίμησης χωρίς να υπάρχει κάποια συγκεκριμένη πλατφόρμα επεξεργαστή ως τελικός στόχος αποδίδει τα παρακάτω αποτελέσματα όπως αυτά φαίνονται στο Σχήμα 4.20. Τα αποτελέσματα αυτά παρέχουν μία πρώτη εποπτεία της σχετικής πολυπλοκότητας των υπό υλοποίηση λειτουργιών. Η πρώτη εκτίμηση δίνει κάποια αποτελέσματα· ωστόσο όπως θα αποδειχθεί στη συνέχεια δεν αποτελούν ένα αρκετά ακριβές μέτρο για την απόλυτη πολυπλοκότητα μιας και οι τιμές που παρουσιάζονται απέχουν



Σχήμα 4.20: Ανάλυση απόδοσης του ATM

κατά πολύ από την τελική. Για παράδειγμα στο Σχήμα 4.20 παρουσιάζεται ότι μόνο η επεξεργασία της επικεφαλίδας απαιτεί περίπου εκατό κύκλους επεξεργασίας, τη στιγμή που στη διάθεση μας υπάρχουν περίπου 35 για να εκτελεσθούν όλες οι λειτουργίες.

Για τον λόγο αυτό δεν αναλύονται ποσοτικά οι πολυπλοκότητες που παρουσιάζονται στο Σχήμα 4.20 αλλά παρατίθενται μόνο και μόνο για να εκφραστεί η σχετική πολυπλοκότητα των λειτουργιών μεταξύ τους. Επίσης, μία σημαντική πληροφορία που αντλείται από το γράφημα του Σχήματος 4.20 είναι το γεγονός ότι η πιο περίπλοκη διαδικασία είναι αυτή της διεργασίας του υπολογισμού του CRC32. Ως αποτέλεσμα, απαραίτητη προϋπόθεση είναι τουλάχιστον η διεργασία του CRC32 να μπορεί να εκτελεστεί σε 35 το πολύ κύκλους[12].

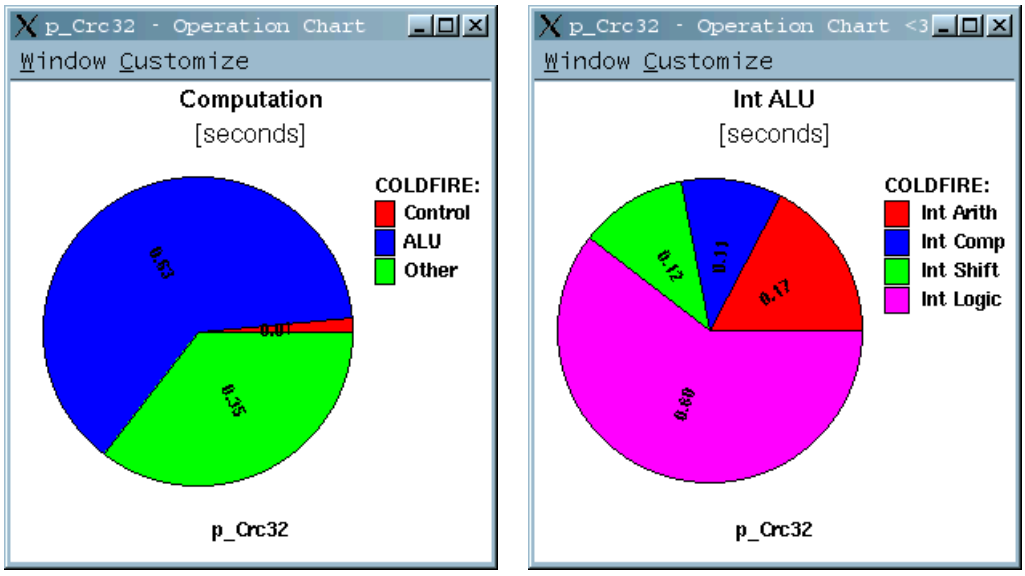
Στη συνέχεια οι λειτουργίες αυτές αντιστοιχίζονται σε έναν επεξεργαστή

[12] Αν και αυτό δεν αποτελεί το απόλυτο μέτρο για την υλοποίηση όλου του ATM.

ώστε να εκτιμηθεί η απόδοσή τους στην εν λόγω πλατφόρμα. Αρχικά χρησιμοποιείται ο επεξεργαστής Motorola Coldfire ο οποίος παρέχεται από τις βιβλιοθήκες του λογισμικού. Με το μοντέλο αυτού του επεξεργαστή υπεισέρχεται και η αρχιτεκτονική του με αποτέλεσμα να έχουμε την εκτίμηση σε επίπεδο κύκλων ρολογιού για μία τέτοια υλοποίηση. Τα αποτελέσματα από την υλοποίηση του μοντέλου στον Coldfire απέδειξε ότι δεν είναι αρκετά ικανός ώστε να εκτελέσει τις λειτουργίες αυτές στον επιθυμητό ρυθμό εφόσον τα αποτελέσματα έδωσαν για την απόδοσή του τα εξής: Για τη διεργασία `p_CamSearch` απαιτούνται 73 κύκλοι, για την `p_Crc32` 220, για την `p_Header` 160 και για την `p_Length` 36 κύκλοι. Τα χαρακτηριστικά του όπως παρατίθενται από την βιβλιοθήκη του λογισμικού παρουσιάζουν ότι η συχνότητα λειτουργίας του είναι στα 100 MHz, η δυνατότητα εκτέλεσης εντολών ανά δευτερόλεπτο[13] είναι στα 100 και οι εντολές του είναι μήκους 32 bit. Ωστόσο, η δοκιμή αυτή μέσω μίας ανάλυσης των λειτουργιών που εκτελούνται όχι πλέον σε επίπεδο κύκλων ρολογιού αλλά σε επίπεδο πράξεων χαμηλού επιπέδου, μας παρέχει μία αξιοσημείωτη επισήμανση. Στο Σχήμα 4.21 φαίνεται ότι ένα μεγάλο μέρος των πράξεων που απαιτούνται για την υλοποίηση της διεργασίας του `p_Crc32` πραγματοποιούνται με τη χρήση εντολών μία αριθμητικής μονάδος (ALU). Στο δεξιό μέρος του Σχήματος 4.21 παρουσιάζεται επίσης και μία βαθύτερη ανάλυση των εντολών τύπου ALU που εκτελούνται στον συγκεκριμένο επεξεργαστή.

Αυτή η πληροφορία οδηγεί τον εκάστοτε σχεδιαστή να χρησιμοποιήσει μία επεξεργαστική μονάδα η οποία ενσωματώνει και μία μονάδα για ψηφιακή επεξεργασία σήματος, η οποία θα αντεπεξέρχεται καλύτερα σε τέτοιου είδους εντολές. Είναι γνωστό ότι τέτοιου είδους επεξεργαστές, τα γνωστά DSP, μπορούν και εκτελούν αρκετά πιο γρήγορα από τους κλασσικούς επεξεργαστές τις πράξεις τύπου MAC, δηλαδή του πολλαπλασιασμού και τις πρόσθεσης.

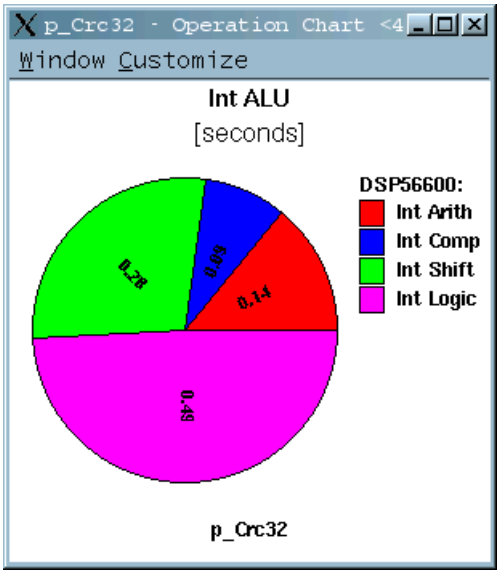
[13] MIPS



Υψηλού επιπέδου Χαμηλού Επιπέδου

Σχήμα 4.21: Τύποι εντολών στον επεξεργαστή Coldfire

Σε ορισμένα DSP αυτές οι λειτουργίες διεκπεραιώνονται σε έναν κύκλο ρολογιού. Επομένως, αναμένεται σημαντική βελτίωση στον χρόνο εκτέλεσης του συστήματος όταν υλοποιηθεί σε τέτοιου είδους επεξεργαστές. Για τον λόγο αυτό πραγματοποιείται μία υλοποίηση σε έναν επεξεργαστή τύπου DSP της Motorola και συγκεκριμένα στον DSP56600. Τα χαρακτηριστικά του όπως παρατίθενται από την βιβλιοθήκη του λογισμικού παρουσιάζουν ότι η συχνότητα λειτουργίας του είναι στα 60 MHz, η δυνατότητα εκτέλεσης εντολών ανά δευτερόλεπτο είναι στα 60 και οι εντολές του είναι μήκους 24 bit. Τα αποτελέσματα της εκτίμησης αυτής ήταν μη ικανοποιητικά για τα κριτήρια που έχουμε θέσει για έναν επεξεργαστή δικτύων PRO^3. Συγκεκριμένα για τη διεργασία `p_CamSearch` απαιτούνται 6 κύκλοι, για την `p_Crc32` 57, για την `p_Header` 32 και για την `p_Length` 5 κύκλοι. Είναι προφανές ότι και αυτή η υλοποίηση δεν καλύπτει τις ανάγκες του συστήματος από πλευράς απόδοσης. Ως αποτέλεσμα



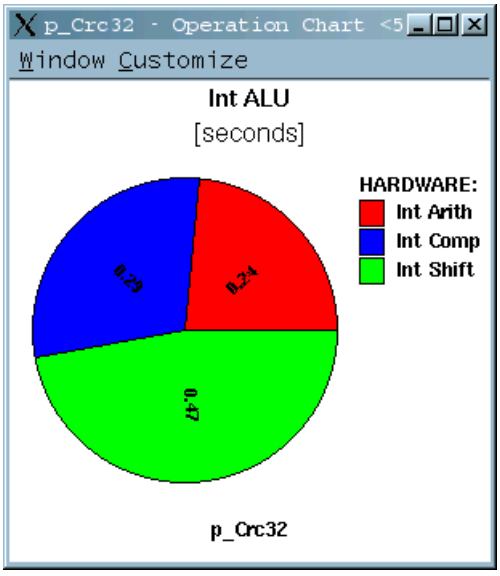
Σχήμα 4.22: Κατανομή εντολών στο DSP56600

Η αναλυτική κατανομή των εντολών στις οποίες καταλήγει η υλοποίηση στο εν λόγω DSP φαίνεται στο Σχήμα 4.22.

Από την δοκιμή με το DSP56600 φαίνεται ότι η χρήση μίας τέτοιας μηχανής θα βελτίωνε κατά πολύ την απόδοση του συστήματος για τέτοιους ρυθμούς, αλλά για τον υψηλό ρυθμό των 2.4 Gbps δεν είναι αρκετό. Η επόμενη λύση είναι να καταφύγουμε σε μεθόδους οι οποίες χρησιμοποιούν υλικό για την υλοποίηση της συγκεκριμένης συνάρτησης. Αν και είναι προτιμότερο να δοκιμάζουμε σταδιακά την εγκατάσταση μίας προς μία των συναρτήσεων στο υλικό για να αναζητήσουμε τη χρυσή τομή, θα προσεγγίσουμε την λύση θέτοντας όλη την λειτουργικότητα σε υλικό και στη συνέχεια θα μετακινούμε λειτουργικότητα προς το λογισμικό. Αυτό γίνεται για τον λόγο του ότι σε αυτό το στάδιο δεν αναζητούμε την βέλτιστη λύση για την κατάτμηση του συστήματος αλλά σε πρώτη φάση αναζητούμε μία λύση η οποία να μπορεί να αντεπεξέρθει στις απαιτήσεις της απόδοσης του συστήματος.

Για τον λόγο αυτό, προσθέτουμε στο μοντέλο του συστήματος μία μονάδα υλικού η οποία θα προσπαθήσει να φιλοξενήσει τις διεργασίες του δέκτη του ATM. Το υλικό αυτό περιγράφεται από την βιβλιοθήκη του λογισμικού SCE ως "Custom Hardware" το οποίο χρονίζεται στα 100 MHz, αποδίδει 100 MIPS και οι εντολές του είναι μήκους 128 bits. Αντιστοιχίζοντας λοιπόν τις λειτουργίες του CRC32 στο υλικό που αναφέραμε, παρατηρούμε ότι το σύστημα τώρα μπορεί να εκτελέσει οριακά τις συναρτήσεις τις οποίες και του αναθέσαμε. Τα αποτελέσματα από αυτό το μοντέλο μας δίνουν ότι για τη διεργασία `p_CamSearch` απαιτούνται 3 κύκλοι, για την `p_Crc32` 16, για την `p_Header` 14 και για την `p_Length` 2 κύκλοι. Ακόμη και μία απλή πρόσθεση των αποτελεσμάτων αυτών οδηγεί σε αποτέλεσμα ίσο με τους 35 διαθέσιμους κύκλους του ATM. Παρόλα αυτά, έχει αναφερθεί ότι λόγω της φύσης της λειτουργίας του CRC32 υπάρχει η δυνατότητα χρήσης μονάδων συνεχούς διοχέτευσης δεδομένων έτσι ώστε να μειωθεί σημαντικά ο συνολικά απαιτούμενος χρόνος για την επεξεργασία των πακέτων με ρυθμό άφιξης τα 2.4 Gbps. Από τα αποτελέσματα που λαμβάνονται από μία τέτοιου είδους κατάτμηση παρατηρούμε ότι η υλοποίηση στο συγκεκριμένο υλικό δεν επιδέχεται περαιτέρω κατάτμηση για εκτέλεση μέρους της λειτουργικότητας σε λογισμικό. Αυτό θα ήταν δυνατόν αν τα αποτελέσματα της απόδοσης (π.χ. οι κύκλοι στη συγκεκριμένη περίπτωση) ήταν κατά πολύ λιγότερα σε σχέση με τους διαθέσιμους πόρους. Για την υλοποίηση ωστόσο, χρησιμοποιείται και ως "οδηγός" ο Πίνακας 4.8 ώστε να επιτευχθεί υλοποίηση των λειτουργιών όσον το δυνατόν πιο αποδοτικά στο συγκεκριμένο υλικό. Τα αποτελέσματα για την ανάλυση των εντολών στο υλικό που χρησιμοποιείται φαίνονται στο Σχήμα 4.23.

Η βελτίωση μεταξύ των τριών βημάτων που ακολουθήθηκαν προκειμένου να καταλήξουμε στη συγκεκριμένη υλοποίηση είναι δραματική. Το Σχήμα 4.24 παρουσιάζει συνολικά την ποσοτική απαίτηση της κάθε μίας από τις τρεις υλο-

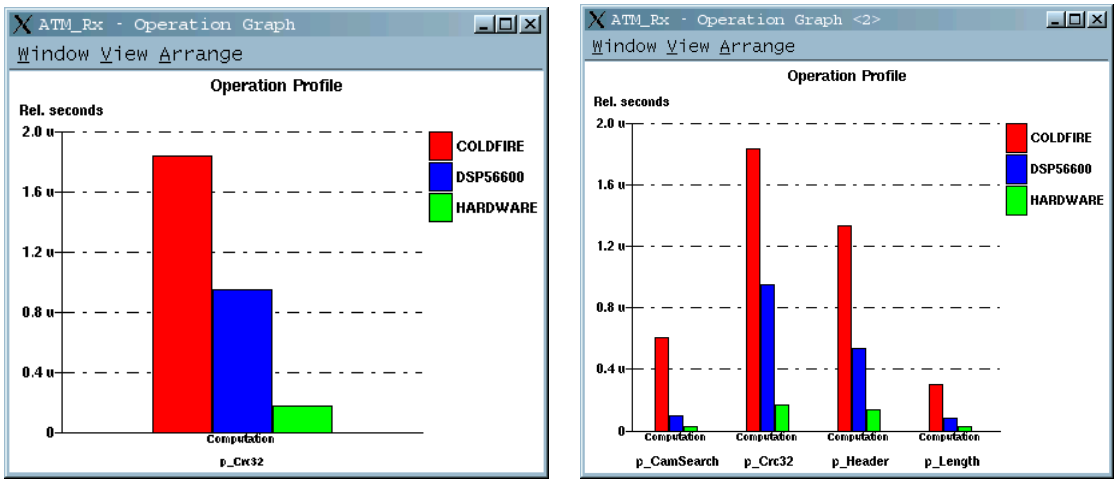


Σχήμα 4.23: Κατανομή εντολών του CRC32 σε υλικό

ποιήσεις για τις πλατφόρμες που αναφέρθηκαν παραπάνω. Είναι φανερό πως εφόσον η υλοποίηση σε υλικό είναι η μοναδική που οριακά καλύπτει και ικανοποιεί τις απαιτήσεις του συστήματος, μία υλοποίηση σε λογισμικό στο DSP56600 θα απαιτούσε περισσότερες από τέσσερις επεξεργαστικές μονάδες για την επίτευξη του στόχου. Επιπλέον, δεν εξετάζεται καθόλου και το γεγονός ότι η λειτουργία των μονάδων αυτών θα εισήγαγε και επιπλέον λογική για τον συγχρονισμό των υποσυστημάτων εφόσον αυτές οι μονάδες θα έπρεπε να λειτουργούν κατά έναν παράλληλο τρόπο.

Επιπλέον, σημαντικό είναι να εξετάσουμε και την συγκριτική απόδοση της κάθε επιμέρους λειτουργίας του CRC32 στις διάφορες πλατφόρμες υλοποίησης. Το δεξιά Σχήμα του Σχήματος 4.24 παρέχει μία ανάλυση των επιμέρους λειτουργιών του CRC32 στις τρεις διαφορετικές πλατφόρμες.

Οι μέθοδοι και οι τεχνικές για κατάτμηση σε τηλεπικοινωνιακά συστήματα δεν περιορίζεται μόνο σε τηλεπικοινωνιακά συστήματα όπως είναι οι επεξερ-



Ανάλυση του CRC32 Ανάλυση όλων των λειτουργιών

Σχήμα 4.24: Σύγκριση του ATM-Rx στις διάφορες πλατφόρμες

γαστές δικτύων· αντιθέτως η εφαρμογή τους βρίσκει εφαρμογή και σε άλλα συστήματα όπως είναι αυτό που παρουσιάζεται στο Κεφάλαιο 5. Στο συγκεκριμένο Κεφάλαιο, παρουσιάζεται πως μπορεί μέσω της κατάτμησης υλικού και λογισμικού να επιτευχθεί βελτίωση στην απόδοση ενός συστήματος ψηφιακής επεξεργασίας σημάτων όπως είναι για παράδειγμα οι καταστολείς της ακουστικής ηχούς.

4.11 Κατάτμηση της ATM εφαρμογής

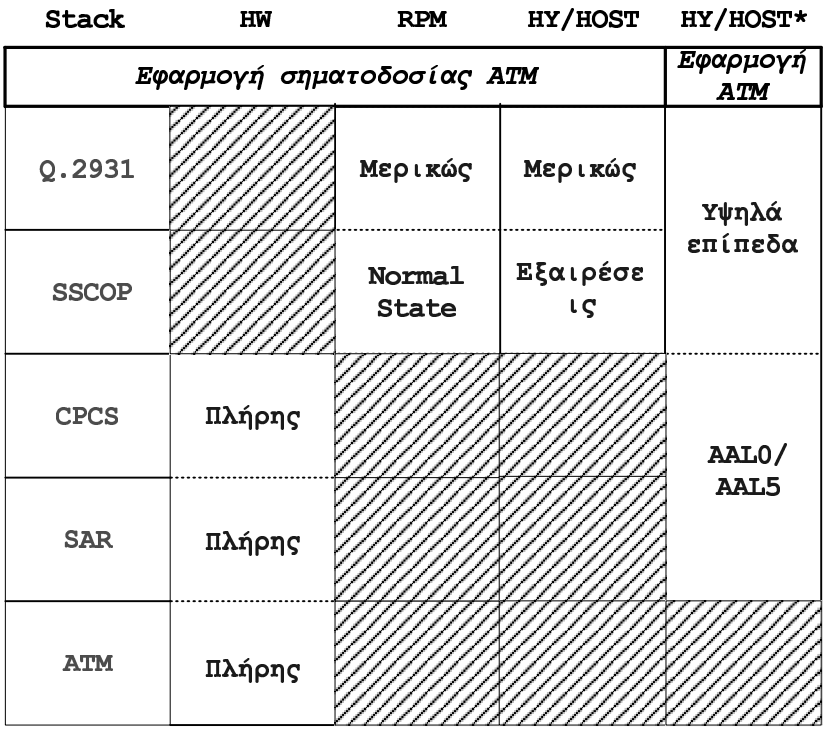
Η κατάτμηση υλικού λογισμικού για την σηματοδοσία του ATM πραγματοποιείται για την επίτευξη της υψηλότερης απόδοσης. Η ανάλυση της απόδοσης της εκτέλεσης του πρωτοκόλλου (βλέπε παράγραφο 4.5), απέδειξε ότι μόνο ένα εξαιρετικά μικρό τμήμα του πρωτοκόλλου είναι ενεργό για περισσότερο από το 90% του συνολικού χρόνου εκτέλεσής του. Οι συναρτήσεις που δεν εμφανίζονται συχνά συσχετίζονται με δραστηριότητες όπως είναι ο χειρισμός

λαθών, η δημιουργία συνδέσεων, ο τερματισμός τους κτλ. Οι ουσιαστικές (και χρονικά κρίσιμες) συναρτήσεις θα υλοποιηθούν απευθείας στο RPM. Άλλα επίπεδα του πρωτοκόλλου θα οδηγηθούν για εκτέλεση στον ενσωματωμένο στο ολοκληρωμένο σύστημα επεξεργαστή της Hyperstone. Η συγκεκριμένη επεξεργαστική μονάδα θα είναι αυτή η οποία θα χειρίζεται τα υψηλότερα επίπεδα του πρωτοκόλλου καθώς επίσης και άλλες λειτουργίες του συστήματος.

Μία μελέτη στα επίπεδα του πρωτοκόλλου QSAAL [Ntua02], αποδεικνύει ότι η μονάδα του RPM είναι ικανή από άποψη επεξεργαστικής ισχύος[14], για την εκτέλεση κάποιων τυπικών ελέγχων σχετικά με την εγκυρότητα της σηματοδοσίας πρόσβασης καθώς επίσης και για να εκτελέσει κάποιες καταστάσεις του πρωτοκόλλου του SSCOP. Αν και αυτές οι λειτουργίες αποτελούν ένα μικρό τμήμα των πρωτοκόλλων του SSCOP και του Q.2931, ωστόσο εκτελούνται σε κάθε πακέτο σηματοδοσίας του πρωτοκόλλου. Δεδομένου ότι αυτές οι συναρτήσεις εκτελούνται πολύ συχνά σε σχέση με τις υπόλοιπες συναρτήσεις του πρωτοκόλλου, η υλοποίηση τους στο RPM θα έχει ως αποτέλεσμα την επιτάχυνση του συνολικού πρωτοκόλλου σηματοδοσίας.

Η πρώτη στήλη του Σχήματος 4.25 παρουσιάζει τα επίπεδα του ATM. Η στήλη που ονομάζεται HW περιγράφει ποια από αυτά τα επίπεδα υλοποιούνται σε υλικό. Οι στήλες με τον τίτλο RPM και HY/HOST περιλαμβάνει πληροφορία σχετικά με την υλοποίηση των επιπέδων στο RPM και στον εσωερικό on-chip επεξεργαστή της Hyperstone. Η στήλη HY/HOST* περιέχει πληροφορία σχετικά με την υλοποίηση των εφαρμογών που δεν ανήκουν στη σηματοδοσία (όπως τα AAL0 και AAL5), στην εξωτερική επεξεργαστική μονάδα. Σύμφωνα πάντα με το Σχήμα 4.25 η εκτέλεση των λειτουργιών όσον αφορά τα επίπεδα του ATM, SAR και CPCS πραγματοποιούνται εξόλοκλήρου σε υλικό. Το μπλοκ του RPM εκτελεί τον κώδικα της κατάσταση κανονικών συνθηκών

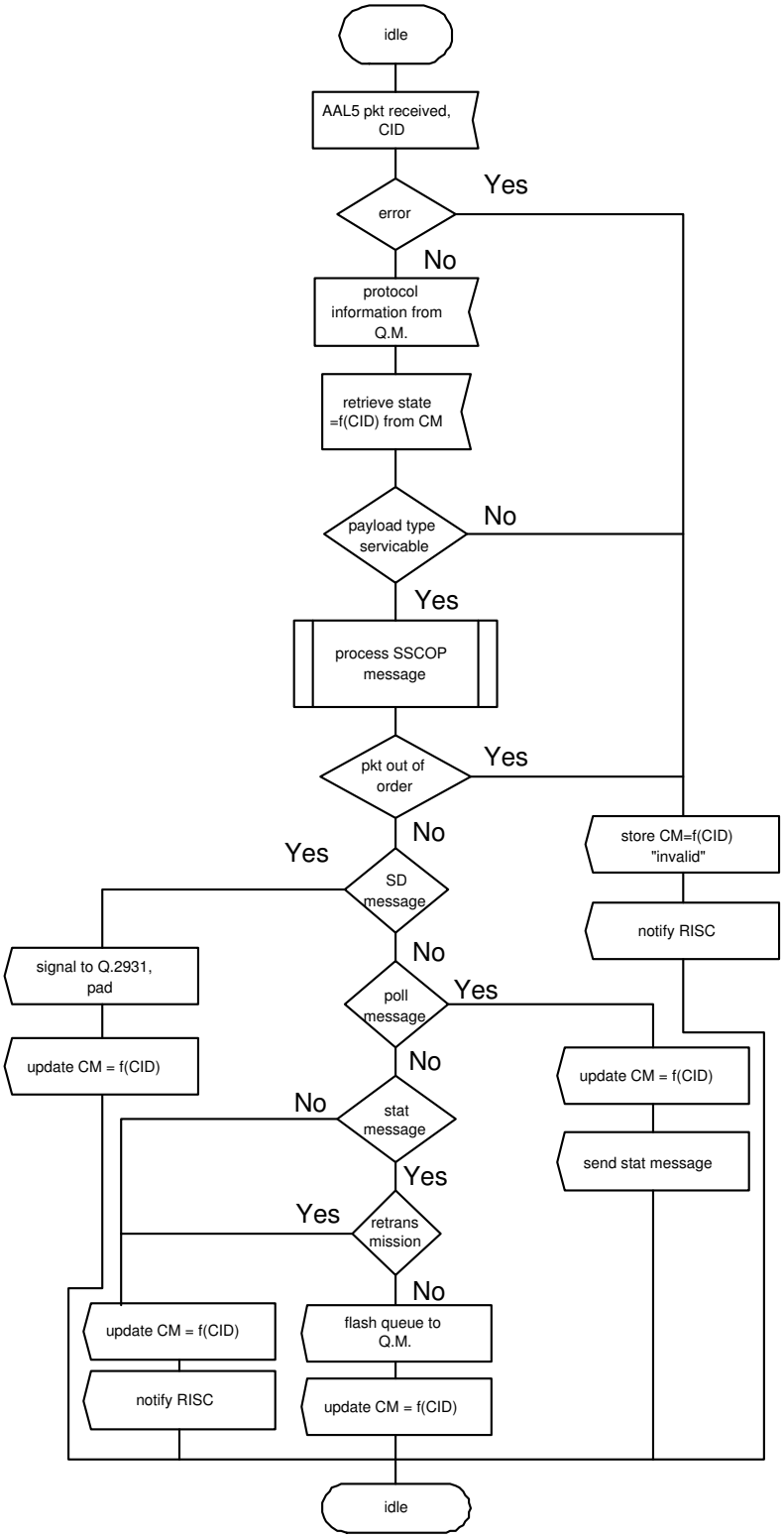
[14]Για τον ρυθμό μετάδοσης του συστήματος που είναι 2.4 Gbps



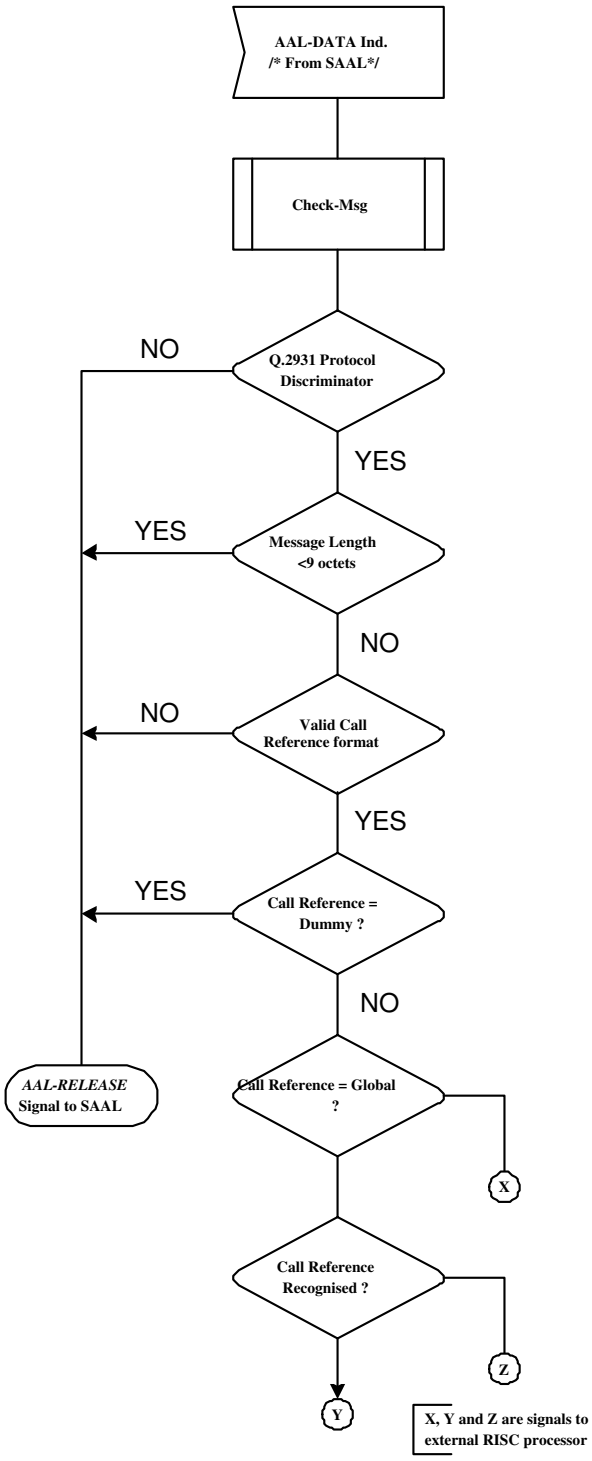
Σχήμα 4.25: Αντιστοίχιση επιπέδων του ATM στο PRO^3

του SSCOP καθώς και συγκεκριμένες συναρτήσεις του Q.2931. Για το επίπεδο του SSCOP η κατάσταση κανονικών συνθηκών είναι αυτή που ονομάζεται "Ετοιμότητας Μεταφοράς Δεδομένων" ή "Data Transfer Ready" όπως περιγράφεται και στις συστάσεις της ITU-T για το Q.2110. Ειδικά για το Q.2931, θα εκτελούνται ρουτίνες για τον έλεγχο της εγκυρότητας και της ορθότητας των δεδομένων του εισερχόμενου πακέτου. Οι υπόλοιπες συναρτήσεις για το SSCOP και το Q.2931 θα εκτελούνται σε στις δύο άλλες επεξεργαστικές μονάδες του συστήματος του PRO^3. Η εξωτερική επεξεργαστική μονάδα είναι επίσης εφοδιασμένη με το λογισμικό για τον χειρισμό του ελέγχου των κλήσεων καθώς και για τον χειρισμό των πακέτων τύπου AAL0. Τα πακέτα τύπου AAL0 θα προωθούνται διαφανώς προς την εξωτερική επεξεργαστική μονάδα χωρίς να περνάνε μέσα από το μπλοκ του RPM.

Αν το πακέτο το οποίο θα επεξεργάζεται από το RPM περιέχει πληροφορία που δεν μπορεί να διαχειρισθεί από το μπλοκ, θα δημιουργηθεί ένα κατάλληλο σήμα προς των χρονοδρομολογητή των διεργασιών. Ο χρονοδρομολογητής διεργασιών με τη σειρά του, θα αποστείλει ένα πακέτο ελέγχου προς την εξωτερική επεξεργαστική μονάδα, η οποία θα διατάζει την λήψη του πακέτου από τον διαχειριστή δεδομένων ώστε να συνεχίσει την επεξεργασία. Η εξωτερική επεξεργαστική μονάδα θα περιέχει το πλήρες πρωτόκολλο για το SSCOP και το Q.2931. Η επικοινωνία μεταξύ των δύο αυτών επιπέδων του πρωτοκόλλου θα γίνεται μέσω της μνήμης ελέγχου που ονομάζεται Control RAM. Αυτή η μνήμη θα αποθηκεύει την πληροφορία σχετικά με την κατάσταση του κάθε πρωτοκόλλου, έτσι ώστε κάθε πρωτόκολλο να μπορεί να εξάγει την απαραίτητη πληροφορία έτσι ώστε να μπορεί να συνεχίσει με την επεξεργασία της συγκεκριμένης ATM σύνδεσης. Τα παρακάτω Σχήματα 4.26 και 4.27 παρουσιάζουν μία τυπική βασική επεξεργασία πακέτεων SSCOP και Q.2931 τα οποία υλοποιούνται στο RPM.



Σχήμα 4.26: Λειτουργία του SSCOP στο RPM



Σχήμα 4.27: Λειτουργία του Q.2931 στο RPM

Οι παραπάνω λειτουργίες των αναφερθέντων πρωτοκόλλων θα επιταχυνθούν με την εκτέλεσή τους στο συγκεκριμένο μπλοκ, συγκρινόμενα πάντα με τις κλασσικές υλοποιήσεις των εφαρμογών βασιζόμενες σε λογισμικό. Οι υπόλοιπες λειτουργίες των πρωτοκόλλων θα υλοποιούνται στις άλλες επεξεργαστικές μονάδες (εσωτερικές ή εξωτερικές).

Κεφάλαιο 5

Κατάτμηση σε συστήματα DSP

Οι συνεχές πρόοδοι στους επεξεργαστές και στις τεχνολογίες ASIC επιτρέπουν την ολοκλήρωση όλο και πιο πολύπλοκων συστημάτων. Ωστόσο, οι βελτιώσεις στις τεχνικές αυτοματοποιημένης σχεδίασης για την παραγωγή συστημάτων αυξάνει πιο αργά από ότι αυξάνουν οι δυνατότητες ολοκλήρωσης των ASIC. Συνεπώς, υπάρχει ένα μεγάλο ενδιαφέρον στα υψηλότερα επίπεδα για την μοντελοποίηση συστημάτων και τη σύνθεσή τους και ειδικότερα στις προσεγγίσεις του ταυτόχρονου σχεδιασμού υλικού/λογισμικού [GV95], [JI95]. Ξεκινώντας από υψηλού επιπέδου προδιαγραφές του συστήματος, οι τεχνικές της ταυτόχρονης σχεδίασης προσπαθούν να βρουν γρήγορα βελτιωμένες προδιαγραφές από αποδοτικές διασταυρωμένες υλοποιήσεις μέσω συστηματικών εξερευνήσεων από διάφορες εξισορροπήσεις των παραγόντων. Η παράλληλη σχεδίαση αποτελείται γενικά από διάφορες διεργασίες όπως για παράδειγμα η κατάτμηση υλικού/λογισμικου και η χρονοδρομολόγηση, η δημιουργία κώδικα υπό περιορισμούς και η σύνθεση υλικού και της επικοινωνίας του.

Η καταστολή της ακουστικής ηχούς, αποτελεί ένα άριστο παράδειγμα για το σχεδιασμό ενσωματωμένων συστημάτων εφόσον η υλοποίηση πραγματικού χρόνου αποτελεί ακόμη και τώρα μία πρόκληση, ειδικά στα συστήματα τηλε-

διάσκεψης ευρείας ζώνης. Ένα από τα πιο σημαντικά θέματα στην αμφίδρομη επικοινωνία φωνής είναι η λαμβανόμενη ποιότητα της επικοινωνίας. Σε συστήματα τηλεδιασκέψεων ακόμη και αν η μεταφορά της εικόνας και του video είναι ικανοποιητική, η χαμηλή ποιότητα στον ήχο οδηγεί πολλές φορές σε αρνητικό αντίκτυπο της ποιότητας της συνολικής επικοινωνίας.

5.1 Καταστολή ακουστικής ηχούς

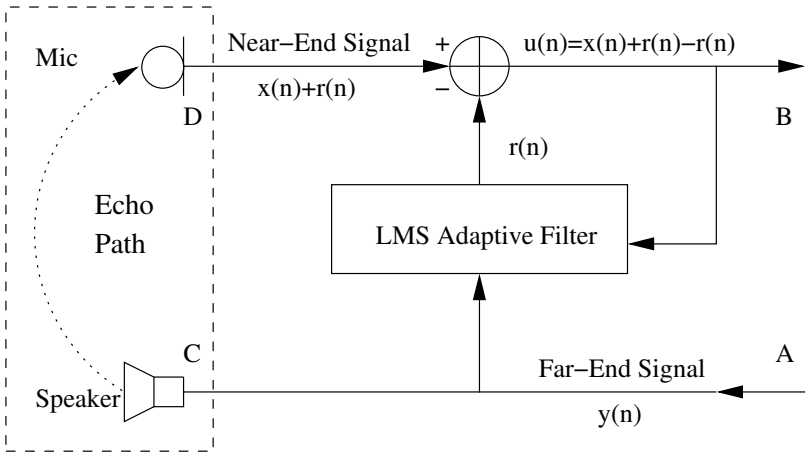
Το πρόβλημα της εμφάνισης της ακουστικής ηχούς έχει σαν αποτέλεσμα την επιστροφή του σήματος στον έναν ή και στους δύο πιθανούς ομιλητές μίας τηλεφωνικής συνδιάλεξης. Το φαινόμενο αυτό λαμβάνει χώρα όταν τουλάχιστον ο ένας από τους δύο συνομιλητές χρησιμοποιεί μία τηλεφωνική συσκευή ανοιχτής ακρόασης. Ο όρος του "ομιλητή κοντινού άκρου" (near end speaker) αντιστοιχεί στο άτομο το οποίο χρησιμοποιεί τη συσκευή ανοιχτής ακρόασης. Εναλλακτικά χρησιμοποιείται και ο όρος "τοπικός" (local). Στην υπό εξέταση περίπτωση, ο "ομιλητής απομακρυσμένου άκρου" (far end speaker) είναι αυτός ο οποίος χρησιμοποιεί μία κανονική τηλεφωνική συσκευή όταν μιλάει με τον τοπικό συνομιλητή. Στις τηλεφωνικές συσκευές ανοιχτής ακρόασης, το ακουστικό σήμα φεύγει από τον απομακρυσμένο ομιλητή και μεταφέρεται μέσω ενός ηλεκτρικού καναλιού, το οποίο τερματίζεται σε ένα μεγάφωνο. Ο σκοπός αυτού του μεγαφώνου είναι να αναπαράγει και να ενισχύσει το ηχητικό σήμα, έτσι ώστε να μπορέσει να φτάσει στον τοπικό ομιλητή σε ικανοποιητικά ακουστικά επίπεδα. Καθώς το ηχητικό σήμα εμφανίζεται στο μεγάφωνο, εκπέμπεται μέσα στο δωμάτιο και ανακλάται από τις συμπαγείς επιφάνειες όπως για παράδειγμα είναι οι τοίχοι, τα έπιπλα, το πάτωμα και το ταβάνι. Κατά συνέπεια, το σήμα διαδίδεται μέσω πολλών διαδρομών, με το κάθε ένα να έχει διαφορετικό ακουστικό μήκος, πριν να φτάσει τελικά στα αυτιά του τελικού ακροατή.

Το απομακρυσμένο σήμα επίσης φτάνει στο μικρόφωνο του τοπικού ακροατή το οποίο είναι εγκατεστημένο στην τηλεφωνική συσκευή ανοιχτής ακρόασης. Τα κανάλια και οι διαδρομές από το μεγάφωνο του τοπικού ομιλητή προς το μικρόφωνό του ονομάζονται ως "διαδρομές ηχούς" (echo paths). Λόγω του διαφορετικού ακουστικού μήκους των διαδρομών, υπάρχει ένας πεπερασμένος διακριτός χρόνος μεταξύ της μικρότερης και της μεγαλύτερης ακουστικής διαδρομής. Όταν η διαδρομή της ηχούς καθυστερεί χρονικά παραπάνω από ένα συγκεκριμένο κατώφλι, τότε αυτό είναι αντιληπτό από το ανθρώπινο αυτί στον απομακρυσμένο ομιλητή [G.165] και αυτό το ενοχλητικό φαινόμενο ονομάζεται ακουστική ηχώ. Δύο προσεγγίσεις έχουν αναπτυχθεί για την επίλυση του προβλήματος της ηχούς· ο ένας είναι η χρησιμοποίηση ημιαμφίδρομης επικοινωνίας και η άλλη είναι η χρησιμοποίηση καταστολέων ηχούς στα συστήματα ανοιχτής ακρόασης. Σε αντίθεση με τα συστήματα ημιαμφίδρομης επικοινωνίας, ο στόχος της αμφίδρομης επικοινωνίας είναι να παρέχει συνομιλία απαλλαγμένη από ηχώ.

5.1.1 Μοντέλο καταστολέα ηχούς

Για την επίλυση του παραπάνω προβλήματος οι σχεδιαστές χρησιμοποιούν συνήθως τεχνικές ψηφιακής επεξεργασίας σήματος για την καταστολή της ακουστικής ηχούς έτσι ώστε να σταματήσουν την ανάδραση και να επιτρέψουν την αμφίδρομη επικοινωνία. Για την καταστολή της ηχούς, το σύστημα του καταστολέα (AEC[1]) πρέπει να μοντελοποιήσει το κανάλι του χώρου μεταξύ του μεγαφώνου και του μικροφώνου. Το κανάλι αυτό είναι μία συνάρτηση του μεγαφώνου και του μικροφώνου καθώς και της τοποθέτησής τους μέσα στον χώρο, της ακουστικής του δωματίου συμπεριλαμβανομένου των υλικών κατασκευής, των διαστάσεων, των επίπλων και της τοποθέτησής τους καθώς και

[1] Acoustic Echo Canceller



Σχήμα 5.1: Μοντέλο καταστολέα ακουστικής ηχούς

της μετακίνησης των ατόμων μέσα στο δωμάτιο αυτό.

Το Σχήμα 5.1 παρουσιάζει την αρχή ενός καταστολέας ακουστικής ηχούς για μία κατεύθυνση μετάδοσης. Ας θεωρήσουμε ως $y(n)$ το σήμα του απομακρυσμένου σήματος, $r(n)$ την ανεπιθύμητη ηχώ και $x(n)$ το σήμα της ομιλίας του τοπικού ομιλητή. Το σήμα του τοπικού ομιλητή υπερτίθεται με την ανεπιθύμητη ηχώ στην πόρτα D. Το ληφθέν σήμα του απομακρυσμένου χρήστη είναι διαθέσιμο στο εν λόγω μοντέλο ως σήμα αναφοράς για τον καταστολέα της ηχούς έτσι ώστε να αναπαράγει ένα μοντελοποιημένο αντίγραφο από την ηχώ η οποία θα αναφέρεται συμβολικά ως $\hat{r}(n)$. Αυτό το αντίγραφο αφαιρείται στη συνέχεια από το σήμα του τοπικού ομιλητή το οποίο αδιακρίτως περιλαμβάνει και το σήμα της ηχούς, ώστε να οδηγήσει ως σήμα μετάδοσης το σήμα τοπικού ομιλητή το $u(n)$ όπου $u(n) = x(n) + r(n) - \hat{r}(n)$. Ιδανικά, η εναπομείναν ηχώ σαν σήμα λάθους ορίζεται ώς

$$e(n) = r(n) - \hat{r}(n) \tag{5.1}$$

η οποία θα είναι πολύ μικρής τιμής μετά την καταστολή της ηχούς και κατά συνέπεια θα είναι $u(n) \approx x(n)$.

5.1.2 LMS και θεωρία καταστολής

Οι αλγόριθμοι Ελαχίστων Μέσων Τετραγώνων LMS (Least Mean Square αποτελούν μία ομάδα συχνά χρησιμοποιούμενων αλγορίθμων προσαρμογής. Η διάδοση αυτών των αλγορίθμων οφείλεται κυρίως λόγω της μικρής τους πολυπλοκότητας από άποψη επεξεργαστικής ισχύς που απαιτούν. Αυτοί οι αλγόριθμοι, όπως φαίνεται και από το όνομά τους, προσπαθούν να ελαχιστοποιήσουν την αναμενόμενη τιμή του τετραγώνου του εκάστοτε λάθους (στην περίπτωση που μελετάμε το λάθος είναι το σήμα με την παραμένουσα ηχώ). Η εγκάρσια δομή επιλέγεται για τα προσαρμοζόμενα φίλτρα εξαιτίας της απλότητας που τη χαρακτηρίζει. Οι αλγόριθμοι αυτοί ξεκινούν από μία αρχική τιμή τυχαία επιλεγμένη (συνήθως μηδενική) για το διάνυσμα βαρών, καθώς τα βάρη προσαρμόζονται σύμφωνα με τον στοχαστικό αλγόριθμο της απότομης κατάβασης, και βελτιώνονται με τις διάφορες επαναλήψεις. Η τελική τιμή που υπολογίζεται για το διάνυσμα βαρών των συντελεστών συγκλίνει στην μέση τιμή της λύσης Wiener. Η λειτουργία της τάξης αλγορίθμων του LMS περιγράφεται από ένα σύστημα ελέγχου ανάδρασης. Βασικά, αποτελείται από ένα συνδυασμό δύο βασικών λειτουργιών:

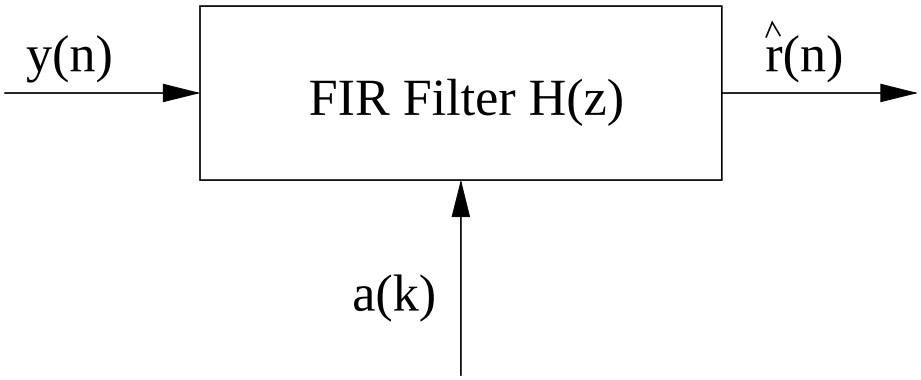
- Μία προσαρμοζόμενη διεργασία, η οποία εμπλέκει την αυτοματοποιημένη κατά μία έννοια ρύθμιση και προσαρμογή των βαρών του φίλτρου.

- Μία διεργασία φιλτραρίσματος η οποία δημιουργεί το γινόμενο μίας ομάδας συντελεστών εισόδου με την αντίστοιχη ομάδα του διανύσματος βαρών το οποίο προκύπτει από τη διεργασία της προσαρμογής. Το αποτέλεσμα είναι η εκτίμηση της επιθυμητής εξόδου με την σύγκριση της πραγματικής απόκρισης με την συγκεκριμένη εκτίμηση.

Η θεωρία καταστολής βασίζεται σε ένα προσαρμοζόμενο μοντέλο του χώρου όπου ακούγεται το σήμα χρησιμοποιώντας προσαρμόζόμενα φίλτρα με πεπερα-

σμένες αποκρίσεις (FIR). Ο αλγόριθμος των προσαρμόσιμων φίλτρων χρησιμοποιεί τις τιμές των προηγουμένων δειγμάτων φωνής και των υπολογισμένων λαθών ώστε να ενημερώσει τους συντελεστές του FIR φίλτρου, βασιζόμενοι σε έναν αλγόριθμο ελαχίστων μέσων τετραγώνων (LMS). Στη συνέχεια, χρησιμοποιεί τους ενημερωμένους συντελεστές και τις τελευταίες τιμές δειγμάτων για να υπολογισθούν οι έξοδοι του FIR φίλτρου. Ένα προσαρμόσιμο φίλτρο διακρίνει δείγματα από ένα σήμα εισόδου $y(n)$, ένα σύνολο από συντελεστές του φίλτρου $a_k(n)$ και παράγει ως έξοδο ένα σήμα $\hat{r}(n)$ το οποίο ονομάζεται "επιθυμητή απόκριση". Το σήμα της επιθυμητής απόκρισης παρέχει ένα πλαίσιο αναφοράς για την προσαρμογή των συντελεστών του φίλτρου. Στην περίπτωση του AEC το σήμα εισόδου $y(n)$ είναι το σήμα το οποίο αποστέλλεται στο τοπικό μεγάφωνο και το σήμα της επιθυμητής απόκρισης ενώ το $\hat{r}(n)$ είναι μία εκτίμηση της ανεπιθύμητης ηχούς η οποία εισάγεται στο τοπικό μικρόφωνο. Όταν το άτομο που βρίσκεται τοπικά δεν μιλάει ενώ ο απομακρυσμένος ομιλητής μιλάει, το σήμα του τοπικού ομιλητή εμπεριέχει μόνο την ηχώ η οποία πρέπει να αφαιρεθεί. Όταν μιλάει λοιπόν μόνο ο απομακρυσμένος ομιλητής, τότε ο στόχος του προσαρμοστικού αλγόριθμου είναι να οδηγήσει το σήμα του λάθους $e(n)$ στο μηδέν έτσι ώστε να μην ακούγεται καμία ηχώ στον απομακρυσμένο ομιλητή. Ένας σημαντικός παράγοντας για μια σωστή υλοποίηση ενός καταστολέα ακουστικής ηχούς, είναι η ακριβής αναγνώριση της κατάστασης της ομιλίας που μπορεί να υφίσταται κάθε φορά στη γραμμή. Αυτή η αναζήτηση επιτυγχάνεται με τον υπολογισμό της ενέργειας και στη συνέχεια με τη σύγκρισή της με ένα προσαρμοστικό κατώφλι.

Στο Σχήμα 5.2 παρουσιάζεται ένα FIR φίλτρο. Η εξίσωση 5.2 περιγράφει την συνάρτηση μεταφοράς ενός τέτοιου φίλτρου. Το φίλτρο χρησιμοποιείται για να εκτιμήσει την εκάστοτε πραγματική συνάρτηση μεταφοράς του καναλιού



Σχήμα 5.2: Φίλτρο τύπου FIR

μεταφοράς του σήματος στον χώρο.

$$\hat{r}(n) = \sum_{k=0}^{N-1} a(k) \cdot y(n-k) \tag{5.2}$$

όπου N είναι ο αριθμός των συντελεστών του φίλτρου, y είναι το σήμα το οποίο απαιτεί φιλτράρισμα και α είναι οι συντελεστές του φίλτρου. Ας σημειωθεί ότι τα y και α είναι διανύσματα.

$$\begin{aligned} \boldsymbol{\alpha} &= [a(0), a(1), ..., a(N-1)] \\ y(n) &= [y(n), y(n-1), ..., y(n-N+1)] \end{aligned}$$

Ένα τέτοιο φίλτρο πρέπει να είναι τόσο μεγάλο σε μήκος έτσι ώστε να μπορεί να προσεγγίσει την απόκριση και την συσχετιζόμενη καθυστέρηση της συνάρτησης μεταφοράς την οποία προσπαθεί να προσομοιώσει. Ο όρος "μεγάλο μήκος" ακουστικού καναλιού, προσδιορίζει την μεγαλύτερη καθυστέρηση η οποία μπορεί να αντισταθμιστεί. Αυτή η συνάρτηση είναι στην πραγματικότητα μία συνάρτηση της γεωμετρίας και των διαστάσεων του χώρου στον οποίο λειτουργεί η συσκευή ανοιχτής ακρόασης. Το ακουστικό περιβάλλον ενός τυπικού

γραφείου έχει μία απόκριση με καθυστέρηση των 50 ms περίπου. Για δεδομένο ρυθμό δειγματοληψίας αυτόν των 5000 δειγμάτων ανά δευτερόλεπτο, αυτό οδηγεί στη χρήση 400 συντελεστών στο φίλτρο του καταστολέα της ηχούς.

Οι συντελεστές του προσαρμοζόμενου φίλτρου προσδιορίζονται κάνοντας χρήση του σήματος το οποίο επιστρέφει στον απομακρυσμένο ομιλητή. Κατά συνέπεια το φίλτρο εξαρτάται άμεσα από το επιστρεφόμενο σήμα. Όταν υπάρχει στο κανάλι λαμβανόμενη ομιλία και ο τοπικός ομιλητής ξεκινά να μιλά (κατάσταση ταυτόχρονης συνομιλίας). ένας καταστολέας ηχούς μπορεί να ερμηνεύσει το σήμα αυτό ως ένα νέο σήμα ηχούς και η προσπάθειά του να το προσαρμόσει να οδηγήσει σε παραμόρφωση της συνομιλίας. Για αυτό το λόγο συνοψίζουμε ότι ένας καταστολέας ηχούς θα πρέπει:

- Να συγκλίνει σχετικά γρήγορα.
- Να επιστρέφει την ελάχιστη ποσότητα ηχούς κατά την μονόδρομη ομιλία.
- Να αποκλίνει ελάχιστα κατά την ταυτόχρονη συνομιλία.

Η διαδικασία του επαναϋπολογισμού του διανύσματος $\mathbf{a}$ έτσι ώστε να επιτευχθεί η οδήγηση του $\hat{r}(n) = 0$ ονομάζεται προσαρμογή του φίλτρου. Αν το φίλτρο αρχίσει να προσαρμόζεται σε λάθος χρονική στιγμή, όπως για παράδειγμα στην περίπτωση που το μικρόφωνο λαμβάνει σήμα από τον τοπικό ομιλητή, το φίλτρο μπορεί πολύ εύκολα να συγκλίνει εντελώς λανθασμένα δημιουργώντας έντονα ηχητικά λάθη τα οποία τελικά θα μεταδοθούν και στον απομακρυσμένο ομιλητή. Το πρόβλημα δημιουργείται εξαιτίας του γεγονότος ότι η ακουστική συμπεριφορά ενός χώρου δεν μπορεί να είναι σταθερή. Αλλαγές στο μοντέλο του χώρου ενός δωματίου μπορεί να συμβεί από μετακινήσεις των συμμετασχόντων σε ένα χώρο ή ακόμη και από μετακινήσεις του μικροφώνου σε σχέση με τα γύρω αντικείμενα ή από την ένταση του ήχου στην συσκευή ανοιχτής ακρόασης.

Ο επαναληπτικός αλγόριθμος των μέσων ελαχίστων τετραγώνων παρέχει έναν χαμηλού κόστους τρόπο ώστε να προσδιορίσει τους συντελεστές του φίλτρου χωρίς ωστόσο να υπολογίζει κάθε φορά την αυτοσυσχέτιση και την διασυσχετισιμότητα των σημάτων [Hyk86]. Ο στόχος είναι η ελαχιστοποίηση του αθροίσματος των μέσων τετραγώνων E.

$$E = \mathcal{E}\left\{\sum_{n=0}^{M} e^2(n)\right\} = \mathcal{E}\left\{\sum_{n=0}^{M}\left(r(n) - \sum_{k=0}^{N-1} a(k)y(n-k)\right)^2\right\} \tag{5.3}$$

όπου $\mathcal{E}$ είναι ο όρος για την μέση τιμή, $e(n)$ είναι το παραμένων σήμα λάθους και S είναι το βήμα το οποίο προσδιορίζει την ταχύτητα της σύγκλισης του αλγορίθμου.

Μετά την ελαχιστοποίηση της παραπάνω σχέσης, πρέπει να βρεθεί ένας αλγόριθμος επαναληπτικού βαθμωτού ανύσματος ο οποίος προσδιορίζει την μικρότερη τιμή του E. Η διεργασία ξεκινά με μία τυχαία επιλογή για τις αρχικές τιμές των συντελεστών του φίλτρου - συνήθως επιλέγονται έτσι ώστε να είναι όλοι αρχικά στην μηδενική τιμή. Ύστερα από την είσοδο του κάθε νέου δείγματος $y(n)$, εισάγεται στο προσαρμοζόμενο FIR φίλτρο η αντίστοιχη έξοδος $\hat{r}(n)$ και το σήμα του λάθους $e(n)$ και υπολογίζονται και οι συντελεστές του φίλτρου οι οποίοι ανανεώνονται σύμφωνα με τον παρακάτω τύπο:

$$a_k(n+1) = a_k(n) + \Delta e(n)y(n) \tag{5.4}$$

όπου k =0,1,...,N και n =1,2,...,Δ είναι η παράμετρος του βήματος και $y(n)$ είναι το δείγμα εισόδου.

Αυτή η σχέση χρησιμοποιείται στον απλό LMS αλγόριθμο. Στον αλγόριθμο NLMS[2], όπου πραγματοποιείται μία απλή κανονικοποίηση του βήματος του αλγορίθμου, ο παραπάνω τύπος γίνεται ως εξής:

$$a_k(n+1) = a_k(n) + \frac{\mu}{P(n)} e(n)y(n) \tag{5.5}$$

[2]Κανονικοποιημένος LMS

όπου $P(n)$ είναι μία εκτίμηση της ισχύος του σήματος εισόδου $y(n)$ και μ είναι το βήμα της εξίσωσης. Ας σημειωθεί ότι για ένα δεδομένο δείγμα n η ποσότητα $\frac{\mu}{P(n)}e(n)$ είναι σταθερή και χρειάζεται να υπολογίζεται μία φορά για κάθε ανανέωση μίας ομάδας συντελεστών. Οι ιδιότητες της σύγκλισης του αλγορίθμου NLMS προσδιορίζονται κατά ένα μεγάλο μέρος από την παράμετρο του βήματος καθώς και από την ισχύ του σήματος $P(n)$. Ο αλγόριθμος AEC μπορεί να ανανεώνει τους εν λόγω συντελεστές έως και μετά από κάθε νέο δείγμα εισόδου (π.χ. ως και 8000 φορές το δευτερόλεπτο[3]).

Ο NLMS αλγόριθμος απαιτεί μία πρόσθεση και έναν πολλαπλασιασμό ανά δείγμα. οπότε ένα φίλτρο n συντελεστών έχει μία πολυπλοκότητα της τάξης $O(n)$. Μοντελοποιώντας το πρόβλημα της ελαχιστοποίησης του λάθους έχουμε:

$$e(n) = r(n) - \alpha^T(n)y(n) \tag{5.6}$$

το οποίο οδηγεί στο μέσο τετράγωνο του λάθους να είναι μία δευτεροβάθμια συνάρτηση των συντελεστών του φίλτρου.

Γενικότερα, η αύξηση της τιμής της παραμέτρου του βήματος επιταχύνει την σύγκλιση του αλγορίθμου, ενώ η μείωση της τιμής της μειώνει την ασυμπτωτική της καταστολής της ηχούς. Η σταθερά της σύγκλισης του χρόνου είναι αντιστρόφως ανάλογη προς την ισχύ του σήματος $y(n)$ και κατά συνέπεια ο αλγόριθμος συγκλίνει πολύ αργά για σήματα χαμηλής ισχύος. Η γενικότερη σχέση για τον υπολογισμό της μέσης τιμής της ισχύος του σήματος εισόδου είναι

$$P_y(n) = L_y^2(n) \tag{5.7}$$

[3] Σε φυσιολογικές περιπτώσεις, για 8KHz PCM κωδικοποίηση φωνητικών δειγμάτων οι συντελεστές του φίλτρου μπορεί να ανανεώνονται με μη προβλέψιμο και κανονικό χρονικά τοποθετημένα τρόπο, όταν το εναπομείναν σήμα λάθους $e(n)$ ξεπερνάει ένα κατώφλι, ή όταν αλλάζει η κατάσταση της ομιλίας στο κανάλι επικοινωνίας.

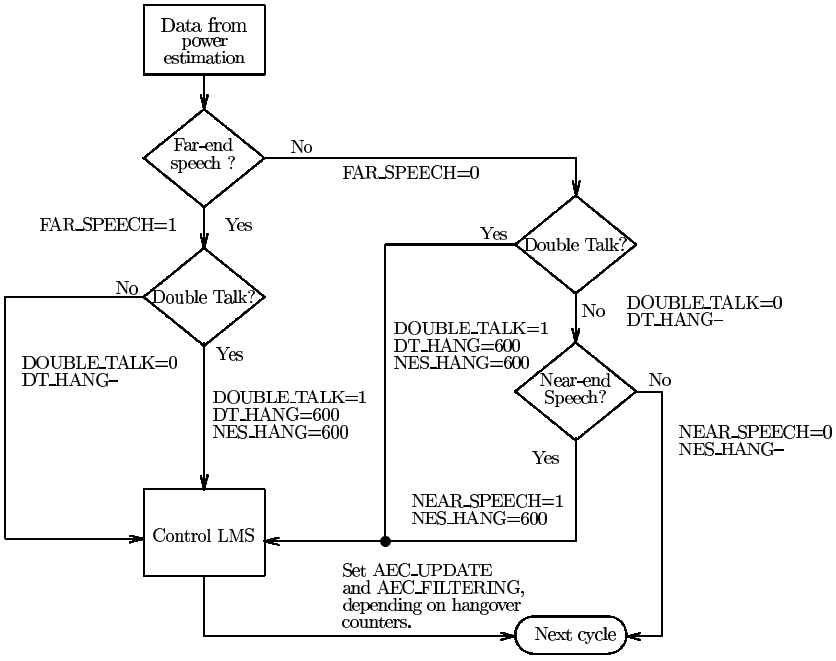
όπου το L_y^n παρουσιάζεται στη σχέση 5.8. Ο όρος ρ είναι μία σταθερά η οποία μπορεί και μεταβάλλεται μεταξύ του μηδενός και της μονάδας.

$$L_y(n) = (1-\rho)L_y(n-1) + \rho|y(n-1)| \tag{5.8}$$

5.2 Ανάλυση του αλγορίθμου LMS

Η χρησιμοποίηση του LMS για την περίπτωση της καταστολής της ηχούς αποτελεί ένα παράδειγμα το οποίο παραδίδει ικανοποιητικά αποτελέσματα όσον αφορά την ποιότητα της φωνής. Αν και δεν είναι η πλέον λύση ακριβείας όσον αφορά την ποιότητα, ο εν λόγω αλγόριθμος είναι μία χαμηλού κόστους λύση. Στις παρακάτω παραγράφους πραγματοποιείται μία ανάλυση του αλγορίθμου σε τμήματα τα οποία μπορούν να θεωρηθούν ως ανεξάρτητες οντότητες μέσα στον αλγόριθμο. Το σημείο εκκίνησης για την ανάλυση αυτού του αλγορίθμου είναι η κατάτμηση του σε ανεξάρτητες συναρτήσεις λειτουργιών και την απεικόνιση τους σε ένα γράφημα ροής όπως αυτό φαίνεται στο Σχήμα 5.3. Για την απλοποίηση του γραφήματος το μπλοκ "Control LMS" είναι αυτό που περιλαμβάνει τις λειτουργίες του φιλτραρίσματος και της ανανέωσης των συντελεστών. Ονομαστικά ο αλγόριθμος αποτελείται από τις εξής οντότητες, μη υπολογίζοντας φυσικά τις συναρτήσεις αρχικοποίησης και τις μη σχετικές με τον αλγόριθμο λειτουργίες της συγκεκριμένης υλοποίησης.

- `power()`: Η συνάρτηση αυτή χρησιμοποιείται για τον υπολογισμό της ισχύος των σημάτων φωνής. Συγκεκριμένα υπολογίζονται οι ισχύς των σημάτων του απομακρυσμένου ομιλητή, η ισχύς του σήματος του τοπικού ομιλητή και η ισχύς του παραμένοντος σήματος μετά την προσπάθεια καταστολής της ηχούς. Ειδικά για το σήμα του τοπικού ομιλητή οι έλεγχοι της ισχύος γίνονται με τρία διαφορετικά βάρη έτσι ώστε να εκτιμηθεί με



Σχήμα 5.3: Γράφημα ροής της καταστολής με LMS

μεγαλύτερη ακρίβεια το μέγεθος της ισχύος ανάλογα με το κανάλι της ηχούς. Η ισχύς υπολογίζεται για μικρό, μεσαίο και μεγάλο κανάλι ηχούς.

- `SP_detect()`: Με αυτήν την συνάρτηση υπολογίζεται η ύπαρξη φωνής καθώς και ο τύπος της στο κανάλι της επικοινωνίας. Ένας πολύ σημαντικός παράγοντας για την απόφαση ή όχι της εφαρμογής της καταστολής της ηχούς σε ένα κανάλι είναι η παρουσία ή η απουσία ταυτόχρονης συνομιλίας μεταξύ των ομιλητών. Γενικότερα ο αλγόριθμος LMS εφαρμόζεται όταν μιλάει μόνο ο απομακρυσμένος ομιλητής εφόσον όταν μιλάει ο τοπικός ομιλητής δεν υπάρχει λόγος για καταστολή όταν υπάρχει μόνο στο τοπικό άκρο η τηλεφωνική συσκευή ανοιχτής ακρόασης. Αν και κατά τη διάρκεια της ταυτόχρονης συνομιλίας υπάρχει μεγάλη πιθανότητα να υπάρχει το φαινόμενο της επιστροφής του σήματος και άρα η ύπαρξη ηχούς

στο κανάλι, δεν εφαρμόζεται η καταστολή λόγω του ότι είναι πρακτικά αδύνατο για ένα σύστημα τέτοιου τύπου να διαχωρίσει τα σήματα του τοπικού και του απομακρυσμένου ομιλητή. Η ITU-T υποστηρίζει πως κατά τη διάρκεια μίας τηλεφωνικής συνομιλίας, στατιστικά υπάρχει μόνο ένα 5% του χρόνου κατά τον οποίο υπάρχει ταυτόχρονη συνομιλία και από τα δύο άκρα. Σε συνδυασμό με το γεγονός ότι η καταληπτότητα της φωνής σε τέτοιες περιπτώσεις είναι από τη φύση της αρκετά μειωμένη και η καταστολή της πιθανής ηχώ δεν θα βελτίωνε ουσιαστικά την ποιότητα, έχει αποφασισθεί να μην πραγματοποιείται καταστολή σε αυτές τις περιπτώσεις. Ειδικά για αυτήν την περίπτωση η εφαρμογή του αλγορίθμου LMS πολύ πιθανόν να οδηγούσε σε απόκλιση του φίλτρου, λαμβάνοντας αρκετά χειρότερο σήμα από ότι θα υπήρχε και χωρίς την εφαρμογή της καταστολής.

- `FIR()`: Όποτε απαιτείται διόρθωση του σήματος λόγω ύπαρξης ηχούς, η συνάρτηση του φίλτρου ενεργοποιείται για να εκτιμήσει την ποσότητα της ηχούς ώστε να δημιουργηθεί ένα αντίγραφο. Το αντίγραφο αυτό στη συνέχεια αφαιρείται από την πραγματική ηχώ και όσο καλύτερη είναι η προσέγγιση τόσο καλύτερη θα είναι η καταστολή. Στη συγκεκριμένη υλοποίηση χρησιμοποιείται φίλτρο με 1024 συντελεστές. Τυπικές τιμές για την υλοποίηση τέτοιου είδους φίλτρων είναι από 64 έως 1024. Το πλήθος των συντελεστών σε περιπτώσεις καταστολέων ηχούς σχετίζονται άμεσα με το μήκος του καναλιού της ηχούς. Συγκεκριμένα φίλτρα με 64 συντελεστές χρησιμοποιούνται σε χώρους μεγέθους αυτοκινήτων και η αύξηση του χώρου αυξάνεται γραμμικά με την αύξηση των συντελεστών.

- `coef_update()`: Κατά την εκκίνηση της λειτουργίας ενός καταστολέα ηχούς, το πρώτο βήμα που πρέπει να πραγματοποιηθεί είναι η αναγνώριση

του χώρου από πλευράς του μήκους του καναλιού της ηχούς. Πιο απλά ο καταστολέας ηχούς θα πρέπει να "εκπαιδευθεί" ώστε να μάθει πιο είναι το μακρύτερο κανάλι ηχούς στον χώρο. Κατά τη διάρκεια που πραγματοποιείται αυτή η διαδικασία, μία συνάρτηση που ονομάζεται `coef_update`, εκτελείται σε κάθε χρονοθυρίδα ώστε να μπορέσει να προσομοιώσει το χώρο. Κάθε μεταβολή που περιγράφηκε παραπάνω, όπως για παράδειγμα μία μετακίνηση των ανθρώπων που βρίσκονται στο χώρο, ή η μετακίνηση αντικειμένων που θα άλλαζαν τη συμπεριφορά του χώρου από πλευράς μήκους καναλιού, ενεργοποιεί τη συνάρτηση αυτή. Λανθασμένες παράμετροι στη συνάρτηση αυτή μπορεί να οδηγήσουν σε συνεχή προσπάθεια εξομοίωσης του χώρου χωρίς επιτυχία. Μία σημαντική παράμετρος που μπορεί να επηρεάσει την ορθή λειτουργία του συστήματος, είναι αυτή η οποία καθορίζει το κατά πόσο υπάρχει πραγματική ανάγκη για επαναπροσαρμογή του μοντέλου του χώρου. Για τον λόγο αυτό, στη συγκεκριμένη υλοποίηση έχει χρησιμοποιηθεί ένας μηχανισμός ο οποίος επιτρέπει την επαναπροσαρμογή του μοντέλου μετά από ένα μη δυναμικό χρονικό όριο. Αυτή η συνάρτηση καθώς και η συνάρτηση του φίλτρου `FIR()` αποτελούν τις πιο απαιτητικές από πλευράς επεξεργαστικής ισχύος.

- `NLP()`: Στις περιπτώσεις που η ενέργεια στο κανάλι επικοινωνίας είναι πολύ χαμηλή, δεν υπάρχει πραγματικός λόγος να εφαρμόζεται ο αλγόριθμος καταστολής. Ακόμη και οι έλεγχοι για φιλτράρισμα του σήματος, απαιτούν ένα σημαντικό κομμάτι επεξεργαστικής δύναμης για την εκτέλεσή τους. Για τον λόγο αυτό έχει υλοποιηθεί μία συνάρτηση μη γραμμικού επεξεργαστή[4], η οποία ελέγχει για ένα χαμηλό κατώφλι ενέργειας το οποίο καθορίζεται πριν την εκκίνηση του αλγορίθμου και όταν απαιτείται μηδενίζει το σήμα στην έξοδο χωρίς να εφαρμόζεται ο αλγόριθμος LMS.

[4]Non-linear processor

- `residual_err()`: Κάθε φορά που το φίλτρο υπολογίζει την νέα εκτίμηση του σήματος της πιθανής ηχούς, πρέπει να αφαιρέσει το αντίγραφο αυτό της εκτίμησης από το σήμα που βρίσκεται στο εν λόγω κανάλι. Συνεπώς, αυτό το `residual_err` σήμα θα πρέπει να οδηγείται όλο και περισσότερο προς το μηδέν, όσο ο αλγόριθμος συγκλίνει. Άλλωστε, αυτός είναι και ο στόχος του LMS αλγορίθμου· η ελαχιστοποίηση του λάθους, το οποίο λάθος έγκειται στην κατά προσέγγιση εκτίμηση του σήματος της ηχούς.

- `control_lms()`: Η συνάρτηση αυτή είναι ίσως από τις πιο κρίσιμες συναρτήσεις διότι καθορίζει το αν και πότε πρέπει να κληθούν οι συναρτήσεις της ανανέωσης των συντελεστών καθώς και το φιλτράρισμα των δεδομένων. Σύμφωνα με το Σχήμα 5.3 αν εντοπισθεί από την συνάρτηση `SP_detect()` σήμα στον τοπικό ομιλητή, μία παράμετρος η οποία ονομάζεται `NEAR_SPEECH` τίθεται στην τιμη '1' και ένας μετρητής `NES_HANG` αρχικοποιείται στην υψηλότερη τιμή του[5]. Οι μετρητές τέτοιου τύπου χρησιμοποιούνται για να υποδείξουν αν υπάρχει ανάγκη για φιλτράρισμα ή ανανέωση των συντελεστών. Για παράδειγμα στην περίπτωση όπου η τιμή του μετρητή `NES_HANG` φτάνει στο μηδέν αυτό σημαίνει ότι δεν υπάρχει σήμα τοπικού ομιλητή για μεγάλο σχετικά χρονικό διάστημα. Αντίστοιχα έχει καθοριστεί και η παράμετρος `DT_HANG` για τον έλεγχο της ταυτόχρονης συνομιλίας. Ο Πίνακας 5.1 παρουσιάζει τις συνθήκες υπό τις οποίες ενεργοποιείται το φιλτράρισμα ή/και η ανανέωση στη συγκεκριμένη υλοποίηση.

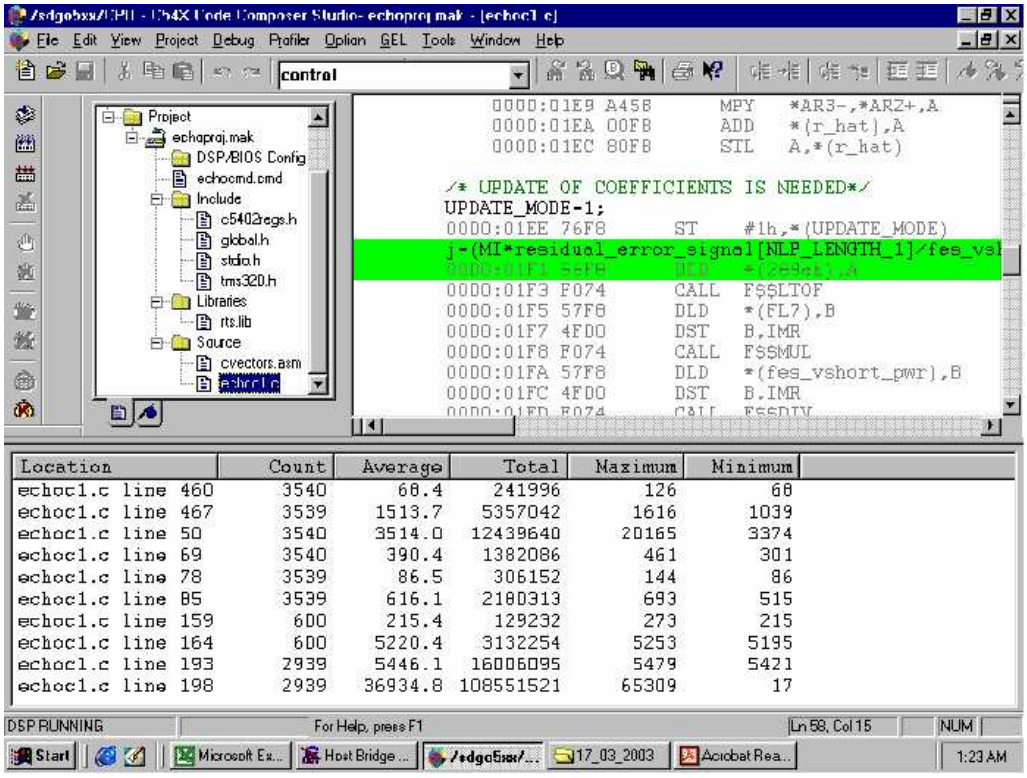
[5] Στην συγκεκριμένη υλοποίηση αυτή η τιμή είναι ιση με 600.

Μετρητής 1	Μετρητής 2	Φιλτράρισμα	Ανανέωση
DT_HANG≥0	NES_HANG≥0	ΝΑΙ	ΟΧΙ
DT_HANG≥0	NES_HANG<0	ΝΑΙ	ΟΧΙ
DT_HANG<0	NES_HANG≥0	ΟΧΙ	ΟΧΙ
DT_HANG≥0	NES_HANG<0	ΝΑΙ	ΝΑΙ

Πίνακας 5.1: Ενέργειες του LMS σύμφωνα με τους μετρητές

5.2.1 Υλοποίηση σε λογισμικό

Συστήματα που επεξεργάζονται με ψηφιακό τρόπο σήματα φωνής υλοποιούνται συνήθως σε επεξεργαστές ψηφιακών σημάτων τα οποία είναι γνωστά και ως DSP. Αυτό συμβαίνει επειδή αυτού του τύπου οι επεξεργαστές έχουν το πλεονέκτημα να εκτελούν πολύ γρήγορα λειτουργίες οι οποίες συναντούνται συχνά σε τέτοιου είδους αλγορίθμους. Για παράδειγμα η λειτουργία του πολλαπλασιασμού, της πρόσθεση και της ανάθεσης (`MAC`), σε μία θέση μνήμης γίνεται με μία μόνο εντολή και απαιτεί ελάχιστους κύκλους. Η υλοποίηση που προτείνεται και παρουσιάζεται εδώ έχει πραγματοποιηθεί σε ένα σύστημα της εταιρίας Spectrum Digital το οποίο είναι εξοπλισμένο με έναν επεξεργαστή τύπου DSP και συγκεκριμένα τον `TMS320C5402` της εταιρείας Texas Instruments. Ο χρονισμός αυτού του επεξεργαστή γίνεται στα 100 MHz και ο διαθέσιμος χρόνο για επεξεργασία με τον δεδομένο αλγόριθμο είναι 4 ms. Αυτός ο χρόνος προκύπτει από το γεγονός ότι η δειγματοληψία γίνεται με ρολόι των 8 KHz. Όπως προκύπτει από τον Πίνακα 5.2 οι χρόνοι για να εκτελεστεί ο αλγόριθμος για φίλτρο μήκους 1024, είναι αρκετά χαμηλότερος από τον διαθέσιμο χρόνο επεξεργασίας από το DSP. Συνεπώς δεν θα υπήρχε κανένα πρόβλημα να υλοποιηθεί ικανοποιητικά ένας τέτοιος αλγόριθμος. Ωστόσο, επειδή φαίνεται ότι περισσεύει αρκετή επεξεργαστική ισχύ στην χρονοθυρίδα των 4 ms, κά-



Σχήμα 5.4: Το περιβάλλον CC Studio και η λήψη των μετρήσεων

νουμε μία εκτίμηση για το πόσα παράλληλα κανάλια φωνής θα μπορούσε να υποστηρίξει ο εν λόγω επεξεργαστής όταν λειτουργεί για καταστολή χώρων που χαρακτηρίζονται από 1024 συντελεστές. Στο Σχήμα 5.4 παρουσιάζεται το περιβάλλον ανάπτυξης του λογισμικού για το συγκεκριμένο DSP και οι λαμβανόμενες τιμές σε επίπεδο κύκλων ρολογιού. Οπότε ο Πίνακας 5.2 έχει προκύψει από εκτέλεση του αλγορίθμου όταν θεωρούμε ότι ο επεξεργαστής που εκτελεί τον κώδικα είναι χρονισμένος στα 100 MHz που είναι και η υψηλότερη συχνότητα λειτουργίας του TMS320C5402. Η στήλη "Κανάλια" παρουσιάζει το πόσα κανάλια μπορεί να εξυπηρετήσει ο επεξεργαστής με αυτήν την υλοποίηση μέσα σε 4 ms. Ο αριθμός των καναλιών σχετίζεται άμεσα με την περισσότερο επεξεργαστικά περίπλοκη συνάρτηση του αλγορίθμου. Σύμφωνα με τον Πίνακα 5.2

Κόμβος	**Συνάρτηση**	**Κύκλοι ρολογιού**	**Χρόνος** μs	**Κανάλια**
0	`power()`	1514	15.14	264
1	`SP_detect()`	616	6.16	649
2	`FIR()`	5220	52.20	77
3	`coef_update()`	36935	369.35	11
4	`NLP()`	390	3.90	1026
5	`residual()`	71	0.71	5634
6	`control_lms()`	83	8.30	4819
Σύνολο	Συναρτήσεων	44829	**448.29**	**8**

Πίνακας 5.2: Χρόνοι εκτέλεσης των συναρτήσεων σε λογισμικό

οι περισσότερο χρονοβόρες συναρτήσεις είναι οι `coef_update()` και η `FIR()`. Οι κύκλοι ρολογιού και οι χρόνοι εκτέλεσης που παρουσιάζονται, προκύπτουν από υλοποίηση του αλγορίθμου μέσω της βιβλιοθήκης πραγματικού χρόνου μαθηματικού συναρτήσεων (`rts.lib`) της Texas Instruments. Επίσης θεωρούμε μία ονοματολογία κόμβων (nodes), για κάθε συνάρτηση του συστήματος όπως φαίνεται και στην πρώτη στήλη.

Από τον Πίνακα 5.2 φαίνεται ότι η συνάρτηση `coef_update()` θα μπορούσε να εκτελεσθεί ικανοποιητικά στο διαθέσιμο χρονικό διάστημα. Ωστόσο, όταν υπεισέρχονται και άλλες συναρτήσεις για την εκτέλεση του αλγορίθμου φαίνεται ότι τελικά μπορούμε να εκτελέσουμε τον αλγόριθμο το πολύ 8 φορές. Η χρησιμοποίηση του επεξεργαστή (utilization), σε αυτήν την περίπτωση προκύπτει ως εξής.

$$U(N) = \frac{MAX\_Channels \times 2 \times (t_0 + t_1 + \ldots + t_k)}{Sampling\_Rate} \tag{5.9}$$

όπου N ο αριθμός των συντελεστών, $MAX\_Channels$ ο μέγιστος αριθμός επιτρεπόμενων καναλιών που θα εκτελούνται σε μία χρονοθυρίδα, $t_0, t_1, \ldots, t_k$

οι χρόνοι εκτέλεσης του k κόμβου και $Sampling\_Rate$ η συχνότητα δειγματοληψίας. Ας σημειωθεί ότι οι συναρτήσεις οι οποίες ο χρόνος επεξεργασίας του αλλάζει σε σχέση με το N είναι οι `coef_update()` και `FIR()`. Το ποσοστό χρησιμοποίησης επί τοις εκατό όπως προκύπτει από τα παραπάνω για αυτήν την περίπτωση είναι **89.66%**, αν θέσουμε όπου $N = 1024$, $k = 6$, $MAX\_Channels = 8$ και $Sampling\_Rate = 8$ KHz.

5.2.2 Κατάτμηση σε υλικό/λογισμικό

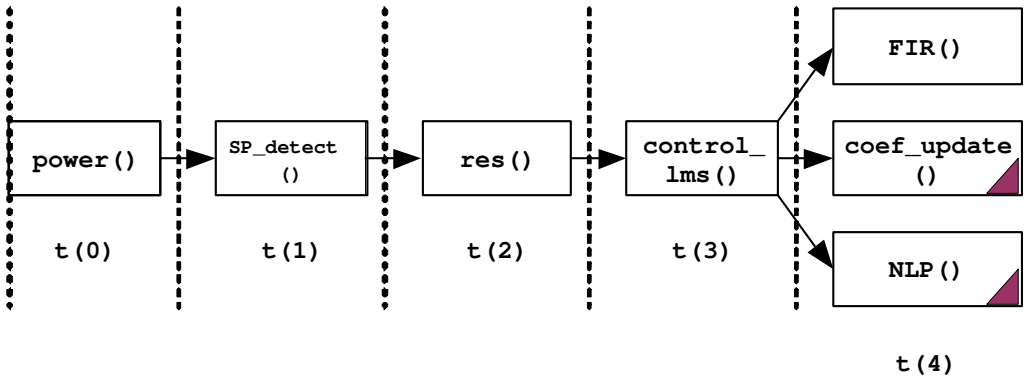
Κατά τη διαδικασία της κατάτμησης υλικού/λογισμικού, οι σχεδιαστές αναζητούν την καλύτερη εξισορρόπηση των παραγόντων μεταξύ των τμημάτων του υλικού και του λογισμικού. Προϋποτίθεται ότι σε συστήματα ροής δεδομένων όπως είναι τα τηλεπικοινωνιακά συστήματα, πρέπει να επιλυθούν τρία βασικά προβλήματα: η *εκχώρηση* (assignment), η *χρονοδρομολόγηση* (scheduling), και η *κατανομή πόρων* (resource allocation).

Ο βασικός στόχος της κατάτμησης υλικού/λογισμικού είναι η ανεύρεση μίας *εκχώρησης* σε υλικό ή λογισμικό για όλα τα τμήματα των προδιαγραφών του συστήματος (όλους τους κόμβους στο συγκεκριμένο παράδειγμα για την καταστολή της ηχούς). Οι χρόνοι της εκτέλεσης αυτών τον κόμβων προσδιορίζονται κατά τη διάρκεια της λειτουργίας της *χρονοδρομολόγησης*. Η μελέτη της χρονοδρομολόγησης επιτρέπει την ικανοποίηση διαφόρων σκοπών. Παραδείγματα τέτοιων σκοπών είναι η ελαχιστοποίηση των πράξεων σε επίπεδο επεξεργασίας, ελαχιστοποίηση της επικοινωνίας μεταξύ υλικού και λογισμικού καθώς και η ελαχιστοποίηση του χρόνου εκτέλεσης. Το πρόβλημα της *κατανομής πόρων* επικεντρώνεται στην ανεύρεση του τύπου και του αριθμού των πόρων που χρησιμοποιούνται για την υλοποίηση του συστήματος. Η προσπάθεια του διαμοιρασμού των πηγών μεταξύ λειτουργιών είναι η βασικότερη δυσκολία. Μία πηγή μπορεί να είναι μία η περισσότερες επεξεργαστικές πράξεις όπως ένα-

ς προσθετής, μία μονάδα που εκτελεί FFT καθώς και στοιχεία μνήμης όπως καταχωρητές FIFO και RAM.

Στη συνέχεια παρουσιάζονται δύο διαφορετικές προσεγγίσεις για την κατάτμηση μεταξύ υλικού και λογισμικού του LMS αλγορίθμου. Στην πρώτη υλοποίηση πραγματοποιείται η ελαχιστοποίηση του υλικού ενώ στη δεύτερη γίνεται η προσπάθεια για υποστήριξη ενός μεγάλου αριθμού καναλιών μετακινώντας ένα σημαντικό μέρος της λειτουργικότητας του συστήματος σε υλικό. Στην πρώτη περίπτωση μετακινείται σε υλικό η λειτουργία της ανανέωσης των συντελεστών του φίλτρου και η επιφάνεια που καταλαμβάνει σε υλικό αυτό το κύκλωμα είναι περίπου 0,16 m^2. Στο Σχήμα 5.5 φαίνεται η χρονική εξάρτηση των παραπάνω υπομονάδων. Παρατηρείται ότι οι περισσότερες λειτουργίες έχουν μία εξάρτηση από κάποια άλλη εφόσον το πρόγραμμα εκτελείται σειριακά. Ωστόσο, είναι φανερό ότι οι κόμβοι του `FIR()` και του `coef_update()` θα μπορούσαν να εκτελούνται όσο τα προηγούμενα βήματα λαμβάνουν χώρα. Παρά το γεγονός ότι υπάρχουν τρεις συναρτήσεις στο χρονικό διάστημα t_4 από τον Πίνακα 5.2 φαίνεται ότι η μετακίνηση των `FIR()` και `coef_update()` θα οδηγούσε σε σημαντικά αποτελέσματα όσον αφορά τη χρονοδρομολόγηση του συστήματος. Το αποτέλεσμα μιας τέτοιας κατάτμησης θα οδηγούσε το σύστημα μαζί με ένα μπλοκ υλικού να διαχειρίζεται πολύ περισσότερα κανάλια από ότι με την κλασσική υλοποίηση σε λογισμικό.

Αντικαθιστώντας στη Σχέση 5.2.1 τις αντίστοιχες τιμές βάσει του Πίνακα 5.3, συμπεραίνουμε πως με μια τέτοια υλοποίηση θα ήταν δυνατή η υποστήριξη 50 καναλιών και η χρησιμοποίηση του επεξεργαστή θα έφτανε στο **98,68%**. Η αύξηση αυτή αντιστοιχεί σε υποστήριξη καναλιών μεγαλύτερη κατά 625% εφόσον ο συνολικός χρόνος εκτέλεσης έχει μειωθεί από $\mathbf{448.29}\mu s$ σε $\mathbf{78.94}\mu s$. Ουσιαστικά αυτή είναι και η διαφορά η οποία δίνει τη δυνατότητα από το συγκεκριμένο DSP να λειτουργεί για 50 διαφορετικά κανάλια στο



Σχήμα 5.5: Χρονική κατανομή εκτέλεσης στον LMS

Κόμβος	**Συνάρτηση**	**Υλικό**	**Λογισμικό**	**Χρόνος**
0	`power()`		$15.14\mu s$	$0 \rightarrow 15.14\mu s$
1	`SP_detect()`		$6.16\mu s$	$15.14 \rightarrow 21.30\mu s$
2	`FIR()`		$52.20\mu s$	$21.30 \rightarrow 73.50\mu s$
3	`coef_update()`	$\sim 12ns$		$73.50 \rightarrow 73.50\mu s$
4	`NLP()`		$3.90\mu s$	$73.50 \rightarrow 77.40\mu s$
5	`residual()`		$0.71\mu s$	$77.40 \rightarrow 78.11\mu s$
6	`control_lms()`		$8.30\mu s$	$78.11 \rightarrow \mathbf{78.94}\mu s$

Πίνακας 5.3: Χρόνοι εκτέλεσης υπό κατάτμηση (`coef_update()`)

Κόμβος	**Συνάρτηση**	**Υλικό**	**Λογισμικό**	**Χρόνος**
0	`power()`		$15.14\mu s$	$0 \rightarrow 15.14\mu s$
1	`SP_detect()`		$6.16\mu s$	$15.14 \rightarrow 21.30\mu s$
2	`FIR()`	$\sim 17ns$		$21.30 \rightarrow 21.30\mu s$
3	`coef_update()`	$\sim 12ns$		$21.30 \rightarrow 21.30\mu s$
4	`NLP()`		$3.90\mu s$	$21.30 \rightarrow 25.20\mu s$
5	`residual()`		$0.71\mu s$	$25.20 \rightarrow 25.91\mu s$
6	`control_lms()`		$8.30\mu s$	$25.91 \rightarrow \mathbf{26.74}\mu s$

Πίνακας 5.4: Χρόνοι εκτέλεσης υπό κατάτμηση (`coef_update()` και `FIR()`)

χρονικό πλαίσιο που είναι διαθέσιμο.

Μία επιπλέον μετακίνηση μιας λειτουργίας στο υλικό θα έδινε ασφαλώς ακόμη καλύτερα αποτελέσματα όσον αφορά τη διαθεσιμότητα του υλικού. Μετακινώντας την αμέσως περισσότερο χρονοβόρα διαδικασία σε υλικό, που αυτή είναι η υλοποίηση του φίλτρου FIR, θα οδηγούσε σε βελτιστοποίηση του χρόνου εκτέλεσης από τον DSP αλλά ταυτόχρονα θα αύξανε και την επιφάνεια του υλικού εφόσον θα απαιτούταν υλικό τόσο για τη λειτουργία ανανέωσης των συντελεστών του φίλτρου αλλά και για την λειτουργία του ιδίου του φίλτρου. Αντικαθιστώντας παρομοίως, όπως και προηγουμένως, τις τιμές στη Σχέση 5.2.1 προκύπτει ο Πίνακας 5.4, θέτοντας την τιμή για το $Max\_Channels$ ίση με 149. Με την επιπλέον αυτή μετακίνηση το κέρδος που αποκομίζεται σε σχέση με την προηγούμενη κατάτμηση είναι της τάξης του 300% εφόσον τώρα μπορούν να υποστηριχθούν έως και 149 κανάλια φωνής. Επίσης, η χρησιμοποίηση του επεξεργαστή με τη δεύτερη κατάτμηση φτάνει στο **99.61%**. Ο Πίνακας 5.5 παρουσιάζει τα αποτελέσματα όσον αφορά τη βελτίωση στο χρόνο επεξεργασίας. Επίσης ο Πίνακας 5.6 παρουσιάζει συνοπτικά ποιο είναι το πραγματικό

Υλικό	**Εκμετάλλευση DSP**	**Κανάλια**	**Βελτίωση DSP**
-	89.66%	8	-
`coef_update()`	98.68%	50	625%
`FIR()`	99.61%	149	1862.5%

Πίνακας 5.5: Χρόνοι εκτέλεσης υπό κατάτμηση (`coef_update()` και `FIR()`)

Συνάρτηση	**Cells**	**SRAM**	**Επιφάνεια** μm^2	**Καθυστέρηση**
`FIR()`	26714	20,5 Kb	16696,25	17,01 ns
`coef_update()`	25126	19,1 Kb	15703,25	11,84 ns

Πίνακας 5.6: Ανάλυση υλικού για τις λειτουργίες `coef_update()` και `FIR()` σε FPGA Xilinx Vertex II

κόστος του συστήματος που μεταφέρουμε σε υλικό, το οποίο παρουσιάζεται σε επιφάνεια και σε καθυστέρηση. Η πλατφόρμα η οποία χρησιμοποιείται για την μελέτη των όσων παρουσιάζονται στον Πίνακα 5.6 είναι τα FPGA Xilinx Vertex II.

Κεφάλαιο 6

Συμπεράσματα

Η κατάτμηση υλικού και λογισμικού σε πολύπλοκα ενσωματωμένα τηλεπικοινωνιακά συστήματα αποτελεί μία πρόκληση για τον σχεδιαστή, ώστε να επιτύχει το καλύτερο δυνατό αποτέλεσμα, με το ανταγωνιστικότερο κόστος και τον λιγότερο χρόνο ανάπτυξης. Η ταχύτατη εξέλιξη στο χώρο, απαιτεί μεθόδους για τον ταυτόχρονο σχεδιασμό συστημάτων που συνδυάζουν υλικό αλλά και λογισμικό, οι οποίες θα είναι αποδοτικές χωρίς την εισαγωγή σημαντικών σφαλμάτων. Ωστόσο, στην παρούσα έρευνα παρουσιάζεται ότι οι διαδικασίες για κατάτμηση υλικού και λογισμικού, δεν αποτελούν διεργασίες οι οποίες μπορεί να εκτελεστούν με εντελώς αυτοματοποιημένο τρόπο. Αν και τα σύγχρονα εργαλεία για το σκοπό αυτό υπόσχονται ικανοποιητικά αποτελέσματα, ο πειραματισμός δείχνει ότι η επέμβαση του Μηχανικού είναι η αναντικατάσταση. Πέρα από την επιλογή της μεθόδου, υπάρχουν πολλά σημεία που η ανθρώπινη παρέμβαση οδηγεί σε λύσεις, εκεί που τα αυτοματοποιημένα εργαλεία μπορεί να έρθουν σε αδιέξοδο ή ακόμη και να παρακάμψουν μικρά "λάθη".

Το σίγουρο είναι ότι η κατάτμηση υλικού και λογισμικού αποτελεί πλέον μία προσέγγιση που πρέπει να λαμβάνεται υπόψη σε κάθε σχεδιασμό ενός πολύπλοκου ενσωματωμένου συστήματος. Αυτό ενισχύεται ακόμη περισσότερο για

την κατηγορία των τηλεπικοινωνιακών συστημάτων που πολλές φορές θέτουν περιορισμένα χρονικά περιθώρια για την διεκπεραίωση περίπλοκων διεργασιών. Η εξέλιξη δείχνει ότι αυτά τα συστήματα τείνουν να ενσωματώσουν ολοένα και περισσότερες λειτουργίες, καθιστώντας τις συσκευές αυτές σαν μικρογραφίες υπολογιστών υψηλής πολυπλοκότητας τη στιγμή που τα σχεδιαστικά λάθη πρέπει να ελαχιστοποιούνται αν όχι να μηδενίζονται. Παρακάτω παρουσιάζονται τα βασικά στοιχεία τα οποία αποτελούν και τα συμπεράσματα της διατριβής. Συνοπτικά αυτά είναι:

- Η επισήμανση της ανάγκης της κατάτμησης υλικού και λογισμικού στα σύγχρονα ενσωματωμένα τηλεπικοινωνιακά συστήματα. Το κόστος ανάπτυξης στο οποίο οδηγούμαστε απουσίας της κατάτμησης μπορεί να είναι σημαντικά αυξημένο, καθώς δεν είναι δυνατή η λήψη αποφάσεων για σημαντικές αλλαγές, τη στιγμή που ο σχεδιασμός έχει προχωρήσει στο στάδιο της ανάπτυξης. Η ενσωμάτωση λειτουργιών σε υλικό χωρίς κριτήρια κόστους μπορεί να οδηγήσει σε σημαντική αύξηση του τελευταίου, χωρίς όμως να είναι πραγματικά απαραίτητη. Επίσης, η κατάτμηση έχει και σαν αποτέλεσμα την σημαντική μείωση του ρίσκου υλοποίησης ενός ενσωματωμένου τηλεπικοινωνιακού συστήματος, λόγω του εντοπισμού λανθασμένων αποφάσεων κατά τη διαδικασία της κατάτμησης.

- Η πρόταση τεχνικών σχεδίασης από τα πολύ πρώιμα στάδια της υλοποίησης. Οι διαδικασίες κατάτμησης που προτείνονται εφαρμόζονται από τα στάδια των προδιαγραφών του συστήματος αποφεύγοντας έτσι πιθανά σφάλματα τη στιγμή που η επέμβαση στο σχεδιασμό δεν θα ήταν πλέον δυνατή.

- Η παρουσίαση ενός ενοποιημένου περιβάλλοντος μέσω του οποίου τα αποτελέσματα της κατάτμησης παρέχονται στον σχεδιαστή, έτσι ώστε να

λαμβάνει εύκολα αποφάσεις σχετικά με την περαιτέρω πρόοδο του σχεδιασμού. Η λειτουργία του περιβάλλοντος δεν περιορίζεται σε αυτοματοποιημένες και μόνο διαδικασίες κοινών αλγορίθμων, αλλά επιτρέπει και την εμπλοκή της κρίσης του σχεδιαστή για ακόμη πιο ακριβή αποτελέσματα. Η επέμβαση του σχεδιαστή, όπως διαπιστώθηκε, αποτελεί σημαντική παρέμβαση για το σύστημα, η αξία της οποίας είναι ανεκτίμητη και αναντικατάστατη.

- Η πρόταση κατάτμησης ενσωματωμένων συστημάτων ψηφιακής επεξεργασίας σημάτων, τα οποία διαφέρουν σημαντικά στη συμπεριφορά από τα κλασσικά ενσωματωμένα συστήματα που συνήθως μελετούνται. Οι μαθηματικές λειτουργίες και η διαφορά της φύσης των εφαρμογών ψηφιακής επεξεργασίας σημάτων, δημιουργούν εξίσου πολύπλοκα συστήματα τα οποία απαιτούν την κατάτμηση για την μείωση του κόστους και την καλύτερη εκμετάλλευση των διαθέσιμων πόρων.

- Η πρόταση τεχνικών κατάτμησης από προδιαγραφές υλικού, από προδιαγραφές λογισμικού αλλά και από ήδη υπάρχουσες πρότυπες ή μη υλοποιήσεις πολύπλοκων λογισμικών όπως είναι για παράδειγμα οι στοίβες της σηματοδοσίας πρωτοκόλλων.

Η έρευνα που παρουσιάζεται δεν σταματά με την περάτωση αυτής της διατριβής αλλά προτείνει μελλοντικούς στόχους και προκλήσεις για τον τομέα της κατάτμησης. Συγκεκριμένα προτείνεται:

- Η δημιουργία μοντέλων πολλών εμπορικών επεξεργαστών οι οποίοι δεν είναι μεν τόσο διαδεδομένοι. Πολλές φορές ένας σχεδιαστής αποφασίζει να χρησιμοποιήσει συγκεκριμένες επεξεργαστικές μονάδες λογισμικού σε ενσωματωμένα συστήματα, εξαιτίας της ύπαρξης υποστήριξης τους είτε

από την ίδια την εταιρεία κατασκευής, είτε από τα εργαλεία που χρησιμοποιούνται για την κατάτμηση. Κλασσικό παράδειγμα αποτελεί η περίπτωση του συστήματος του PRO^3 όπου ο επεξεργαστής που χρησιμοποιήθηκε δεν ήταν ο πλέον εμπορικός τη χρονική στιγμή που έγινε η επιλογή του. Σε αυτές τις περιπτώσεις είτε μοντελοποιείται ο συγκεκριμένος επεξεργαστής είτε προσεγγίζεται από κάποια μονάδα με παρόμοια χαρακτηριστικά.

- Η ανάλυση του επιπλέον φόρτου ενός λειτουργικού συστήματος. Συχνά παρουσιάζονται στατιστικά μοντέλα για επεξεργαστικές μονάδες, προβάλλοντας τις δυνατότητές τους όσον αφορά την ταχύτητα εκτέλεσης ενός αλγορίθμου για παράδειγμα. Η αυξημένη πολυπλοκότητα ενός σύγχρονου ενσωματωμένου συστήματος απαιτεί πλέον την ύπαρξη ενός λειτουργικού συστήματος. Ωστόσο, σπάνια συναντά κανείς ανάλυση επιδόσεων ενός επεξεργαστή για τον φόρτο ενός λειτουργικού συστήματος. Η εγκατάσταση για παράδειγμα ενός διαδεδομένου λειτουργικού συστήματος όπως π.χ. το Linux σε ένα ενσωματωμένο τηλεπικοινωνιακό σύστημα χρησιμοποιεί ένα σημαντικό μέρος από τους πόρους του επεξεργαστή. Η εκτίμηση της ποσότητας αυτής οδηγεί σε περισσότερο ρεαλιστικά μέτρα για την ανάπτυξη επιπλέον λογισμικού σε τέτοιου είδους συστήματα.

- Η εξέταση της απόδοσης λόγω της ύπαρξης διεπαφών μεταξύ των διαφόρων στοιχείων του συστήματος. Η ανάλυση που παρουσιάστηκε επικεντρώνεται στην κατάτμηση λειτουργιών δίνοντας έμφαση στην απόδοση η οποία καταναλώνεται για την διεκπεραίωσή τους. Όταν διασυνδέονται όμως πολλά διαφορετικά υποσυστήματα μεταξύ τους, είναι αναγκαία η ύπαρξη διεπαφών για την μεταξύ τους επικοινωνία. Η διερεύνηση της καθυστέρησης που συμβαίνει στην ανταλλαγή πληροφοριών μέσω των

διεπαφών, μερικές φορές παρέχει πληροφορίες που πρέπει να λαμβάνονται σοβαρά υπόψη κατά τη σχεδίαση του συστήματος.

Βιβλιογραφία

[Up90] *United State Patent*, G.P.Hyatt, Pat. no. 4 942 516, July, 1990.

[Coc71] *Computer on a Chip*, IEEE Computer, January, 1971.

[Ev98] *Alpha EV7: A Scalable Single-chip SMP*, P. Bannon, MicroProcessor Forum, October, 1998.

[Mot97] *PowerPC - MPC 750 RISC Microprocessor Technical Summary*, Motorola Inc., 1997

[Intel97] *Pentium II Processor Developer's Manual*, Intel Inc., October 1997.

[Sam99] *256 MBit SDRAM - 16M x 4 bit x 4 Banks Synchronous DRAM*, Datasheet KM416S16230A, Rev. 0.2, Samsung Electronics Inc., January 1999.

[Clark89] *An Analysis of TCP Processing Overhead*, D. Clark, V. Jacobson, J. Romkey, IEEE Communications Magazine, June 1989.

[Ecunm02] *Application Decomposition for High-Speed Network Processing Platforms*, N. Nikolaou, Th. Orfanoudakis, D. Pollatos, N. Zervos, ECUMN02 conference, April02, Colmar, France.

[Harel88] *STATEMATE: A working environment for the development of complex reactive systems*, D. Harel, H. Lachover, A. Naamad, A. Pnue-

li, M. Politi, R. Sherman, and A. Shtul-Trauring, Proceedings of the International Conference on Software Engineering, 1988.

[Ga91] *High-Level Synthesis: Introduction to Chip and System Design*, D.D. Gajski, N.D. Dutt, C.H. Wu, Y.L. Lin, Kluwer Academic Publishers, Boston, Massachusetts, 1991.

[Dem79] *Structured Analysis and System Specification*, T. DeMarco, Yourdon Press, New York, 1979.

[Dav83] *Tools and Techniques for Stuctureed System Analysis and Design*, W.S. Davis, Addison-Wesley, Reading, Massachusets, 1983.

[Sod90] *Computer System Techniques: Development, Implementation and Software Maintenance*, TAB Professional and Reference Books, Blue Ridge Summit, Pennsylvania, 1990.

[Che77] *The Entity-Relationship Approach to Logical Data Base Design*, P.S. Chen, Q.E.D. Information Sciences, Wellesley, Massachusetts, 1977.

[Teo90] *Database Modeling and Design: The Entity-Relationship Approach*, T.J Teory, Morgan Kaufman Publishers, San Mateo, California, 1990.

[Sut88] *Jackson System Development*, A. Sutcliffe, Prentice-Hall, New York, 1988.

[OG86] *Flow graph reprsentation*, A. Orailoglou, D.D. Gajski, IEEE Design and Test of Computers, 1986.

[LG88] *Synthesis for VHDL*, J. Lis, D.D. Gajski, Proceedings of the International Conference on Computer Design, 1988.

[WM85] *Structured Development for Real-Time Systems*, P.T. Ward, S.J. Mellor, Yourdon Press, New York, 1985.

[YC78] *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*, E. Yourdon, L.L. Constantine, Yourdon Press, New York, 1978.

[Char98] *Partitioning Tools Play a Key Role in Top-Down Design*, Small H. Charles, Computer Design, 1998.

[Doer90] *A survery of Light-Weight Transport Protocols for high speed Networks*, W. Doeringer, D. Dykeman, M. Kaiserswerth, B. Meister, H. Rudin, W. Williamson, IEEE Trans. Comm. Vol. 38, No. 11, pp. 2025-2039, Nov 1990.

[Black93] *Data Link Protocols*, U. Black, Englewood, NJ, Prentice-Hall, Inc. 1993.

[Ntua02] *Efficient Protocol Decomposition and Performance Analysis of the SSCOP Layer*, F.E. Andritsopoulos, G.A. Doumenis, Y.M. Mitsos, D.I. Reisis, WSEAS Conference, Crete, 2002.

[Doum93] *Efficient Implementation of the SAR sublayer and the ATM layer in high speed broadband ISDN terminal adapters*, G. Doumenis, D. Reisis, G. Stassinopoulos, IEEE International Conference on Communications, Geneva, 1993.

[ITU95] *B-ISDN Digital Subscriber Signalling System No. 2, User Network Interface for Basic Call/Connection Control*, ITU-T Recommendation Q.2931, Geneva, Switzerland, February 1995.

[Nug98] *SSCOP Protocol Througput Evaluation - Simulation Based on Estelle Specifications*, C. Nugulescu, E. Borcoci, Proceedings of the 1st International Workshop on the Formal Description Technique Estelle, France, 1998.

[Atm99] *Draft UNI Signalling Performance Test Suite*, ATM Forum Technical Committee, July 1999.

[Nieh97] *Performance Benchmarking of Signallin in ATM Networks*, D. Niehaus, A. Battou, A. McFarland, B. Decina, H. Darby, V. Sirkay, B. Edwards, IEEE Communication Magazine, August 1997.

[Gprof00] *GNU gprof*, http://sources.redhat.com/binutils/docs-2.10/gprof.html, GNU Project, June 2000.

[GD92] *Partitioning of functional models of synchronous digital systems*, R. Gupta, G. DeMicheli, Proceedings of the European Conference on Design Automation (EDAC), Pages 2-7 1992.

[EHB94] *Hardware-Software cosynthesis for microcontrollers*, R. Ernst, J. Henkel, T. Berner, IEEE Design & Test of Computers, p. 64-75, December 1994.

[Joh94] *Hierarchical clustering schemes*, S.C. Johnson, Psychometrica, p. 241-254, September 1967.

[LT91] *Architectural partitioning for system level design*, E.D. Lagnese, D.E. Thomas, In Proceedings of the Design Automation Conference, 1989.

[MK90] *Incorporating bottom-up design into hardware synthesis*, M.C. McFarland, T.J. Kowalski, IEEE Transactions on Computer-Aided Design, September 1990.

[CB87] *Partitioning before logic synthesis*, R. Camposano, R.K. Brayton, Proceedings of the International Conference on Computer-Aided Design, 1987.

[KGV83] *Optimization by simulation annealing*, S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Science, 220(4598):671-680, 1983.

[RSV85] *Probabilistic hill climbing algorithms: Properties and applications*, F. Romeo, A. Sangiovanni-Vincentelli, Proceeding of the 1985 Chapel Hill Conference on VLSI, p. 393-417, 1985.

[Len83] *Combinatorial Algorithms for Integrated Circuit Layout*, T. Lengauer, John Wiley and Sons, England, 1990.

[OvG84] *Floorplan design using simulated annealing*, R. Otten, L. van Ginneken, Proceedings of the International Conference on Computer-Aided Design, p. 96-98, 1984.

[HRSV83] *An efficient general cooling schedule for simulated annealing*, M.D. Hung, F. Romeo, A. Sangiovanni-Vincentelli, Proceedings of the International Conference on Computer-Aided Design, p. 381-384, 1986.

[KV93] *Partitioning for multicomponent synthesis from VHDL specifications*, N. Kumar, R. Vemuri, VHDL International Users' Forum, p. 19-28, 1993.

[GV95] *Specification and design of embedded hardware-software systems*, D. Gajsky, F. Vahid, IEEE Journal Design and Test of Computers, p. 53-67, 1995.

[JI95] *Synthesis steps and design models for codesign*, A. Jerraya, T. Ismail, IEEE Computer 28(2):44-52, February, 1995.

[G.165] *General characteristics of International telephone connections and International telephone circuits*, ITU-T Recommendation G.165.

[Hyk86] *Adaptive filter theory*, S. Haykin, Prentice-Hall, Englewood Cliffs, New-Jersey, 1986.

[Co97] *Hardware/Software Co-Design of Digital Telecommunication Systems*, Ivo Bolsens, Hugo J. De Man, Bill Lin, Karl Van Rompaey, Steven Vercauteren, and Diederik Verkest, Proceedings of the IEEE, Vol. 85, no. 3, March 1997.

[Lyc96] *Lycos: the lyngby co-synthesis system*, J. Madsen, J. Grode, P.V. Knudsen, M.E. Petersen, and A. Haxthausen, Design Automation for Embedded Systems, 1996.

[Cos98] *High-Level Estimation Techniques for Usage in Hardware/Software Co-Design*, J. Henkel and R. Ernst, IEEE/ACM Proc. of Asia and South Pacific Design Automation Conference, pages 353-360, Yokohama, Japan, February 1998.

[Hy03] *Hypersone AG*, http://www.hyperstone.com, 2003.

[DGGP01] *System Design: A Practical Guide With Spec C*, Domer Rainer, Gajski Daniel, Gerstlauer Andreas, Peng Junyu, Cluwer Academic Pub (May 1, 2001) ISBN: 0792373871.

[Ntua03] *Verification of a complex SoC; the PRO3 case-study*, F. Andritsopoulos, G. Doumenis, C. Haropoulos, F. Karoubalis, Y. Mitsos, S.

Perissakis, D. Reisis, I. Theologitou, Design Automation and Test Conference (DATE), Munich 2003.

[Cam03] *A Low-Cost Classification Method for High-speed Network Processors*, F. Andritsopoulos, Y. Mitsos, C. Charopoulos, G. Doumenis, D. Reisis, 9th International Conference on Information Systems Analysis and Synthesis: ISAS '03, Orlando, Florida USA, August 2003.

[SCE03] *System-on-Chip Environment SCE*, Center for Embedded Computer Systems, University of California, Irvine, U.S.A., http://www.cecs.uci.edu/ specc/

ΔΗΜΟΣΙΕΥΣΕΙΣ

[1]. I. Theologitou, G.Doumenis, F. Andritsopoulos, T. Orphanoudakis, S. Perisakis, D. Reisis, "System and application verification framework for complex network processors", submitted for publication to: IEEE Communications Magazine, Special Issue on "Testing and Verification of Communication System-on-Chip devices", February 2003.

[2]. Fotis E. Andritsopoulos, Newton Bomeisel Cardoso, Gregory A. Doumenis, Yannis M. Mitsos, Lambros E. Sarakis, "An accurate Dual Tone Multiple Frequency Detector based on the low-complexity Goertzel algorithm", Tecnologia magazine, Vol 23, number 2/2002, ISSN 0101-8191, September 2002.

[3]. Fotis E. Andritsopoulos, Yannis M. Mitsos, Gregory A. Doumenis, Constantin N. Papaodysseus, "Time distributed update of the NLMS algorithm coefficients for Acoustic Echo Cancellers", submitted for publication to: Elsevier, ... special issue.

[4]. F. Andritsopoulos, G. Doumenis, C. Haropoulos, F. Karoubalis, S. Perissakis, D. Reisis, I. Theologitou, "Verification of a complex SoC; the PRO3 case-study", Design Automation and Test Conference (DATE), Munich 2003.

[5]. F.E. Andritsopoulos, G.A. Doumenis, Y.M. Mitsos, D.I. Reisis, "Efficient Protocol Decomposition and Performance Analysis of the SSCOP layer", In WSEAS CSCC (CSCC 2002) MULTICONFERENCE, Rethymnon Crete, July 2002.

[6]. F. Andritsopoulos, Y. Mitsos, C. Charopoulos, G. Doumenis, D. Reisis, "A Low-Cost Classification Method for High-speed Network Processor-

s", In 9th International Conference on Information Systems Analysis and Synthesis: ISAS '03, Orlando, Florida USA, August 2003.

[7]. Fotis E. Andritsopoulos, Newton Bomeisel Cardoso, Gregory A. Doumenis, Yannis M. Mitsos, Lambros E. Sarakis, "An accurate Dual Tone Multiple Frequency Detector based on the low-complexity Goertzel algorithm", In 19th Brazilian Telecommunication Symposium (SBrT), Brazil, September, 2001.

ΒΙΟΓΡΑΦΙΚΟ ΣΗΜΕΙΩΜΑ

Ο υποψήφιος διδάκτορας κ. Φώτιος Ε. Ανδριτσόπουλος, γεννήθηκε στον Πειραιά στις 7 Απριλίου 1976. Το 1994 εισήχθη στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών της Πολυτεχνικής Σχολής του Δημοκριτείου Πανεπιστημίου Θράκης. Αποφοίτησε τον Σεπτέμβριο του 1999, με χαρακτηρισμό πτυχίου "Πολύ Καλά" (7.6). Το θέμα της διπλωματικής εργασίας που εκπόνησε ήταν "Ασύρματο Δίκτυο στη Ζώνη". Η εργασία εκπονήθηκε υπό την επίβλεψη του Αναπληρωτή Καθηγητή του Δημοκριτείου Πανεπιστημίου Θράκης κ. Χρήστου Κουκουρλή και βαθμολογήθηκε με γενικό χαρακτηρισμό "Άριστα" (10). Η διπλωματική αυτή εργασία έλαβε δύο βραβεία το έτος 2000, ένα από την εταιρεία Ericsson Hellas και ένα από το Τεχνικό Επιμελητήριο Ελλάδος (ΤΕΕ).

Το Δεκέμβριο του 1999, έγινε δεκτός για μεταπτυχιακές σπουδές στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών (ΗΜΜΥ) του Εθνικού Μετσοβίου Πολυτεχνείου (ΕΜΠ).

Από το Σεπτέμβριο του 1999 έχει συμμετάσχει, ως ερευνητής, σε ευρωπαϊκά ερευνητικά προγράμματα, με ιδιαίτερη ενεργή δράση στο πρόγραμμα PRO^3 (IST-1999-11449). Επίσης έχει συμμετάσχει ενεργά σε διάφορα ερευνητικά προγράμματα λογισμικού όπως είναι το πρόγραμμα ACEOS. Έχει συμβάλλει στη σχεδίαση και ανάπτυξη συστημάτων ψηφιακής επεξεργασίας σημάτων για την ανάπτυξη λογισμικού σε συστήματα τηλεφωνικών δικτύων.

Έλαβε μέρος σε πολλές επιστημονικές συναντήσεις και συνεργάσθηκε με έμπειρους μηχανικούς τηλεπικοινωνιακών οργανισμών, εταιρειών ανάπτυξης τηλεπικοινωνιακών συστημάτων και ερευνητικών πανεπιστημιακών ομάδων.

Στα πλαίσια των δραστηριοτήτων του, αποκόμισε γνώση και εμπειρία σε θέματα Αρχιτεκτονικών και Τεχνολογιών Τηλεπικοινωνιακών Δικτύων, Σχεδίασης και Υλοποίησης Λογισμικού Τηλεπικοινωνιακών Εφαρμογών και Συ-

στημάτων, Ελέγχου Κυκλωμάτων Υψηλής Ολοκλήρωσης και ανάπτυξης Λειτουργικών Συστημάτων.

Ο κ. Ανδριτσόπουλος, κατά τη διάρκεια των μεταπτυχιακών του σπουδών παρουσίασε το ερευνητικό του έργο σε συναντήσεις και επιστημονικά συνέδρια. Εργασίες του κ. Ανδριτσόπουλου έχουν δημοσιευθεί σε πρακτικά συνεδρίων και τον επιστημονικό τύπο.