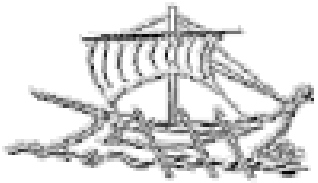


ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ Τ.Ε

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ενσωματωμένα Υπολογιστικά Συστήματα

Γεώργιος Αγγελούδης
Χρήστος Καρούνιας

Εισηγητής: Αναστασία Βελώνη, Καθηγήτρια

ΠΕΙΡΑΙΑΣ
ΙΟΥΝΙΟΣ 2016

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ενσωματωμένα Υπολογιστικά Συστήματα

Γεώργιος Αγγελούδης
Α.Μ. 38261

Χρήστος Καρούνιας
Α.Μ. 35549

Εισηγητής:

Δρ Αναστασία Βελώνη, Καθηγήτρια

Εξεταστική Επιτροπή:

Ημερομηνία εξέτασης:

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό των ενσωματωμένων υπολογιστικών συστημάτων. Την προσπάθειά μας αυτή υποστήριξε η επιβλέπων καθηγήτριά μας, Αναστασία Βελώνη, την οποία θα θέλαμε να ευχαριστήσουμε.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία ασχολείται με την ανάπτυξη, τη σχεδίαση καθώς και την χρήση των ενσωματωμένων υπολογιστικών συστημάτων. Σε αυτό το πεδίο παρουσιάζουμε κάποια παραδείγματα ενσωματωμένης υπολογιστικής, που παρουσιάζονται στην καθημερινότητά μας ως εργαλείο αυτοματοποιήσης των σημερινών πολύπλοκων συστημάτων. Επίσης, λαμβάνεται υπόψη σε κάθε είδος συστήματος, το κόστος, η μείωση κατανάλωσης ενέργειας καθώς και το μέγεθός του.

ABSTRACT

The following research paper attempts to examine embedded computing systems in terms of development, design and practical use. In this field we shall present examples of embedding computing observed in everyday life, whose role is to automate modern complex systems. Moreover, costs, energy outputs and size are taken into account for each type of system.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Ενσωματωμένα Υπολογιστικά Συστήματα
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: μικροεπεξεργαστής, σχεδίαση, μνήμη, αρχιτεκτονική, SoC

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

ABS Anti-lock Braking System

ATM Automated Teller Machine

GPS Global Positioning System

μP Microprocessor (integrated circuit)

μC Microcontroller

SOC System on a Chip

ASIC Application-specific integrated circuit

FPGA Field Programmable Gate Array

USB Universal Serial Bus

UART Universal Asynchronous Receiver/Transmitter

DMA Direct Memory Access

TLM Transaction-level Modeling

CPU Central Processing Unit

GPU Graphics Processing Unit

RAM Random Access Memory

MPSoC Multiprocessor System on a Chip

VLSI Very-large-scale Integration

VHDL VHSIC Hardware Description Language

PLD Programmable Logic Device

PROM Programmable Read-only Memory

CLB Configurable Logic Block

CPLD Complex Programmable Logic DeviceY

LSI Large Scale Integration

ASSP Application-specific Standard Product

NRE Non-recurring engineering

SIC Standard Industrial Classification

ROM Read-only Memory

EPROM Erasable Programmable Read-only Memory

KME Κεντρική Μονάδα Επεξεργασίας

PDA Personal Digital Assistant

DRM Digital Rights Management

LAN Local Area Network

WEP Wired Equivalent Privacy

IPSec Internet Protocol Security

SSL Secure Sockets Layer

OS Operating System

FIPS Federal Information Processing Standards

ARM Advanced RISC Machines

SHARC Super Harvard Architecture Single-chip Computer

RISC Reduced Instruction Set Computer

CISC Complex Instruction Set Computer

SIMD Single Instruction Multiple Data

IEEE Institute of Electrical and Electronics Engineers

UML Unified Modeling Language

OCL Object Constraint Language

ASC+T Automatic Stability Control + Traction

LCD Liquid Crystal Display

ECC Error Correcting Code

ADPCM Adaptive Differential Pulse-code Modulation

FSK Frequency-shifting Keying

ΚΕΦΑΛΑΙΟ ΠΡΩΤΟ

1.1 Εισαγωγή

Η σχεδίαση ψηφιακών συστημάτων έχει εισάγει μία νέα εποχή. Σε μία εποχή που ο σχεδιασμός μικροεπεξεργαστών έχει εξελιχθεί σε μία κλασσική άσκηση βελτιστοποίησης, η σχεδίαση ενσωματωμένων υπολογιστικών συστημάτων, στα οποία οι μικροεπεξεργαστές αποτελούν μόνο ένα τμήμα τους, αποτελεί ανοιχτή πρόκληση. Οι υλοποιήσεις των συστημάτων αυτών τυπικά αποτελούνται από μέρη λογισμικού, προγραμματιζόμενα ή ειδικού σκοπού, και υποσυστήματα επικοινωνίας και μνήμης.

Ωστόσο, η σχεδίαση ενσωματωμένων συστημάτων εξασκείται σήμερα σαν μία δεξιοτεχνία. Μολονότι η γνώση για τα συστατικά του υλικού και τα υποσυστήματα λογισμικού είναι ξεκάθαρη, δεν υπάρχουν μεθοδολογίες σχεδίασης συστήματος σε κοινή χρήση για το συντονισμό της συνολικής διεργασίας σχεδίασης. Η σχεδίαση ενσωματωμένων συστημάτων πραγματοποιείται ακόμα με μη προγραμματισμένο εξαρχής τρόπο στα περισσότερα έργα. Επίσης, κατά το σχεδιασμό ενσωματωμένων συστημάτων απαιτείται συνήθως εξισορρόπηση μεταξύ του κόστους και των επιδόσεων τους. Ακόμα, για τη βελτιστοποίησή τους θα πρέπει να λαμβάνονται υπόψη διάφορα μη συγκρίσιμα και συχνά ανταγωνιστικά μεγέθη, όπως κόστος, απόδοση, κατανάλωση ενέργειας, αξιοπιστία κλπ.

Τέλος, απαιτείται αρκετή προσπάθεια και εργαλεία αυτοματοποίησης σχεδιασμού ώστε να διαχειριστεί κανείς την πολυπλοκότητα των σημερινών ενσωματωμένων συστημάτων και να διενεργήσει τους σωστούς χειρισμούς ώστε να καλυφθούν οι απαιτήσεις της αγοράς.

1.2 Ενσωματωμένα Συστήματα

Με μια πρώτη ερμηνεία, ένα ενσωματωμένο σύστημα δεν είναι τίποτα περισσότερο από μια εξειδικευμένη υπολογιστική μηχανή.

Με τον όρο υπολογιστική μηχανή -ή απλά υπολογιστής –συνήθως αναφερόμαστε σε ένα σύστημα που περιλαμβάνει εισόδους, εξόδους και μια υπολογιστική μονάδα. Η μηχανή είναι απαραίτητο να είναι συνδεδεμένη με μια πηγή ενέργειας, ώστε να της παρέχει την απαραίτητη, για τη λειτουργία της, ενέργεια. Εκτός από την επεξεργασία της πληροφορίας, η υπολογιστική μηχανή εκλύει θερμότητα που επιστρέφει στο περιβάλλον. Οι είσοδοι, οι

οποίοι εισέρχονται από το περιβάλλον, καθορίζουν τις εξόδους, οι οποίες με τη σειρά τους μεταφέρονται από τη μηχανή πίσω στο περιβάλλον.[5]

1.2.1 Ενσωματωμένα Συστήματα και Συστήματα Γενικού Σκοπού

Μια υπολογιστική μηχανή, μπορούμε να την ταξινομήσουμε είτε ως ενσωματωμένη, είτε ως γενικού σκοπού. Ένα ενσωματωμένο υπολογιστικό σύστημα ειδικού σκοπού, είναι μια υπολογιστική μηχανή η οποία αποτελεί συνήθως τμήμα ενός ευρύτερου συστήματος. Ο σκοπός της είναι στενά επικεντρωμένος στο να υποστηρίζει το σύστημα αυτό.

Το περιβάλλον ενός ενσωματωμένου συστήματος, είναι το ευρύτερο σύστημα στο οποίο και ανήκει. Ο τελικός χρήστης του προϊόντος κατά κανόνα δεν αλληλεπιδρά με το ενσωματωμένο σύστημα ή αλληλεπιδρά με πολύ περιορισμένο τρόπο. Ακόμα και αν ο υπολογιστής είναι σε θέση να εκτελέσει περισσότερες λειτουργίες, ένα ενσωματωμένο σύστημα συνήθως περιορίζεται στο να έχει ένα συγκεκριμένο ρόλο στο τελικό σύστημα, όπως για παράδειγμα τον έλεγχο της συμπεριφοράς του ευρύτερου συστήματος.

Αντίθετα, ένα υπολογιστικό σύστημα γενικού σκοπού είναι ένα σύστημα/προϊόν από μόνο του και ως εκ τούτου, ο τελικός χρήστης αλληλεπιδρά άμεσα με αυτό. Χαρακτηριστικό των συστημάτων αυτών, είναι ότι έχουν σχετικά λίγες, τυποποιημένες εισόδους και εξόδους (π.χ πληκτρολόγια, ποντίκια, συνδέσεις δικτύου, οθόνες και εκτυπωτές).

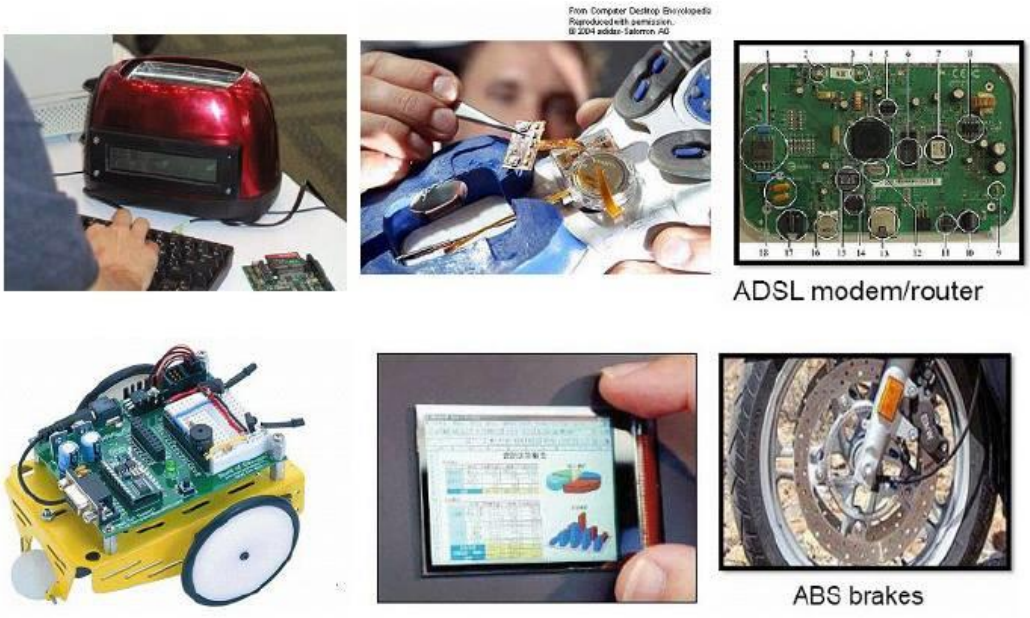
Τα ενσωματωμένα συστήματα μπορεί να έχουν κάποια από αυτά τα περιφερειακά, αλλά επίσης περιλαμβάνουν πολύ περισσότερα και εξειδικευμένα. Επιπλέον, ο τρόπος με τον οποίο κωδικοποιούνται οι είσοδοι και οι έξοδοι μπορεί να διαφέρει από ένα ενσωματωμένο σύστημα σε ένα άλλο. Από την άλλη πλευρά, οι υπολογιστές γενικού σκοπού χρησιμοποιούν καθορισμένη κωδικοποίηση.[5]

1.2.2. Χρήση των ενσωματωμένων συστημάτων

Μερικές εφαρμογές και χρήσεις των ενσωματωμένων συστημάτων είναι σε:

- Κινητήρες οχημάτων και αυτόματα συστήματα φρένων (ABS)
- Μηχανήματα αναλήψεως (ATM)
- Οικιακές συσκευές (φούρνοι μικροκυμάτων, συσκευές DVD, Μαγνητόφωνα κλπ)

- Δικτυακές συσκευές (switches και routers)
- Παιχνιδομηχανές
- Συστήματα αεροναυπηγικής
- Πυραύλους
- Περιφερειακά υπολογιστών (εκτυπωτές, πληκτρολόγια, ποντίκια)
- Αυτοματοποιήσεις σπιτιού, συναγερμοί, θερμοστάτες
- Παγκόσμιο σύστημα εύρεσης θέσεως (GPS)
- Στρατιωτικές εφαρμογές, αυτόματα συστήματα επεξεργασίας
- Αριθμομηχανές
- Ιατρικούς εξοπλισμούς



Εικόνα 1.1: Ενσωματωμένα συστήματα γύρω μας

1.3 Οι επεξεργαστές των ενσωματωμένων συστημάτων

Οι ενσωματωμένοι επεξεργαστές χωρίζονται σε δυο κύριες κατηγορίες. Η πρώτη κατηγορία αποτελείται από τους συνήθεις μικροεπεξεργαστές (μP), οι οποίοι χρησιμοποιούν ξεχωριστά ολοκληρωμένα κυκλώματα για την μνήμη και ξεχωριστά για τα περιφερειακά. Η δεύτερη κατηγορία αποτελείται από τους μικροελεγκτές (μC), οι οποίοι έχουν πολύ περισσότερα περιφερειακά πάνω στο chip μειώνοντας με αυτόν τον τρόπο την κατανάλωση ενέργειας, το μέγεθος και το κόστος του ενσωματωμένου συστήματος. Σε αντίθεση με τους προσωπικούς υπολογιστές η αρχιτεκτονική των επεξεργαστών, η οποία χρησιμοποιείται στα ενσωματωμένα συστήματα, είναι ανάλογη με το λογισμικό που χρησιμοποιείται για εφαρμογές ειδικού σκοπού. Μπορεί να χρησιμοποιηθεί η αρχιτεκτονική Von Neumann ή η αρχιτεκτονική Harvard. Επιπλέον, ο σχεδιασμός του επεξεργαστή μπορεί να είναι τύπου RISC ή non-RISC. Το μέγεθος των λέξεων ποικίλλει από 4 bit έως 64 bit και άνω. Συνήθως στα ενσωματωμένα συστήματα χρησιμοποιείται μέγεθος λέξης 8 bit ή 16 bit. Τέλος πρέπει να αναφερθεί πως για τα ενσωματωμένα συστήματα έχουν σχεδιαστεί σε μεγάλο αριθμό μικροελεγκτές. Σε πολλές περιπτώσεις οι μικροελεγκτές είναι προτιμότεροι από τους μικροεπεξεργαστές, διότι οι μικροεπεξεργαστές χρειάζονται μεγαλύτερη υποστήριξη από επιπλέον κυκλώματα σε σχέση με τους μικροελεγκτές.

1.4 Πλήρες ενσωματωμένο σύστημα

Τα πλήρη ενσωματωμένα συστήματα είναι συστήματα τα οποία έχουν διαμορφωθεί για λειτουργίες μεγάλου όγκου. Αυτά τα συστήματα ονομάζονται “συστήματα σε ένα τσιπ” (SOC). Τέτοια συστήματα μπορούν να υλοποιηθούν σε ASIC κυκλώματα ή με την χρήση FPGAs.

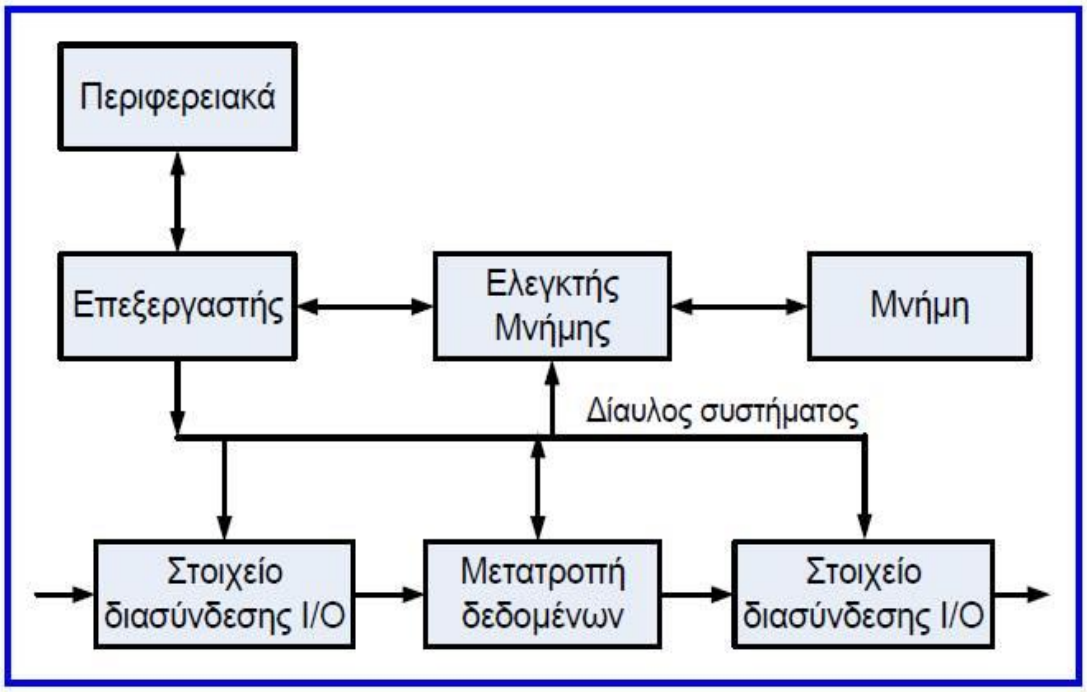
1.4.1 System on a chip (SoC)

Μία συνηθισμένη δομή ενός ενσωματωμένου συστήματος είναι το λεγόμενο SoC (system on a chip) όπου όλα τα στοιχεία του συστήματος ολοκληρώνονται σε ένα και μόνο chip. Ένα SoC εμπεριέχει τόσο το απαιτούμενο για την εφαρμογή hardware (έναν τουλάχιστον μικροελεγκτή, διάφορα blocks μνήμης, κάποιες περιφερειακές μονάδες, στοιχεία για χρονισμό του κυκλώματος, buses για τη διασύνδεση των παραπάνω στοιχείων καθώς και διάφορα εξωτερικά interfaces –σε κάποιες περιπτώσεις-

όπως USB, UART) όσο και το software που ελέγχει τον μικροεπεξεργαστή, τα περιφερειακά και τα διάφορα interfaces. Γενικά, ο σχεδιασμός ενός SoC έχει ως στόχο την παράλληλη ανάπτυξη του υλικού και του λογισμικού.

Ένα βασικό στοιχείο της διαδικασίας σχεδίασης ενός SoC είναι η προσομοίωσή του μέσω της οποίας ελέγχεται η σωστή και απρόσκοπτη λειτουργία του συστήματος. Για να συμβεί αυτό, το hardware «χαρτογραφείται» πάνω σε μία πλατφόρμα προσομοίωσης η οποία βασίζεται στη φιλοσοφία των FPGA (Field Programmable Gate Array) ενώ το software φορτώνεται στα blocks μνήμης που υπάρχουν στην εν λόγω πλατφόρμα. Με τον κατάλληλο προγραμματισμό μπορεί να γίνει έλεγχος σωστής λειτουργίας και debugging τόσο για το hardware όσο και για το software και μάλιστα πολύ κοντά στην πραγματική ταχύτητα λειτουργίας του συστήματος. Στη συνέχεια και αφού ολοκληρωθεί η φάση αυτή του ελέγχου, ακολουθεί η φάση της τοποθέτησης των διαφόρων δομικών μονάδων του υλικού στη σχεδιαστική επιφάνεια και η φάση της διασύνδεσης των μονάδων αυτών.

Μετά το πέρας και αυτού του σταδίου ο σχεδιαστής προχωράει στην κατασκευή του συστήματος –την ουσιαστική δηλαδή υλοποίησή του- με χρήση διαφόρων τεχνολογιών. Μία από αυτές είναι τα FPGAs ενώ μία άλλη είναι η μέθοδος σχεδίασης ASICs (Application Specific Integrated Circuits) με χρήση στοιχείων κυρίως ψηφιακής λογικής.

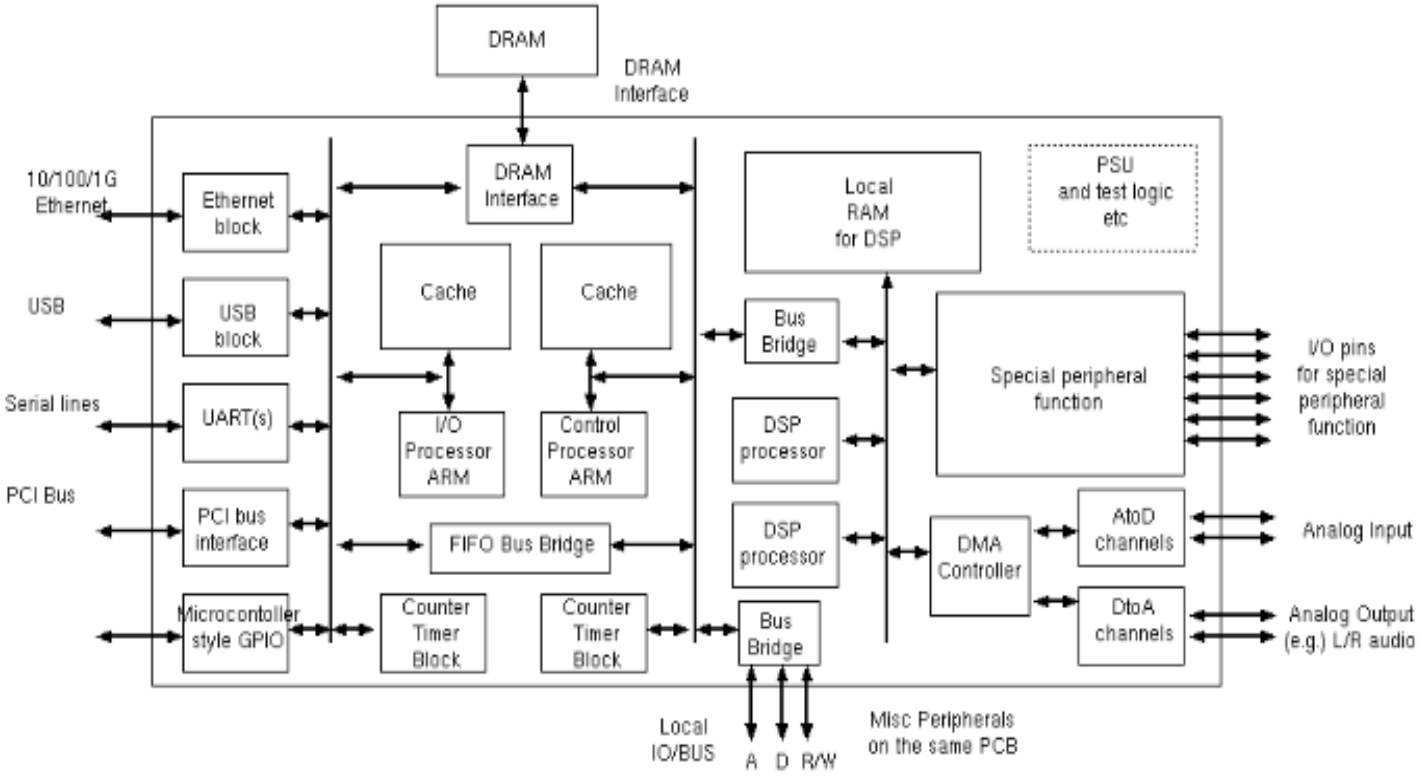


Σχήμα 1.1: Τυπικό σύστημα σε ολοκληρωμένο κύκλωμα

Ένα σύστημα σε ένα τσιπ χρησιμοποιεί συνήθως 70-140 mm^2 του πυριτίου. Ένα SoC είναι ένα πλήρες σύστημα σε ένα τσιπ. Ένα τέτοιο σύστημα περιλαμβάνει έναν μικροεπεξεργαστή, μνήμη και περιφερειακά.

Ο επεξεργαστής μπορεί να είναι ένας πρόχειρος ή πρότυπο μικροεπεξεργαστών, ή θα μπορούσε να είναι ένας εξειδικευμένος επεξεργαστής για τον ήχο, το μόντεμ ή βίντεο εφαρμογές. Μπορεί να υπάρχουν και άλλοι επεξεργαστές, καθώς και άλλες κατηγορίες στον κύκλο των διαύλων, όπως οι ελεγκτές DMA.

Οι ελεγκτές DMA μπορεί να είναι σύνθετοι, και διακρίνονται μόνο από επεξεργαστές με πλήρη ή μερική έλλειψη οδηγιών. Οι επεξεργαστές διασυνδέονται χρησιμοποιώντας μια ποικιλία μηχανισμών, συμπεριλαμβανομένης της κοινής μνήμης και οντότητες υλικού για ανταλλαγή μηνυμάτων, όπως εξειδικευμένα κανάλια. Τα συστήματα σε τσιπ βρίσκονται σε κάθε καταναλωτικό προϊόν, όπως modems, κινητά τηλέφωνα, DVD players, τηλεοράσεις και iPod.



Σχήμα 1.2: Μία πιο αναλυτική παρουσίαση ενός System On Chip (SoC).[15]

Σχεδιασμός Ροής

Ο σχεδιασμός ροής χωρίζεται από το επίπεδο των διαρθρωτικών RTL (Register Transfer Language) σε:

- **Front End:** καθορισμός, εξερεύνηση, σχεδιασμός, σύλληψη, σύνθεση -> διαρθρωτικών RTL

- **Back End:** Διαρθρωτικά RTL -> τόπος, δρομολόγηση, κατασκευή μάσκας, παραγωγή

Front End

Η αρχιτεκτονική εξερεύνησης θα δοκιμάσει διαφορετικούς συνδυασμούς επεξεργαστών, μνημών και δομών των διαύλων ώστε να βρεθεί μια εφαρμογή με καλή δύναμη και εξισορρόπηση φορτίου. Ένα μοντέλο υψηλού επιπέδου και «χαλαρού» χρόνου αρκεί για να υπολογίσει την απόδοση μιας αρχιτεκτονικής.

Τα μοντέλα υψηλού επιπέδου που δεν προορίζονται να είναι συνθέσιμα και δοκιμαστικά εξαρτήματα θα πρέπει να κωδικοποιηθούν, χρησιμοποιώντας συνήθως SystemC.

Back End

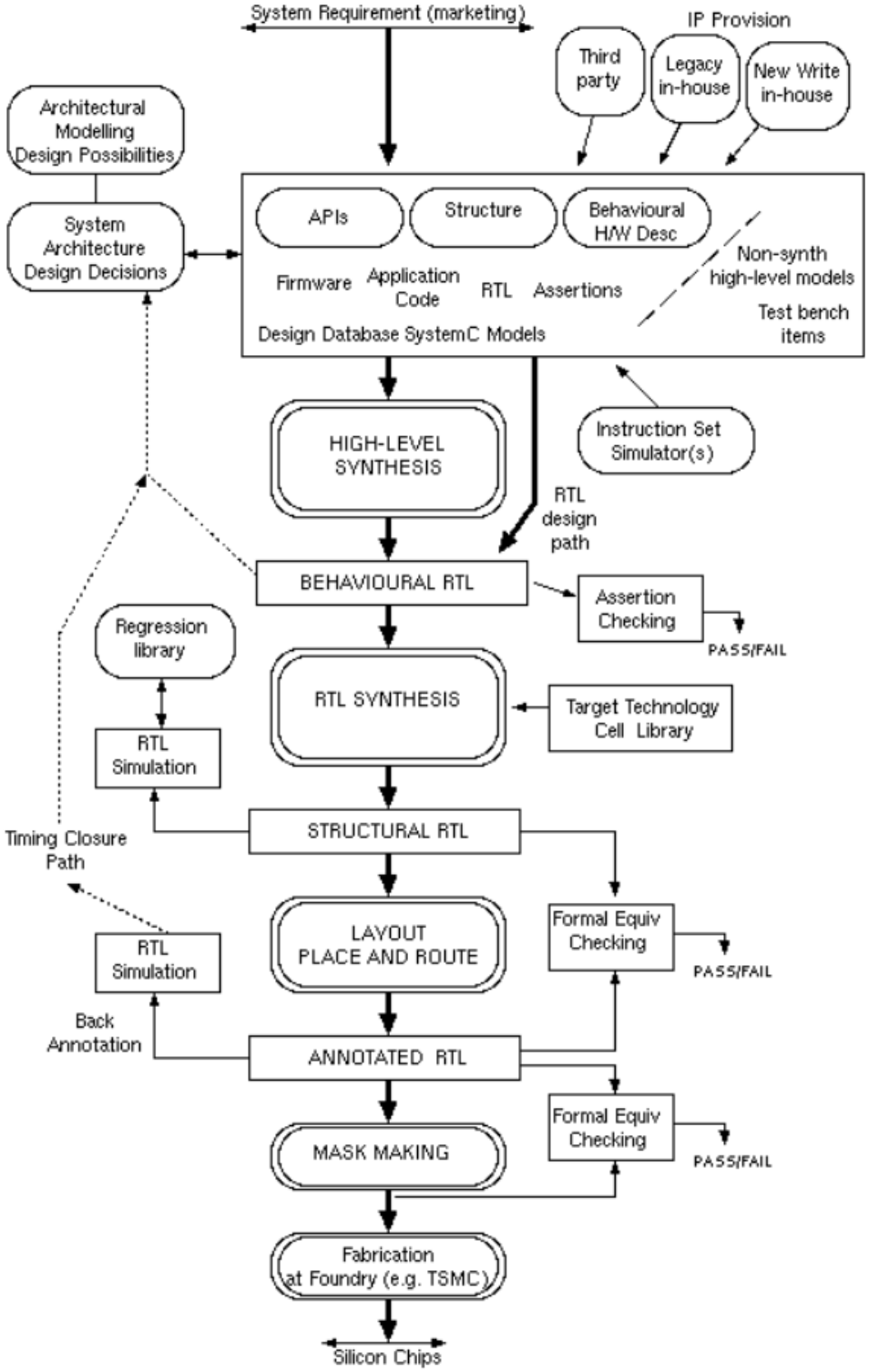
Μετά τη σύνθεση του RTL χρησιμοποιώντας μια βιβλιοθήκη στοχευόμενης τεχνολογίας, υπάρχει μια διαρθρωτική λίστα που δεν έχει καθυστερήσεις.

Ο τόπος και η διαδρομή δίνουν δισδιάστατες (2-D) συντεταγμένες για κάθε συσκευή, προσθέτουν εξωτερικές λαβές I/O (εισόδου-εξόδου) και βάζουν καλωδίωση μεταξύ των συστατικών. Το RTL σημειώνεται με τις πραγματικές καθυστερήσεις στην υλοποίηση των πυλών, δίνει μια ακριβή ισχύ και μοντέλο επιδόσεων. Αν η απόδοση δεν είναι μέχρι την ισοτιμία, οι αλλαγές στο σχεδιασμό είναι απαραίτητες.

Η κατασκευή των μασκών είναι συνήθως το πιο ακριβό βήμα (π.χ. 1 εκατομμύριο), έτσι πρέπει να είναι σωστό την πρώτη φορά. Η κατασκευή γίνεται εσωτερικά από ορισμένες μεγάλες εταιρείες (π.χ. Intel, Samsung), αλλά οι

περισσότερες εταιρείες χρησιμοποιούν μικρές επιχειρήσεις (π.χ. UMC, TSMC).

Σε όλα τα στάδια (front and back end), μια βιβλιοθήκη τυποποιημένων δοκιμών θα πρέπει να εκτελείται κάθε νύχτα και τυχόν αλλαγές που προκαλούν αποτυχία σε ένα προηγουμένως επιτυχημένο τεστ (παλινδρομήσεις) θα πρέπει να αναφέρονται αυτόματα στο πρόγραμμα διαχειριστή.



Σχήμα 1.3: Ο σχεδιασμός και η λειτουργία του SoC.[15]

Επίπεδα Μοντελοποίησης

Το σύστημα μοντελοποίησης πρέπει να στηρίζει όλες τις φάσεις της διαδικασίας σχεδιασμού, από την έναρξη του σχεδιασμού μέχρι την κατασκευή. Θα πρέπει να γίνει μίξη των συστατικών που χρησιμοποιούν διαφορετικά επίπεδα αφαίρεσης σε μία προσομοίωση.

Τα επίπεδα που χρησιμοποιούνται συχνότερα είναι τα εξής:

1. Functional Modelling:

Η «έξοδος» από την εκτέλεση της προσομοίωσης είναι ακριβής.

2. Memory Accurate Modelling:

Το περιεχόμενο και η διάταξη της μνήμης είναι ακριβής.

3. Untimed TLM (Transactional Level Modelling) :

Οι χρονοσφραγίδες δεν καταγράφονται στις συναλλαγές.

4. Loosey-Timed TLM:

Ο αριθμός των συναλλαγών είναι ακριβής, αλλά ο σκοπός να είναι λάθος.

5. Approximately-Timed TLM:

Ο αριθμός και η σειρά των συναλλαγών είναι ακριβής.

6. Cirle-Accurate Level Modelling:

Ο αριθμός των κύκλων ρολογιού που καταναλώνεται είναι ακριβής.

7. Event-Level Modelling:

Η σειρά των καθαρών αλλαγών μέσα σε ένα κύκλο ρολογιού είναι ακριβής

Άλλοι όροι που χρησιμοποιούνται είναι οι εξής:

- **Programmer View Accurate**: Τα περιεχόμενα της μνήμης και των καταχωρητών είναι σύμφωνα με το πραγματικό υλικό, αλλά ο συγχρονισμός μπορεί να είναι ανακριβής και κάποιες άλλες εγγραφές ή συνδυαστικά δίκτυα τα οποία δεν έχουν σχεδιαστεί ως μέρος της

«εικόνας του προγραμματιστή» μπορεί να μη μοντελοποιηθούν με ακρίβεια.

- **Behavioural Modelling**: Χρησιμοποιώντας ένα πακέτο νημάτων, ή άλλης βιβλιοθήκης (π.χ. SystemC), τα «χειροποίητα» (hand-crafted) προγράμματα γράφτηκαν για να περιγράψουν τη συμπεριφορά του κάθε στοιχείου ή του υποσυστήματος. Τα σημαντικότερα στοιχεία του υλικού, όπως δίαυλοι, κρύπτες ή ελεγκτές DRAM μπορεί να αγνοηθούν σε ένα τέτοιο μοντέλο.

Η οπτική του προγραμματιστή (programmers view) είναι συχνά συντομογραφία ως «PV» και αν προστεθεί το χρονοδιάγραμμα αυτό ονομάζεται «PV + T».

Το PV περιέχει μόνο τα σημαντικά αρχιτεκτονικά μητρώα όπως εκείνα που το προγραμματιστικό λογισμικό μπορεί να χειριστεί βάσει οδηγιών. Άλλοι καταχωρητές σε μια συγκεκριμένη εφαρμογή υλικού, όπως τα στάδια του αγωγού και οι καταχωρητές κρατήσεων για την αντιμετώπιση των διαρθρωτικών κινδύνων, δεν αποτελούν μέρος του PV. [15]

1.4.1.1 Διαφορές SoC και CPU

Το νούμερο ένα πλεονέκτημα ενός SoC είναι το μέγεθός του: Ένα SoC είναι μόνο λίγο μεγαλύτερο από μία CPU, και όμως περιέχει πολύ περισσότερες λειτουργίες. Η CPU, είναι πολύ δύσκολο να καταστήσει έναν υπολογιστή μικρότερο από 10 εκατοστά (4 ίντσες) στο τετράγωνο, αποκλειστικά και μόνο λόγω του αριθμού των μεμονωμένων chips που πρέπει να συμπιεστούν σε αυτή. Χρησιμοποιώντας SoCs, μπορέι να μπει ένας πλήρης υπολογιστής σε smartphones και tablets, και να εξακολουθούν να έχουν άφθονο χώρο για τις μπαταρίες.

Λόγω του πολύ υψηλού επιπέδου της ολοκλήρωσης και την πολύ μικρότερη καλωδίωση, ένα SoC χρησιμοποιεί επίσης σημαντικά λιγότερη ενέργεια όπου αυτό είναι πού σημαντικό όταν πρόκειται για mobile computing. Η μείωση του αριθμού των φυσικών chips σημαίνει ότι είναι πολύ φθηνότερο να χτίσει ένας υπολογιστής χρησιμοποιώντας ένα SoC.

Εικόνα 1.2: Συμβατική μητρική πλακέτα του υπολογιστή (αριστερά) έναντι του κυκλώματος iPad 3 (δεξιά). Αυτή η εικόνα είναι περίπου σε κλίμακα.

Το μόνο πραγματικό μειονέκτημα ενός SoC είναι η πλήρης έλλειψη ευελιξίας. Στον ηλεκτρονικό υπολογιστή μπορεί να μπει αργότερα μια άλλη CPU, GPU, RAM ανά πάσα στιγμή, όμως δεν μπορεί να γίνει το ίδιο και στο smartphone . Στο μέλλον ίσως να είναι σε θέση να μπορεί να γίνει αυτό, αλλά επειδή τα πάντα θα είναι ενσωματωμένα, αυτό θα είναι αρκετά δαπανηρό.

1.4.1.2 MultiProcessor SoC (MPSoC) & MultiCore Processors

Το πολλαπλών επεξεργαστών system-on-chip (MPSoC) χρησιμοποιεί πολλαπλές CPU μαζί με άλλα υποσυστήματα υλικού για να εφαρμόσει ένα σύστημα. Ένα ευρύ φάσμα των αρχιτεκτονικών MPSoC έχουν αναπτυχθεί την τελευταία δεκαετία.

Ο Hammond et al. πρότεινε μία αρχιτεκτονική CMP (chip multiprocessor) πριν την εμφάνιση της αρχιτεκτονικής Lucent Daytona. Η προτεινόμενη μηχανή τους περιλάμβανε οκτώ πυρήνες CPU, ο καθένας με τη δική του μνήμη πρώτου επιπέδου και μια κοινή μνήμη cache δευτέρου επιπέδου. Χρησιμοποίησαν αρχιτεκτονικές μεθόδους προσομοίωσης για να συγκρίνουν την αρχιτεκτονική CMP με τις multithreading και superscalar ταυτόχρονα.

Βρήκαν ότι η CMP παρείχε υψηλότερες επιδόσεις στα περισσότερα σημεία αναφοράς τους και υποστήριξαν ότι η σχετική απλότητα του υλικού της CMP

γίνεται πιο ελκυστική για την εφαρμογή VLSI.

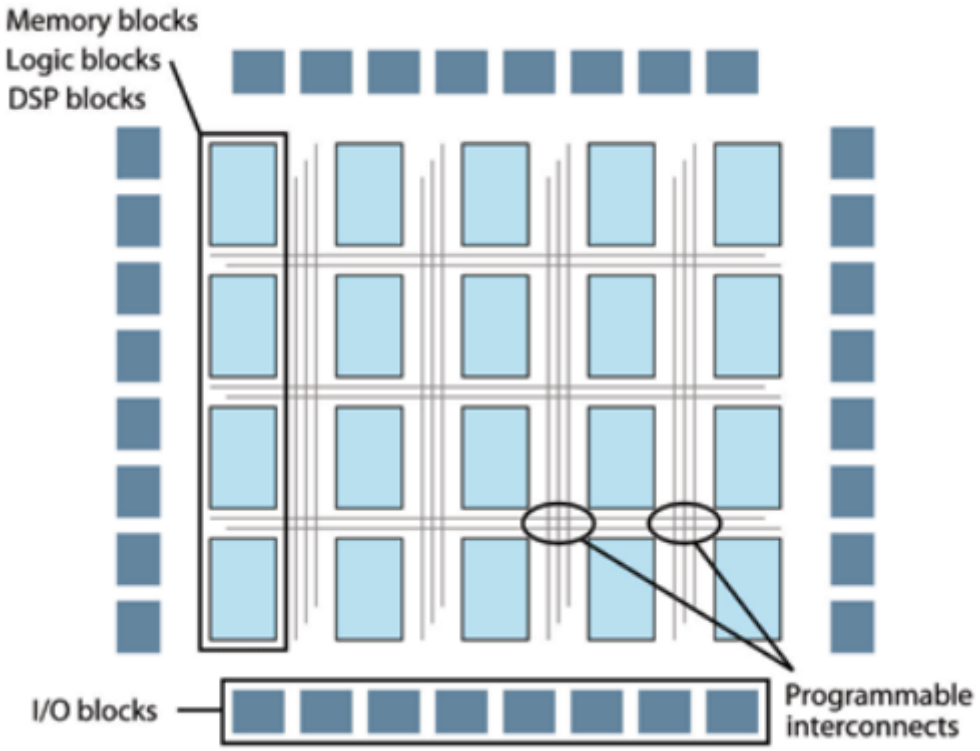
Οι εμπορικοί πολυπύρηνοι επεξεργαστές έχουν πολύ μικρότερη ιστορία από αυτή των εμπορικων MPSoCs. Οι πρώτοι πολυπύρηνοι επεξεργαστές γενικού σκοπού εισήχθησαν το 2005. Ο επεξεργαστής Intel Core Duo συνδυάζει δύο ενισχυμένους πυρήνες επεξεργαστή Pentium M σε ένα ενιαίο καλούπι. Οι επεξεργαστές είναι σχετικά χωριστά, αλλά μοιράζονται μια μνήμη L2 cache, καθώς και λογική διαχείριση ενέργειας.

Ο multicore επεξεργαστής AMD Opteron διπλού πυρήνα προσφέρει ξεχωριστή L2 κρυφή μνήμη για τους επεξεργαστές της, αλλά ένα αίτημα κοινής διεπαφής του συστήματος συνδέει τους επεξεργαστές με τον ελεγκτή μνήμης και HyperTransport.

Ο επεξεργαστής Niagra SPARC έχει οκτώ συμμετρικούς multi-thread πυρήνες τεσσάρων κατευθύνσεων.

1.4.2 Τι είναι μια FPGA

Μια FPGA, ή προγραμματιζόμενη στο πεδίο συστοιχία πυλών, είναι ένα ολοκληρωμένο κύκλωμα το οποίο είναι παραμετροποιήσημο από τον χρήστη. Η παραμετροποίηση μιας FPGA γίνεται με την χρήση μιας γλώσσας υλικού, η οποία μπορεί να είναι ή VHDL ή Verilog. Τα σύγχρονα FPGAs αποτελούνται από πολλές λογικές πύλες και block μνήμης (RAM) με σκοπό να υλοποιούν πολύπλοκους ψηφιακούς υπολογισμούς. Μια FPGA αποτελείται από λογικές πράξεις (όπως AND, OR κτλ.), από blocks εισόδου - εξόδου για την συνδεσιμότητα με τα PINS και από επαναπρογραμματιζόμενες διασυνδέσεις, ώστε όλα τα στοιχεία μέσα σε μια FPGA να είναι διασυνδεμένα. Όπως βλέπουμε και στην Εικόνα 1.3 τα λογικά blocks είναι τοποθετημένα σε έναν πίνακα δύο διαστάσεων και οι διασυνδέσεις είναι οργανωμένες ως οριζόντια και κάθετα κανάλια δρομολόγησης ανάμεσα από τα λογικά blocks. Τα κανάλια δρομολόγησης εμπεριέχουν καλώδια και επαναπρογραμματιζόμενους διακόπτες, επιτρέποντας στα λογικά blocks να διασυνδέονται με πολλούς διαφορετικούς τρόπους μεταξύ τους. Η FPGA έχει παρόμοιο πεδίο εφαρμογών με άλλα προγραμματιζόμενα ολοκληρωμένα ψηφιακά κυκλώματα, όπως τα PLD (Programmable Logic Device) και τα ASIC. [2]



Εικόνα 1.3: Παράδειγμα αρχιτεκτονικής μιας FPGA(c)

Χαρακτηριστικά της FPGA είναι τα εξής :

- Χάνει τον προγραμματισμό της κάθε φορά που διακόπτεται η τάση τροφοδοσίας της. Επομένως απαιτεί εξωτερικό μικροεπεξεργαστή ή μνήμη με μόνιμη συγκράτηση δεδομένων (non - volatile memory) από τα οποία θα προγραμματίζεται κάθε φορά που επανέρχεται η τάση τροφοδοσίας.
- Ο προγραμματισμός της μπορεί να αλλάζει κάθε φορά που τροποποιείται το λογισμικό του μικροεπεξεργαστή ή τα δεδομένα της μνήμης που το ελέγχει.
- Δεν υπάρχει όριο στο πόσες φορές μπορεί να επαναπρογραμματιστεί.

- Η κατανάλωση ισχύος είναι σημαντικά αυξημένη σε σχέση με τα ASIC.

1.4.2.1 Η ιστορία της FPGA

Η βιομηχανία των FPGA προήλθε από PROM μνήμες (programmable read only memory) και PLDs (programmable logic devices). Πρόσφεραν τη δυνατότητα να προγραμματίζονται μαζικά σε ένα εργοστάσιο. Ωστόσο η λογική του προγραμματισμού, ήταν πολύ δύσκολη μεταξύ λογικών κυκλωμάτων.

Οι ιδρυτές της Xilinx, Ross Freeman και Bernard Vonderschmitt, εφηύραν την πρώτη εμπορικά εφαρμόσιμη FPGA το 1985 - την XC2064. Η XC2064 διέθετε προγραμματιζόμενες πύλες καθώς και προγραμματιζόμενες διασυνδέσεις μεταξύ των πυλών και υπήρξε η απαρχή για μια καινούρια τεχνολογία και μια καινούρια αγορά.

Περιείχε 64 διαμορφούμενα λογικά block (CLBs), με δύο πίνακες αναζήτησης τριών εισόδων.

Στα τέλη της δεκαετίας του 1980 το υπουργείο των ναυτικών ενόπλων δυνάμεων της Αμερικής χρηματοδότησε ένα πείραμα του οποίου εισηγητής ήταν ο Steve Casselman. Σκοπός του πειράματος ήταν να αναπτυχθεί ένα σύστημα με 600.000 πύλες. Η Xilinx συνέχισε την εξέλιξη χωρίς ανταγωνισμό από το 1985 μέχρι και τα μέσα της δεκαετίας του ’90, όταν άρχισαν να ξεπετάγονται ανταγωνιστικές εταιρίες οι οποίες σκοπό είχαν να καταλάβουν ένα σημαντικό κομμάτι της αγοράς. Το 1993 η Actel εξυπηρετούσε το 18% της αγοράς.

Η δεκαετία του ’90 υπήρξε μια τρομερή περίοδος ανάπτυξης για τις FPGA και ως προς την εξέλιξη αλλά και ως προς το πλήθος των ανθρώπων που άρχισαν να ασχολούνται με αυτόν τον τομέα. Ενώ στις αρχές τις δεκαετίας οι FPGA χρησιμοποιούνταν κυρίως στις τηλεπικοινωνίες και τα δίκτυα, προς το τέλος τρύπωσαν στις καταναλωτικές, αυτοκινητιστικές και βιομηχανικές εφαρμογές.

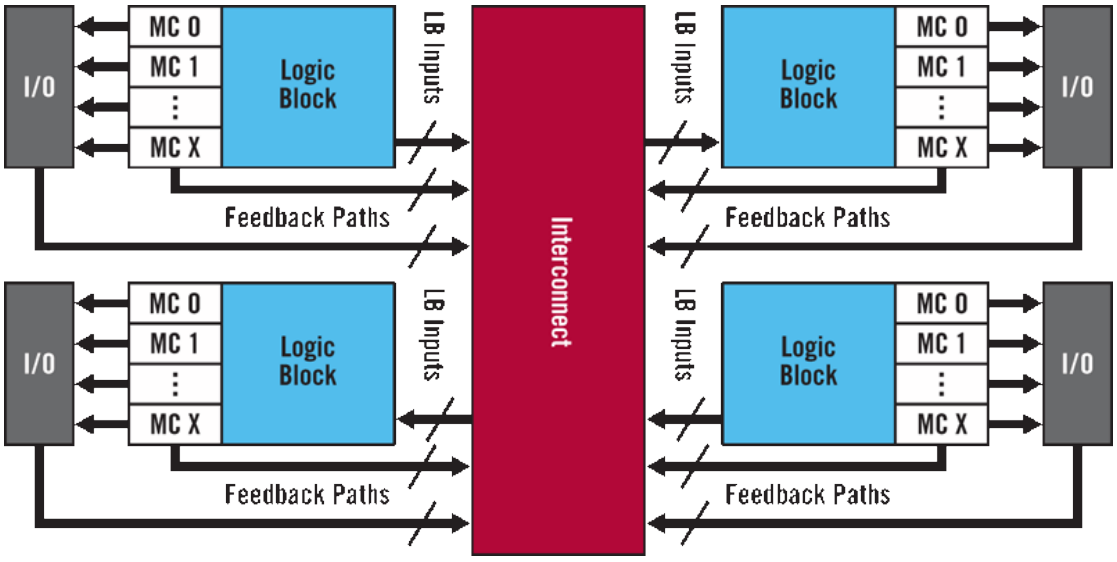
Οι FPGA είδαν για λίγο το φως της δημοσιότητας όταν το 1997, ο Adrian Thomson συνένωσε τεχνολογίες γενετικών αλγορίθμων και FPGAs για να κατασκευάσει μια συσκευή αναγνώρισης προτύπων ήχου. Ο αλγόριθμος του Thomson επέτρεπε σε μια FPGA της Xilinx που διέθετε έναν πίνακα 64 x 64

κελιών να υπολογίζει τις συνθέσεις που χρειάζονταν για να διεκπεραιώνει μια διεργασία αναγνώρισης ήχων.[2]

1.4.2.2 Τι είναι μία CPLD

Η CPLD (Complex Programmable Logic Device) είναι ένας συνδυασμός μίας πλήρως προγραμματιζόμενης σειράς AND/OR και μιας ομάδας μακροκυψελών. Η σειρά αυτή είναι επαναπρογραμματιζόμενη και μπορεί να εκτελέσει ένα πλήθος λογικών λειτουργιών. Οι μακροκυψέλες είναι λειτουργικά τμήματα που εκτελούν συνδυαστική ή συνεχή λογική.

Οι CPLDs χρησιμοποιούσαν αναλογικούς ενισχυτές αίσθησης ώστε να αυξηθεί η απόδοσή τους. Αυτή η αύξηση της απόδοσης είχε υψηλές απαιτήσεις και έτσι οι ενισχυτές αφαιρέθηκαν. Η αφαίρεση των ενισχυτών κάνει η αρχιτεκτονική επεκτάσιμη, επιτρέποντας την μείωση του κόστους και τη δυνατότητα βελτίωσης για την κάθε γενιά διαδικασιών.



Εικόνα 1.4: Η δομή του CPLD (b)

1.4.2.3 Διαφορά μεταξύ των FPGAs & CPLDs

Οι κυριότερες διαφορές ανάμεσα στα CPLDs και τις FPGAs είναι σε θέματα αρχιτεκτονικής.

Ένα CPLD έχει περιορισμένη δομή διότι αποτελείται από έναν ή περισσότερους λογικούς πίνακες τροφοδοτώντας έτσι ένα σχετικά μικρό αριθμό από χρονισμένους καταχωρητές. Αυτό έχει ως αποτέλεσμα τη μείωση της ευελιξίας στο σύστημα, μπορούμε όμως πιο εύκολα να προβλέψουμε τις χρονικές καθυστερήσεις.

Η αρχιτεκτονική των FPGA, από την άλλη, βασίζεται στις διασυνδέσεις. Αυτό τις κάνει πολύ πιο ευέλικτες (μεγαλύτερη γκάμα λογικών συνδυασμών που μπορούν να προκύψουν) καθώς και πιο περίπλοκες στο σχεδιασμό τους.

Άλλη μία διαφορά ανάμεσα στα CPLDs και τις FPGAs είναι η παρουσία κάποιων ενσωματωμένων μνημών και ενσωματωμένων λειτουργιών υψηλού επιπέδου στις περισσότερες FPGAs, καθώς και το να διαθέτουν λογικά block για τη σύνθεση κάποιων αποκωδικοποιητών ή μαθηματικών συναρτήσεων.[1]

1.4.3. Τι είναι τα ASIC

Η ανάπτυξη της τεχνολογίας των ολοκληρωμένων κυκλωμάτων πέραν του επιπέδου LSI οδήγησε στα κυκλώματα πολύ μεγάλης κλίμακας ολοκλήρωσης (VLSI). Έτσι, ο όρος VLSI κατέληξε να αναφέρεται περισσότερο στον τρόπο με τον οποίο σχεδιάζονταν τα ολοκληρωμένα κυκλώμα τα παρά στον αριθμό των τρανζίστορ που περιείχαν. Σημαίνει αναλυτική σχεδίαση σε επίπεδο κυκλώματος (circuit design) των ολοκληρωμένων κυκλωμάτων σε αντίθεση με τη σχεδίαση σε επίπεδο συστήματος (system – level design). Η εκτεταμένη διαθεσιμότητα των εργαλείων CAD για τη σχεδίαση VLSI και η ανάπτυξη της βιομηχανίας σε υπηρεσίες κατασκευής ολοκληρωμένων κυκλωμάτων έχει κάνει τώρα εφαρμόσιμη την κατασκευή ολοκληρωμένων κυκλωμάτων για ευρύ φάσμα εφαρμογών.

Χρησιμοποιούμε τον όρο ολοκληρωμένο κύκλωμα ειδικού σκοπού, ή ASIC, για να αναφερθούμε σε ένα ολοκληρωμένο κύκλωμα που κατασκευάζεται για μια συγκεκριμένη εφαρμογή. Αυτό δεν είναι για να πούμε ότι ένα ASIC κατασκευάζεται απαραίτητα μόνο για έναν πελάτη ή για ένα έργο. Μάλλον, σχεδιάζεται για να ικανοποιήσει ένα συγκεκριμένο σύνολο απαιτήσεων, και έτσι

περιέχει κυκλώματα που έχουν προσαρμοστεί σε αυτές τις απαιτήσεις. Μπορεί να σχεδιαστεί για ένα συγκεκριμένο τελικό προϊόν που παρέχεται από έναν κατασκευαστή ή για χρήση σε ένα εύρος προϊόντων που παρέχονται από τους κατασκευαστές για ένα συγκεκριμένο τμήμα της αγοράς. Αυτού του είδους τα ASIC μερικές φορές ονομάζονται πρότυπα προϊόντα εξειδικευμένα για εφαρμογές, ή ASSP, αφού αντιμετωπίζονται ως πρότυπα κομμάτια μέσα σε αυτό το τμήμα της αγοράς, αλλά δε χρησιμοποιούνται έξω από αυτό. Τέτοια παραδείγματα περιλαμβάνουν ολοκληρωμένα κυκλώματα για κινητά τηλέφωνα, τα οποία χρησιμοποιούνται από έναν αριθμό ανταγωνιστών στην κατασκευή κινητών τηλεφώνων, αλλά δεν χρησιμοποιούνται για παράδειγμα σε κυκλώματα ελέγχου οχημάτων.

Ένας από τους βασικούς λόγους για τους οποίους θα αναπτύσσαμε ένα ASIC για ένα προϊόν είναι ότι, επειδή είναι προσαρμοσμένο στις απαιτήσεις αυτής της εφαρμογής, έχει χαμηλότερο κόστος ανά ολοκληρωμένο κύκλωμα από ότι ένα προγραμματιζόμενο στοιχείο, όπως μια FPGA. Ωστόσο, για να επιτύχουμε αυτό το επίπεδο προσαρμογής στις απαιτήσεις της εφαρμογής, χρειάζεται να επενδύσουμε πολύ περισσότερη προσπάθεια για τη σχεδίαση και την επαλήθευση. Πρέπει να αποσβέσουμε το μη επαναλαμβανόμενο κόστος τεχνικής μελέτης (non - recurring engineering – NRE) σε όλα τα κομμάτια του προϊόντος που θα πουληθούν. Επομένως, έχει νόημα να χρησιμοποιήσουμε ένα ASIC μόνο εάν ο αριθμός πωλήσεων του προϊόντος μας είναι αρκετά μεγάλος.

Υπάρχουν δύο βασικές τεχνικές σχεδίασης και κατασκευής για ASIC, που διαφοροποιούνται ως προς το βαθμό προσαρμογής στις απαιτήσεις της εφαρμογής. Πρώτον, τα πλήρως προσαρμοσμένα (fully custom) ολοκληρωμένα κυκλώματα εμπεριέχουν αναλυτική σχεδίαση όλων των τρανζίστορ και των συνδέσεων σε ένα ASIC. Αυτό επιτρέπει πιο αποδοτική χρήση των πόρων του υλικού σε ένα ολοκληρωμένο κύκλωμα και έχει ως αποτέλεσμα υψηλότερη απόδοση, αλλά έχει υψηλό κόστος NRE και απαιτεί υψηλή τεχνική κατάρτιση στη σχεδίαση VLSI εντός της σχεδιαστικής ομάδας. Αυτό έχει ως επακόλουθο, τα πλήρως προσαρμοσμένα SIC να σχεδιάζονται συνήθως μόνο για προϊόντα που πωλούνται σε μεγάλες ποσότητες, όπως CPU και ολοκληρωμένα κυκλώματα για καταναλωτικές συσκευές.

Δεύτερον, τα ASIC με πρότυπα κελιά (standard cell) εμπεριέχουν επιλογή βασικών κελιών, όπως πύλες και flip - flop, από μια βιβλιοθήκη για να σχηματιστεί το κύκλωμα. Τα κελιά έχουν προηγουμένως σχεδιαστεί από έναν κατασκευαστή ολοκληρωμένων κυκλωμάτων ή έναν προμηθευτή ASIC, και χρησιμοποιούνται από το εργαλείο σύνθεσης κατά τη διάρκεια της διαδικασίας σχεδίασης για να υλοποιήσουν τη σχεδίαση. Η αξία αυτής της προσέγγισης είναι ότι το κόστος NRE για κάθε σχεδίαση ASIC μειώνεται σημαντικά, αφού γίνεται απόσβεση του κόστους σχεδίασης της βιβλιοθήκης των κελιών σε έναν αριθμό σχεδιάσεων ASIC. Ο συμβιβασμός είναι ότι το ASIC μπορεί να μην είναι τόσο πυκνό ή να μην έχει την απόδοση ενός πλήρως προσαρμοσμένου ASIC.

Με την πάροδο του χρόνου το μέγεθος των ASIC έχει μειωθεί σημαντικά, τα εργαλεία σχεδιασμού του ASIC έχουν βελτιστοποιηθεί και η πολυπλοκότητα των ASIC κυκλωμάτων έχει αυξηθεί από τις 5.000 πύλες στις 100 εκατομμύρια πύλες. Τα ASIC τελευταίας γενιάς εμπεριέχουν μικροεπεξεργαστές, block μνήμης όπως RAM, ROM, EPROM και Flash. Οι σχεδιαστές των ASIC κυκλωμάτων χρησιμοποιούν γλώσσες περιγραφής υλικού (HDL) για τον σχεδιασμό και την περιγραφή λειτουργίας των ASIC , όπως είναι η VHDL ή η Verilog. Σε αντίθεση με τις FPGA τα ASIC δεν είναι επαναπρογραμματιζόμενα. Γι’ αυτόν το λόγο ονομάζονται και κυκλώματα ειδικού σκοπού. Αλλά αυτό που προσφέρουν είναι μειωμένο μέγεθος, σημαντικά μειωμένη κατανάλωση ενέργειας, ενώ το κόστος που έχουν επιτρέπει τη μαζική παραγωγή τους.

ΚΕΦΑΛΑΙΟ ΔΕΥΤΕΡΟ

2.1 Εισαγωγή

Η σχεδίαση των ενσωματωμένων συστημάτων είναι μία πολύπλοκη διαδικασία, πιο πολύ από αυτή της ανάπτυξης προγραμμάτων για έναν προσωπικό υπολογιστή. Αυτό συμβαίνει επειδή πρέπει να ικανοποιηθούν πολλοί σχεδιαστικοί περιορισμοί. Τέτοιοι περιορισμοί είναι οι περιορισμοί απόδοσης, κόστους, εξοικονόμησης ισχύος κ.τ.λ.

Το σύστημα πρέπει να πληρεί κάποιες συγκεκριμένες απαιτήσεις. Μπορεί επίσης να πρέπει να εκτελεί τις εργασίες μέσα σε συγκεκριμένο χρονικό διάστημα, να έχει περιορισμένη κατανάλωση ενέργειας, να έχει συγκεκριμένο μέγεθος, και να ικανοποιεί ένα σύνολο από άλλες μη λειτουργικές απαιτήσεις.

Η υλοποίηση ενός ενσωματωμένου συστήματος μπορεί να οδηγήσει σε ένα σύστημα που δε λειτουργεί. Για να αποφευχθεί ένα τέτοιου είδους πρόβλημα, όπως και άλλα παρόμοια με αυτό προβλήματα, πρέπει ο σχεδιαστής να διαθέτει τις κατάλληλες γνώσεις και δεξιότητες για τη σχεδίαση του συστήματος.

Στο κεφάλαιο αυτό θα εξηγήσουμε τη διαδικασία σχεδίασης ενός ενσωματωμένου συστήματος καθώς και τα προβλήματα και τις προκλήσεις που μπορεί να αντιμετωπίσουν οι σχεδιαστές.

2.2 Προκλήσεις στη σχεδίαση των ενσωματωμένων συστημάτων

Μία σημαντική πηγή δυσκολίας στη σχεδίαση των ενσωματωμένων συστημάτων είναι οι εξωτερικοί περιορισμοί (Constraints). Ορισμένοι από αυτούς είναι:

- Η ποσότητα του υλικού που χρειάζεται να χρησιμοποιηθεί.
- Η επιλογή του υλικού είναι σημαντική, διότι πρέπει να ικανοποιηθεί ο περιορισμός της απόδοσης του συστήματος και να υπολογιστεί το κόστος κατασκευής. Αν το υλικό είναι λίγο, τότε το σύστημα αποτυγχάνει να ικανοποιήσει τις προθεσμίες του, ενώ αν είναι πάρα πολύ, το σύστημα γίνεται πολύ ακριβό.
- Η ικανοποίηση της προθεσμίας.

- Ο πιο «ωμός» τρόπος ικανοποίησης της προθεσμίας είναι η επιτάχυνση του υλικού ώστε να εκτελείται ταχύτερα το πρόγραμμα.
- Η ελαχιστοποίηση της κατανάλωσης ενέργειας.

Ένας τρόπος ελαχιστοποίησης της κατανάλωσης ενέργειας είναι να το κάνουμε να λειτουργεί πιο αργά. Πρέπει όμως να δοθεί ιδιαίτερη προσοχή στους περιορισμούς της προθεσμίας και της απόδοσης έτσι ώστε να μην παραβιαστούν:

1. Ο σχεδιασμός με δυνατότητα αναβάθμισης.
2. Η πλατφόρμα υλικού μπορεί να χρησιμοποιείται για χρόνια από τους χρήστες και για αυτό το λόγο πρέπει να έχει ο σχεδιαστής τη δυνατότητα να προσθέτει και άλλα χαρακτηριστικά.
3. Η αξιοπιστία.
4. Ο πελάτης, όπως είναι προφανές, περιμένει ότι το προϊόν που θα αγοράσει θα λειτουργεί. Η αξιοπιστία είναι ιδιαίτερα σημαντική σε ορισμένες εφαρμογές όπως για παράδειγμα οι εφαρμογές των συστημάτων ασφαλείας.

Άλλο ένα σύνολο προκλήσεων προέρχεται από τα χαρακτηριστικά των συστατικών των ίδιων των συστημάτων.

Μερικοί από τους τρόπους με τους οποίους η φύση των ενσωματωμένων συστημάτων κάνει τη σχεδίασή τους πιο δύσκολη είναι οι εξής:

- Πολύπλοκη δοκιμή
- Περιορισμένες δυνατότητες παρατήρησης
- Περιορισμένα περιβάλλοντα ανάπτυξης [6]

2.3 Η μεθοδολογία σχεδίασης ενός ενσωματωμένου συστήματος

Η μεθοδολογία ή διεργασία σχεδίασης είναι σημαντική επειδή χωρίς αυτή, δε μπορούμε να παρέχουμε αξιόπιστα τα προϊόντα που θέλουμε να δημιουργήσουμε. Τα περισσότερα ενσωματωμένα υπολογιστικά συστήματα είναι πολύπλοκα, έτσι για να σχεδιαστούν και να κατασκευαστούν, πρέπει να ληφθούν υπόψη κάποιες διεργασίες σχεδίασης.

Η μεθοδολογία σχεδίασης ενός ενσωματωμένου συστήματος είναι σημαντική, κυρίως για τους εξής λόγους:

1. Βοηθάει το σχεδιαστή να καταγράψει όλα τα απαραίτητα βήματα για τη σχεδίαση του συστήματος. Με αυτόν τον τρόπο βεβαιώνεται ότι έχει κάνει ότι είναι αναγκαίο για το σύστημα.
2. Επιτρέπει την ανάπτυξη εργαλείων σχεδίασης με τη βοήθεια του υπολογιστή.
3. Η επικοινωνία των μελών της ομάδας σχεδίασης είναι ευκολότερη. Τα μέλη της ομάδας μπορούν να κατανοήσουν πιο εύκολα το κομμάτι της σχεδίασης που πρόκειται να υλοποιήσουν και τι πρέπει να λάβουν από τα άλλα μέλη της ομάδας. Εφόσον τα περισσότερα συστήματα σχεδιάζονται από ομάδες, αυτός είναι ίσως ο σημαντικότερος ρόλος μιας μεθοδολογίας σχεδίασης.

Οι διεργασίες αναπτύσσονται με τη πάροδο του χρόνου και αλλάζουν εξαιτίας εξωτερικών και εσωτερικών δυνάμεων. Παραδείγματα τέτοιων δυνάμεων είναι οι αλλαγές των απαιτήσεων του πελάτη, τα διαθέσιμα εξαρτήματα, το διαθέσιμο ανθρώπινο δυναμικό κτλ.

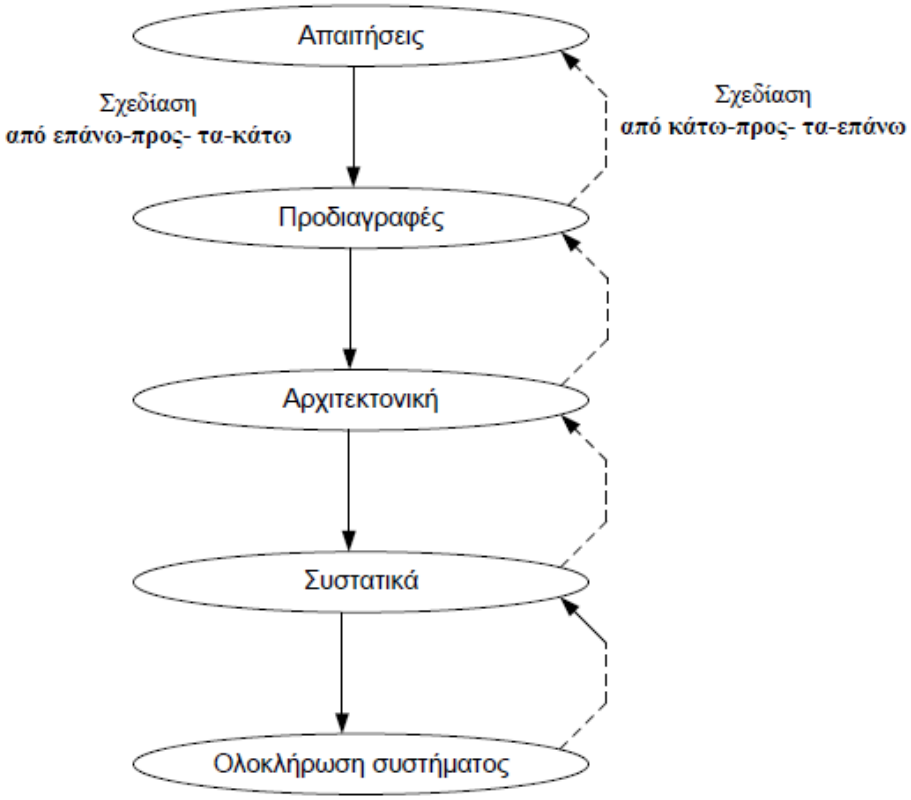
Μία καλή μεθοδολογία είναι κρίσιμη για την κατασκευή ενός συστήματος που δουλεύει σωστά. Η παράδοση συστημάτων με σφάλματα οδηγεί σε δυσαρεστημένους πελάτες.

Οι στόχοι σχεδίασης είναι εξίσου σημαντικοί και αναφέρονται παρακάτω:

- Το κόστος κατασκευής
- Η κατανάλωση ισχύος
- Η απόδοση του συστήματος (τόσο από πλευράς προθεσμίας, όσο και από πλευρά ταχύτητας)
- Η διασύνδεση με το χρήστη

Πρέπει επίσης να εξεταστούν οι εργασίες που εκτελούνται σε κάθε βήμα της διαδικασίας σχεδίασης ενός ενσωματωμένου συστήματος. Αρχικά, πρέπει να αναλύσουμε τη σχεδίαση σε κάθε βήμα έτσι ώστε να καθοριστεί ο τρόπος με τον οποίο θα ικανοποιήσουμε όλους τους περιορισμούς και τις προδιαγραφές του συστήματος. Έπειτα χρειάζεται να προστεθούν λεπτομέρειες έτσι ώστε να

τελειοποιηθεί η σχεδίαση. Τέλος, πρέπει να εξασφαλίσουμε ότι ο σύστημα που σχεδιάστηκε λειτουργεί. Αυτό θα γίνει με την επαλήθευση της σχεδίασης του συστήματος. Ο σκοπός της τελευταίας εργασίας είναι να εξασφαλίσουμε ότι ικανοποιούνται όλοι οι στόχοι που θέσαμε εξ αρχής για το σύστημα και έχουμε αναφέρει παραπάνω.



Σχήμα 2.1: Τα κυριότερα στάδια στη διαδικασία σχεδίασης ενός ενσωματωμένου συστήματος

Στο παραπάνω σχήμα φαίνονται τα στάδια της σχεδίασης ενός ενσωματωμένου συστήματος. Επίσης φαίνονται δύο διαδικασίες: η από πάνω-προς-τα-κάτω και η από κάτω-προς-τα-πάνω. Στην πρώτη διαδικασία, ξεκινάμε με την πιο αφαιρετική περιγραφή και καταλήγουμε στην πιο λεπτομερή. Από τη στιγμή που γνωρίζουμε τα συστατικά που χρειαζόμαστε για τη σχεδίαση του συστήματος, μπορούμε να τα σχεδιάσουμε χρησιμοποιώντας επίσης τόσο μονάδες λογισμικού όσο και μονάδες εξειδικευμένου υλικού. Με τη βοήθεια αυτών μπορούμε να σχεδιάσουμε ένα πλήρες σύστημα. Η δεύτερη διαδικασία

είναι μια εναλλακτική προσέγγιση της πρώτης, με τη διαφορά ότι εδώ ξεκινάμε με τα συστατικά σχεδίασης του συστήματος. Η διαδικασία αυτή είναι απαραίτητη διότι με τη χρήση της γνωρίζουμε την πορεία της σχεδίασης του συστήματος στα τελευταία στάδια της σχεδίασής του. Όσο μικρότερη εμπειρία έχει ο σχεδιαστής του συστήματος, τόσο περισσότερο θα πρέπει να βασίζεται στη χρήση της από κάτω-προς-τα-πάνω σχεδίασης.

Στις παρακάτω ενότητες θα εξηγηθεί πιο αναλυτικά το κάθε κομμάτι της διαδικασίας σχεδίασης.

2.3.1 Οι απαιτήσεις του συστήματος

Ο καθορισμός των απαιτήσεων είναι ιδιαίτερα σημαντικός, διότι έτσι γνωρίζει ο σχεδιαστής τις απαραίτητες πληροφορίες για το σύστημα που καλείται να σχεδιάσει. Οι απαιτήσεις βοηθάνε στη σωστή δημιουργία της αρχιτεκτονικής των συστατικών του συστήματος.

Οι απαιτήσεις μπορούν να είναι λειτουργικές ή μη λειτουργικές. Οι τυπικές μη λειτουργικές απαιτήσεις περιλαμβάνουν:

- Απόδοση
- Κόστος
- Φυσικό μέγεθος και βάρος
- Κατανάλωση ισχύος
- Χρόνος που απαιτείται για τον υπολογισμό της εξόδου κτλ.

Αφού καθοριστούν οι απαιτήσεις, περνάμε στο στάδιο της επικύρωσης. Η διαδικασία αυτή απαιτεί την κατανόηση τόσο αυτού που θέλει ο πελάτης όσο και του τρόπου με τον οποίο επικοινωνούν αυτές τις ανάγκες. Για τη διευκόλυνση του σχεδιαστή, δημιουργείται ένα πρωτότυπο (mock-up). Το πρωτότυπο χρησιμοποιεί καταγεγραμμένα δεδομένα για να προσομοιώσει τη λειτουργικότητα του συστήματος μία συγκεκριμένη χρονική στιγμή και μπορεί να εκτελεστεί σε έναν υπολογιστή ή σε ένα σταθμό εργασίας.

Σε περίπτωση που το σύστημα που καλείται μια ομάδα σχεδίασης να σχεδιάσει είναι μεγάλο και πολύπλοκο, χρησιμοποιείται μια φόρμα απαιτήσεων, η οποία μπορεί να συμπληρωθεί κατά την έναρξη του έργου. Η φόρμα αυτή

μπορεί επίσης να χρησιμοποιηθεί ως λίστα ελέγχου έτσι ώστε να βεβαιωθούν οι σχεδιαστές ότι δεν έχουν παραλείψει κάποιο βασικό χαρακτηριστικό. Ένα παράδειγμα φόρμας απαιτήσεων φαίνεται στο επόμενο σχήμα.

Όνομα:

Σκοπός:

Είσοδοι:

Έξοδοι:

Λειτουργίες:

Απόδοση:

Κόστος κατασκευής:

Ισχύς:

Φυσικό μέγεθος και βάρος:

Σχήμα 2.2: Φόρμα απαιτήσεων

2.3.2 Οι προδιαγραφές του συστήματος (Specifications)

Ο διαχωρισμός ανάμεσα στον καθορισμό των απαιτήσεων και των προδιαγραφών ενός συστήματος έχει μεγάλη σημασία στη διαδικασία σχεδίασης. Αυτό συμβαίνει διότι η περιγραφή του συστήματος που θέλει ο πελάτης έχει μεγάλη διαφορά από τα στοιχεία που χρειάζεται ο σχεδιαστής. Ένας τρόπος «μετάφρασης» της γλώσσας του πελάτη στη γλώσσα του σχεδιαστή είναι η καταγραφή ενός συνόλου απαιτήσεων σε μια τυπική προδιαγραφή.

Ο ρόλος της προδιαγραφής είναι ουσιαστικός για την ανάπτυξη των συστημάτων που λειτουργούν χωρίς μεγάλη προσπάθεια στη σχεδίασή τους. Οι προδιαγραφές πρέπει να είναι σαφείς και κατανοητές, έτσι ώστε ο σχεδιαστής να μπορεί να επαληθεύσει ότι το σύστημα λειτουργεί ικανοποιώντας τις απαιτήσεις του συστήματος και του πελάτη. Εάν οι προδιαγραφές έχουν λανθασμένα χαρακτηριστικά, υπάρχει πιθανότητα η αρχιτεκτονική του συστήματος να είναι

ελλιπής και να μην μπορεί να ικανοποιηθεί κάποιο στάδιο υλοποίησης του συστήματος. [6]

2.3.3 Η σχεδίαση των συστατικών υλικού/λογισμικού

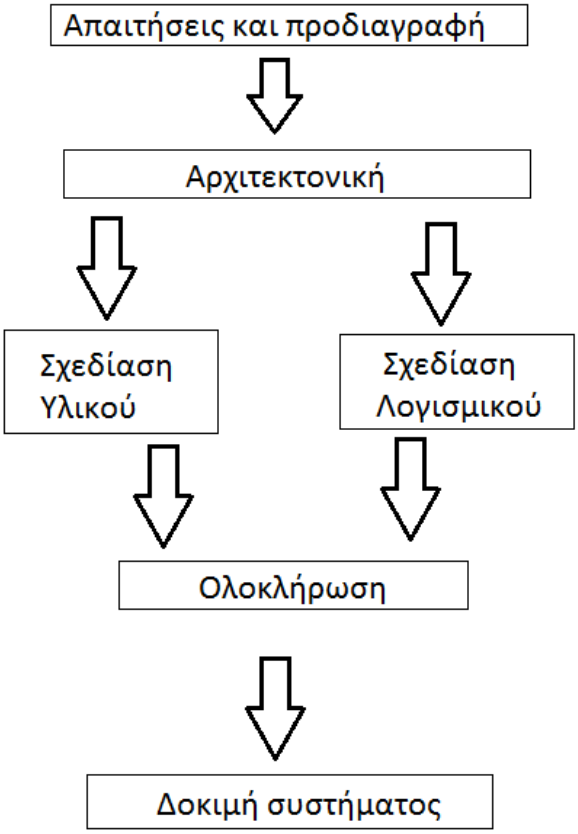
Στο στάδιο της σχεδίασης των συστατικών στοιχείων του συστήματος, κατασκευάζονται τα συστατικά της αρχιτεκτονικής. Τα συστατικά αυτά περιλαμβάνουν τόσο μονάδες υλικού, όπως π.χ. FPGA, όσο και μονάδες λογισμικού.

Ορισμένα από τα συστατικά στοιχεία του συστήματος που θα χρησιμοποιηθούν θα είναι ήδη έτοιμα, ενώ κάποια άλλα θα πρέπει να τα σχεδιαστούν από την ομάδα σχεδίασης του συστήματος. Παράδειγμα έτοιμων συστατικών στοιχείων είναι η ΚΜΕ και τα chip μνήμης.

Επίσης, στη σχεδίαση των συστατικών υλικού και λογισμικού μπορεί να χρησιμοποιηθούν μονάδες τυποποιημένου λογισμικού, των οποίων η χρήση βοηθάει στην εξοικονόμηση χρόνου σχεδίασης και στη γρήγορη υλοποίηση εξειδικευμένων λειτουργιών.

Κατά την ανάπτυξη των μονάδων ενσωματωμένου λογισμικού πρέπει να εξασφαλιστεί ότι το σύστημα λειτουργεί σωστά και χωρίς σφάλματα. Πρέπει οι σχεδιαστές να είναι πολύ προσεκτικοί και έμπειροι ώστε να εξασφαλίσουν ότι το σύστημα δεν καταλαμβάνει περισσότερο χώρο στη μνήμη απ’ όσο πρέπει και ότι η κατανάλωση ισχύος είναι σε χαμηλά επίπεδα.

Παρακάτω φαίνεται μια απλή μεθοδολογία σχεδίασης των συστατικών υλικού/λογισμικού:



Σχήμα 2.3: Απλή μεθοδολογία σχεδίασης συστατικών υλικού/λογισμικού

2.3.4 Ολοκλήρωση του συστήματος και έλεγχος σφαλμάτων

Αφού ολοκληρωθεί η σχεδίαση και η κατασκευή των συστατικών του συστήματος, τα τοποθετούμε όλα μαζί και ελέγχουμε αν το σύστημα λειτουργεί. Στο στάδιο αυτό βλέπουμε αν υπάρχουν σφάλματα και βρίσκουμε τρόπους να τα εξαλείψουμε.

Ο εντοπισμός των σφαλμάτων μπορεί να γίνει πιο εύκολος εάν ο σχεδιασμός του συστήματος είναι αρκετά καλός. Για τη διευκόλυνσή μας μπορούμε να «σπάσουμε» το σύστημα σε μικρότερα κομμάτια και να εντοπίσουμε και να αντιμετωπίσουμε τα σφάλματα στο κάθε κομμάτι ξεχωριστά. Με τη διόρθωση αυτών των σφαλμάτων μπορούν να διορθωθούν και άλλα, πιο πολύπλοκα σφάλματα που εντοπίζονται μόνο με σκληρή δοκιμή του συστήματος.

Συνοψίζοντας, στο στάδιο της ολοκλήρωσης του συστήματος πρέπει να καθορίσει ο σχεδιαστής γιατί το σύστημα λειτουργεί με το συγκεκριμένο τρόπο

και να διορθώσει τυχόν σφάλματα που θα εμφανιστούν. Η προσεκτική εισαγωγή των κατάλληλων ευκολιών αποσφαλμάτωσης βοηθάει στα προβλήματα ολοκλήρωσης του συστήματος.

2.4 Ανάπτυξη και αποσφαλμάτωση ενός ενσωματωμένου συστήματος

Στην ενότητα αυτή θα μελετήσουμε πως μπορούμε να δημιουργήσουμε αποδοτικά μέσα για τον προγραμματισμό και τον έλεγχο των ενσωματωμένων συστημάτων με τη χρήση ενός υπολογιστή εργασίας (Host) .

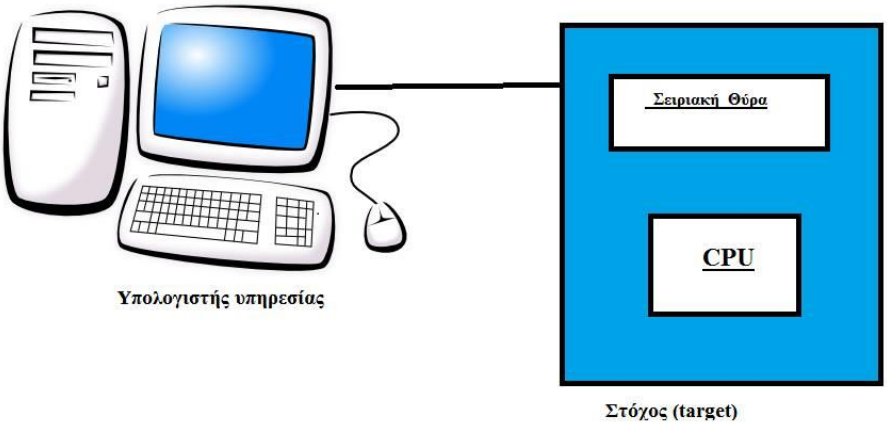
2.4.1 Το περιβάλλον ανάπτυξης

Λόγω περιορισμένης ποσότητας του εξοπλισμού ανάπτυξης του ενσωματωμένου συστήματος, αναπτύσσουμε ένα μέρος του λογισμικού σε έναν υπολογιστή ή σε κάποιο σταθμό εργασίας, γνωστό ως υπολογιστή υπηρεσίας ή host.

Ένας υπολογιστής υπηρεσίας πρέπει να:

- Φορτώνει το πρόγραμμα στον στόχο
- Ξεκινά και να τερματίζει την εκτέλεσή του
- Εξετάζει την μνήμη και τους καταχωρητές της CPU

Στόχος καλείται το υλικό στο οποίο θα τρέχει ο κώδικας και συνδέεται σειριακά με τον host, όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 2.1: Σύνδεση μεταξύ host και στόχου

2.4.1 Αποσφαλμάτωση

Ένας host προσφέρει ένα πιο φιλικό περιβάλλον στον σχεδιαστή του ενσωματωμένου συστήματος, απ' ότι μία τυπική πλατφόρμα ενσωματωμένης υπολογιστικής. Όμως, όταν ο κώδικας που βοηθά στην αποσφαλμάτωση επικοινωνεί με συσκευές I/O (εισόδου/εξόδου), ενδέχεται να παρουσιαστούν προβλήματα. Αυτό συμβαίνει διότι στον υπολογιστή υπηρεσίας, ο κώδικας λειτουργεί διαφορετικά για την κάθε συσκευή. Για το λόγο αυτό, δημιουργούμε ένα πρόγραμμα «πάγκου δοκιμής» (testbench program). Ο πάγκος δοκιμής παράγει εισόδους οι οποίες εξομοιώνουν τις ενέργειες των συσκευών εισόδου και συγκρίνει τις τιμές εξόδου που υπάρχουν με τις αναμενόμενες.

Ένα επίσης σημαντικό εργαλείο αποσφαλμάτωσης είναι το σημείο διακοπής (breakpoint). Η απλούστερη μορφή σημείου διακοπής είναι να καθορίζει ο χρήστης μια διεύθυνση όπου η εκτέλεση του προγράμματος πρέπει να διακοπεί. Όταν ο υπολογιστής φτάσει σε αυτό το σημείο, τότε ο έλεγχος επιστρέφει στο πρόγραμμα παρακολούθησης. Από το πρόγραμμα αυτό, ο χρήστης μπορεί να εξετάσει και να τροποποιήσει τους καταχωρητές της CPU, και κατόπιν να συνεχίσει την εκτέλεση του προγράμματος.

Σημαντικό εργαλείο αποσφαλμάτωσης αποτελεί και ο εξομοιωτής σε κύκλωμα (In-Circuit Emulator - ICE). Στο εσωτερικό του εξομοιωτή υπάρχει μία έκδοση μικροεπεξεργαστή που επιτρέπει να διαβάζονται οι εσωτερικοί καταχωρητές του όταν σταματά ο εξομοιωτής. Ο εξομοιωτής περιβάλλει με επιπλέον λογικά κυκλώματα αυτή την έκδοση του μικροεπεξεργαστή, για να μπορεί ο χρήστης να ορίζει τα σημεία διακοπής, να εξετάζει και να τροποποιεί την κατάσταση της CPU.

Ένα βοηθητικό εργαλείο στην αποσφαλμάτωση των συστημάτων είναι το μοντέλο σφάλματος (fault model). Το μοντέλο αυτό περιγράφει τους τύπους προβλημάτων που μπορούν να ανακύψουν στα συστατικά του συστήματος. Όταν παράγουμε δοκιμές μπορούμε να απαριθμήσουμε τι είδους σφάλματα μπορούν να συμβούν και σε ποια συστατικά. Οι δοκιμές πρέπει να εκτελούνται σε κάθε μονάδα που εξέρχεται από τη γραμμή παραγωγής και έτσι το κόστος δοκιμής μπορεί να αποτελέσει σημαντικό τμήμα του κόστους κατασκευής, συνεπώς οι αποδοτικές δοκιμές είναι ιδιαίτερα σημαντικές.

Εφόσον γνωρίζουμε τους τύπους των σφαλμάτων στους οποίους θέλουμε να εστιάσουμε, μπορούμε να απαριθμήσουμε όλες τις πιθανές περιπτώσεις σφάλματος για μια συγκεκριμένη σχεδίαση. Πρέπει να ελέγξουμε επίσης όλα τα πιθανά σφάλματα ώστε να δημιουργήσουμε μία σειρά δοκιμών που θα εφαρμοστούν στο σύστημα και θα εκθέσουν τα αποτελέσματα των σφαλμάτων.

2.5 Δοκιμή της κατασκευής

Ο ρόλος της δοκιμής της κατασκευής (manufacturing test) είναι να εξασφαλίσουμε ότι δεν εισάγονται λάθη στο σύστημα που έχουμε φτιάξει. Ακόμα και αν η κατασκευή μας είναι τέλεια, πάντα θα υπάρχει ενδεχόμενο λαθών στην κατασκευή κάθε αντίγραφου του συστήματος που βγαίνει από τη γραμμή παραγωγής.

Στην κατασκευή πολύπλοκων συστημάτων, είναι απαραίτητο να λαμβάνουμε υπ’ όψιν τις δοκιμές κατασκευής κατά τη διάρκεια της σχεδίασης. Η σχεδίαση με δυνατότητα δοκιμής (design for testability) εξασφαλίζει ότι η σχεδίαση δεν είναι δύσκολο να δοκιμαστεί και οι δοκιμές αυτές είναι πιο ικανοποιητικές.

Η αναλογία των λειτουργικών συσκευών που παράγονται από τη γραμμή παραγωγής και λειτουργούν σωστά ονομάζεται απόδοση (yield). Κάθε μονάδα

πρέπει να δοκιμάζεται, συνεπώς πρέπει ο χρόνος που απαιτείται για τη δοκιμή να είναι όσο γίνεται λιγότερος. Παράλληλα με τη μείωση χρόνου πρέπει να εξασφαλιστεί ότι έχουμε εντοπίσει όσα περισσότερα σφάλματα μπορούμε, προτού αποσταλεί το σύστημα στον πελάτη. Σε περίπτωση που το προϊόν παρουσιάσει σφάλματα στον πελάτη, επιστρέφεται με αποτέλεσμα να υπάρχει κόστος σε χρόνο και χρήμα.

Κατά τη δοκιμή ενός συστήματος πρέπει να είμαστε σε θέση να ξεχωρίσουμε τις ελαττωματικές συσκευές από αυτές που λειτουργούν κανονικά. Στην ιδανική περίπτωση, θα πρέπει να καλύψουμε όλες τις εμφανίσεις ενός σφάλματος ενός συγκεκριμένου τύπου. Η αναλογία των σφαλμάτων που μπορούμε να εντοπίσουμε σε μία σειρά δοκιμών ονομάζεται κάλυψη σφάλματος (fault coverage).

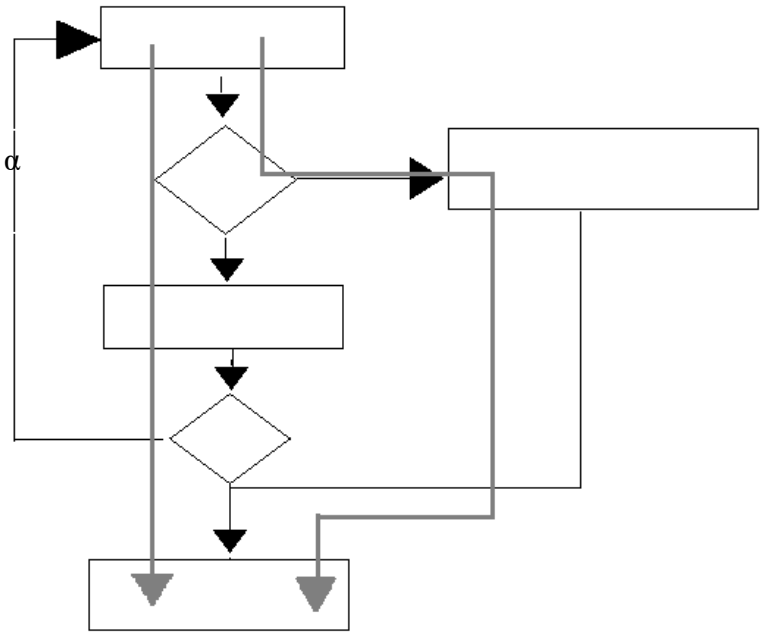
2.5.1 Επιλογή διαδρομής για δοκιμή ενός προγράμματος

Για να δοκιμάσουμε ένα σύνολο διαδρομών για τη δοκιμή ενός προγράμματος που προορίζεται για εκτέλεση σε ένα ενσωματωμένο σύστημα, υπάρχουν δύο επιλογές:

1. Να εκτελέσουμε την κάθε εντολή τουλάχιστον μία φορά και
2. Να εκτελέσουμε κάθε κατεύθυνση μίας διακλάδωσης τουλάχιστον μία φορά.

Για τις δομημένες γλώσσες προγραμματισμού, οι δύο συνθήκες που αναφέραμε είναι ισοδύναμες, όμως για μη δομημένο κώδικα είναι διαφορετικές. Ένα παράδειγμα μη δομημένου κώδικα είναι η συμβολική γλώσσα.

Παρακάτω φαίνεται ένα διάγραμμα, με τη βοήθεια του οποίου κατανοείται η διαφορά μεταξύ της κάλυψης εντολής και της κάλυψης διακλάδωσης. Μπορούμε να εκτελέσουμε κάθε πρόταση τουλάχιστον μία φορά εκτελώντας το πρόγραμμα κατά μήκος δύο ξεχωριστών διαδρομών.



Σχήμα 2.4: Διαδρομές εκτέλεσης προγράμματος

Έτσι, αυτό αφήνει τον κλάδο «α» από τη χαμηλότερη συνθήκη, ακάλυπτο. Για να εξασφαλίσουμε ότι έχουμε εκτελέσει κατά μήκος κάθε ακμής του διαγράμματος, πρέπει να εκτελέσουμε και μία Τρίτη διαδρομή μέσω του προγράμματος. Αυτή η διαδρομή δεν ελέγχει νέες εντολές αλλά αναγκάζει τον κλάδο «α» να εκτελεστεί. [1]

2.6 Προκλήσεις στην σχεδίαση ασφαλούς ενσωματωμένου συστήματος.

Οι σχεδιαστές μεγάλου μεγέθους και αυξανόμενου αριθμού ενσωματωμένων συστημάτων πρέπει να υποστηρίζουν διάφορες λύσεις ασφαλείας έτσι ώστε να ασχοληθούν με μία ή περισσότερες απαιτήσεις ασφαλείας. Οι απαιτήσεις αυτές παρουσιάζουν σημαντικά εμπόδια κατά τη διάρκεια της διαδικασίας του σχεδιασμού του συστήματος και περιγράφονται συνοπτικά παρακάτω:

Processing Gap: Οι υπάρχουσες αρχιτεκτονικές ενσωματωμένων συστημάτων δεν είναι σε θέση να συμβαδίζουν με τις υπολογιστικές απαιτήσεις των διεργασιών ασφαλείας, λόγω της αύξησης του ρυθμού των δεδομένων και την πολυπλοκότητα των πρωτοκόλλων ασφαλείας. Αυτές οι «ατέλειες» είναι

αισθητές κυρίως σε συστήματα που πρέπει να επεξεργαστούν πολύ υψηλούς ρυθμούς δεδομένων ή έναν μεγάλο αριθμό των συναλλαγών (π.χ., δρομολογητές του δικτύου, firewalls, και διακομιστές web), καθώς και σε συστήματα με περιορισμένους πόρους για μνήμη και επεξεργασία (π.χ., PDAs, ασύρματα τηλέφωνα και έξυπνες κάρτες).

Battery Gap: Η ενεργειακή κατανάλωση της μπαταρίας σε ένα ασφαλές ενσωματωμένο σύστημα με περιορισμένη μπαταρία είναι πολύ υψηλή. Ο ρυθμός ανάπτυξης της χωρητικότητας της μπαταρίας (5-8% ετησίως) είναι εύκολο να ξεπεραστεί από τις αυξανόμενες ενεργειακές απαιτήσεις των διεργασιών ασφαλείας, οδηγώντας στο λεγόμενο «κενό μπαταρίας». Διάφορες μελέτες δείχνουν ότι το διευρυνόμενο κενό μπαταρίας θα απαιτούσε από τους σχεδιαστές να κάνουν τις επιλογές για το σχεδιασμό της μπαταρίας με χαμηλή κατανάλωση ενέργειας (όπως βελτιστοποιημένα πρωτόκολλα ασφάλειας, προσαρμοσμένο υλικό ασφαλείας, και ούτω καθεξής) για ασφάλεια.

Ευελιξία (Flexibility): Ένα ενσωματωμένο σύστημα συχνά χρειάζεται για την εκτέλεση πολλών και διαφορετικών πρωτοκόλλων και προτύπων ασφαλείας ώστε να υποστηρίζει:

(i) πολλαπλούς στόχους ασφάλειας (π.χ., ασφαλείς επικοινωνίες, DRM, και ούτω καθεξής),

(ii) τη διαλειτουργικότητα σε διαφορετικά περιβάλλοντα (π.χ., μια φορητή συσκευή ότι πρέπει να εργαστεί και σε 3G και σε ασύρματο LAN), και

(iii) εγγύηση μεταποίησης σε διαφορετικά επίπεδα της στοίβας πρωτοκόλλων δικτύου (π.χ., ένα ασύρματο LAN, που είναι ενεργοποιημένο σε PDA και πρέπει να συνδεθεί σε ένα εικονικό ιδιωτικό δίκτυο και να υποστηρίξει ασφαλή web περιήγηση, μπορεί να χρειαστεί να εκτελέσει WEP, IPSec και SSL)

Επιπλέον, με τα πρωτόκολλα ασφάλειας να είναι συνεχώς στόχος των hackers, δεν είναι έκπληξη το γεγονός ότι εξακολουθούν να εξελίσσονται. Είναι,

ως εκ τούτου, σκόπιμο να πρέπει η αρχιτεκτονική ασφαλείας να είναι ευέλικτη (προγραμματιζόμενη) τόσο ώστε να προσαρμόζεται εύκολα στις μεταβαλλόμενες απαιτήσεις. Ωστόσο, η ευελιξία μπορεί επίσης να καταστήσει πιο δύσκολη την εξασφάλιση της ασφάλειας ενός σχεδίου.

Tamper Resistance: Επιθέσεις λόγω κακόβουλου λογισμικού, όπως ιούς και «δούρειους ίππους» είναι οι πιο κοινές απειλές σε οποιοδήποτε ενσωματωμένο σύστημα που είναι ικανό να εκτελεί εφαρμογές που έχετε κατεβάσει. Αυτές οι επιθέσεις μπορούν να εκμεταλλεύονται τα τρωτά σημεία του λειτουργικού συστήματος (OS) ή λογισμικού εφαρμογών, να αποκτούν πρόσβαση στα εσωτερικά του συστήματος, και να διαταράξουν την κανονική λειτουργία του. Επειδή αυτές οι επιθέσεις μπορούν να χειριστούν τα ευαίσθητα δεδομένα ή διεργασίες, να αποκαλύψουν εμπιστευτικές πληροφορίες (επιθέσεις της ιδιωτικής ζωής), και/ή να αρνηθούν την πρόσβαση στους πόρους του συστήματος (επιθέσεις διαθεσιμότητα), είναι απαραίτητο να αναπτυχθούν διάφορα μέτρα για το υλικό και το λογισμικό ενάντια σε αυτές τις επιθέσεις. Σε πολλά ενσωματωμένα συστήματα, όπως έξυπνες κάρτες, τις νέες και εξελιγμένες τεχνικές επίθεσης, όπως η διερεύνηση των διαύλων, ανάλυση χρονισμού, πρόκληση βλάβης, την ανάλυση ενέργειας, την ηλεκτρομαγνητική ανάλυση, και ούτω καθεξής, έχουν αποδειχθεί επιτυχείς στην εύκολη παραβίαση ασφάλειας.

Assurance Gap: Είναι γνωστό ότι όντως αξιόπιστα συστήματα είναι πολύ πιο δύσκολο να οικοδομηθούν από εκείνους που εργάζονται απλώς τις περισσότερες φορές. Αξιόπιστα συστήματα πρέπει να είναι σε θέση να χειριστούν το ευρύ φάσμα των καταστάσεων που μπορεί να συμβεί κατά τύχη. Τα ασφαλή συστήματα αντιμετωπίζουν μια ακόμα μεγαλύτερη πρόκληση: πρέπει να συνεχίσουν να λειτουργούν αξιόπιστα παρά τις επιθέσεις από αντίπαλους που. Δεδομένου ότι τα συστήματα γίνονται πιο περίπλοκα, υπάρχουν αναπόφευκτα πιο πιθανοί τρόποι αστοχίας που πρέπει να αντιμετωπιστούν.

Κόστος: Ένας από τους θεμελιώδεις παράγοντες που επηρεάζουν την αρχιτεκτονική ασφαλείας ενός ενσωματωμένου συστήματος είναι το κόστος. Για να κατανοήσουμε τις επιπτώσεις της ασφάλειας που σχετίζονται με την επιλογή

του σχεδιασμού σχετικά με το συνολικό κόστος του συστήματος, πρέπει να σκεφτούμε την απόφαση της ενσωμάτωσης των φυσικών μηχανισμών ασφαλείας σε ένα κρυπτογραφικό single-chip module.

Το Federal Information Processing Standard (FIPS 140-2) προσδιορίζει τέσσερα αυξημένα επίπεδα των φυσικών (καθώς και άλλων) απαιτήσεων ασφάλειας οι οποίες μπορούν να καλυφθούν με ένα ασφαλές σύστημα. Το επίπεδο ασφάλειας 1 απαιτεί ελάχιστη φυσική προστασία, το επίπεδο 2 απαιτεί την προσθήκη σφραγισμένων μηχανισμών όπως μια σφραγίδα ή περίβλημα, ενώ το επίπεδο 3 ορίζει ισχυρότερους μηχανισμούς ανίχνευσης και αντίδρασης.

Τέλος, το επίπεδο 4 δίνει εντολές για την προστασία του περιβάλλοντος από αποτυχίες (EFP και EFT), καθώς και εξαιρετικά αυστηρές διαδικασίες επανασχεδιασμού. Έτσι, μπορεί να επιλεχθεί η παροχή αυξανόμενων επιπέδων ασφάλειας που χρησιμοποιούν όλο και πιο προηγμένα μέτρα, έστω και με υψηλότερο κόστος του συστήματος, την προσπάθεια σχεδιασμού, και το χρόνο σχεδίασης. Είναι ευθύνη του σχεδιαστή να εξισορροπούνται οι απαιτήσεις ασφαλείας ενός ενσωματωμένου συστήματος με το κόστος της εφαρμογής των σχετικών μέτρων ασφαλείας [14]

ΚΕΦΑΛΑΙΟ ΤΡΙΤΟ

Στο κεφάλαιο 3 θα γίνει η μελέτη των μικροεπεξεργαστών εστιάζοντας στα σύνολα εντολών των ARM και SHARC. Οι δύο μικροεπεξεργαστές διαφέρουν σημαντικά. Η κατανόηση των λεπτομερειών του συνόλου εντολών είναι σημαντική τόσο από άποψη πληρότητας αλλά και προκειμένου να κατανοηθούν οι επιπτώσεις των χαρακτηριστικών της αρχιτεκτονικής του στην απόδοση του συστήματος, καθώς και σε άλλες ιδιότητες του.

3.1 Ταξινόμηση αρχιτεκτονικής υπολογιστών

Πρωτού ερευνήσουμε τις λεπτομέρειες των συνόλων εντολών των μικροεπεξεργαστών, είναι χρήσιμη η ανάπτυξη κάποιας βασικής ορολογίας. Αυτό θα γίνει με την ταξίνομηση των βασικών τρόπων με τους οποίους μπορεί να οργανωθεί ένας υπολογιστής.

3.1.1 Η αρχιτεκτονική von Neumann

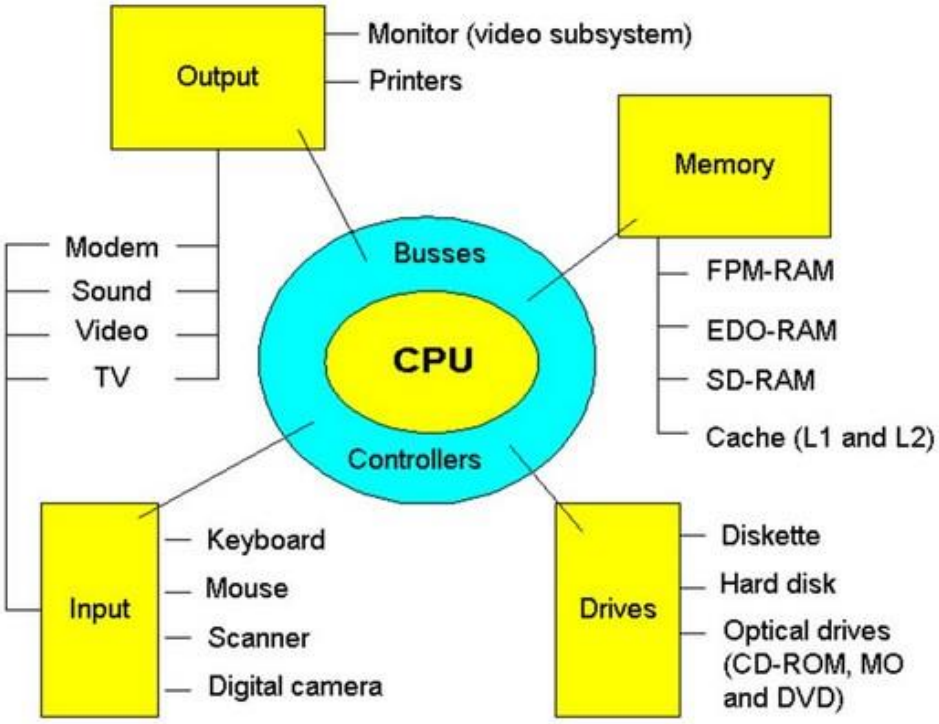
Σύμφωνα με τον von Neumann, έναν από τους πρωτοπόρους της σχεδίασης των πρώτων υπολογιστικών συστημάτων στη δεκαετία του 1940, ένα υπολογιστικό σύστημα:

1. Πρέπει να έχει διακριτές μονάδες για την αποθήκευση και επεξεργασία πληροφοριών.
2. Δεδομένα και εντολές αναπαρίστανται ως δυαδικά ψηφία για να μπορούν να αποθηκεύονται στις μονάδες αποθήκευσης.

Με βάση τη λογική της διάκρισης της επεξεργασίας από την αποθήκευση πληροφορίας, ένα υπολογιστικό μπορεί να διαχωρισθεί σε πέντε βασικά τμήματα:

1. τη Μονάδα Μνήμης, η οποία περιέχει τα δεδομένα αλλά και τις εντολές για την επεξεργασία τους,
2. τη Μονάδα Εισόδου, μέσω της οποίας εισάγονται τα δεδομένα στο εσωτερικό του υπολογιστικού συστήματος,

3. τη Μονάδα Εξόδου, μέσω της οποίας μεταφέρονται τα αποτελέσματα της επεξεργασίας των δεδομένων από το εσωτερικό του υπολογιστικού συστήματος στον εξωτερικό κόσμο,
4. την Αριθμητική και Λογική Μονάδα, η οποία εκτελεί αριθμητικές και λογικές πράξεις στα δεδομένα (σύμφωνα με τις αποθηκευμένες στη μνήμη εντολές), και
5. τη Μονάδα Ελέγχου, η οποία συντονίζει τη λειτουργία όλων των υπολοίπων μονάδων και διασφαλίζει την ομαλή συνεργασία μεταξύ τους.



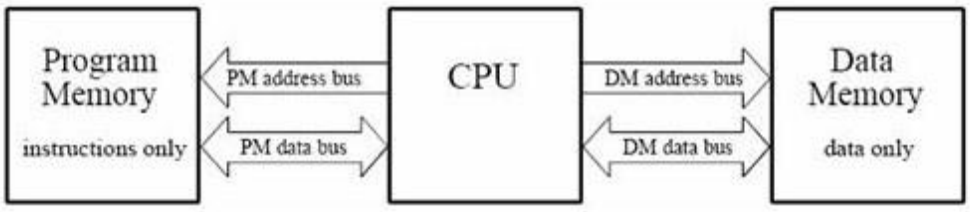
Σχήμα 3.1: Η αρχιτεκτονική von Neumann σε σχέση με το υλικό ενός σύγχρονου Η/Υ

Στο παραπάνω σχήμα οι μονάδες κύριας μνήμης, περιφερειακής μνήμης, εισόδου και εξόδου επικοινωνούν με την ΚΜΕ μέσω διαδρόμων επικοινωνίας (διαύλων - busses) και ειδικών συσκευών οι οποίοι ονομάζονται ελεγκτές (controllers).

3.1.2 Η αρχιτεκτονική Harvard.

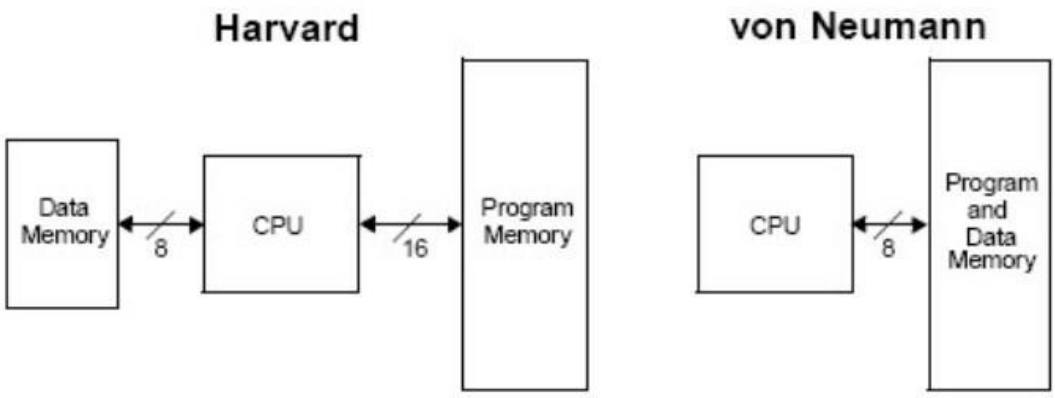
Σε αντιδιαστολή με την αρχιτεκτονική von Neumann υπάρχει η αρχιτεκτονική Harvard (Harvard architecture) της οποίας η βασική διαφορά είναι ότι διαχωρίζει τον χώρο αποθήκευσης εντολών και δεδομένων.

Πιο συγκεκριμένα, έχει τη μνήμη προγράμματος και τη μνήμη δεδομένων ως χωριστές μνήμες που προσεγγίζεται από χωριστούς διαύλους (bus). Αυτό βελτιώνει το εύρος ζώνης πέρα από την παραδοσιακή αρχιτεκτονική Von Neumann στην οποία το πρόγραμμα και τα στοιχεία προσκομίζονται από την ίδια μνήμη χρησιμοποιώντας τον ίδιο δίαυλο. Για να εκτελέσει μια οδηγία, μια μηχανή Von Neumann πρέπει να κάνει γενικά περισσότερες προσβάσεις στον δίαυλο για να προσκομίσει την πληροφορία. Κατόπιν τα στοιχεία μπορεί να πρέπει να μεταφερθούν μέσω του διαύλου, να χρησιμοποιηθούν στην αριθμητική και λογική μονάδα, και ενδεχομένως να τοποθετηθούν σε μια νέα θέση μνήμης. Όπως μπορεί κανείς να δει από αυτήν την περιγραφή, η αρχιτεκτονική αυτή μπορεί να δημιουργήσει κυκλοφοριακή συμφόρηση ή ακόμη και κορεσμό. Με την αρχιτεκτονική Harvard, η πληροφορία προσκομίζεται σε έναν απλό κύκλο ρολογιού. Ενώ η μνήμη προγράμματος προσπελαύνεται, η μνήμη δεδομένων είναι σε έναν ανεξάρτητο δίαυλο και μπορεί να διαβαστεί και να γραφτεί. Αυτοί οι χωρισμένοι δίαυλοι επιτρέπουν σε μια εντολή την εκτέλεση, ενώ η επόμενη εντολή προσκομίζεται.



Σχήμα 3.2: Αρχιτεκτονική Harvard

Στο παρακάτω σχήμα βλέπουμε μία σύγκριση των δύο αρχιτεκτονικών

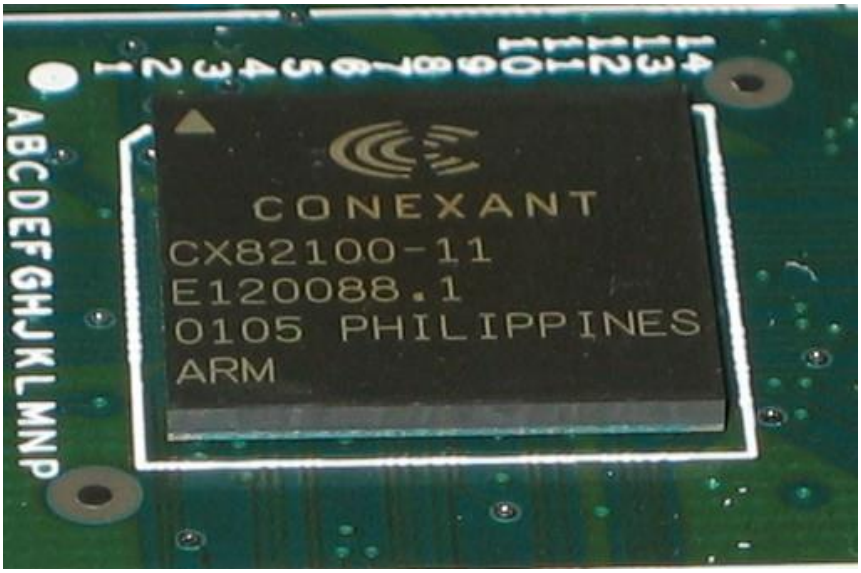


Σχήμα 3.3: Σύγκριση αρχιτεκτονικών

3.1.3 Αρχιτεκτονική RISC έναντι CISC

Οι μελέτες πραγματικών προγραμμάτων, η ταχύτερη μνήμη RAM, οι βελτιώσεις στην τεχνολογία των μεταγλωτιστών και το μικρότερο κόστος αποθηκευτικού χώρου είναι μερικές από τις δυνάμεις που οδήγησαν στη σχεδίαση Μηχανών Περιορισμένου Συνόλου Εντολών (Reduced Instruction Set Computer) (RISC).

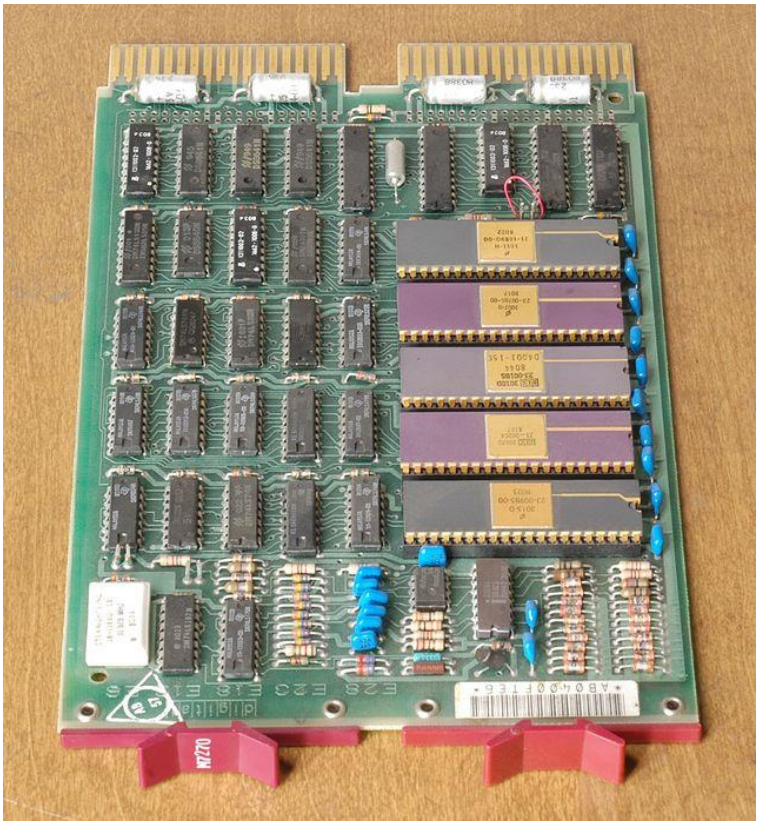
Η στρατηγική στην οποία βασίζεται η αρχιτεκτονική RISC είναι η ύπαρξη ενός μικρού συνόλου εντολών, οι οποίες πραγματοποιούν ένα ελάχιστο πλήθος απλών λειτουργιών. Οι σύνθετες εντολές προσομοιώνονται με τη χρήση ενός υποσυνόλου απλών εντολών. Ο προγραμματισμός σε RISC είναι πιο δύσκολος και χρονοβόρος από ότι στον άλλο σχεδιασμό, επειδή οι πιο σύνθετες εντολές προσομοιώνονται από απλές.



Εικόνα 3.1: Η αρχιτεκτονική ARM είναι παράδειγμα εφαρμογής του σχεδιασμού RISC

Η στρατηγική στην οποία βασίζεται η αρχιτεκτονική CISC(Complex instruction set computer) είναι η ύπαρξη ενός μεγάλου συνόλου εντολών, στο οποίο συμπεριλαμβάνονται και σύνθετες. Οι προγραμματιστές στην CISC δεν χρειάζεται να γράφουν σύνολα εντολών για να πραγματοποιήσουν κάποια σύνθετη εργασία.

Η πολυπλοκότητα του συνόλου εντολών έχει ως αποτέλεσμα το ηλεκτρονικό κύκλωμα της ΚΜΕ και της μονάδας ελέγχου να είναι υπερβολικά σύνθετα. Για την μείωση αυτής της πολυπλοκότητας, οι σχεδιαστές της αρχιτεκτονικής CISC έχουν καταλήξει στην ακόλουθη λύση: Ο προγραμματισμός γίνεται σε δύο επίπεδα. Οι εντολές σε γλώσσα μηχανής δεν εκτελούνται κατευθείαν από την ΚΜΕ. Αυτό προϋποθέτει την προσθήκη ενός ειδικού τύπου μνήμης, τη μικρομνήμη, στην οποία αποθηκεύεται το σύνολο των εντολών κάθε σύνθετης εντολής του μηχανήματος. Ένα μειονέκτημα της αρχιτεκτονικής CISC είναι ο επιπλέον φόρτος που σχετίζεται με το μικροπρογραμματισμό και την προσπέλαση της μικρομνήμης.



Εικόνα 3.2: Η αρχιτεκτονική PDP-11 είναι παράδειγμα εφαρμογής του σχεδιασμού CISC

Πέρα από το βασικό χαρακτηριστικό RISC/CISC, μπορούμε να ταξινομήσουμε τους υπολογιστές από διάφορα χαρακτηριστικά των συνόλων εντολών τους. Το σύνολο εντολών του υπολογιστή καθορίζει την διασύνδεση μεταξύ των μονάδων λογισμικού και του υποκείμενου υλικού οι εντολές καθορίζουν τι θα κάνει το υλικό κάτω από τις συγκεκριμένες συνθήκες. Οι εντολές μπορούν να έχουν μια ποικιλία χαρακτηριστικών, τα οποία περιλαμβάνουν:

- Σταθερό έναντι μεταβλητού μήκους
- Τρόπους διευθυνσιοδότησης (addressing modes)
- Αριθμούς τελεστών (operands)
- Τύπους υποστηριζόμενων λειτουργιών

Το σύνολο των καταχωρητών οι οποίοι είναι διαθέσιμοι για χρήση από τα προγράμματα καλείται μοντέλο προγραμματισμού (programming model), επίσης γνωστό ως μοντέλο προγραμματιστή (programmer model).

Μπορούν να υπάρξουν διαφορετικές υλοποιήσεις μιας αρχιτεκτονικής. Στην πραγματικότητα, ο ορισμός της αρχιτεκτονικής χρησιμεύει για να καθορίζει εκείνα τα χαρακτηριστικά τα οποία πρέπει να ισχύουν για όλες τις υλοποιήσεις και αυτά τα οποία μπορεί να ποικίλουν από υλοποίηση σε υλοποίηση. Διαφορετικές CPUs μπορεί να προσφέρουν διαφορετικές ταχύτητες ρολογιών, διαφορετικές διευθετήσεις κρυφής μνήμης (cache configurations), αλλαγές στο δίαυλο (bus) ή στις γραμμές διακοπών (interrupt lines), και πολλές άλλες αλλαγές οι οποίες μπορούν να κάνουν ένα μοντέλο CPU ελκυστικότερο από ένα άλλο για μια συγκεκριμένη εφαρμογή.

3.2 Συμβολική γλώσσα

Μια συμβολική γλώσσα (assembly language) είναι μια χαμηλού επιπέδου γλώσσα προγραμματισμού, δηλαδή μια γλώσσα πολύ κοντά στη γλώσσα μηχανής και στο υλικό του υπολογιστή. Κάθε συγκεκριμένη αρχιτεκτονική συνόλου εντολών, δηλαδή κάθε οικογένεια επεξεργαστών, έχει τη δική της συμβολική γλώσσα, η οποία δίνεται συνήθως από τον κατασκευαστή της.

Ένα πρόγραμμα σε γλώσσα μηχανής είναι ένα μοτίβο από bits στα οποία κωδικοποιούνται εντολές του επεξεργαστή και δεδομένα. Αυτό γίνεται πιο ευανάγνωστο αντικαθιστώντας τις ακολουθίες των bits με μνημονικά σύμβολα.

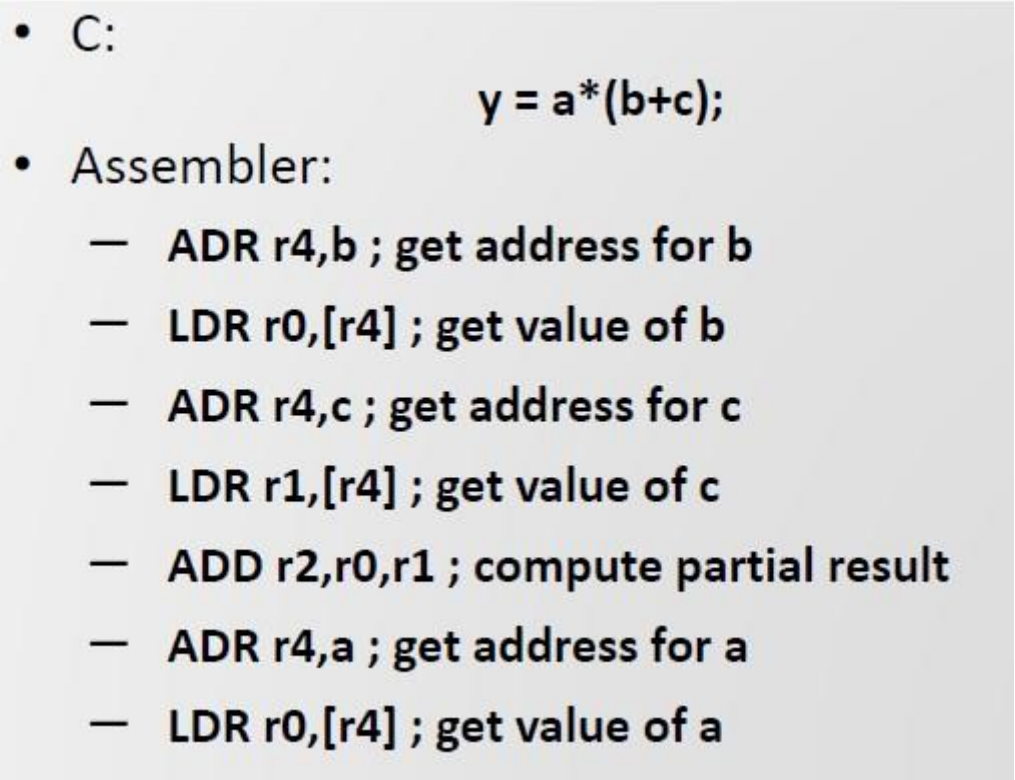
Οι συμβολικές γλώσσες μοιράζονται συνήθως τα ίδια χαρακτηριστικά :

- Μια εντολή εμφανίζεται ανά γραμμή
- Οι ετικέτες (labels), οι οποίες δίνουν ονόματα στις θέσεις μνήμης , αρχίζουν στην πρώτη στήλη
- Οι εντολές πρέπει να αρχίζουν στη δεύτερη στήλη ή μετά ώστε να διακρίνονται από τιε ετικέτες
- Σχόλια τα οποία ξεκινούν από κάποιο καθορισμένο χαρακτήρα σχολίου (; στην περίπτωση του ARM) και φτάνουν μέχρι το τέλος της γραμμής

Μορφή εντολής σε συμβολική γλώσσα (assembly)

[Ετικέτα] [Εντολή] [Όρισμα] ; [Σχόλια]

Παρακάτω υπάρχει ένα παράδειγμα σε συμβολική γλώσσα



Εικόνα 3.3: Παράδειγμα σε συμβολίκη γλώσσα

Η μετατροπή ενός προγράμματος από συμβολική γλώσσα σε γλώσσα μηχανής γίνεται από ένα συμβολομεταφραστή (assembler) και το αντίστροφο γίνεται από έναν αντισυμβολομεταφραστή (disassembler).

Οι συμβολομεταφραστές πρέπει επίσης να παρέχουν μερικές ψευδολειτουργείες (pseudo-ops) για να βοηθούν τους προγραμματιστές ν δημιουργούν ολοκληρωμένα προγράμματα συμβολικής γλώσσας. Ένα παράδειγμα ψυεδο-λειτουργίας είναι μία η οποία επιτρέπει σε τιμές δεδομένων να φορτωθούν σε θέσεις μνήμης. Αυτές επιτρέπουν, για παράδειγμα, να τίθενται στη μνήμη.

3.3 Ο επεξεργαστής ARM

Ο ARM (Advanced RISC Machine) στην πραγματικότητα είναι μια οικογένεια αρχιτεκτονικών RISC, η οποία έχει αναπτυχθεί από την ARM Holdings. Η αρχιτεκτονική ARM είναι η πιο συχνά χρησιμοποιούμενη αρχιτεκτονική συνόλου εντολών 32-bit όσον αφορά τους επεξεργαστές που παράγονται.

Οι επεξεργαστές ARM είναι σχετικά απλοί, κάτι που τους κάνει κατάλληλους για εφαρμογές χαμηλής ισχύος. Αυτό έχει ως αποτέλεσμα να έχουν υπερισχύσει στις αγορές των κινητών και των ενσωματωμένων συστημάτων, σαν μικροί και σχετικά χαμηλού κόστους μικροεπεξεργαστές και μικροελεγκτές. Το 2005, περίπου το 98% των πάνω από ένα δισεκατομμύριο κινητών τηλεφώνων που πωλούνται κάθε χρόνο είχαν τουλάχιστον έναν επεξεργαστή ARM. Το 2009 οι επεξεργαστές ARM αντιστοιχούσαν περίπου στο 90% όλων των ενσωματωμένων επεξεργαστών RISC 32-bit και χρησιμοποιούνται σε μεγάλο βαθμό σε καταναλωτικά ηλεκτρονικά προϊόντα, συμπεριλαμβανομένων των προσωπικών ψηφιακών βοηθών (personal digital assistants, PDAs), των κινητών τηλεφώνων, των συσκευών ψηφιακής μουσικής και πολυμέσων, των φορητών κονσόλων βιντεοπαιχνιδιών, των αριθμομηχανών και περιφερειακών υπολογιστών όπως οι σκληροί δίσκοι και οι δρομολογητές.

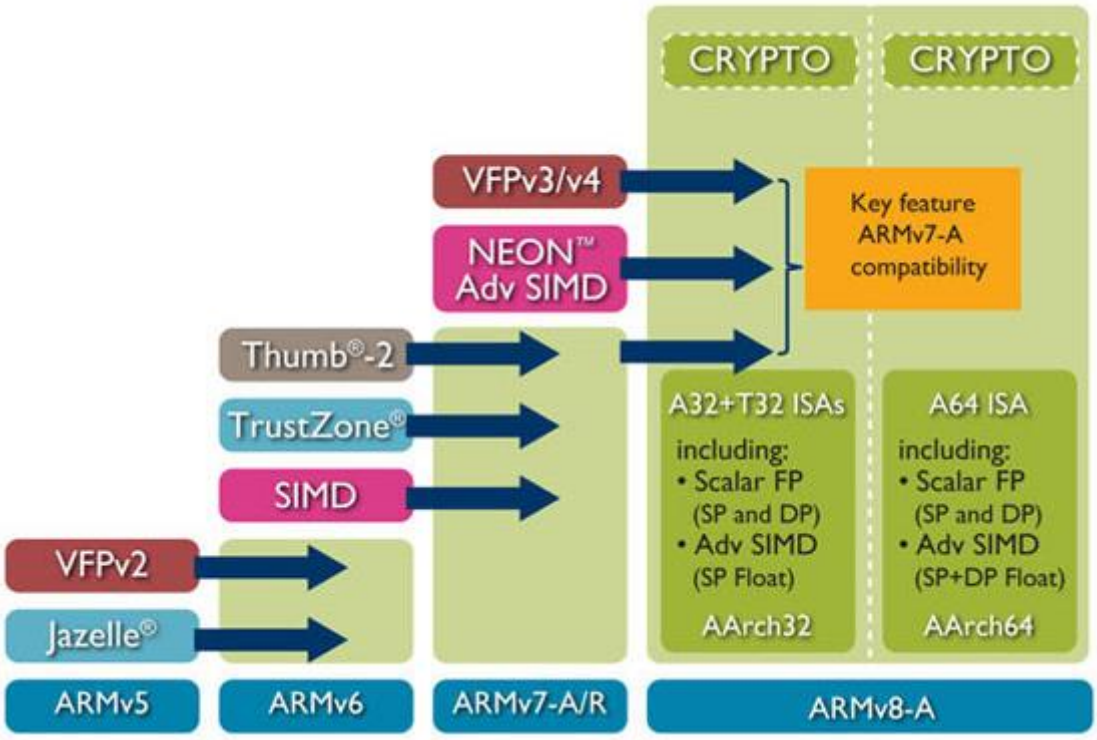
Η αρχιτεκτονική ARM υποστηρίζει δύο βασικούς τύπους δεδομένων:

- Η τύπική λέξη του ARM είναι 32-bits
- Η λέξη μπορεί να διαιρεθεί σε τέσσερα bytes των 8 bits.

Η ARM αρχιτεκτονική αποτελεί τη βάση για κάθε επεξεργαστή ARM. Με την πάροδο του χρόνου, η αρχιτεκτονική ARM έχει εξελιχθεί για να περιλάβει αρχιτεκτονικά χαρακτηριστικά ώστε να ανταποκριθεί στην αυξανόμενη ζήτηση για νέες λειτουργίες, υψηλές επιδόσεις και τις ανάγκες των νέων και αναδυόμενων αγορών.

Υπάρχουν επί του παρόντος δύο ARMv8 προφίλ, το προφίλ ARMv8-A για αγορές υψηλής απόδοσης, όπως τα κινητά και τις επιχειρήσεις, καθώς και το προφίλ ARMv8-R για ενσωματωμένες εφαρμογές στην αυτοκινητοβιομηχανία και βιομηχανικό έλεγχο.

Η αρχιτεκτονική ARM υποστηρίζει ένα ευρύ φάσμα των σημείων επίδοσης, καθιστώντας την αρχιτεκτονική αυτή ως την «πρότυπη» αρχιτεκτονική σε πολλά τμήματα της αγοράς και οδηγεί σε πολύ μικρές και αποτελεσματικές υλοποιήσεις των επεξεργαστών ARM, με χρήση μικρο-αρχιτεκτονικής τελευταίας τεχνολογίας. Έχουν αναπτυχθεί επεκτάσεις στην αρχιτεκτονική οι οποίες παρέχουν υποστήριξη για Java επιτάχυνση, ασφάλεια, SIMD, και Advanced SIMD τεχνολογίες. Η ARMv8-αρχιτεκτονική προσθέτει μία κρυπτογραφική επέκταση ως προαιρετικό χαρακτηριστικό.



Εικόνα 3.4: Η εξέλιξη των επεξεργαστών ARM(e)

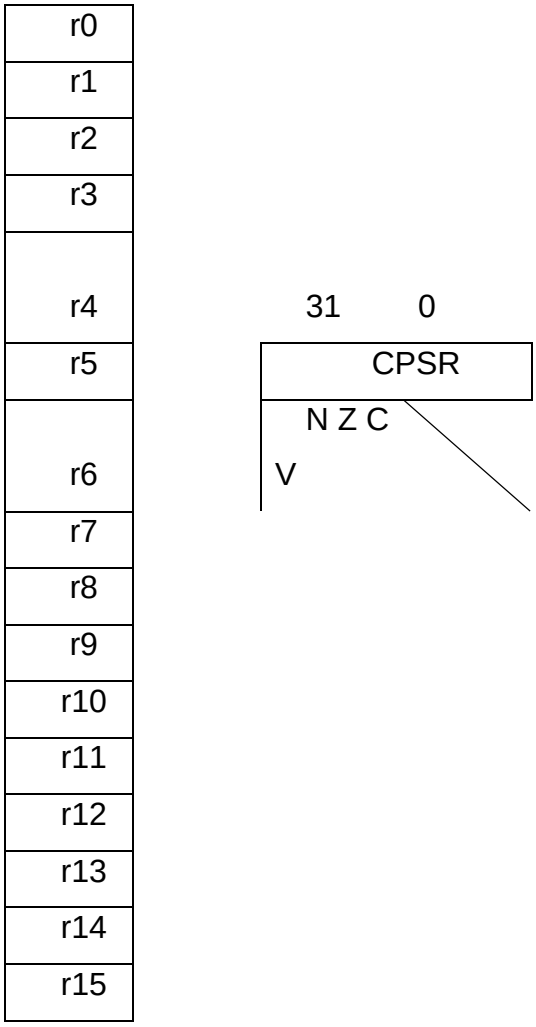
Η αρχιτεκτονική ARM είναι παρόμοια με την αρχιτεκτονική RISC, οποία θα εξηγηθεί παρακάτω, καθώς ενσωματώνει αυτά τα τυπικά χαρακτηριστικά της RISC αρχιτεκτονικής:

- Ένα ενιαίο αρχείο μητρώου αρχιτεκτονική φορτίο / κατάστημα, όπου η επεξεργασία των δεδομένων λειτουργεί μόνο στο περιεχόμενο μητρώο, δεν συνδέονται άμεσα με το περιεχόμενο της μνήμης.

- Απλή αντιμετώπιση των τρόπων, με όλες τις διευθύνσεις φορτίο / κατάστημα προσδιορίζεται από τα περιεχόμενα του μητρώου και μόνο τομείς διδασκαλίας.

Βελτιώσεις στη βασική αρχιτεκτονική RISC επιτρέπει στους επεξεργαστές ARM για να αποκτούν μια καλή ισορροπία των υψηλών επιδόσεων, μικρού μεγέθους κώδικα, χαμηλή κατανάλωση ενέργειας και μικρή επιφάνεια πυριτίου.

Στον επεξεργαστή ARM, οι αριθμητικές και λογικές λειτουργίες δεν μπορούν να εκτελεσθούν άμεσες θέσεις μνήμης. Ο ARM είναι μια αρχιτεκτονική φόρτωσης-αποθήκευσης που οι τελεστές δεδομένων πρέπει ο πρώτα να φορτώνονται στην CPU και στην συνέχεια από εκεί να αποθηκεύονται πίσω στην κύρια μνήμη για την αποθήκευση των αποτελεσμάτων. Ο ARM έχει 16 καταχωρητές γενικού σκοπού, r0 έως r15. Εκτός από τον r15 που χρησιμεύει σαν μετρητής προγράμματος, είναι όλοι ίδιοι. Ο άλλος σημαντικός βασικός καταχωρητής στο μοντέλο προγραμματισμού είναι ο καταχωρητής τρέχουσας κατάστασης προγράμματος (current program status register- CPSR).



Σχήμα 3.4: Το βασικό μοντέλο προγραμματισμού του ARM

3.4 Ο επεξεργαστής SHARC

Η οικογένεια των επεξεργαστών ψηφιακού συστήματος SHARC [ADI97] χρησιμοποιεί τη αρχιτεκτονική Harvard. Ο SHARC είναι ένα καλό συμπλήρωμα της CPU του ARM επειδή χρησιμοποιούν αρκετά διαφορετικές τεχνικές σε πολλές πτυχές των αρχιτεκτονικών τους. Οι εντολές SHARC γράφονται μια ανα γραμμή και έχουν την παρακάτω μορφή:

R1= DM(M0,I0), R2= PM(M8,I8); ! ένα σχόλιο

label: R3= R1 + R2;

Ο SHARC χρησιμοποιεί διαφορετικά μεγέθη λέξης και μεγέθη χώρων διευθύνσεων για εντολές και δεδομένα. Μια εντολή SHARC αποτελείται από 48bits, μια βασική λέξη δεδομένων, από 32bits και μία διεύθυνση, από 32bits. Η οικογένεια SHARC περιλαμβάνει μια σημαντική ποσότητα εσωτερικής μνήμης. Η εσωτερική μνήμη είναι ομοιόμορφα χωρισμένη μεταξύ της μνήμης προγράμματος (program memory – PM) και της μνήμης δεδομένων (data memory – DM), και περοσσότερη μνήμη μπορεί να προστεθεί εξωτερικά (έξω από το ολοκλήρωμένο κύκλωμα του επεξεργαστή –off-chip).

Ο SHARC, στην πραγματικότητα, υποστηρίζει τους ακόλουθους τύπους δεδομένων:

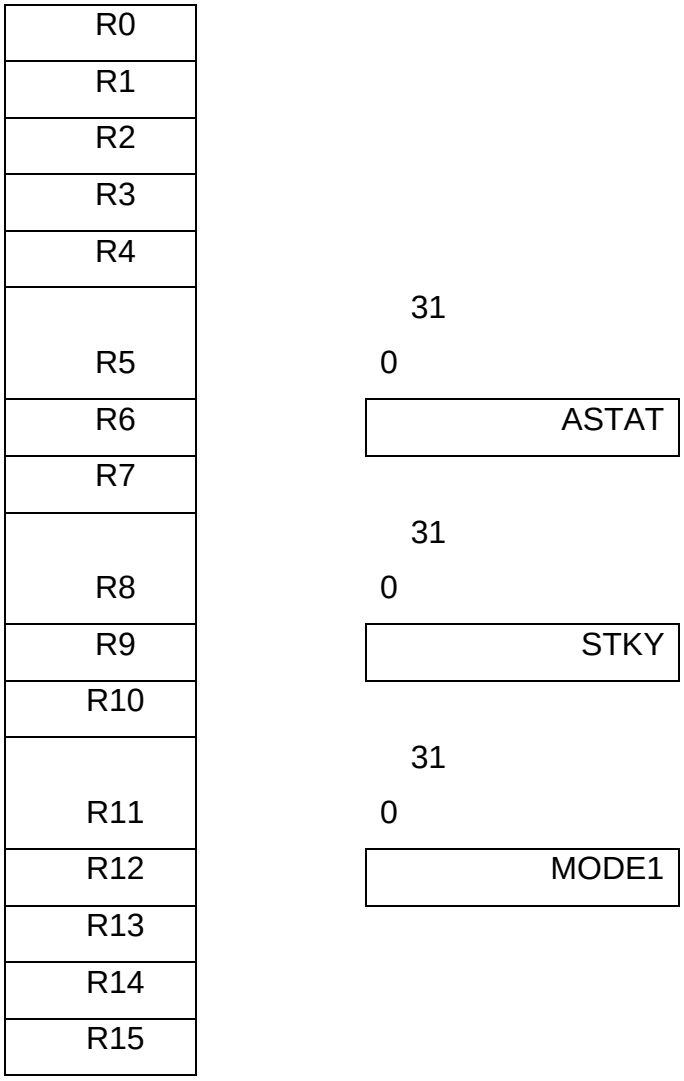
- 32-bit κινητής υποδιαστολής IEEE μονής ακρίβειας
- 40-bit κινητής υποδιαστολής IEEE εκτεταμένης ακρίβειας
- 32-bit ακέραιους

Ο SHARC είναι μια τροποποιημένη αρχιτεκτονική Harvard που επιτρέπει στη μνήμη προγράμματος να αποθηκεύει τόσο δεδομένα όσο και εντολές. Αυτό επιτρέπει μερικές φορές στα δεδομένα να προσκομιστούν και από τις δύο μνήμες παράλληλα.

Το μοντέλο προγραμματισμού για τον SHARC είναι μάλλον μεγάλο και σύνθετο. Οι κύριοι καταχωρητές δεδομένων έχουν δύο διαφορετικά ονόματα.

- R0 έως R15 όταν χρησιμοποιούνται για λειτουργίες ακέραιων και
- f0 έως f15 όταν χρησιμοποιείται για λειτουργίες κινητής υποδιαστολής.

Όλοι οι παραπάνω καταχωρητές δεδομένων είναι μεγέθους 40bits για την διαχείριση του μεγαλύτερου τύπου δεδομένων, την εκτεταμένης ακρίβειας κινητής υποδιαστολής 40-bit τιμή.



Σχήμα 3.5: Το βασικό μοντέλο προγραμματισμού του SHARC

Όταν τύποι δεδομένων 32-bit αποθηκεύονται στους καταχωρητές, τοποθετούνται στα περισσότερο σημαντικά bits του καταχωρητή.

Η CPU έχει τρείς κύριες λειτουργικές μονάδες δεδομένων:

- μια αριθμητική λογική μονάδα
- ένα πολλαπλασιαστή
- και έναν ολισθητή.

Οι παραπάνω μονάδες εκτελούν ένα ευρύ φάσμα λειτουργιών, όπως ταιριάζει σε ένα μικροεπεξεργαστή σχεδιασμένο για εντατικά από άποψη αριθμητικής προγράμματα.

Οι τρείς σημαντικότεροι καταχωρητές τρόπου λειτουργίας για τις λειτουργίες δεδομένων είναι οι:

1. arithmetic status (ASTAT)
2. sticky (STKY)
3. mode 1 (MODE1)

ΚΕΦΑΛΑΙΟ ΤΕΤΑΡΤΟ

4.1 Εισαγωγή στη UML

Η UML χρησιμοποιείται για τη μοντελοποίηση μεγάλου εύρους συστημάτων. Ο στόχος της UML είναι να περιγράφει κάθε τύπο συστήματος, μέσα από αντικειμενοστραφή διαγράμματα. Η πιο συνήθης χρήση της είναι η παραγωγή μοντέλων συστημάτων λογισμικού, πέρα απο αυτό όμως χρησιμοποιείται για την περιγραφή συστημάτων που δεν αφορούν λογισμικό, όπως για παράδειγμα μηχανικών συστημάτων. Μία κατηγορία συστημάτων στην οποία χρησιμοποιείται η UML είναι τα ενσωματωμένα συστήματα πραγματικού χρόνου.

Η UML στοχεύει στο σχεδιασμό αντικειμενοστραφών συστημάτων. Το σχέδιο είναι μια απλοποιημένη παράσταση της πραγματικότητας.

Έτσι δημιουργώντας ένα σχέδιο επιτυγχάνουμε τέσσερεις στόχους:

1. παριστάνουμε οπτικά το σύστημα που έχουμε ή θέλουμε να κατασκευάσουμε,
2. προσδιορίζουμε τη δομή και τη συμπεριφορά του συστήματος,
3. δημιουργούμε ένα πρότυπο για να βασίσουμε την κατασκευή του συστήματος,
4. τεκμηριώνουμε τις αποφάσεις που λάβαμε.

Σε όλους τους τεχνολογικούς τομείς ο σχεδιασμός βασίζεται σε τέσσερεις βασικές αρχές:

1. η επιλογή του είδους του σχεδίου έχει επίπτωση στον τρόπο και την μορφή επίλυσης του προβλήματος,
2. όλα τα σχέδια εκφράζονται σε διαφορετικές βαθμίδες ακρίβειας,
3. τα καλύτερα σχέδια σχετίζονται με την πραγματικότητα,
4. ένα είδος σχεδίων δεν είναι ποτέ αρκετό.

Η UML περιλαμβάνει τρία βασικά στοιχεία:

1. Οντότητες
2. Σχέσεις
3. Διαγράμματα

Τα πιο βασικά θεμελιώδη στοιχεία της UML είναι το αντικείμενο (object) και η κλάση (class).Το αντικείμενο είναι ένα στιγμιότυπο (instance) μιας κλάσης. Μια

κλάση έχει ιδιότητες (attributes) και συμπεριφορές (behaviors). Ένα πακέτο (package) είναι μια οργανωτική μονάδα του συστήματος που μπορεί να περιέχει ορισμούς κλάσεων, αντικειμένων και ούτω καθεξής. Μια κατάσταση (state) χρησιμοποιείται στα διαγράμματα κατάστασης για την περιγραφή συμπεριφοράς. Ένας φυσικός επεξεργαστής είναι ένα στοιχείο υλικού. Το συστατικό είναι το φυσικό τμήμα του συστήματος το οποίο υλοποιεί ένα σύνολο διασυνδέσεων(interfaces). [10][11]

4.2 Τμήματα της UML

Οψεις: Δείχνουν διαφορετικά χαρακτηριστικά του συστήματος που μοντελοποιούνται. Μια όψη αποτελείται απο ένα σύνολο διαγραμμάτων

Σχήμα 4.1: Είδη όψεων.

Στοιχεία μοντέλου: Είναι οι έννοιες που χρησιμοποιούνται στα διαγράμματα για να αναπαραστήσουν τις κλάσεις, τα αντικείμενα και τις μεταξύ τους συσχετίσεις

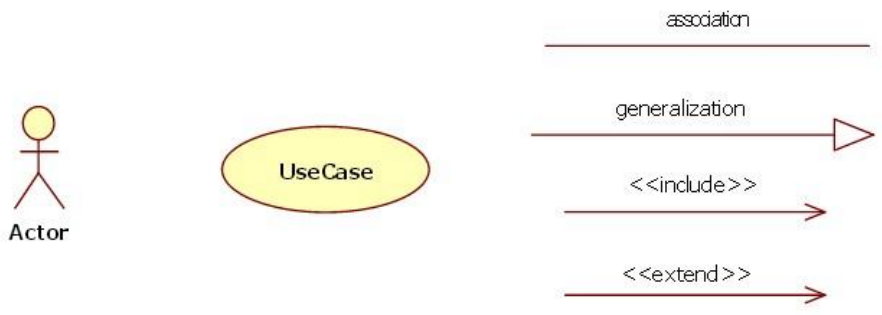
Διαγράμματα: Περιγράφουν τα περιεχόμενα μιας όψης. Υπάρχουν διαφορετικά διαγράμματα που χρησιμοποιούνται σε συνδυασμό για να δώσουν όλες τις όψεις του σύστημα.

Η UML ορίζει τα παρακάτω διαγράμματα:

- Διάγραμμα περιπτώσεων χρήσης (Use case diagram)
- Διάγραμμα κλάσεων (Class diagram)
- Διάγραμμα ακολουθίας (Sequence diagram)
- Διάγραμμα συνεργασίας (Collaboration diagram)
- Διάγραμμα καταστάσεων(Statechart diagram)
- Διάγραμμα δραστηριότητας(Activity diagram)
- Διάγραμμα συστατικών (Component diagram)
- Διάγραμμα ανάπτυξης(Deployment diagram)

Διάγραμμα περιπτώσεων χρήσης

Χρησιμοποιείται για την μοντελοποίηση της λειτουργικότητας ενός συστήματος, όπως αυτή γίνεται αντιληπτή από τον εξωτερικό χρήστη. Τα διαγράμματα αυτά διαμερίζουν τη λειτουργικότητα του συστήματος σε συναλλαγές που έχουν νόημα για τους χρήστες του συστήματος ή αλλιώς χειριστές (actors). Τα επιμέρους τμήματα της λειτουργικότητας ονομάζονται περιπτώσεις χρήσης (use cases). Το σύνολο των περιπτώσεων χρήσης συνιστούν τη συμπεριφορά του συστήματος. Τα βασικά διαγραμματικά στοιχεία του διαγράμματος περιπτώσεων χρήσης είναι το σύστημα, ο χειριστής, η περίπτωση χρήσης και οι σχέσεις μεταξύ τους.

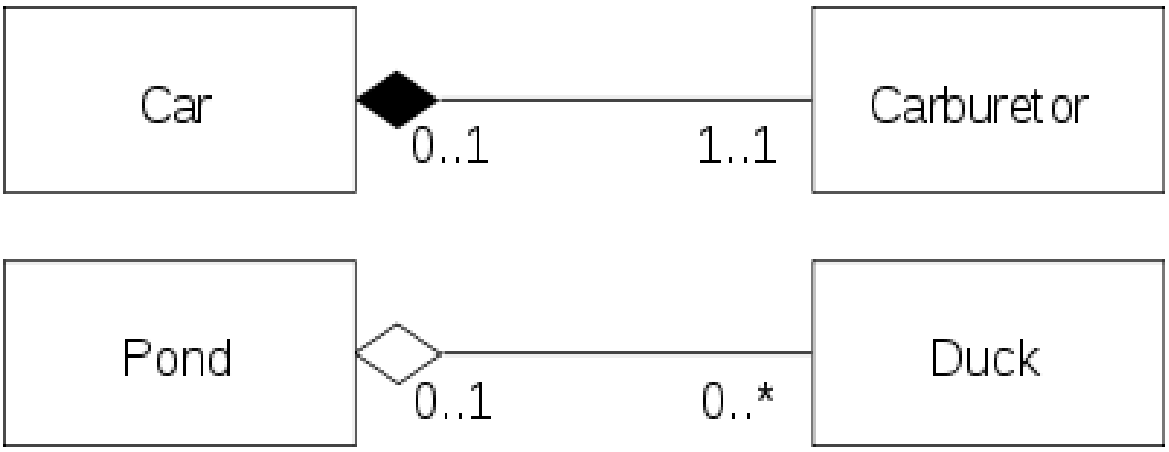


Σχήμα 4.2: Παράδειγμα Διαγράμματος περιπτώσεων χρήσης

Ο σημαντικότερος ρόλος του συγκεκριμένου διαγράμματος είναι ότι αποτελεί ένα μέσο επικοινωνίας μεταξύ πελατών και σχεδιαστών, όσον αφορά στη λειτουργικότητα του συστήματος. Η απλότητα των συμβολισμών το καθιστά ιδανικό για αυτό το σκοπό, παρέχοντας τη δυνατότητα εύκολης αντίληψης του συνόλου των λειτουργιών καθώς και εύκολης τροποποίησής τους.

Διάγραμμα κλάσεων

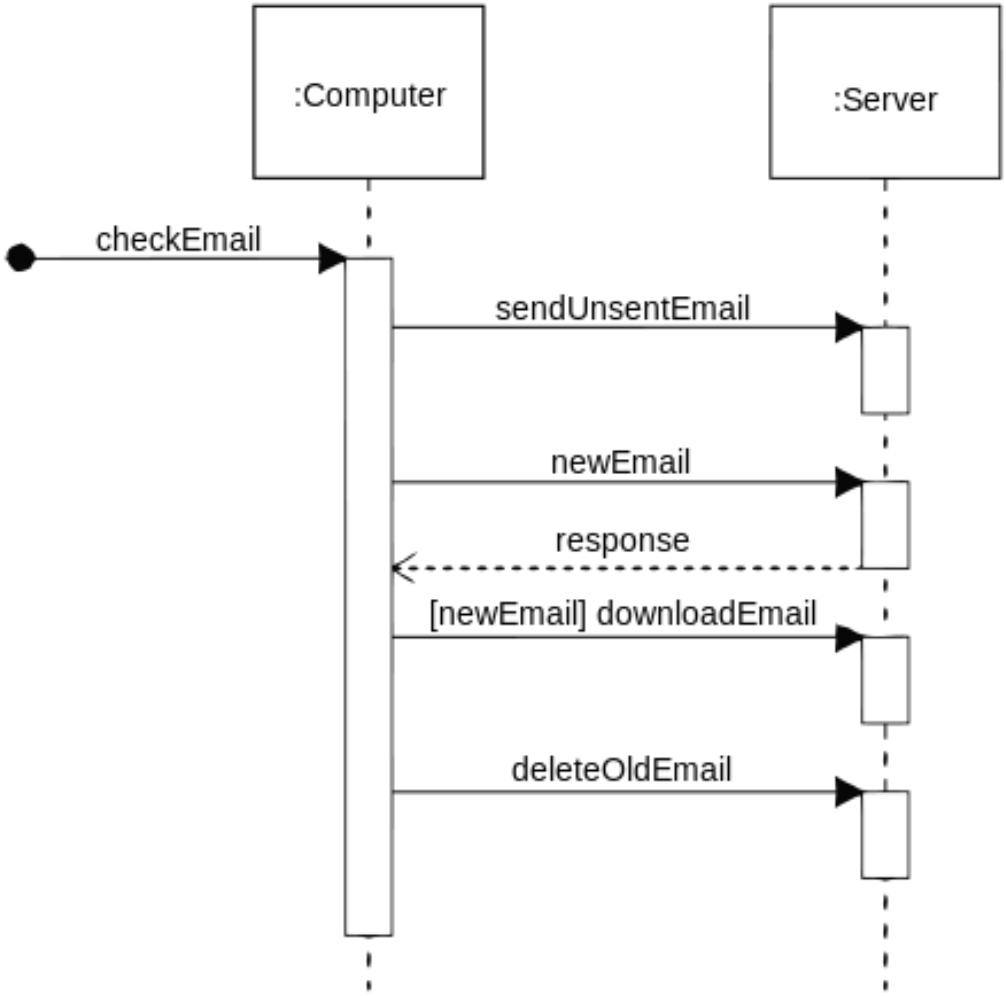
Το διάγραμμα κλάσεων αποτελείται από τις κλάσεις του συστήματος και τις μεταξύ τους συσχετίσεις, περιγράφοντας με αυτό τον τρόπο τη στατική δομή του συστήματος. Το διάγραμμα κλάσεων μπορεί να χρησιμοποιηθεί σε διάφορες φάσεις της ανάπτυξης του συστήματος. Στο αρχικό στάδιο της ανάλυσης απαιτήσεων οι κατασκευαστές αρχίζουν να αποκτούν γνώση για το πεδίο του προβλήματος του συστήματος. Αυτή η αρχική κατανόηση των εννοιών του πεδίου του προβλήματος καταγράφεται σε ένα διάγραμμα κλάσεων, το οποίο ονομάζεται μοντέλο του πεδίου προβλήματος (problem domain model). Στο μοντέλο αυτό καταγράφονται ως κλάσεις οι έννοιες του πεδίου του προβλήματος και οι μεταξύ τους συσχετίσεις. Έπειτα, στο στάδιο της ανάλυσης, με οδηγό το μοντέλο του πεδίου προβλήματος, κατασκευάζεται ένα διάγραμμα κλάσεων, το οποίο αναπαριστά τη βασική αρχιτεκτονική δομή του συστήματος. Σε αυτό το στάδιο οι κλάσεις πρέπει να επιδιώκουν την αναπαράσταση του συστήματος που μοντελοποιείται με την ελάχιστη δυνατή πληροφορία, χωρίς να επιχειρείται αναφορά σε θέματα υλοποίησης. Στη συνέχεια, μεταβαίνοντας στο στάδιο της σχεδίασης, η περιγραφή των κλάσεων συμπληρώνεται με τις λειτουργίες που υλοποιούν τη συμπεριφορά των αντικειμένων και με επιπρόσθετες ιδιότητες ή συσχετίσεις, που επιβάλλονται από το περιβάλλον υλοποίησης. Τέλος, κατά την υλοποίηση του συστήματος, είναι δυνατόν να επέλθουν τροποποιήσεις στη δομή των κλάσεων λόγω απαιτήσεων που σχετίζονται με απόκρυψη πληροφορίας, ορατότητα και άλλες μη λειτουργικές απαιτήσεις, όπως π.χ. απόδοση και ασφάλεια.



Σχήμα 4.3: Παράδειγμα Διαγράμματος κλάσεων

Διαγράμματα ακολουθίας

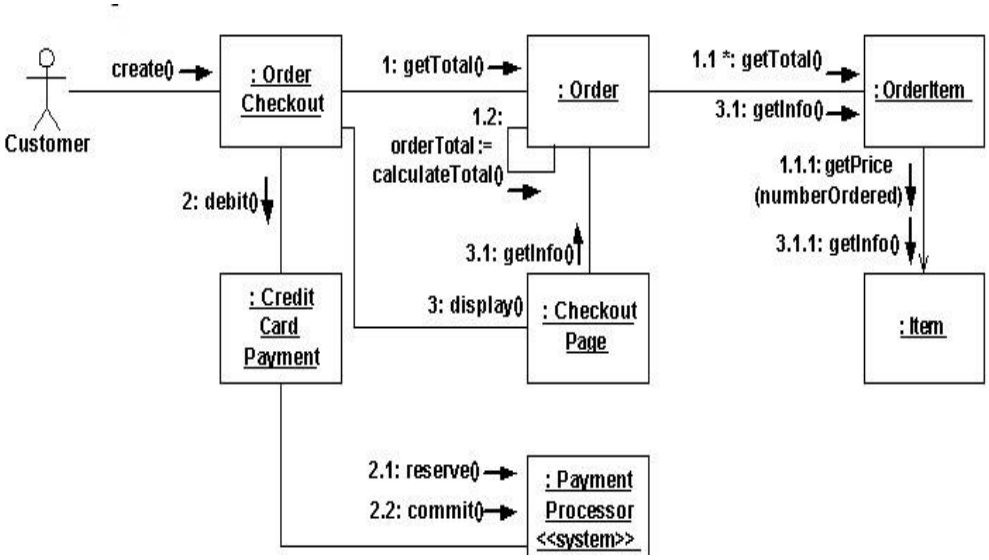
Παρουσιάζει την αλληλεπίδραση μεταξύ αντικειμένων σε δύο διαστάσεις. Η κάθετη διάσταση αντιστοιχεί στην κλίμακα του χρόνου, ενώ στην οριζόντια διάσταση συμβολίζονται τα ανεξάρτητα αντικείμενα. Τα αντικείμενα συμβολίζονται με παραλληλόγραμμα μέσα στα οποία μπορεί να σημειωθεί το όνομα του στιγμιότυπου του αντικειμένου που συμμετέχει στο σενάριο που απεικονίζεται και ακολουθεί μετά από άνω - κάτω τελεία το όνομα της κλάσης στην οποία ανήκει το αντικείμενο. Σε κάθε αντικείμενο αντιστοιχεί μια κάθετη γραμμή που ονομάζεται γραμμή ζωής (lifeline). Τα αντικείμενα ανταλλάσουν μηνύματα, τα οποία στην επίσημη ορολογία της UML ονομάζονται ερεθίσματα (stimuli). Ένα μήνυμα που αποστέλλεται μεταξύ των αντικειμένων συμβολίζεται ως ένα βέλος από τη γραμμή ζωής ενός αντικειμένου προς τη γραμμή ζωής ενός άλλου.



Σχήμα 4.4: Παράδειγμα Διαγράμματος ακολουθίας

Διάγραμμα συνεργασίας

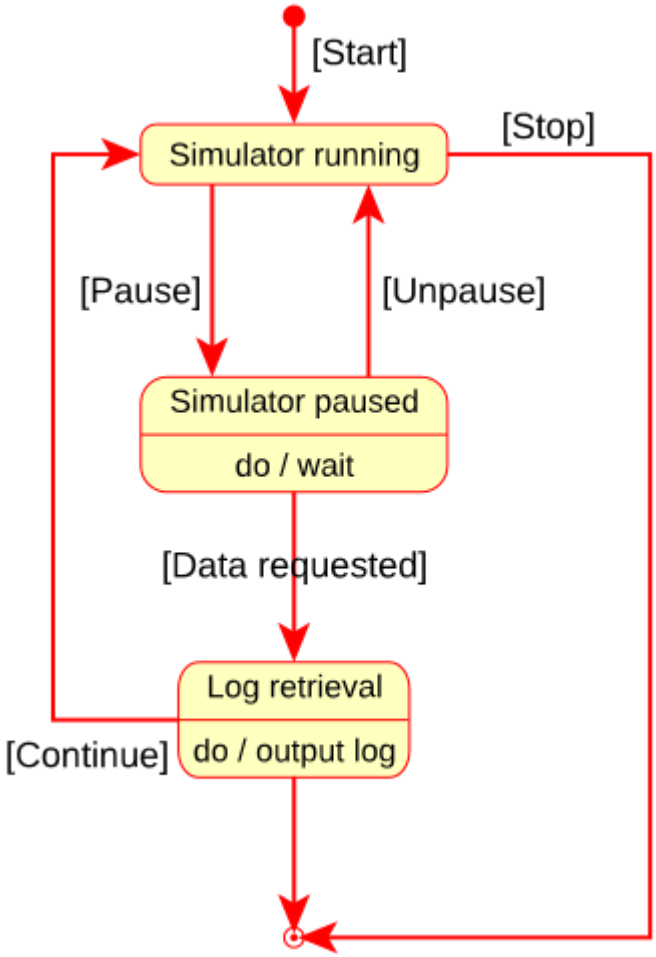
Απεικονίζονται τα συνεργαζόμενα αντικείμενα και οι συσχετίσεις μεταξύ τους. Επίσης, χρησιμοποιούνται για να παρουσιάσουν τις σχέσεις μεταξύ αντικειμένων. Πλησίον των συνδέσεων εμφανίζονται ως μικρότερες ακμές τα μηνύματα που αποστέλλονται. Για να απεικονιστεί η ακολουθία των μηνυμάτων που ανταλλάσσονται χρησιμοποιείται αρίθμηση των μηνυμάτων. Τα διαγράμματα ακολουθίας και συνεργασίας θεωρούνται συμπληρωματικά, καθώς περιέχουν την ίδια πληροφορία άλλα κάθε ένα δίνει μια διαφορετική οπτική γωνία.



Σχήμα 4.5: Παράδειγμα Διαγράμματος συνεργασίας

Διάγραμμα καταστάσεων

Χρησιμοποιείται για την περιγραφή της ροής του ελέγχου σε ένα σύστημα εστιάζοντας στις αλλαγές κατάστασης που λαμβάνουν χώρα σε ένα αντικείμενο. Πολύ συχνά οι προδιαγραφές ενός συστήματος μπορούν να καθοριστούν βάσει μιας μηχανής πεπερασμένων καταστάσεων (finite state machine) ή απλά μηχανής καταστάσεων. Συνήθως, μια μηχανή καταστάσεων περιγράφεται ως ένας γράφος όπου οι κόμβοι αντιστοιχούν σε καταστάσεις και τα βέλη υποδηλώνουν τη μετάβαση από μια κατάσταση σε μια άλλη. Εν γένει, οι μηχανές πεπερασμένων καταστάσεων είναι κατάλληλες για την περιγραφή σύγχρονων συστημάτων. Συνήθως, ένα διάγραμμα καταστάσεων είναι προσαρτημένο σε μια κλάση και αποτελεί ένα μοντέλο όλων των δυνατών κύκλων ζωής ενός αντικειμένου της κλάσης. Κάθε αντικείμενο αντιμετωπίζεται ως ξεχωριστή οντότητα που επικοινωνεί με το περιβάλλον ανιχνεύοντας γεγονότα και αντιδρώντας σε αυτά. Όταν λαμβάνει χώρα ένα ανιχνεύσιμο γεγονός, το αντικείμενο αποκρίνεται με βάση την κατάσταση στην οποία βρίσκεται. Η εκτέλεση μιας ενέργειας μπορεί να οδηγήσει σε μετάβαση σε μια άλλη κατάσταση.

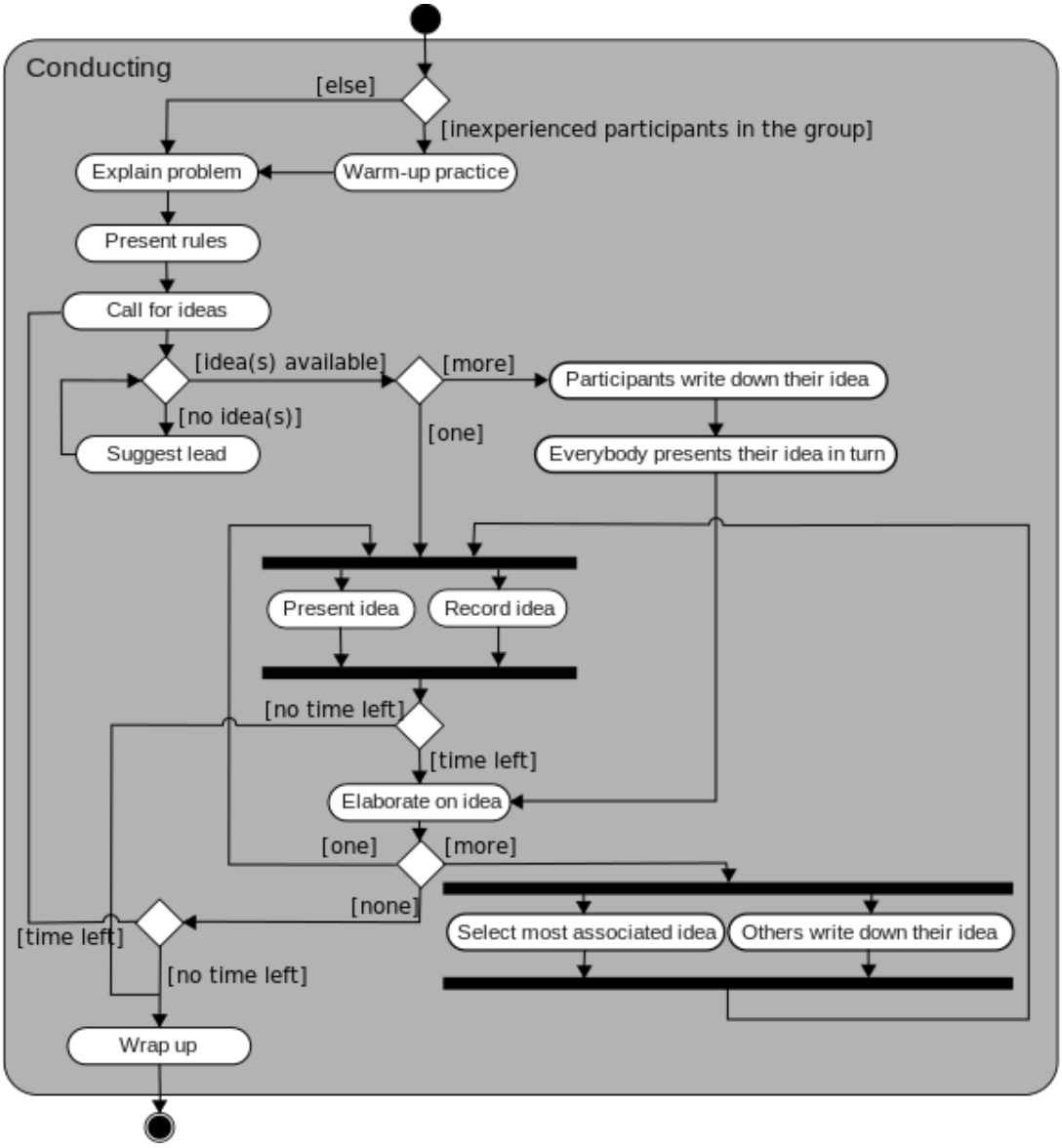


Σχήμα 4.6: Παράδειγμα Διαγράμματος καταστάσεων

Διάγραμμα δραστηριότητας

Ένας γράφος δραστηριότητας είναι μια ειδική μορφή μηχανής καταστάσεων που έχει ως στόχο τη μοντελοποίηση των υπολογισμών και της ροής της εργασίας. Οι καταστάσεις του γράφου δραστηριότητας αναπαριστούν τις καταστάσεις εκτέλεσης ενός υπολογισμού, όχι τις καταστάσεις των αντικειμένων που συμμετέχουν. Υπό κανονικές συνθήκες, ένας γράφος δραστηριότητας προϋποθέτει ότι οι υπολογισμοί πραγματοποιούνται χωρίς εξωτερικές γεγονοδηγούμενες διακοπές, αλλιώς είναι προτιμότερο ένα διάγραμμα καταστάσεων. Μια κατάσταση δραστηριότητας δεν αναμένει την εμφάνιση ενός γεγονότος, αλλά την ολοκλήρωση της διαδικασίας που περιγράφει, για τη μετάβαση στην επόμενη δραστηριότητα. Ένας γράφος δραστηριότητας μπορεί να περιλαμβάνει διακλάδωση της δραστηριότητας σε ταυτόχρονα νήματα

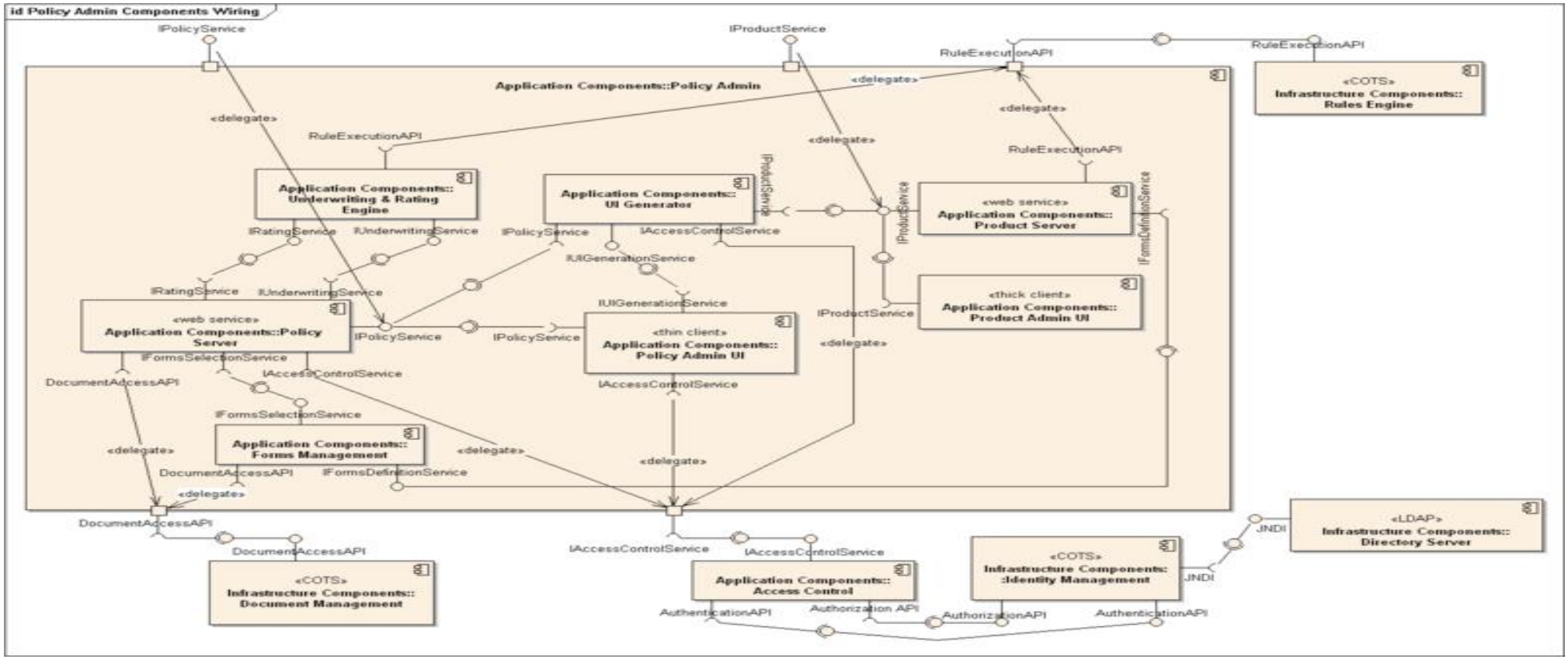
εκτέλεσης. Ένα διάγραμμα δραστηριότητας είναι ο συμβολισμός ενός γράφου δραστηριότητας στη UML. Περιγράφει τις συνθήκες που καθορίζουν ποιές δραστηριότητες θα εκτελεστούν σε κάθε σημείο του προγράμματος, ποιές δραστηριότητες μπορούν να λαμβάνουν χώρα παράλληλα καθώς και τυχόν επαναληπτικές δομές που περιλαμβάνονται.



Σχήμα 4.7: Παράδειγμα Διαγράμματος δραστηριότητας

Διάγραμμα συστατικών

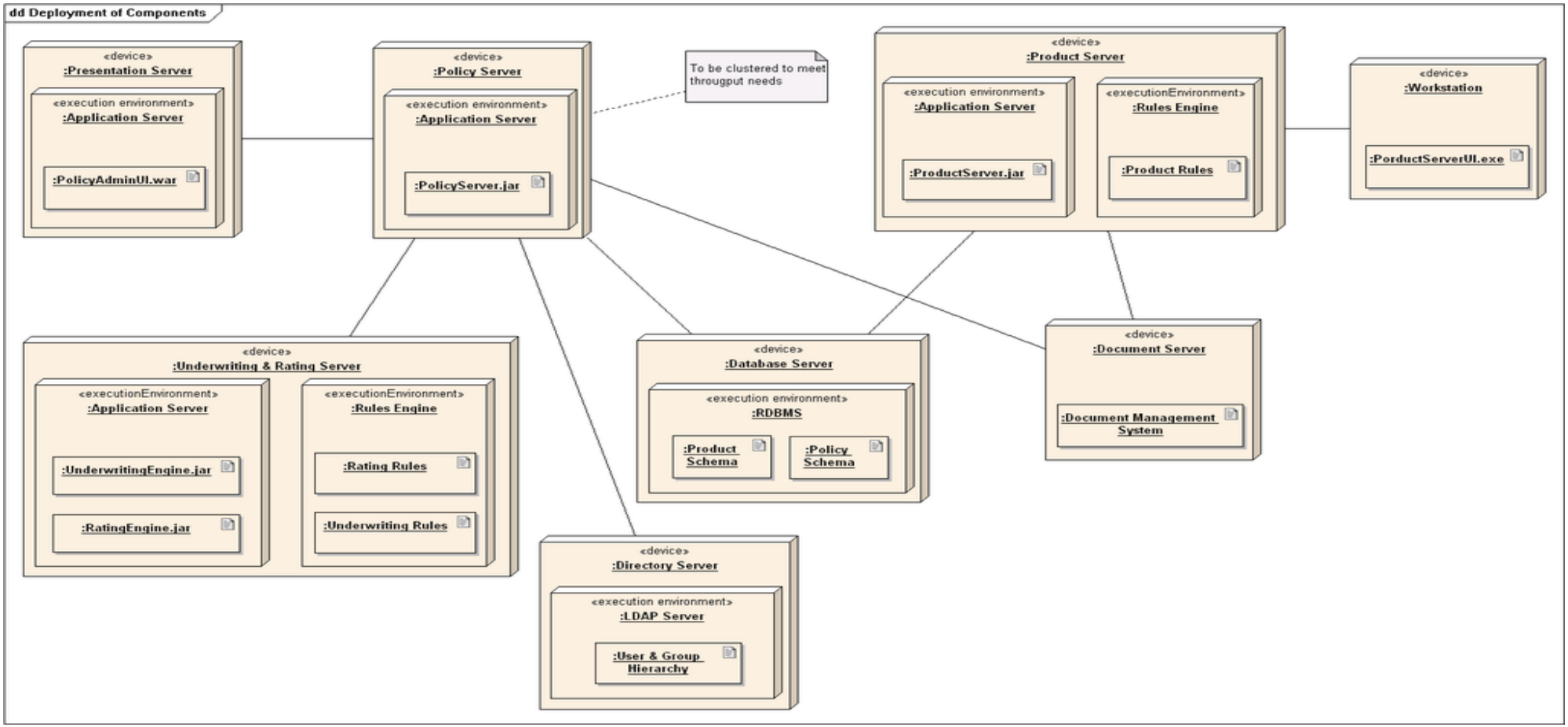
Ένα συστατικό (component) είναι μια φυσική μονάδα υλοποίησης κώδικα με σαφώς προσδιορισμένες διασυνδέσεις, η οποία αποτελεί επαναχρησιμοποιήσιμο τμήμα του συστήματος. Σε ένα αντικειμενοστραφές σύστημα ένα συστατικό ενσωματώνει την υλοποίηση μίας ή περισσοτέρων κλάσεων. Καλά σχεδιασμένα συστατικά δε θα πρέπει να εξαρτώνται άμεσα από άλλα συστατικά αλλά μόνο από διασυνδέσεις. Οι εξαρτήσεις έχουν επίδραση στη συντήρηση ενός συστήματος λογισμικού. Αν κάποιο συστατικό Α εξαρτάται από κάποιο άλλο συστατικό Β, οποιαδήποτε αλλαγή στο Β μπορεί να επηρεάσει το Α. Υπό την ίδια έννοια, οι εξαρτήσεις καθορίζουν την ευκολία επαναχρησιμοποίησης ενός συστατικού. Στην περίπτωση όπου ένα συστατικό στο σύστημα μπορεί να αντικατασταθεί από κάποιο άλλο, που υποστηρίζει τις ίδιες διασυνδέσεις, δεν επιφέρονται αλλαγές στο υπόλοιπο σύστημα. Ένα διάγραμμα συστατικών απεικονίζει το δίκτυο των εξαρτήσεων μεταξύ των συστατικών του συστήματος. Μια εξάρτηση μεταξύ δύο συστατικών υποδηλώνει ότι για την ορθή λειτουργία του ενός συστατικού απαιτείται η ύπαρξη ενός άλλου. Το συστατικό συμβολίζεται ως ένα ορθογώνιο ενώ οι εξαρτήσεις συμβολίζονται ως διακεκομμένες ακμές με κατεύθυνση από το εξαρτώμενο συστατικό προς αυτό που παρέχει τις λειτουργίες. Στα διαγράμματα συστατικών υπάρχει η δυνατότητα απεικόνισης λογικών τμημάτων ενός συστήματος με τη χρήση των πακέτων. Ένα πακέτο περιλαμβάνει ένα σύνολο από συστατικά τα οποία έχουν λειτουργική συνάφεια. Ο συμβολισμός ενός πακέτου στη UML επιτυγχάνεται με τη χρήση ενός φακέλου. [12][13]



Σχήμα 4.8: Παράδειγμα Διαγράμματος συστατικών

Διάγραμμα ανάπτυξης

Το διάγραμμα ανάπτυξης περιγράφει την οργάνωση των επεξεργαστικών πόρων (κόμβων) του συστήματος και την αντιστοίχιση των συστατικών λογισμικού στους κόμβους αυτούς. Κατά κύριο λόγο απεικονίζουν την τοπολογία του υλικού επί του οποίου εκτελείται το σύστημα λογισμικού. Ένας κόμβος (node) είναι ένα φυσικό αντικείμενο που αναπαριστά έναν υπολογιστικό πόρο, ο οποίος στη γενική περίπτωση έχει τουλάχιστον μνήμη και δυνατότητα επεξεργασίας. Οι κόμβοι μπορούν να αντιστοιχίζονται σε στερεότυπα ώστε να διακρίνονται διαφορετικά είδη πόρων, όπως κεντρικές μονάδες επεξεργασίας, μνήμες, εξυπηρετητές για βάσεις δεδομένων και συσκευές διασύνδεσης με άλλα συστήματα. Ένας κόμβος συμβολίζεται ως ένας τρισδιάστατος κύβος με το όνομα του κόμβου και ενδεχομένως ένα στερεότυπο που εκφράζει την κατηγορία στην οποία ανήκει. Η τοπολογία του συστήματος απεικονίζεται συνδέοντας τους κόμβους με γραμμές συσχέτισης, οι οποίες μπορούν να υποδηλώνουν ρητά το πρωτόκολλο επικοινωνίας ή να χαρακτηρίζουν το σύστημα μεταφοράς δεδομένων με κάποιο τρόπο. Το διάγραμμα ανάπτυξης χρησιμοποιείται από μηχανικούς συστημάτων για τη μοντελοποίηση ενσωματωμένων συστημάτων, συστημάτων πελάτη/εξυπηρετητή, όπου υπάρχει σαφής διαχωρισμός μεταξύ των εφαρμογών που εκτελούνται στο σύστημα του πελάτη και των μονίμων δεδομένων που φιλοξενούνται στον εξυπηρετητή, καθώς και πλήρως κατανεμημένων συστημάτων που περιλαμβάνουν συνήθως πολλαπλά επίπεδα εξυπηρετητών και συνήθως φιλοξενούν πολλαπλές εκδόσεις των συστατικών λογισμικού στους κόμβους τους.



Σχήμα 4.9: Παράδειγμα Διαγράμματος ανάπτυξης

4.3 Η UML στην ανάπτυξη ενσωματωμένων συστημάτων

Η πλούσια γραφική σημειολογία της UML σε συνδυασμό με τις δυνατότητες μοντελοποίησης που παρέχει, επιτρέπει τον προσδιορισμό και την οπτικοποίηση της δομής και της συμπεριφοράς του συστήματος σε πολλαπλά επίπεδα αφαίρεσης. Τα στοιχεία αυτά την καθιστούν ικανή να χρησιμοποιηθεί στην ανάπτυξη ενσωματωμένων συστημάτων για τη μοντελοποίηση και την τεκμηρίωσή τους.

Η UML παρέχει ένα πλούσιο σύνολο στοιχείων μοντελοποίησης που δίνουν τη δυνατότητα να μοντελοποιηθούν τα πιο σημαντικά χαρακτηριστικά των κατανεμημένων συστημάτων πραγματικού χρόνου, όπως η απόδοση (χρησιμοποιώντας tagged attributes ή την OCL [10]), οι φυσικοί πόροι (χρησιμοποιώντας διαγράμματα ανάπτυξης) και ο χρόνος (χρησιμοποιώντας classifiers και tagged attributes). Ωστόσο, υπάρχουν επιπλέον παράγοντες που πρέπει να ληφθούν υπόψη. Η μοντελοποίηση εφαρμογών συγκεκριμένου πεδίου είναι ευκολότερη αν χρησιμοποιείται μια πιο εξειδικευμένη (domain-specific) σημειολογία, η οποία αναπαριστά τα βασικά στοιχεία και πρότυπα (patterns) του συγκεκριμένου πεδίου.

Επιπλέον, απαιτείται μια σαφώς ορισμένη σημασιολογία για το συγκεκριμένο πεδίο, προκειμένου να αποφευχθεί η διαφορετική ερμηνεία ίδιων μοντέλων και να υποστηριχθούν τα εργαλεία ανάλυσης.

Τέλος, πολλαπλά διαγράμματα μπορούν να χρησιμοποιηθούν για να περιγράψουν διαφορετικές όψεις του συστήματος. Η δυνατότητα περιγραφής του ίδιου αντικειμένου από διαφορετικές οπτικές γωνίες διευκολύνει μεν την τεκμηρίωση του συστήματος, αλλά μπορεί να καταλήξει σε ασυνέπειες μεταξύ των διαγραμμάτων, όταν η χρήση της UML δε συνδέεται με μια αυστηρώς ορισμένη μέθοδο σχεδιασμού.

Για τους παραπάνω λόγους, για την ανάπτυξη εφαρμογών ενσωματωμένων συστημάτων, αλλά και γενικότερα εφαρμογών συγκεκριμένου πεδίου, απαιτείται [11]

- μια γλώσσα συγκεκριμένου πεδίου, η οποία ονομάζεται profile

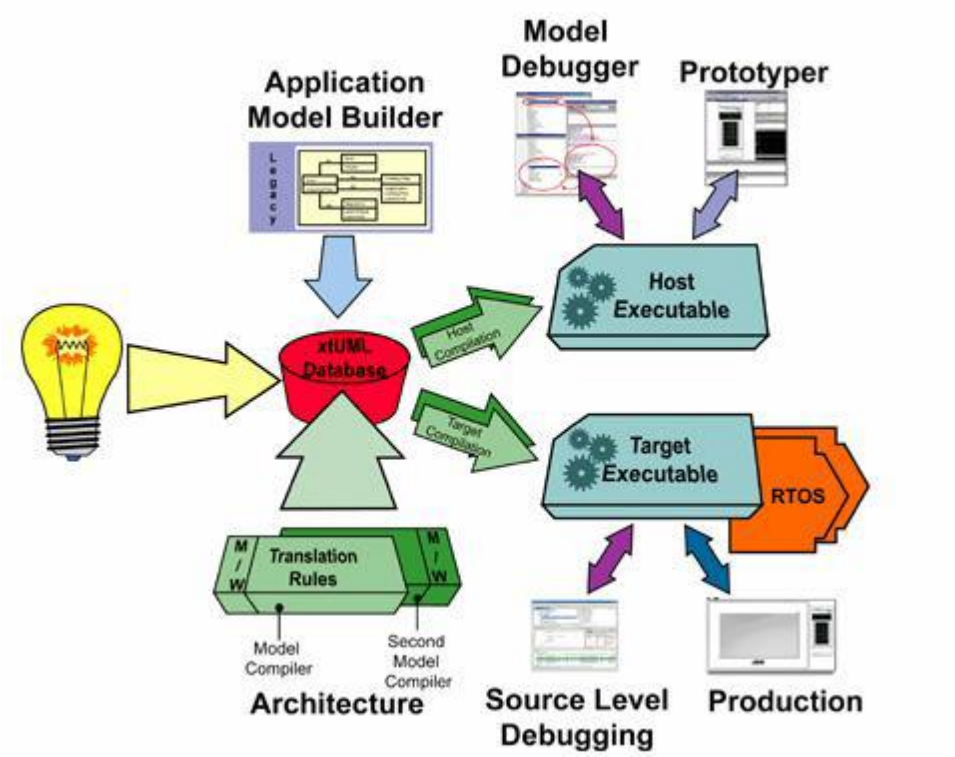
και βασίζεται στη βασική υποδομή της UML και περιλαμβάνει domainspecific building blocks (που ορίζονται χρησιμοποιώντας την έννοια του στερεοτύπου)

- μια μεθοδολογία, η οποία ορίζει το πώς και πότε η profile σημειολογία πρέπει να χρησιμοποιείται.

Ένα UML profile για τα ενσωματωμένα συστήματα πρέπει να περιλαμβάνει εξειδικευμένη σημειολογία για την αναπαράσταση της δομής και της συμπεριφοράς των platform resources και των υπηρεσιών που παρέχουν, με ιδιαίτερη έμφαση σε θέματα απόδοσης και κόστους. Πρέπει επίσης να παρέχει τη δυνατότητα οπτικοποίησης πολλαπλών εναλλακτικών υλοποιήσεων για να διευκολύνει τη γρήγορη σύγκρισή τους. [11]

Το UML Platform profile είναι μια γραφική γλώσσα για την τεκμηρίωση των πλατφόρμων των ενσωματωμένων συστημάτων. Περιλαμβάνει domain-specific classifiers και σχέσεις εξειδικευμένες με στερεότυπα, που συμπληρώνουν τη σημειολογία που είναι ορισμένη στη UML και στο UML Profile for Schedulability, Performance and Time Specification (ή αλλιώς Real-Time UML Profile) [12]. Έτσι επιτυγχάνεται η μοντελοποίηση της δομής και της συμπεριφοράς των ενσωματωμένων συστημάτων και η αναπαράσταση της σχέσης μεταξύ των platforms σε διαφορετικά επίπεδα αφαίρεσης.

Στη συνέχεια περιγράφεται μια προσέγγιση για την ανάπτυξη συστημάτων βασισμένη στα executable models [13], η οποία δείχνει το πώς η UML μπορεί να χρησιμοποιηθεί για την ανάπτυξη ενσωματωμένων συστημάτων. Στην εικόνα 4.1 παρακάτω φαίνεται η διαδικασία ανάπτυξης του συστήματος με βάση αυτή την προσέγγιση.

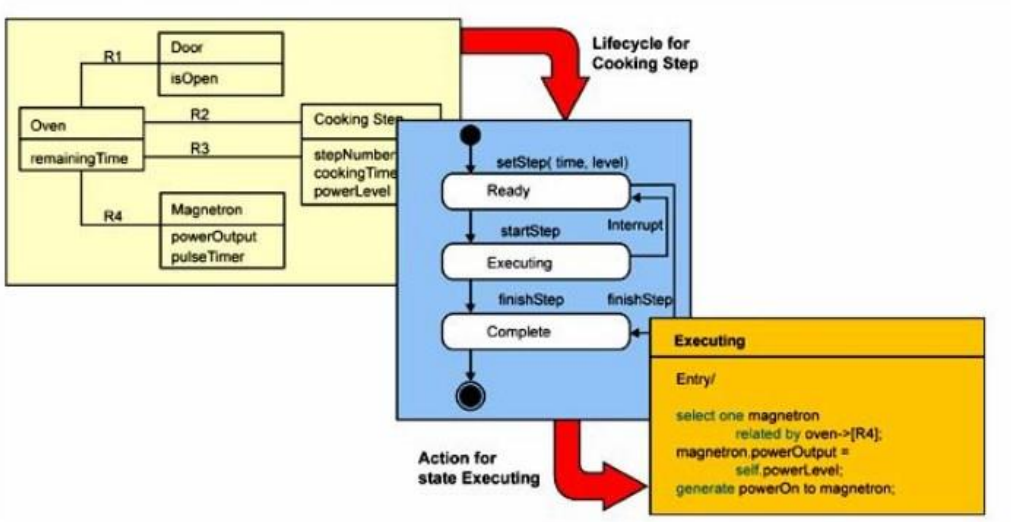


Εικόνα 4.1: Η συνολική διαδικασία ανάπτυξης [13]

Το πρώτο βήμα της διαδικασίας ανάπτυξης (το οποίο φαίνεται στην εικόνα 4.1 ως μια λάμπα) αναπαριστά το concept πίσω από την ανάπτυξη ενός ενσωματωμένου συστήματος. Το concept αυτό διαμορφώνεται με βάση την επιθυμητή λειτουργικότητα, η οποία εκφράζεται από τη συμμετοχή της αγοράς, και τη διαθεσιμότητα, η οποία εκφράζεται από τη συμμετοχή ειδικών υλικού και λογισμικού. Ο καθορισμός του concept πίσω από ένα προϊόν είναι συχνά ένας συμβιβασμός και μια αντιστάθμιση ανάμεσα σε πιθανά χαρακτηριστικά που είναι επιθυμητά από την αγορά και τις δυνατότητες που μπορούν να προσφερθούν από μια δεδομένη αρχιτεκτονική υλικού-λογισμικού. Σε αυτό το σημείο η διαδικασία ανάπτυξης χωρίζεται στον τομέα της εφαρμογής και αυτόν της αρχιτεκτονικής.

Πρώτα περιγράφεται ο τομέας της εφαρμογής.

Τα application models είναι εκτελέσιμα και αποτελούνται από τρία πρωτεύοντα διαγράμματα, όπως φαίνεται στην εικόνα 4.2 παρακάτω:



Εικόνα 4.2: Ένα παράδειγμα application model [13]

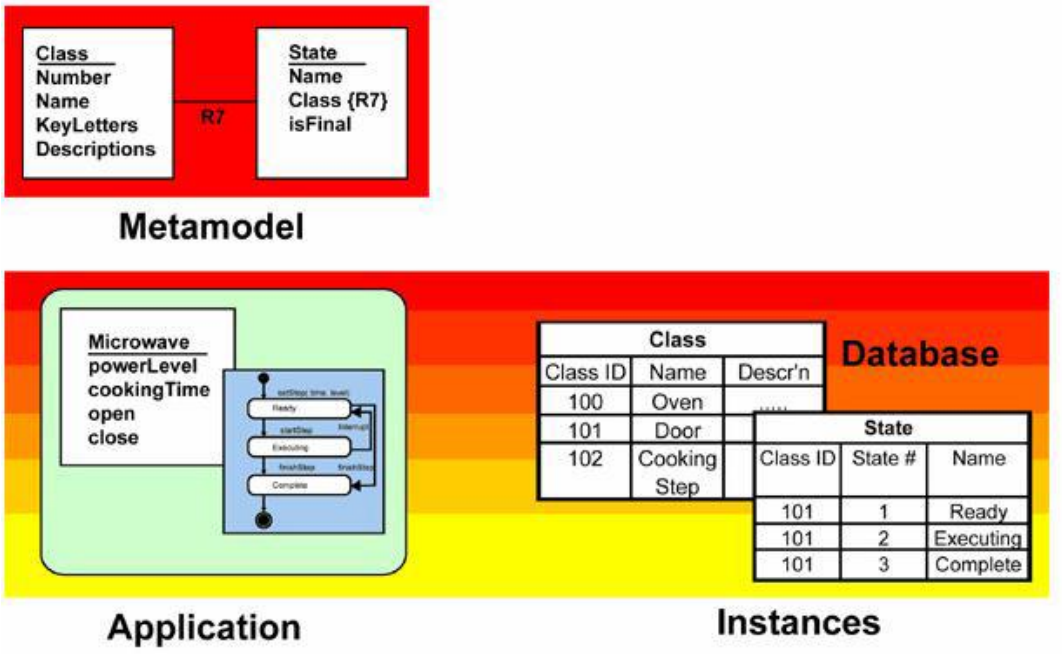
Το πρώτο μέρος είναι η δήλωση των εννοιολογικών αντικειμένων του συγκεκριμένου συστήματος, η οποία γίνεται με τη βοήθεια του διαγράμματος κλάσεων της UML. Πρέπει να σημειωθεί ότι αυτή η χρήση του διαγράμματος κλάσεων δεν έχει καμία σχέση με τη δομή του λογισμικού. Σε ένα translatable model, ένα διάγραμμα κλάσεων αναπαριστά μόνο την εννοιολογική ομαδοποίηση πληροφορίας και συμπεριφοράς [13].

Διαγράμματα καταστάσεων μπορούν να κατασκευαστούν για τα στιγμιότυπα των κλάσεων μεμονωμένα, ή και συνολικά για ένα σύνολο στιγμιοτύπων. Έτσι, μέσα από τη μηχανή καταστάσεων μοντελοποιείται όλη η δυναμική συμπεριφορά του application model.

Ένα εκτελέσιμο μοντέλο είναι ασήμαντο χωρίς την ύπαρξη κανόνων, οι οποίοι ορίζουν την εκτέλεση. Στην executable UML, κάθε μηχανή καταστάσεων εκτελείται ταυτόχρονα με όλες τις άλλες. Οι μηχανές αυτές επικοινωνούν στέλνοντας σήματα, τα οποία ορίζουν σχέσεις προτεραιότητας ανάμεσα σε ακολουθίες ενεργειών. Όπως και με τα διαγράμματα κλάσεων, το διάγραμμα καταστάσεων αυτό δεν έχει να κάνει με την υλοποίηση, μιας και τα σήματα μπορούν να υλοποιηθούν με οποιοδήποτε τρόπο ικανοποιεί την επιθυμητή προτεραιότητα των ενεργειών. Οι ενέργειες (actions) είναι το

τελευταίο από τα πρωτεύοντα κατασκευάσματα της executable UML. Οι ενέργειες είναι επίσης ανεξάρτητες της υλοποίησης, με την έννοια ότι δεν υποδηλώνουν τη δόμηση του λογισμικού.

Η αρχιτεκτονική είναι ένα σύνολο κανόνων που ορίζουν το πώς μπορεί να μετατραπεί μια εφαρμογή γραμμένη σε executable UML σε υλοποίηση. Ο ορισμός της αρχιτεκτονικής μπορεί να γίνει σε οποιοδήποτε χρόνο συγκριτικά με τον ορισμό της εφαρμογής. Ένα συνεπές σύνολο κανόνων που προορίζονται για μια αρχιτεκτονική καλείται model compiler. Όταν παράγεται ένα executable UML model, η σημασία του μοντέλου αποθηκεύεται σε μια βάση δεδομένων. Το διάγραμμα που φαίνεται στην εικόνα 4.2 παραπάνω δημιουργεί στιγμιότυπα στη βάση δεδομένων όπως φαίνεται στην εικόνα 4.3 παρακάτω.



Εικόνα 4.3: Μεταμοντέλο [13]

Η δομή της βάσης δεδομένων δημιουργείται από τα elements της executable UML. Έτσι, υπάρχει ένας πίνακας Class για την αποθήκευση της

πληροφορίας σχετικά με τις κλάσεις, και ένας πίνακας State για την αποθήκευση πληροφορίας σχετικά με τις καταστάσεις. Η δομή αυτής της βάσης δεδομένων καλείται μεταμοντέλο (metamodel).

Ο model compiler εφαρμόζει τους κανόνες αρχιτεκτονικής στην εφαρμογή, έτσι όπως είναι αποθηκευμένοι στη βάση δεδομένων. Οι κανόνες διατρέχουν τη βάση σύμφωνα με την επιθυμητή έξοδο και παράγουν κείμενο στη γλώσσα επιλογής, το οποίο μπορεί να μεταφραστεί. Σε αυτό το σημείο είναι που γίνονται οι απαραίτητες διορθώσεις. Αν ο κώδικας είναι πολύ αργός ή πολύ γρήγορος σχετικά με την επιθυμητή απόδοση, μπορεί εύκολα να εντοπιστεί η αιτία και να γραφεί ο κανόνας που θα τον διορθώσει.

Αφού παραχθεί το κείμενο από τον model compiler μπορεί αν μεταφραστεί σε κάποια γλώσσα επιλογής C, C++ ή Assembly. Αυτό αναπαριστά τη μετάβαση από τις αφηρημένες model-driven αρχιτεκτονικές στο περιβάλλον ανάπτυξης λογισμικού ενσωματωμένων συστημάτων. Ένας open model compiler κάνει τη διαδικασία ελεγχόμενη. Ένας περαιτέρω έλεγχος του κώδικα που παρήχθη μπορεί να γίνει με τη χρήση ενός debugger. Με τη βοήθεια ενός prototyper μπορεί να γίνει η εξομοίωση του πλήρους συστήματος, προτού συγκεντρωθεί το απαραίτητο hardware.

Συνοψίζοντας, η προσέγγιση αυτή δίνει τη δυνατότητα στο σχεδιαστή να εξακριβώσει αν ο σχεδιασμός επιλύει τα προβλήματα που πρέπει να επιλυθούν τρέχοντας ένα executable UML model και κάνοντας debugging του μοντέλου, προτού παραχθεί κώδικας. Αν υπάρχει κάποιο πρόβλημα με τη συμπεριφορά της εφαρμογής, τότε μεταβάλλεται αντίστοιχα και το μοντέλο. Αν υπάρχει κάποιο πρόβλημα με την απόδοση, τότε οι κανόνες προσαρμόζονται. Αυτός ο διαχωρισμός της εφαρμογής από την αρχιτεκτονική οδηγεί σε ένα περισσότερο συντηρήσιμο και αποδοτικό σύστημα. [13]

ΚΕΦΑΛΑΙΟ ΠΕΜΠΤΟ

Στο κεφάλαιο αυτό θα παρουσιάσουμε κάποια παραδείγματα ενσωματωμένης υπολογιστικής, που παρουσιάζονται στην καθημερινότητά μας.

5.1 Το σύστημα ελέγχου φρένων και σταθερότητας της BMW 850i

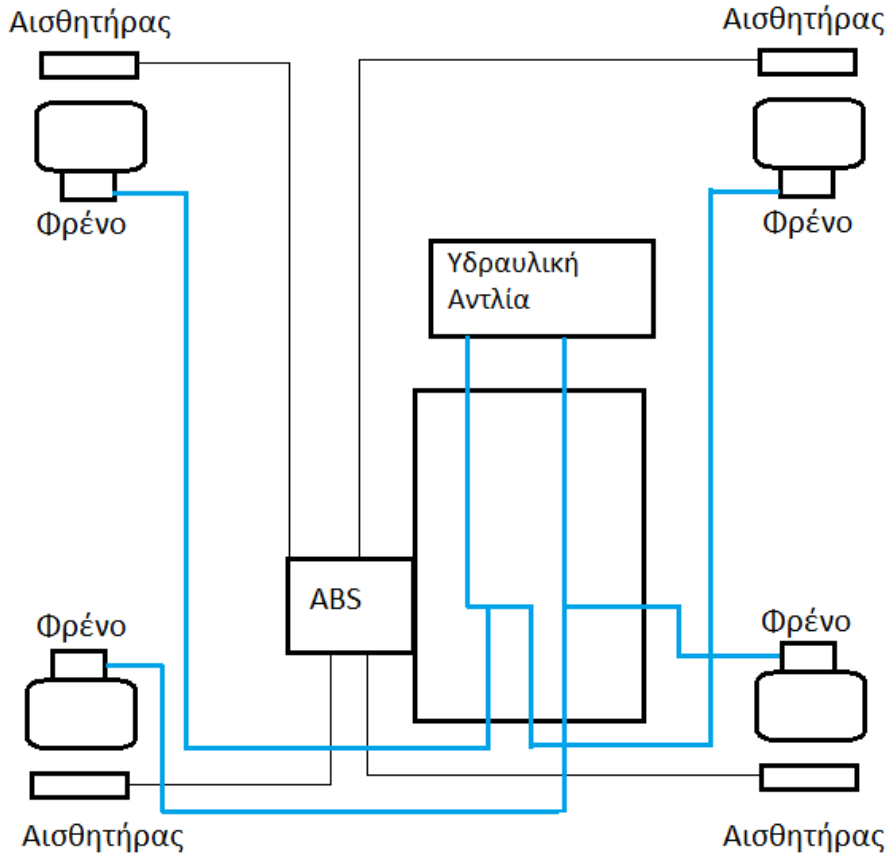
Το μοντέλο αυτό διαθέτει ένα σύστημα ελέγχου για τους τροχούς του αυτοκινήτου, το οποίο δεν είναι απλό. Το σύστημα αντιμπλοκαρίσματος των τροχών (ABS) μειώνει την ολίσθηση με την αυτόματη μεταβολή της πίεσης των φρένων. Το σύστημα ελέγχου σταθερότητας (ASC+T) παρεμβαίνει στη μηχανή κατά τη διάρκεια των ελιγμών του αυτοκινήτου, ώστε να υπάρχει η βέλτιστη σταθερότητα. Με λίγα λόγια, τα συστήματα αυτά διαχειρίζονται τα κρίσιμα συστήματα του αυτοκινήτου.

Αρχικά, θα εξετάσουμε το ABS και τη χρήση του. Σκοπός του συστήματος αυτού, είναι να απελευθερώνει προσωρινά το φρένο σε έναν τροχό όταν αυτός περιστρέφεται πολύ αργά. Όταν ο τροχός σταματά να περιστρέφεται, τότε το αυτοκίνητο ολισθαίνει και δυσκολεύεται ο οδηγός να το ελέγξει. Το ABS χρησιμοποιεί αισθητήρες σε κάθε τροχό για τη μέτρηση της ταχύτητάς του. Οι ταχύτητες των τροχών χρησιμοποιούνται από το σύστημα για τον υπολογισμό της μεταβολής της πίεσης του υδραυλικού υγρού για την αποφυγή της ολίσθησης των τροχών.

Το σύστημα ASC+T ελέγχει την ισχύ του κινητήρα και των φρένων ώστε να βελτιστοποιηθεί η σταθερότητα του αυτοκινήτου. Ελέγχει, επίσης, τη ρυθμιστική βαλβίδα, τα διαφορικά φρένα, τον χρονισμό ανάφλεξης και την αλλαγή των ταχυτήτων. Το σύστημα αυτό μπορεί να απενεργοποιηθεί από τον οδηγό σε ειδικές περιπτώσεις, όταν π.χ. το αυτοκίνητο κινείται με αλυσίδες

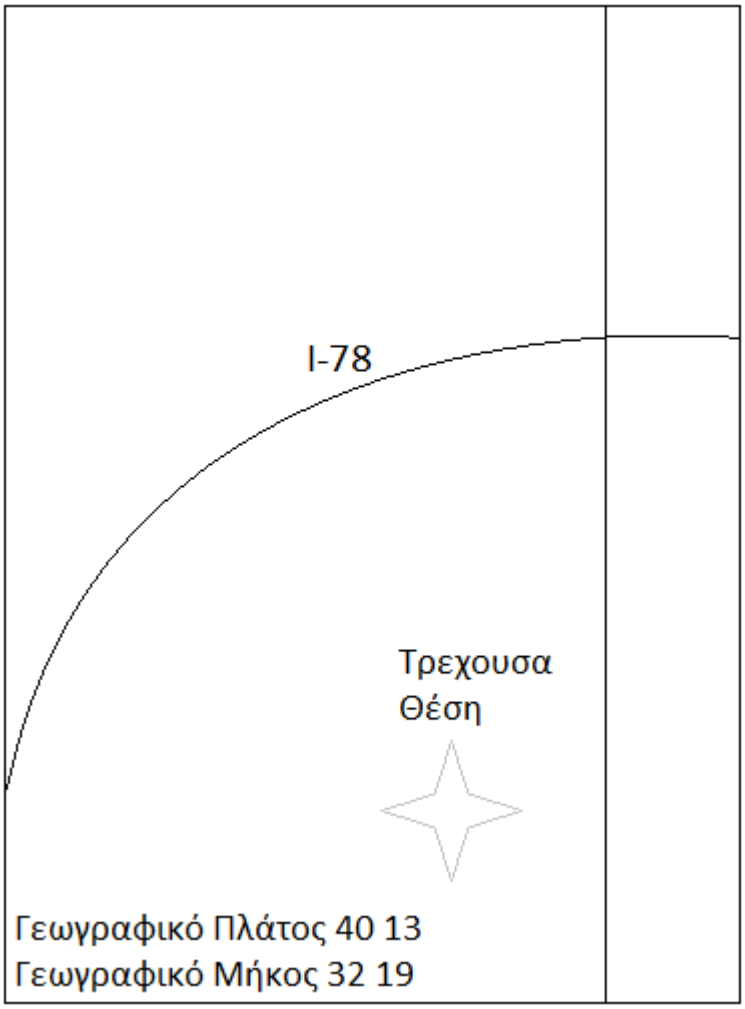
Τα δύο αυτά συστήματα πρέπει να επικοινωνούν μεταξύ τους επειδή το ASC+T αλληλεπιδρά με το σύστημα των φρένων. Οι μονάδες διαχείρισης και ελέγχου περιλαμβάνουν την ηλεκτρονικά ελεγχόμενη ρυθμιστική βαλβίδα, την

ψηφιακή διαχείριση της μηχανής και τον ηλεκτρονικό έλεγχο μετάδοσης. Παρακάτω φαίνονται οι συνδέσεις των δύο συστημάτων με τα φρένα του αυτοκινήτου.



Σχήμα 5.1: Σύνδεση μεταξύ ABS και ACS+T

5.2 Ανάλυση απαιτήσεων ενός κινούμενου χάρτη παγκόσμιου συστήματος εντοπισμού θέσης (GPS)



Σχήμα 5.2: Η οθόνη ενός κινούμενου χάρτη

Το σχήμα 5.2 δείχνει πως μοιάζει η οθόνη ενός κινούμενου χάρτη. Ο κινούμενος χάρτης είναι μία συσκευή η οποία δείχνει στον χρήστη έναν χάρτη της περιοχής γύρω από την τρέχουσα θέση του, και αλλάζει καθώς ο χρήστης και η συσκευη μετακινούνται.

Οι απαιτήσεις που έχουμε για τον κινούμενο χάρτη είναι οι εξής:

1. **Λειτουργικότητα**: Το σύστημα πρέπει να εμφανίζει τους κύριους δρόμους και άλλα σημεία αναφοράς διαθέσιμα σε τυποποιημένες τοπογραφικές βάσεις δεδομένων.

2. **Διασύνδεση με το χρήστη**: Η οθόνη πρέπει να έχει ανάλυση 400x600 pixels και να έχει ένα σύστημα μενού που να εμφανίζεται στην οθόνη όταν πατηθεί κάποιο κουμπί ώστε να μπορεί ο χρήστης να ελέγξει το σύστημα.
3. **Απόδοση**: Ο χάρτης θα πρέπει κατά την εκκίνηση να εμφανίζεται το πολύ σε 1 sec. Το σύστημα θα πρέπει να επιβεβαιώνει τη θέση του χρήστη και να εμφανίζει τον αντίστοιχο χάρτη μέσα σε 15 sec.
4. **Κόστος**: Το κόστος δε θα πρέπει να ξεπερνάει τα 500€
5. **Φυσικό μέγεθος και βάρος**: Η συσκευή θα πρέπει να χωράει στην παλάμη του χεριού.
6. **Κατανάλωση ισχύος**: Η συσκευή θα πρέπει να μπορεί να λειτουργεί για 8 ώρες με 4 μπαταρίες ΑΑ.

Οι απαιτήσεις αυτές θα πρέπει να «μεταφραστούν» σε κάτι που να μπορεί να χρησιμοποιηθεί από τους σχεδιαστές. Η διατήρηση ενός αρχείου με της απαιτήσεις του πελάτη μπορεί να βοηθήσει την επίλυση αποριών σχετικά με την προδιαγραφή οι οποίες ενδέχεται να προκύψουν αργότερα στη σχεδίαση. Με βάση αυτά, καταγράφουμε τις απαιτήσεις του συστήματος με όρους μηχανικής σε μία φόρμα, η οποία φαίνεται παρακάτω:

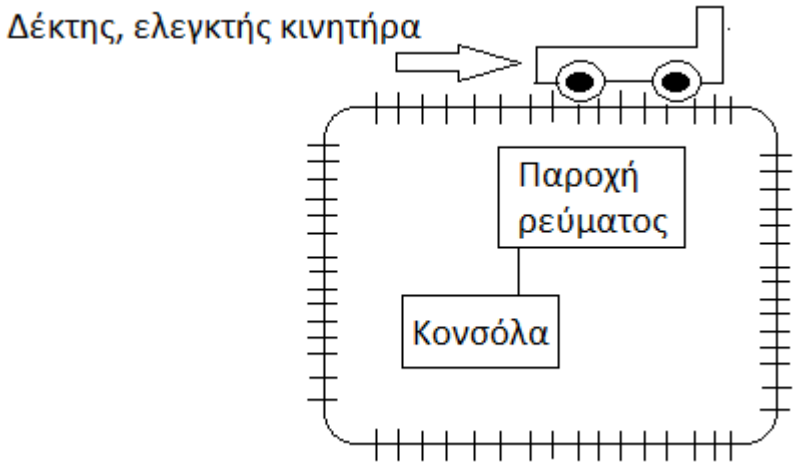
Πίνακας 5.1: Φόρμα απαιτήσεων συστήματος GPS

Όνομα	Κινούμενος χάρτης παγκοσμίου συστήματος εντοπισμού θέσης (Global Positioning System – GPS)
Σκοπός	Καταναλωτικής κατηγορίας κινούμενος χάρτης για χρήση κατά την οδήγηση
Είσοδοι	Ένα πλήκτρο ισχύος, δύο πλήκτρα ελέγχου

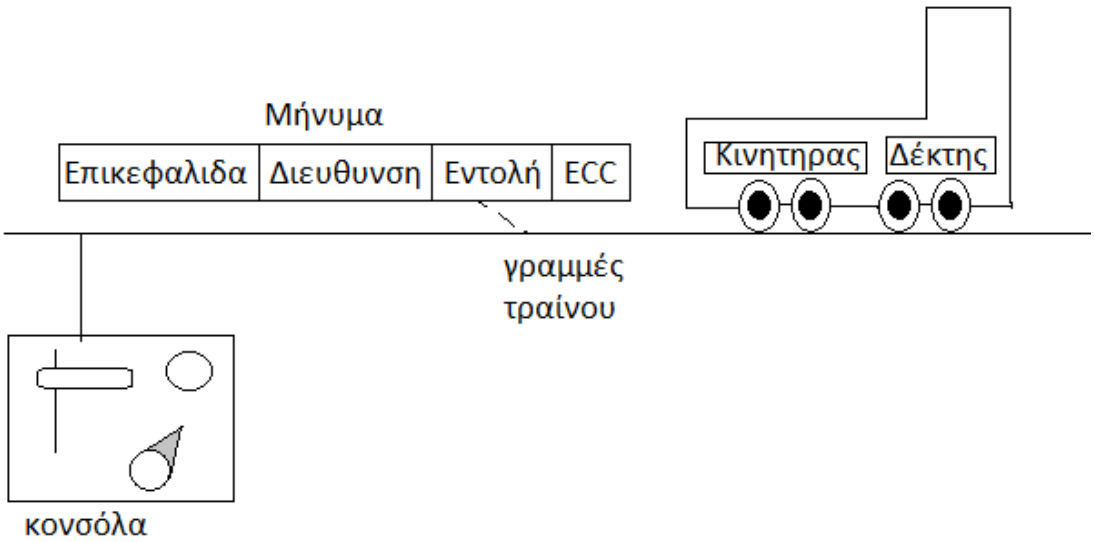
Έξοδοι	Φωτιζόμενη οθόνη υγρών κρυστάλλων (LCD) με ανάλυση 400x600
Λειτουργίες	Χρησιμοποιεί GPS πέντε δεκτών. Δείχνει πάντα το τρέχον γεωγραφικό μήκος και το τρέχον γεωγραφικό πλάτος.
Απόδοση	Ενημερώνει την οθόνη κάθε 0,25 sec. σε κίνηση
Κόστος κατασκευής	100€
Ισχύς	100mWatt
Φυσικό μέγεθος και βάρος	Όχι μεγαλύτερο από 5cm x 15cm, 340 γραμμάρια.

5.3 Μοντέλο ελεγκτή τραίνων

Ο χρήστης στέλνει μηνύματα στο τραίνο μέσω ενός χειριστηρίου ελέγχου το οποίο συνδέεται με τις γραμμές. Το χειριστήριο διαθέτει πλήκτρα ελέγχου όπως η βαλβίδα ρύθμισης, το πλήκτρο τερματισμού λειτουργίας σε κατάσταση εκτάκτου ανάγκης κλπ. Το τραίνο λαμβάνει την ηλεκτρική ισχύ από τις δύο ράγες της γραμμής ώστε να μπορεί το χειριστήριο να στέλνει σήματα στο τραίνο μέσω της γραμμής διαμορφώνοντας την τάση τροφοδοσίας. Το τραίνο διαθέτει ένα σύστημα ελέγχου ώστε να ρυθμίζεται η ταχύτητα του κινητήρα και η κατεύθυνση του τραίνου και αναλογικά ηλεκτρονικά για την ανίχνευση των μεταδιδόμενων bits. Στο σύστημα υπάρχει και ένα πακέτο κώδικά διόρθωσης σφάλματος (ECC – Error Correction Code).



Σχήμα 5.3: Το σύστημα του ελεγκτή



Σχήμα 5.4: Σηματοδοσία προς το τραίνο

Αρχικά, θα αναλύσουμε τις απαιτήσεις για τον ελεγκτή τραίνου και στη συνέχεια θα αναλύσουμε τις προδιαγραφές. Οι πιο βασικές απαιτήσεις του συστήματος παρουσιάζονται παρακάτω:

- Η κονσόλα θα πρέπει να ελέγχει το πολύ μέχρι 8 τραίνα σε μία γραμμή.
- Θα πρέπει να υπάρχει ένα σχήμα ανίχνευσης λαθών το οποίο χρησιμοποιείται για μετάδοση μηνυμάτων.

- Η ταχύτητα του τραίνου θα ελέγχεται από μία βαλβίδα ρύθμισης σε 63 διαφορετικά επίπεδα (προς τα εμπρός και αντίστροφα).
- Θα πρέπει να υπάρχει ένα πλήκτρο τερματισμού λειτουργίας σε κατάσταση εκτάκτου ανάγκης.
- Πρέπει, τέλος, να υπάρχει ένας μηχανισμός που θα επιτρέπει στο χρήστη να ρυθμίζει τη δυνατότητα απόκρισης του τραίνου σε εντολές αλλαγής ταχύτητας. Ο μηχανισμός αυτός καλείται μηχανισμός ελέγχου αδράνειας και πρέπει να παρέχει τουλάχιστον 8 διαφορετικά επίπεδα.

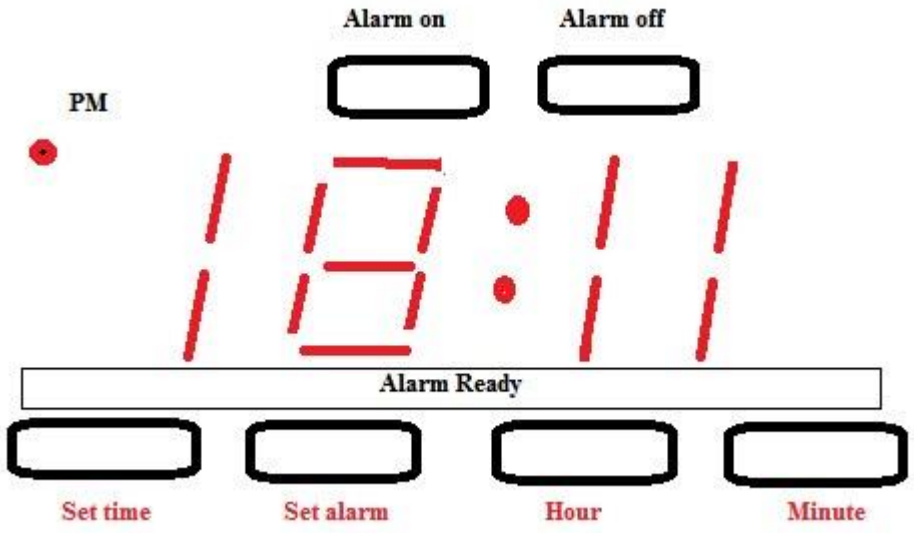
Σχηματικά, οι απαιτήσεις αυτές παρουσιάζονται ως εξής:

Πίνακας 5.2: Απαιτήσεις μοντέλου ελεγκτή τραίνου

Όνομα	Μοντέλο ελεγκτή τραίνου
Σκοπός	Έλεγχος ταχύτητας έως 8 διαφορετικών μοντέλων τραίνων
Είσοδοι	Βαλβίδα, ρύθμιση αδράνειας, τερματισμός κατάστασης εκτάκτου ανάγκης, αριθμός τραίνου
Έξοδοι	Σήματα ελέγχου τραίνων
Λειτουργίες	Καθορισμός της ταχύτητας της μηχανής με βάση τις ρυθμίσεις της αδράνειας και απόκριση σε σταμάτημα εκτάκτου ανάγκης
Απόδοση	Να είναι δυνατή η ενημέρωση της ταχύτητας του τραίνου τουλάχιστον 10 φορές/sec
Κόστος κατασκευής	Γύρω στα 50€
Ισχύς	10 Watt (από πρίζες τοίχου)
Φυσικό μέγεθος και βάρος	Να έχει περίπου το μέγεθος ενός τυποποιημένου πληκτρολογίου και βάρος λιγότερο από 1 κιλό

5.4 Ρολόι ξυπνητήρι

Ένα ακόμα παράδειγμα σχεδίασης συστήματος είναι ένα ρολόι ξυπνητήρι. Για να διαβάσουμε τα πλήκτρα του ρολογιού και να ενημερώνουμε την ώρα στην οθόνη χρησιμοποιούμε ένα μικροεπεξεργαστή. Στην εικόνα που υπάρχει παρακάτω βλέπουμε μια πρόσοψη ενός τέτοιου ρολογιού. Ο χρόνος αναπαριστάται σε δωδεκάωρη βάση με τέσσερα ψηφία και χρησιμοποιείται μια φωτεινή ένδειξη για αν είναι AM ή PM. Χρησιμοποιούμε επίσης διάφορα πλήκτρα για να θέσουμε την ώρα αφύπνισης. Όταν πιέζουμε τα πλήκτρα hour και minute, αυξάνουμε την ώρα και τα λεπτά, αντίστοιχα κατά ένα. Για την ρύθμιση της ώρας, πρέπει να κρατήσουμε πατημένο το πλήκτρο set time ενώ πιέζουμε τα πλήκτρα hour και minute. Αντίστοιχα δουλεύει και το πλήκτρο set alarm. Το πλήκτρο alarm on και off είναι για να ενεργοποιούμε και να απενεργοποιούμε το ξυπνητήρι. Όταν είναι ενεργοποιημένο το ξυπνητήρι είναι φωτεινη η ένδειξη alarm ready. Ένα ηχείο παρέχει τον ήχο αφύπνισης.



Εικόνα 5.1: Το μπροστινό τμήμα για το ρολόι ξυπνητήρι

Πίνακας 5.3: Απαιτήσεις για το ρολόι ξυπνητήρι

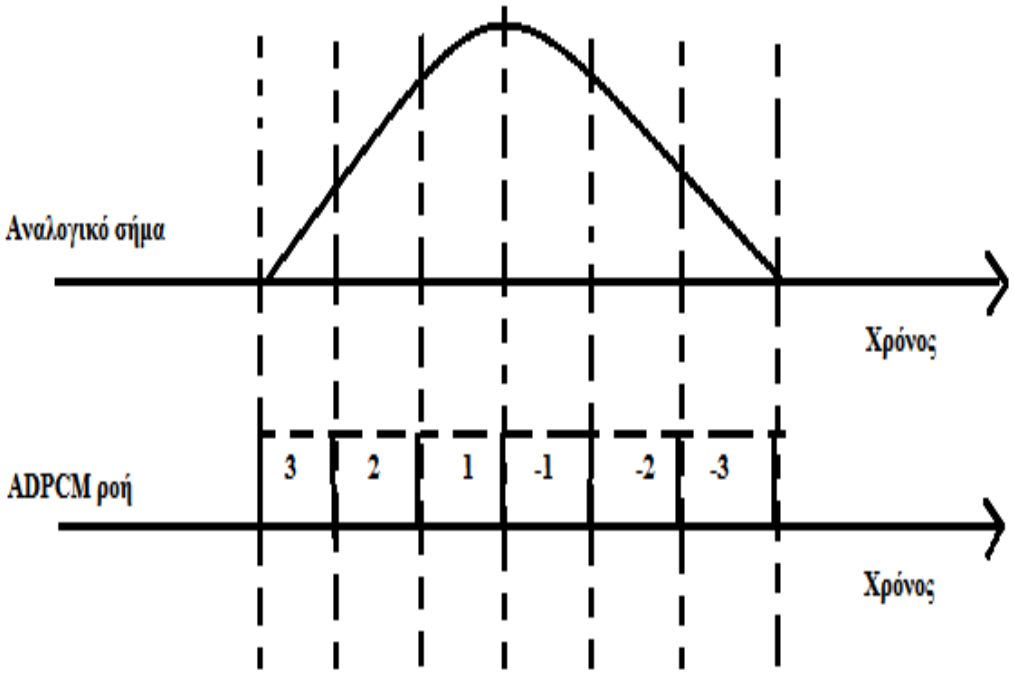
Όνομα	Ρολόι ξυπνητήρι.
Σκοπός	Ένα 24ωρο ψηφιακό ρολόι με ένα απλό ξυπνητήρι.
Είσοδοι	6 πλήκτρα: set time, set alarm, hour, minute, alarm on, alarm off.
Έξοδοι	Τετραψήφια, έξοδος όπως στα ρολόγια. Φωτεινή ένδειξη PM ή AM. Φωτεινή ένδειξη alarm ready. Βομβητής
Λειτουργίες	Προεπιλεγμένος τρόπος λειτουργίας: η οθόνη παρουσιάζει την τρέχουσα ώρα. Η φωτεινή ένδειξη PM είναι αναμμένη από το μεσημέρι έως τα μεσάνυχτα. Τα πλήκτρα hour και minute χρησιμοποιούνται για να αυξήσουν την ώρα και το ξυπνητήρι, αντίστοιχα. Πίεση πλήκτρου set time: Αυτό το πλήκτρο πιέζεται ενώ τα πλήκτρα hour/minute είναι πατημένα, για να θέσει την ώρα. Η νέα ώρα εμφανίζεται αυτόματα στην οθόνη. Πίεση του πλήκτρου set alarm: Ενώ αυτό το πλήκτρο πιέζεται, η οθόνη αλλάζει προσωρινά στην ρύθμιση του χρόνου αφύπνισης. Πιέζοντας τα πλήκτρα hour/minutes αλλάζει η ώρα αφύπνισης κατά τρόπο

	παρόμοιο με τον καθορισμό της ώρας. Alarm on: θέτει το ρολόι σε κατάσταση αφύπνησης, αναγκάζει το ρολόι να ενεργοποιήσει τον βομβητή όταν φθάσει η ώρα αφύπνισης και ανοίγει την φωτεινή ένδειξη alarm ready. Alarm off: κλείνει τον βομβητή, βγάζει το ρολόι από την κατάσταση αφύπνισης και σβήνει την φωτεινή ένδειξη alarm ready.
Απόδοση	Δείχνει ώρες και λεπτά αλλά όχι δευτερόλεπτα. Το ρολόι πρέπει να είναι ακριβές, στα πλαίσια της ακρίβειας που παρέχεται από το σήμα του ρολογιού ενός μικροεπεξεργαστή. (Η υπερβολική ακρίβεια μπορεί να αυξήσει υπέρμετρα το κόστος δημιουργίας ενός ακριβού ρολογιού)
Κόστος κατασκευής	Στην κλίμακα καταναλωτικών προϊόντων. Το κόστος θα εξαρτηθεί κατά κύριο λόγο από το σύστημα του μικροεπεξεργαστή, και όχι από τα πλήκτρα ή την οθόνη.
Ισχύς	Τροφοδοτείται από εναλλασσόμενο ρεύμα μέσω μιας τυποποιημένης παροχής τροφοδοσίας ρεύματος.
Φυσικό μέγεθος και βάρος	Αρκετά μικρό να χωρά σε ένα κομοδίνο, ενώ το βάρος του θα

	πρέπει να είναι το αναμενόμενο βάρος για ένα ρολόι ξυπνητήρι

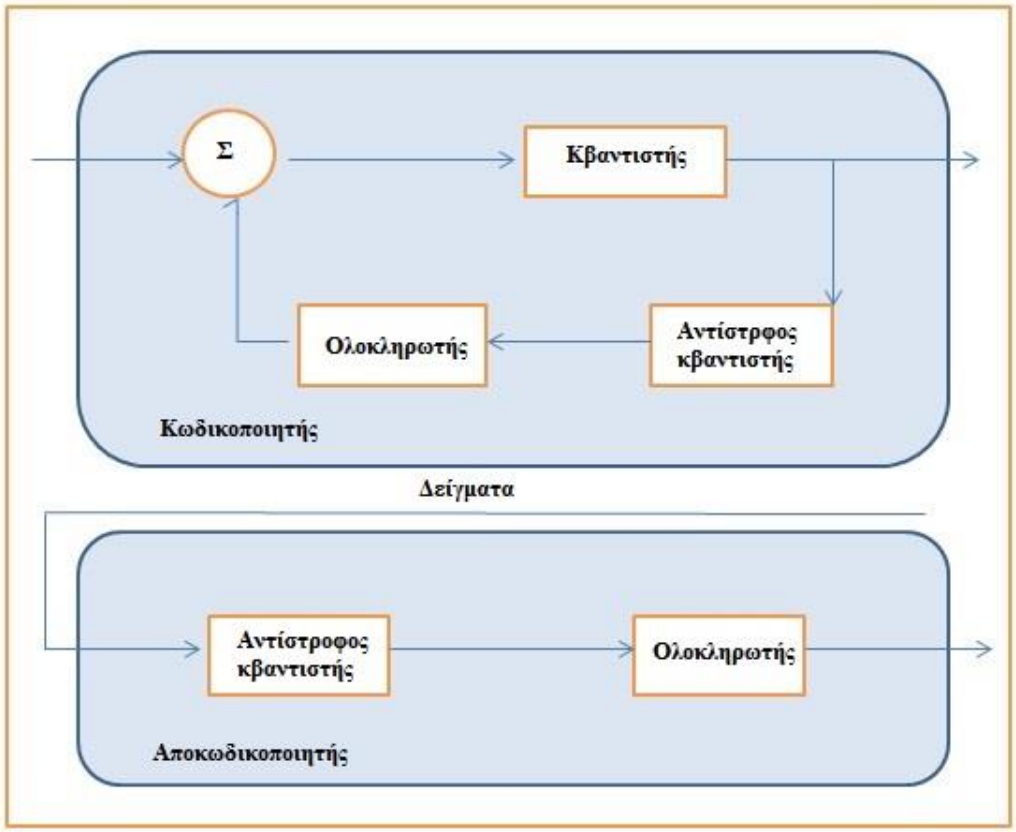
5.5 Ψηφιακός αυτόματος τηλεφωνητής

Ένας ψηφιακός αυτόματος τηλεφωνητής είναι ένα σύστημα που αποθηκεύει μηνύματα σε ψηφιακή μορφή αντί σε αναλογική ταινία. Αυτό γίνεται με την χρήση ενός απλού αλγόριθμου για την συμπίεση των δεδομένων της φωνής έτσι ώστε να μπορούμε να κάνουμε περισσότερο αποδοτική χρήση της περιορισμένης ποσότητας διαθέσιμης μνήμης. Το σχήμα συμπίεσης που χρησιμοποιείται (σχήμα 5.5) είναι γνωστό σαν προσαρμοστική διαφορική παλμοκωδική διαμόρφωση (Adaptive Differential Pulse Code Modulation- ADPCM). Το ADPCM κωδικοποιεί αλλαγές στο σήμα, αντίθετα με την παραδοσιακή δειγματοληψία, στην οποία κάθε δείγμα παρουσιάζει το μέγεθος του σήματος σε μια συγκεκριμένη χρονική στιγμή. Τα δείγματα εκφράζονται με ένα αλφάβητο κωδικοποίησης το οποίο εκτείνεται τόσο σε αρνητικές όσο και σε θετικές τιμές {-3,-2,-1,1,2,3}. Κάθε δείγμα χρησιμοποιείται για την πρόβλεψη της τιμής του σήματος στην παρούσα στιγμή από την προηγούμενη τιμή. Σε κάθε χρονική στιγμή το δείγμα επιλέγεται ώστε το σφάλμα μεταξύ της προβλεφθείσας τιμής και της πραγματικής τιμής του σήματος να ελαχιστοποιείται.



Σχήμα 5.5: Το ADPCM σχήμα κωδικοποίησης

Στο παρακάτω σχήμα βλέπουμε ένα σύστημα ADPCM συμπίεσης, το οποίο συμπεριλαμβάνει έναν κωδικοποιητή και έναν αποκωδικοποιητή. Τόσο ο κωδικοποιήτης όσο και ο αποκωδικοποιητής χρησιμοποιούν έναν ολοκληρωτή για την εκ νέου δημιουργία κυματομορφής από τα δείγματα. Ο ολοκληρωτής υπολογίζει απλά ένα τρέχον άθροισμα της ιστορίας των δειγμάτων.Ο κωδικοποιήτης συγκρίνει την εισερχόμενη κυματομορφή με την προβλεπόμενη κυματομορφή. Ο κβαντιστής κωδικοποιεί αυτήν την διαφορά σαν την καλύτερη πρόβλεψη της επόμενης τιμής της κυματομορφής. Ο αντίστροφος κβαντιστής επιτρέπει να απεικονιστούν σύμβολα επιπέδου bit πάνω σε πραγματικές αριθμητικές τιμές. Ο αποκωδικοποιητής χρησιμοποιεί απλά ένα αντίστροφο κβαντιστή και έναν ολοκληρωτή για την μετατροπή των διαφορικών δειγμάτων στην κυματομορφή.



Σχήμα 5.6: Το ADPCM σύστημα συμπίεσης

Ο αυτόματος τηλεφωνητής θα συνδεθεί τελικά με μια τηλεφωνική συνδρομητική γραμμή που στο άλλο άκρο της είναι το κεντρικό γραφείο. Όλη η πληροφορία μεταφέρεται στην τηλεφωνική γραμμή με αναλαογική μορφή μέσα από ένα ζευγάρι καλωδίων. Για να γίνει η μετατροπή του αναλογικού σήματος σε ψηφιακό και το αντίστροφο χρειαζόμαστε να ανιχνεύσουε δύο άλλα χαρακτηριστικά της γραμμής.

- Σήμα κλήσης: Το κεντρικό γραφείο στέλνει ένα σήμα κλήσης στο τηλέφωνο όταν μια κλήση βρίσκεται σε αναμονή.
- Κατάσταση εκτός αγκίστρου (off-hook): Ο όρος της τηλεφωνικής βιομηχανίας για την απάντηση μιας κλήσης είναι η μετακίνηση εκτός αγκίστρου. Ο τεχνικός όρος για το κλείσιμο του τηλεφώνου είναι η μετακίνηση εντός αγκίστρου (on-hook). Η διασύνδεση θα στείλει ένα ψηφιακό σήμα για τη μετακίνηση της τηλεφωνικής γραμμής εκτός αγκίστρου, κάτι που θα αναγκάσει το αναλογικό κύκλωμα να κάνει την απαραίτητη σύνδεση έτσι ώστε τα

δεδομένα φωνής να μπορούν να αποσταλούν και να ληφθούν κατά την διάρκεια τους.

Πίνακας 5.4: Απαιτήσεις ψηφιακού αυτόματου τηλεφωνητή

Όνομα	Ψηφιακός αυτόματος τηλεφωνητής
Σκοπός	Αυτόματος τηλεφωνητής με ψηφιακή μνήμη, που χρησιμοποιεί συμπίεση φώνης
Είσοδοι	*Τηλέφωνο*: δείγματα φωνής, δείκτης κλήσης *Διασύνδεση χρήστη*: μικρόφωνο, κοθμπί αναπαραγωγής μηνυμάτων, κουμπί εγγραφής εξερχόμενου μηνύματος.
Έξοδοι	*Τηλέφωνο*: δείγματα φωνής, εντολή εντός-εκτός αγκίστρου. *Διασύνδεση χρήστη*: Μεγάφωνο, δεικτης αριθμού μηνυμάτων. Λαμπάκι μηνύματων
Λειτουργίες	*Προκαθορισμένη λειτουργία*: όταν η μηχανή λαμβάνει δείκτη κλήσης, σηματοδοτεί μετακίνηση εκτός αγκίστρου, αναπαράγει το εξερχόμενο μήνυμα, και στη συνέχεια καταγράφει το εισερχόμενο μήνυμα. Το μέγιστο μήκος καταγραφής για το εισερχόμενο μήνυμα είναι 30 δευτερόλεπτα, χρονικό σημείο στο οποίο ο τηλεφωνητής κλείνει το τηλέφωνο.

	Αν εξαντλείται η μνήμη του τηλεφωνητή, το εξερχόμενο μήνυμα αναπαράγεται και ο τηλεφωνητής στη συνέχεια κλείνει το τηλέφωνο χωρίς καταγραφή μηνύματος. *Λειτουργία αναπαραγωγής ήχου*: όταν το κουμπί αναπαραγωγής πιέζεται, ο τηλεφωνητής αναπαράγει όλα τα μηνύματα. Αν το κουμπί αναπαραγωγής πιέζεται πάλι μέσα σε πέντε δευτερόλεπτα, τα μηνύματα αναπαράγονται πάλι. Τα μηνύματα σβήνονται μετά από την αναπαραγωγή. *Λειτουργία έκδοσης εξερχόμενου μηνύματος:* όταν ο χρήστης χτυπά το κουμπί εγγραφής εξερχόμενου μηνύματος, ο τηλεφωνητής καταγράφει ένα εξερχόμενο μήνυμα το πολύ 10 δευτερολέπτων. Όταν ο χρήστης κρατάει πατημένο το κουμπί εγγραφής εξερχόμενων μηνυμάτων και πιέζει το κουμπί αναπαραγωγής, το εξερχόμενο μήνυμα αναπαράγεται.
Απόδοση	Πρέπει να είναι ικανός να καταγράφει περίπου 30 λεπτά συνολικής φωνής, συμπεριλαμβανομένων εισερχόμενων και εξερχόμενων μηνυμάτων. Τα δεδομένα φωνής

	δειγματοληπτούνται με τον τυπικό ρυθμό των 8 kHz.
Κόστος κατασκευής	Εύρος καταναλωτικών προϊόντων: περίπου 50 ευρώ
Ισχύς	Τροφοδοσία από εναλλασόμενο ρεύμα μέσω μιας τυπικής παροχής
Φυσικό μέγεθος και βάρος	Συγκρίσιμο στο μέγεθος και στο βάρος με ένα τηλέφωνο γραφείου

5.6 Ελεγκτής Ανελκυστήρα

Για την σχεδίαση ενός ελεγκτή ανελκυστήρα θα χρησιμοποιηθούν οι αρχές σχεδίασης ενός κατανεμημένου συστήματος. Τα συστατικά φυσικά κατανέμονται ανάμεσα στους ανελκυστήρες και στα πατώματα του κτιρίου, και το σύστημα πρέπει να ικανοποιεί τόσο τις αυστηρές προθεσμίες (εξασφάλιση ότι ο ανελκυστήρας σταματάει στο σωστό σημείο) όσο και τις χαλαρές προθεσμίες (συμμόρφωση στις απαιτήσεις για ανελκυστήρες).

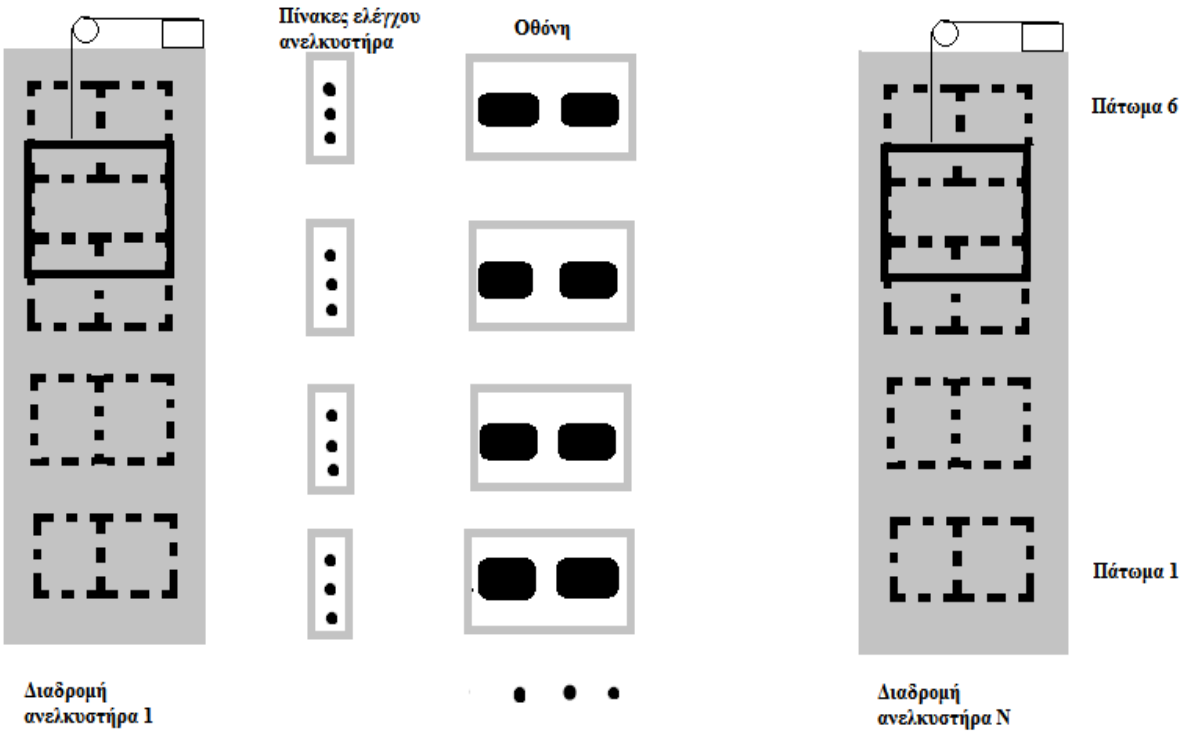
Σχεδιάζουμε ένα σύστημα πολλαπλών ανελκυστήρων για να αυξήσουμε την πρόκληση. Η διευθέτηση μιας σειράς ανελκυστήρων παρουσιάζεται στην παρακάτω εικόνα.

- Ο θάλαμος του ανελκυστήρα είναι μονάδα που τρέχει πάνω και κάτω στη διαδρομή του ανελκυστήρα μεταφέροντας επιβάτες.
- Κάθε θάλαμος τρέχει σε μια διαδρομή ανελκυστήρα και μπορεί να σταματήσει σε οποιαδήποτε από τα 6 πατώματα.
- Κάθε θάλαμος ανελκυστήρα έχει ένανα πίνακα ελέγχου θαλάμου ο οποίος επιτρέπει στους επιβάτες να επιλέγουν τα πατώματα στα οποία θα σταματήσουν.
- Κάθε πάτωμα έχει ένα πίνακα ελέγχου πατώματος ο οποίος καλεί τονα ανελκυστήρα.

- Κάθε πάτωμα επίσης έχει ένα σύνολο οθονών για την παρουσιάση της τρέχουσας κατάστασης των συστημάτων ανελκυστήρων.

Η διασύνδεση χρήστη αποτελείται από τους πίνακες ελέγχου των ανελκυστήρων, τους πίνακες ελέγχου των πατωμάτων και τις οθόνες.

- Οι πίνακες ελέγχου των θαλάμων έχουν 6 κουμπία για την επιλογή των πατωμάτων συν ένα κουμπί στάσης για επείγουσα ανάγκη.
- Κάθε πίνακας ελέγχου πατώματος έχει ένα κουμπί προς τα πάνω και ένα κουμπί προς τα κάτω τα οποία ζητούν έναν ανελκυστήρα να πηγαίνει στην επιλγμένη κατεύθυνση.
- Υπάρχει μια οθόνη για κάθε διαδρομή ανλεκυστήρα σε κάθε πάτωμα.
- Κάθε οθόνη έχει ένα φως για την κατεύθυνση προς τα κάτω και προς τα πάνω ένω αν είναι αδρανής ο ανελκυστήρας δεν είναι αναμμένο το φως.
- Οι οθόνες για μια διαδρομή ανελκυστήρα πάντα παρουσιάζουν την ίδια κατάσταση σε όλα τα πατωμάτα.



Σχήμα 5.7: Μια σειρά ανελκυστήρων

Το σύστημα ελέγχου του ανελκυστήρα αποτελείται από δύο τύπους συστατικών:

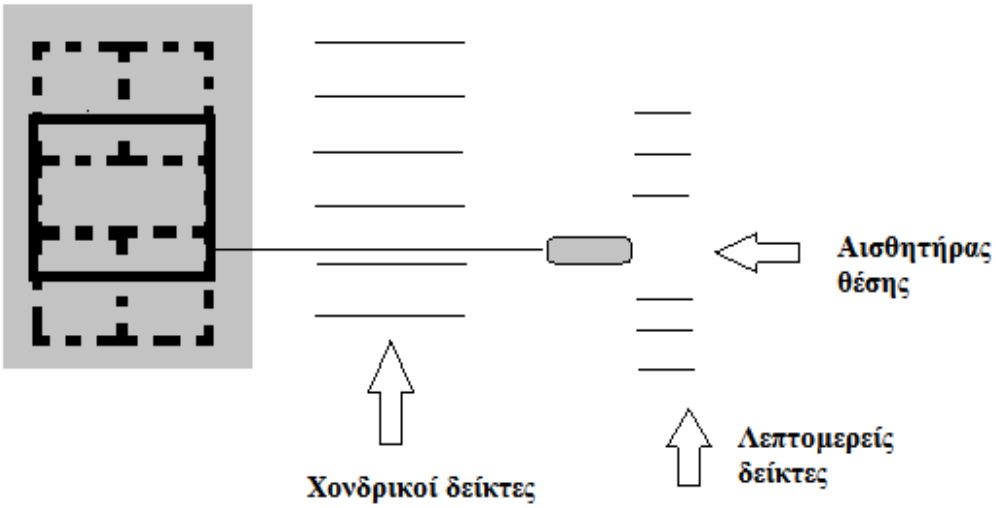
- Έναν μόνο κύριο ελεγκτή οποίος διοικεί τη συνολική συμπεριφορά όλων των ανελκυστήρων
- Σε κάθε ανελκυστήρα έναν ελεγκτής θαλάμου ο οποίος εκτελεί οτιδήποτε πρέπει να γίνει μέσα στο θάλαμο.

Ο ελεγκτής θαλάμου πρεπέι να ανιχνεύει τα πατήματα των κουμπιών στον πίνακα ελέγχου του θαλάμου αλλά και την τρέχουσα θέση του ανελκυστήρα.

Όπως βλέπουμε στην παρακάτω εικόνα ο ελεγκτής του θαλάμου διαβάζει δυο σύνολα δεικτών στον τοίχο της διαδρομής του ανελκυστήρα για την ανίχνευση της θέσης, Οι χονδρικοί δείκτες τρέχουν σε όλο το μήκος της διαδρομής του ανελκυστήρα και ένας αισθητήρας καθορίζει πότε ο ανελκυστήρας περνά από το καθένα. Οι λεπτομερείς δείκτες τοποθετούνται

μόνο γύρω από το σημείο στάσης για κάθε πάτωμα. Το σύστημα ανελκυστήρα μπορεί να σταματήσει στη σωστή θέση με την μέτρηση των χονδρικών και των λεπτομερών δεικτών.

Η κίνηση του ανελκυστήρα ελέγχεται από δύο εισόδους ελέγχου κινητήρα: μία για πάνω και μία για κάτω. Όταν και οι δύο είναι ανενεργές, ο ανελκυστήρας δεν κινείται. Το σύστημα δεν θα πρέπει να ενεργοποιεί τόσο το πάνω όσο και το κάτω σε μια διαδρομή ανελκυστήρα ταυτόχρονα.



Σχήμα 5.8: Ανίχνευση θέσης ανελκυστήρα

Ο κύριος ελεγκτής έχει αρκετές εργασίες:

- Να διαβάζει τις εισόδους από τους πίνακες έλεγχου πατώματος
- Να στέλνει σήματα στα φώτα των οθονών πατώματος
- Να διαβάζει τις αιτήσεις πατώματος από τους ελεγκτές
- Να λαμβάνει εισόδους από τους αισθητήρες θαλάμου
- Να λέει στους ανελκυστήρες πότε να κινούνται και πότε να σταματούν

- Να χρονοπρογραμματίζει τους ανελκυστήρες για να απαντούν αποτελεσματικά στις αιτήσεις των επιβατών

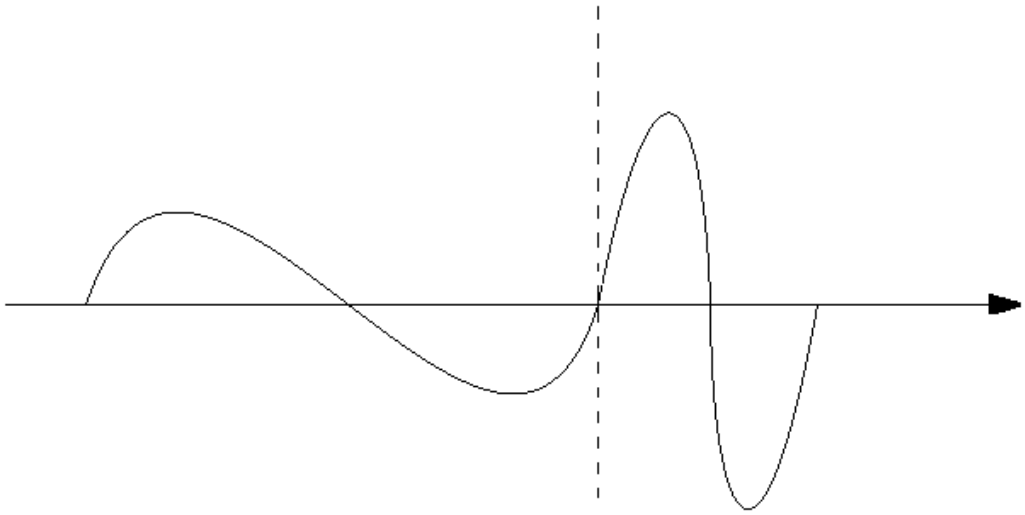
Οι βασικές απαιτήσεις είναι οι παρακάτω:

Πίνακας 5.5: Απαιτήσεις συστήματος ανελκυστήρα

Όνομα	Σύστημα ανελκυστήρα
Είσοδοι	6 εισόδοι ελέγχου πατώματος, Ν αισθητήρες θέσης, Ν πίνακες ελέγχου θαλάμου, ένας πίνακας κυρίου ελέγχου
Έξοδοι	6 οθόνες, Ν ελεγκτές κινητήρων
Λειτουργίες	Απόκριση στους πίνακες έλεγχου πατώματος, θαλάμου και κύριου-ασφαλής λειρουργία θαλάμου
Απόδοση	Ο έλεγχος των ανελκυστήρων είναι κρίσιμος από άποψη χρόνου
Κόστος κατασκευής	Το κόστος των ηλεκτρονικών είναι μικρό σε σύγκριση με τα μηχανολογικά συστήματα
Ισχύς	Δεν είναι σημαντική
Φυσικό μέγεθος και βάρος	Η καλωδίωση είναι η κυριότερη ανησυχία

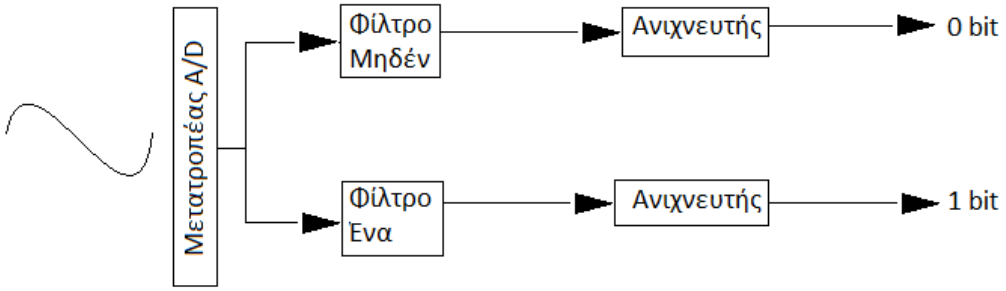
5.7 Σχεδίαση Modem Λογισμικού

Το modem του παραδείγματος θα χρησιμοποιεί διαμόρφωση FSK (μετατόπισης συχνότητας). Το σχήμα FSK μεταδίδει ημιτονικούς τόνους με 0 και 1 αναθεμένα σε διαφορικές συχνότητες. Οι ημιτονικοί τόνοι είναι καταλληλότεροι για μετάδοση πάνω από αναλογικές τηλεφωνικές γραμμές σε σχέση με τις παραδοσιακές τάσεις (χαμηλές ή υψηλές) των ψηφιακών κυκλωμάτων.



Σχήμα 5.10: Η διαμόρφωση μετατόπισης συχνότητας. Δεξιά βρίσκεται το φίλτρο ένα και αριστερά το φίλτρο μηδέν.

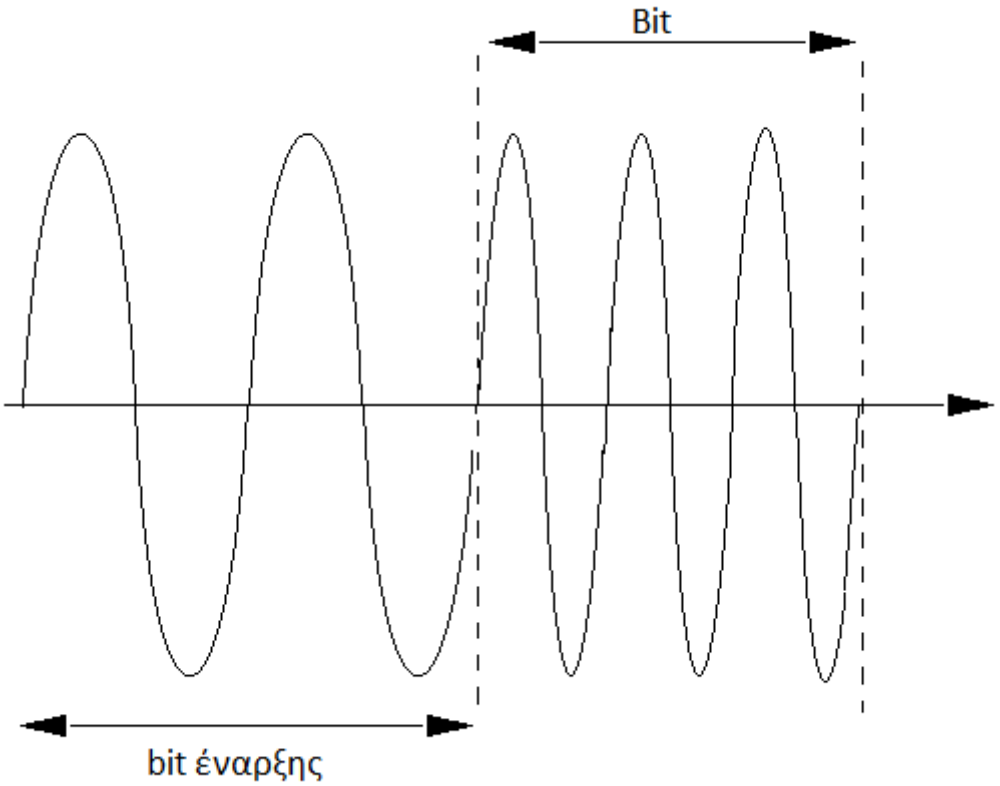
Το παρακάτω σχήμα χρησιμοποιείται για τη μετάφραση της ακουστικής εισόδου σε ένα ρεύμα bits (bit stream). Η αναλογική είσοδος δειγματοληπτείται και το ρεύμα που παράγεται στέλνεται σε δύο ψηφιακά φίλτρα. Το ένα φίλτρο περνάει συχνότητες στο εύρος που παριστάνει ένα 0 και απορρίπτει τις συχνότητες του 1, και το άλλο φίλτρο κάνει το αντίστροφο. Οι έξοδοι των δύο φίλτρων στέλνονται σε ανιχνευτές, οι οποίοι υπολογίσουν τη μέση τιμή του σήματος στα τελευταία N δείγματα. Όταν η ενέργεια περνάει πάνω από μία τιμή κατωφλίου, το κατάλληλο bit ανιχνεύεται.



Σχήμα 5.10: Σχήμα ανιχνευτή FSK

Έστω ότι θέλουμε να στείλουμε δεδομένα σε μονάδες των 8 bits. Τα modems που θα επικοινωνήσουν μεταξύ τους ώστε να εκπέμψουν και να λάβουν, έχουν συνεννοηθεί εκ των προτέρων για το χρόνο που θα διαρκέσει η διαδικασία μετάδοσης του 1 bit (baud rate). Το modem που λαμβάνει δε γνωρίζει πότε ο πομπός άρχισε αν στέλνει 1 byte. Ακόμη και αν ο δέκτης ανιχνεύσει μια εκπομπή, οι συχνότητες των ρολογιών πομπού και δέκτη διαφέρουν, και έτσι υπάρχει απώλεια συγχρονισμού.

Κατά τη διεργασία λήψης, ο δέκτης ανιχνεύει την αρχή ενός byte ψαχνοντας το bit έναρξης (είναι πάντα 0). Ο δέκτης γνωρίζει που να ψάξει για την αρχή του byte με βάση τη διάρκεια του bit έναρξης. Στη συνέχεια τρέχει έναν αλγόριθμο ανίχνευσης στο σημείο που έχει προβλέψει ότι ξεκινάει το bit, ώστε να προληφθεί η πιθανότητα εσφαλμένης πρόβλεψης.



Σχήμα 5.11: Λήψη bits στο modem

Το modem θα υλοποιήσει μια διασύνδεση υλικού σε μία τηλεφωνική γραμμή ή το λογισμικό για την κλήση ενός τηλεφωνικού αριθμού. Σ'αυτό το σημείο θα υποθέσουμε ότι έχουμε αναλογικές ηχητικές εισόδους και εξόδους για αποστολή και λήψη. Επίσης, θα τρέξουμε σε ρυθμό bit μικρότερο από 1200 baud για να διευκολυνθούμε στην υλοποίηση. Κατόπιν, θα βάλουμε το μήνυμα του πομπού στη μνήμη και θα σώσουμε το αποτέλεσμα του δέκτη πάλι στη μνήμη. Με βάση αυτά συμπληρώνεται ο πίνακας απαιτήσεων ο οποίος φαίνεται παρακάτω.

Πίνακας 5.6: Απαιτήσεις Modem

Όνομα	Modem
Σκοπός	Ένα Modem σταθερού baud rate με διαμόρφωση frequency-shift keying (FSK)
Είσοδοι	Είσοδος αναλογικού ήχου, κουμπί reset
Έξοδοι	Έξοδος αναλογικού ήχου, οθόνη LED bit
Λειτουργίες	**Πομπός**: Στέλνει δεδομένα που φυλάσσονται στη μνήμη του επεξεργαστή σε bytes των 8 bit. Στέλνει bit έναρξης για κάθε byte ίσο με ένα bit. **Δέκτης**: Ανιχνεύει αυτόματα τα bytes και φυλάσσει τα αποτελέσματα στην κεντρική μνήμη. Εμφανίζει τυχόν bit σε LED.
Απόδοση	1200 baud
Κόστος κατασκευής	Κυριαρχείται από τον μικροεπεξεργαστή και την αναλογική είσοδο/έξοδο
Ισχύς	Τροφοδοτείται με AC μέσω ενός τυπικού τροφοδοτικού
Φυσικό μέγεθος και βάρος	Αρκετά μικρό και ελαφρύ ώστε να χωράει σε ένα γραφείο

ΣΥΜΠΕΡΑΣΜΑΤΑ

Τα ενσωματωμένα συστήματα αναπτύσσονται συνεχώς. Απαιτείται αρκετή προσπάθεια και εργαλεία αυτοματοποίησης σχεδιασμού ώστε να διαχειριστεί κανείς την πολυπλοκότητα των σημερινών ενσωματωμένων συστημάτων.

Αυτό έχει ως αποτέλεσμα να φέρνει αντιμέτωπους τους μηχανικούς με νέα προβλήματα όταν καλούνται να προδιαγράψουν, προσομοιώσουν, σχεδιάσουν και βελτιστοποιήσουν τέτοια σύνθετα συστήματα. Οι υλοποιήσεις ενσωματωμένων συστημάτων αποτελούνται συνήθως από προγραμματιζόμενα στοιχεία γενικού ή ειδικού σκοπού, στοιχεία υλικού ειδικού σκοπού, καθώς και στοιχεία επικοινωνίας και μνήμης.

Λόγω της εξέλιξης της τεχνολογίας και της αύξησης της πολυπλοκότητας των σημερινών εφαρμογών, οδηγείσαι στην ανάπτυξη των SoC για την υλοποίηση ενσωματωμένων συστημάτων. Τα προγραμματιζόμενα στοιχεία σε ένα ενσωματωμένο σύστημα παρέχουν την απαιτούμενη ευελιξία, ενώ τα στοιχεία υλικού ειδικού σκοπού αποσκοπούν στην ικανοποιητική υλοποίηση χρονικά κρίσιμων διεργασιών και στη μείωση της κατανάλωσης ενέργειας.

Ένα σημαντικό πρόβλημα στο σχεδιασμό SoC είναι η παραγωγικότητα σχεδιασμού. Μία λύση είναι η επαναχρησιμοποίηση στοιχείων IP και η χρήση αποδοτικών μεθοδολογιών και εργαλείων σχεδιασμού. Οι πλατφόρμες προτυποποίησης χρησιμοποιούνται μετά την προσομοίωση ώστε να αναπτυχθεί το πρωτότυπο του συστήματος που πρόκειται να υλοποιηθεί σε ολοκληρωμένο κύκλωμα.

Ως εναλλακτική λύση των στοιχείων FPGA και των SoC ειδικού σκοπού, οι μηχανικοί-σχεδιαστές σήμερα έχουν τη δυνατότητα χρησιμοποίησης ολοκληρωμένων κυκλωμάτων με ενσωματωμένα επαναπροσδιοριζόμενα στοιχεία καθώς και ενσωματωμένων επαναπροσδιοριζόμενων στοιχείων

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] C. Alippi, Intelligence for Embedded Systems, Springer, 2014.

[2] C. Baumann, “Field Programmable Gate Array (FPGA)”, *Summary paper for the seminar “Embedded System Architecture”*, University of Innsbruck, 2010.

[3] S. Heath, “Embedded systems design. EDN series for design engineers (2 ed.)”, "An embedded system is a microprocessor based system that is built to control a function or a range of functions.", 2003.

[4] S. Brown, Z. Vranesic, Fundamentals of Digital Logic with Verilog design, McGraw-Hill Companies, Inc., 2003.

[5] Κ. Καλοβρέκτης, Βασικές Δομές Ενσωματωμένων Συστημάτων, Εκδόσεις Βαρβαρήγου, 2012.

[6] W. Wolf, Αρχές Σχεδίασης Ενσωματωμένων Υπολογιστικών Συστημάτων, Εκδόσεις Νέων Τεχνολογιών, 2008.

[7] J. Krumm, Ubiquitous Computing Fundamentals, Chapman and Hall/CRC, 2009.

[8] S. Poslad, Ubiquitous Computing: Smart Devices, Environments and Interactions, Wiley; 1 edition, 2009.

[9] A. Greenfield, Everyware: The Dawning Age of Ubiquitous Computing, New Riders Publishing; 1st edition, 2006.

[10] J. Warmer, A. Kleppe, The Object Constraint Language: Precise Modeling with UML, Object Technology Series, Addison-Wesley, 1999.

[11] R. Chen, M. Sgro., L. Lavagno, G. Martin, A. Sangiovanni-Vincentelli, J. Rabaey, UML and Platform-based Design, IEEE Computer Society 2002.

[12] B. Selic, “A Generic Framework for Modeling Resources with UML”, *IEEE Computer*, June 2000.

[13] S. Ravi and A. Raghunathan, “ Security in Embedded Systems:Design challenges”

[14] Dr. D. J. Greaves, “System on Chip Design and Modelling”, University of Cambridge Computer Laboratory.

ΗΛΕΚΤΡΟΝΙΚΕΣ ΠΗΓΕΣ

[1] http://www.xilinx.com/fpga/asic.htm

[2] http://www.xilinx.com/cpld/

[3] http://www.electronicproducts.com/Digital_ICs/Standard_and_Programmable_Logic/The_evolution_of_FPGA_coprocessing.aspx

[4] http://fedoraproject.org/wiki/Architectures/ARM

[5] http://www.arm.com/products/processors/instruction-set-architectures/index.php

[6] S. Mellor, “From UML to embedded system: Is it possible?”, http://www.eetimes.com/esc/showArticle.jhtml?articleID=184417422