Κεφάλαιο

1

Ενσωματωμένη Υπολογιστική

- Γιατί ενσωματώνουμε μικροεπεξεργαστές στα συστήματα.
- Ποιες είναι οι δυσκολίες σχετικά με την ενσωμάτωση των μικροεπεξεργαστών.
- Μεθοδολογίες σχεδίασης.
- Περιγραφή σχεδίασης με τη UML.
- Καθοδηγούμενη περιήγηση στο βιβλίο.

1.1 Εισαγωγή

Σε αυτό το κεφάλαιο θέτουμε τη βάση για τη μελέτη της σχεδίασης ενσωματωμένων υπολογιστικών συστημάτων (embedded computing systems). Προκειμένου να γίνουν κατανοητές οι διαδικασίες σχεδίασης, πρέπει αρχικά να καταλάβουμε πώς και γιατί οι μικροεπεξεργαστές χρησιμοποιούνται για τον έλεγχο, τη διασύνδεση με το χρήστη (user interface), την επεξεργασία σήματος (signal processing), και πολλές άλλες εργασίες (tasks). Ο **μικροεπεξεργαστής** είναι πλέον τόσο συνηθισμένος που εύκολα μπορεί κάποιος να ξεχάσει πόσο δύσκολο είναι να γίνουν κάποια πράγματα χωρίς αυτόν.

Αρχικά, εξετάζουμε τις διάφορες χρήσεις των μικροεπεξεργαστών. Στη συνέχεια εξετάζουμε τους κυριότερους λόγους για τους οποίους οι μικροεπεξεργαστές χρησιμοποιούνται στη σχεδίαση συστημάτων – παραδίδοντας πολύπλοκες συμπεριφορές, γρήγορους χρόνους διεκπεραίωσης σχεδίασης και ούτω καθεξής. Στη συνέχεια, στην Ενότητα 1.3, παρουσιάζουμε το παράδειγμα της σχεδίασης ενός συστήματος για την κατανόηση των κύριων βημάτων της σχεδίασης ενός συστήματος. Η Ενότητα 1.4 περιλαμβάνει μια λεπτομερή εξέταση των τεχνικών προδιαγραφής ενσωματωμένων συστημάτων – χρησιμοποιούμε αυτές τις τεχνικές σε όλο το βιβλίο. Στην Ενότητα 1.5, χρησιμοποιούμε το μοντέλο ενός ελεγκτή τραίνου σαν ένα παράδειγμα για την εφαρμογή των τεχνικών

προδιαγραφής που εισάγονται στην Ενότητα 1.4. Η Ενότητα 1.6 παρέχει μια περιήγηση στο βιβλίο ανά κεφάλαιο.

1.2 Πολύπλοκα Συστήματα και Μικροεπεξεργαστές

Τι είναι ένα **ενσωματωμένο υπολογιστικό σύστημα (embedded computing system)**; Ένας απλός ορισμός είναι ότι είναι οποιαδήποτε συσκευή, η οποία περιλαμβάνει έναν προγραμματιζόμενο υπολογιστή, ο οποίος όμως δεν είναι υπολογιστής γενικού σκοπού. Συνεπώς, ένας προσωπικός υπολογιστής (personal computer – PC) δεν είναι ένα ενσωματωμένο υπολογιστικό σύστημα, μολονότι συχνά οι προσωπικοί υπολογιστές χρησιμοποιούνται για την υλοποίηση ενσωματωμένων υπολογιστικών συστημάτων, όπως για παράδειγμα το βασισμένο σε προσωπικό υπολογιστή τηλεφωνικό σύστημα της Ενότητας 9.7. Αλλά μια μηχανή φαξ ή ένα ρολόι που υλοποιείται από έναν μικροεπεξεργαστή είναι ένα ενσωματωμένο υπολογιστικό σύστημα.

Αυτό σημαίνει ότι η σχεδίαση ενσωματωμένων υπολογιστικών συστημάτων είναι μια χρήσιμη δεξιότητα για την ανάπτυξη πολλών τύπων προϊόντων. Τα αυτοκίνητα, οι προσωπικοί ψηφιακοί βοηθοί (personal digital assistants - PDAs), ακόμη και οι οικιακές συσκευές κάνουν εκτεταμένη χρήση των μικροεπεξεργαστών. Οι σχεδιαστές σε πολλά πεδία πρέπει να έχουν την ικανότητα να αναγνωρίζουν που μπορούν να χρησιμοποιηθούν οι μικροεπεξεργαστές, να σχεδιάζουν μια πλατφόρμα υλικού (hardware platform) με συσκευές εισόδου/εξόδου οι οποίες μπορούν να υποστηρίξουν τις απαιτούμενες εργασίες (tasks), και να υλοποιούν λογισμικό το οποίο εκτελεί την απαιτούμενη επεξεργασία. Η μηχανική των υπολογιστών (computer engineering), όπως η μηχανολογική σχεδίαση ή η θερμοδυναμική, είναι ένας θεμελιώδης επιστημονικός κλάδος ο οποίος μπορεί να εφαρμοστεί σε πολλές διαφορετικές περιοχές. Αλλά φυσικά, η σχεδίαση ενσωματωμένων υπολογιστικών συστημάτων δεν είναι αυθύπαρκτη. Πολλές από τις προκλήσεις που συναντώνται κατά τη σχεδίαση ενός ενσωματωμένου υπολογιστικού συστήματος δεν είναι μηχανική υπολογιστών – για παράδειγμα, μπορεί να είναι μηχανολογικά προβλήματα ή προβλήματα αναλογικών ηλεκτρονικών. Σε αυτό το βιβλίο εστιάζουμε κύρια στον ίδιο τον ενσωματωμένο υπολογιστή, επομένως θα επικεντρωθούμε στο υλικό και το λογισμικό που υλοποιούν τις επιθυμητές λειτουργίες στο τελικό προϊόν.

1.2.1 Ενσωμάτωση Υπολογιστών

Οι υπολογιστές έχουν ενσωματωθεί σε εφαρμογές από τις πρώτες ημέ-

ρες της υπολογιστικής. Ένα παράδειγμα είναι ο Whirlwind, ένας υπολογιστής που σχεδιάστηκε στο MIT (Massachusetts Institute of Technology) στα τέλη της δεκαετίας του '40 και στις αρχές της δεκαετίας του '50. Ο Whirlwind ήταν επίσης ο πρώτος υπολογιστής που σχεδιάστηκε για την υποστήριξη λειτουργίας **πραγματικού χρόνου** (real time) και η αρχική ιδέα ήταν να χρησιμοποιηθεί σαν ένας μηχανισμός για τον έλεγχο ενός προσομοιωτή (simulator) αεροσκαφών. Μολονότι ήταν υπερβολικά μεγάλος σε φυσικές διαστάσεις συγκριτικά με τους σημερινούς υπολογιστές (περιείχε περισσότερες από 4.000 λυχνίες κενού – vacuum tubes, για παράδειγμα), η πλήρης σχεδίαση του από τα συστατικά μέχρι το σύστημα εναρμονίστηκε με τις ανάγκες της ενσωματωμένης υπολογιστικής πραγματικού χρόνου (real time embedded computing). Η χρησιμότητα των υπολογιστών στην αντικατάσταση μηχανολογικών ή ανθρώπινων ελεγκτών ήταν εμφανής από την αρχή της εποχής των υπολογιστών – για παράδειγμα, η χρήση υπολογιστών για τον έλεγχο χημικών διεργασιών είχε προταθεί στα τέλη της δεκαετίας του '40 [Sto95].

Ένας μικροεπεξεργαστής είναι μια κεντρική μονάδα επεξεργασίας (central processing unit – CPU) σε ένα ολοκληρωμένο κύκλωμα ή τσιπ. Η τεχνολογία VLSI (Very Large Scale Integration - ολοκλήρωση πολύ μεγάλης κλίμακας) έχει επιτρέψει την τοποθέτηση μιας πλήρους CPU σε ένα μόνο ολοκληρωμένο κύκλωμα/τσιπ από τη δεκαετία του '70, αλλά οι CPU εκείνης της εποχής ήταν πολύ απλές. Ο πρώτος μικροεπεξεργαστής, ο 4004 της Intel, σχεδιάστηκε για μια ενσωματωμένη εφαρμογή, συγκεκριμένα μια αριθμομηχανή. Η αριθμομηχανή δεν ήταν ένας υπολογιστής γενικού σκοπού – απλά παρείχε τις βασικές αριθμητικές λειτουργίες. Ωστόσο, ο Ted Hoff της Intel συνειδητοποίησε ότι ένας υπολογιστής γενικού σκοπού κατάλληλα προγραμματισμένος θα μπορούσε να υλοποιήσει την απαιτούμενη λειτουργία, και ότι ο υπολογιστής–σε–ένα–τσιπ (computer-on-a-chip) θα μπορούσε να επαναπρογραμματιστεί για χρήση και σε άλλα προϊόντα. Εφόσον η σχεδίαση ολοκληρωμένων κυκλωμάτων ήταν (και είναι ακόμη) μια ακριβή και χρονοβόρα διαδικασία, η ικανότητα επαναχρησιμοποίησης της σχεδίασης του υλικού με αλλαγή του λογισμικού ήταν μια εντυπωσιακή ανακάλυψη. Ο HP–35 ήταν η πρώτη αριθμομηχανή χειρός η οποία πραγματοποιούσε υπερβατικές λειτουργίες [Whi72]. Παρουσιάστηκε το 1972, και επομένως χρησιμοποιούσε αρκετά ολοκληρωμένα κυκλώματα/τσιπ για την υλοποίηση της CPU, αντί για έναν μικροεπεξεργαστή ενός τσιπ. Ωστόσο, η ικανότητα ανάπτυξης προγραμμάτων για την πραγματοποίηση μαθηματικών αντί για την ανάγκη σχεδίασης ψηφιακών κυκλωμάτων για την πραγματοποίηση λειτουργιών όπως οι τριγωνομετρικές συναρτήσεις, ήταν κρίσιμη για την επιτυχή σχεδίαση της αριθμομηχανής.

Οι σχεδιαστές αυτοκινήτων ξεκίνησαν να χρησιμοποιούν τους μικρο-

επεξεργαστές σύντομα, αφού έγιναν διαθέσιμες CPU σε ένα ολοκληρωμένο κύκλωμα/τσιπ. Η πιο σημαντική και πιο σύνθετη χρήση των μικροεπεξεργαστών στα αυτοκίνητα ήταν ο έλεγχος της μηχανής: προσδιορισμός του πότε το μπουζί παράγει σπινθήρα, έλεγχος του μίγματος καυσίμου/αέρα, και ούτω καθεξής. Γενικά, υπήρχε μια τάση προς τα ηλεκτρονικά στα αυτοκίνητα – ηλεκτρονικές συσκευές μπορούσαν να χρησιμοποιηθούν για να αντικατασταθεί ο μηχανολογικός διανομέας. Αλλά η μεγάλη ώθηση στον έλεγχο μηχανής που βασίζεται στο μικροεπεξεργαστή, προήλθε από δύο σχεδόν ταυτόχρονες εξελίξεις: Την πετρελαϊκή κρίση της δεκαετίας του '70 η οποία ώθησε τους καταναλωτές να δώσουν πολύ μεγαλύτερη αξία στην οικονομία καυσίμων, και το φόβο της ρύπανσης που οδήγησε σε νόμους για τον περιορισμό των εκπομπών ρύπων από τις μηχανές των αυτοκινήτων. Ο συνδυασμός χαμηλής κατανάλωσης καυσίμων και χαμηλών εκπομπών ρύπων είναι πολύ δύσκολο να επιτευχθεί. Για να ικανοποιηθούν αυτοί οι στόχοι χωρίς συμβιβασμούς στην απόδοση των μηχανών, οι κατασκευαστές αυτοκινήτων στράφηκαν προς τη χρήση σύνθετων αλγορίθμων που θα μπορούσαν να υλοποιηθούν μόνο με μικροεπεξεργαστές.

Οι μικροεπεξεργαστές έχουν διαφορετικά επίπεδα πολυπλοκότητας. Συνήθως ταξινομούνται με το μέγεθος της λέξης τους. Ένας **μικροελεγκτής** (microcontroller) 8–bit σχεδιάζεται για εφαρμογές χαμηλού κόστους και περιλαμβάνει μνήμη και συσκευές εισόδου/εξόδου· ένας μικροελεγκτής 16–bit συχνά χρησιμοποιείται για πιο σύνθετες εφαρμογές οι οποίες μπορεί να απαιτούν μεγαλύτερα μήκη λέξης ή είσοδο/έξοδο και εξωτερική (εκτός του ολοκληρωμένου κυκλώματος του μικροελεγκτή) μνήμη (off-chip memory)· και ένας μικροεπεξεργαστής μειωμένου συνόλου εντολών (Reduced Instruction Set Computer - RISC) 32–bit προσφέρει πολύ υψηλή επίδοση για υπολογιστικά εντατικές εφαρμογές.

Δεδομένης της μεγάλης ποικιλίας των τύπων μικροεπεξεργαστών που είναι διαθέσιμοι, δεν πρέπει να θεωρείται έκπληξη το γεγονός ότι οι μικροεπεξεργαστές χρησιμοποιούνται με πολλούς τρόπους. Υπάρχουν πολλές οικιακές χρήσεις των μικροεπεξεργαστών. Ο τυπικός φούρνος μικροκυμάτων έχει τουλάχιστον έναν μικροεπεξεργαστή για τον έλεγχο της λειτουργίας του. Πολλά σπίτια έχουν προηγμένα συστήματα θερμοστατών, τα οποία αλλάζουν το επίπεδο θερμοκρασίας σε διαφορετικές χρονικές στιγμές κατά τη διάρκεια της ημέρας. Η σύγχρονη φωτογραφική μηχανή είναι ένα κύριο παράδειγμα των ισχυρών χαρακτηριστικών τα οποία μπορούν να προστεθούν κάτω από τον έλεγχο ενός μικροεπεξεργαστή. Η φωτογραφική μηχανή Canon EOS 3 περιέχει τρεις μικροεπεξεργαστές, συμπεριλαμβανομένης και μιας μειωμένου συνόλου εντολών (RISC) CPU 32–bit, η οποία ελέγχει τα συστήματα αυτόματης εστίασης και ελέγχου ματιών [Pho99].

Η ψηφιακή τηλεόραση κάνει εκτενή χρήση ενσωματωμένων επεξεργαστών (embedded processors). Σε μερικές περιπτώσεις, εξειδικευμένες CPUs έχουν σχεδιαστεί για την εκτέλεση σημαντικών αλγορίθμων – ένα παράδειγμα είναι η CPU που σχεδιάστηκε για επεξεργασία ήχου στην ομάδα τσιπ (chip set) της SGS Thomson για το DirecTV [Lie98]. Αυτός ο επεξεργαστής σχεδιάστηκε για την αποδοτική υλοποίηση προγραμμάτων αποκωδικοποίησης ψηφιακού ήχου. Η προγραμματιζόμενη CPU χρησιμοποιήθηκε αντί μιας καλωδιωμένης (hardwired) μονάδας για δύο λόγους: πρώτον, έκανε ευκολότερη τη σχεδίαση και την αποσφαλμάτωση (debugging) του συστήματος και δεύτερον, επέτρεπε τη δυνατότητα ενημερώσεων (upgrades) και τη χρήση της CPU για άλλους σκοπούς.

Ένα προηγμένο αυτοκίνητο μπορεί να έχει 100 μικροεπεξεργαστές, αλλά ακόμη και τα οικονομικότερα αυτοκίνητα χρησιμοποιούν σήμερα πολλούς μικροεπεξεργαστές. Κάποιοι από αυτούς τους μικροεπεξεργαστές κάνουν πολύ απλά πράγματα όπως η ανίχνευση του αν χρησιμοποιούνται οι ζώνες ασφαλείας. Άλλοι ελέγχουν κρίσιμες λειτουργίες όπως η ανάφλεξη και τα συστήματα φρεναρίσματος. Το Παράδειγμα Εφαρμογής 1–1 περιγράφει κάποιους από τους μικροεπεξεργαστές που χρησιμοποιούνται στην BMW 850i.

Παράδειγμα Εφαρμογής 1–1

Το Σύστημα Ελέγχου Φρένων και Σταθερότητας της BMW 850i

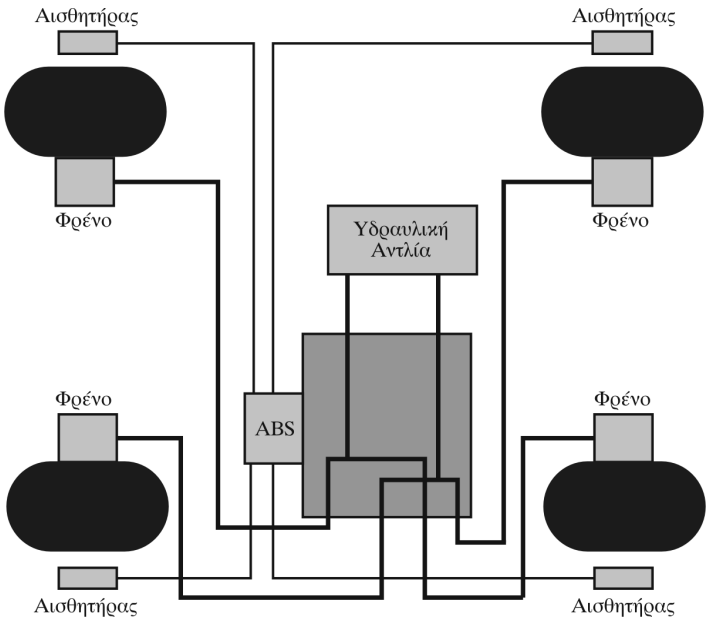
Το μοντέλο BMW 850i παρουσιάστηκε με ένα σύνθετο σύστημα για τον έλεγχο των τροχών του αυτοκινήτου. Ένα σύστημα αντιμπλοκαρίσματος τροχών (Antilock Brake System - ABS) μειώνει την ολίσθηση με την αυτόματη μεταβολή της πίεσης των φρένων. Ένα αυτόματο σύστημα ελέγχου σταθερότητας (automatic stability control - ASC+T) παρεμβαίνει στη μηχανή κατά τη διάρκεια των ελιγμών, για τη βελτίωση της σταθερότητας του αυτοκινήτου. Αυτά τα συστήματα ελέγχουν ενεργά κρίσιμα συστήματα του αυτοκινήτου· ως συστήματα ελέγχου, απαιτούν εισόδους από και έξοδο προς το αυτοκίνητο.

Αρχικά ας εξετάσουμε το ABS. Ο σκοπός ενός ABS είναι να απελευθερώνει προσωρινά το φρένο σε ένα τροχό όταν αυτός περιστρέφεται πολύ αργά – όταν ο τροχός σταματά να περιστρέφεται, το αυτοκίνητο αρχίζει να ολισθαίνει και είναι δύσκολο να ελεγχθεί. Το σύστημα τοποθετείται μεταξύ της υδραυλικής αντλίας, η οποία παρέχει ισχύ στα φρένα, και των ίδιων των φρένων όπως φαίνεται στο ακόλουθο διάγραμμα. Αυτή η διάταξη επιτρέπει στο ABS να ρυθμίζει τα φρένα έτσι ώστε οι τροχοί να μην μπλοκάρουν. Το ABS χρησιμοποιεί αισθητήρες σε κάθε τροχό για τη μέτρηση της ταχύτητας του. Οι ταχύτητες των τροχών χρησιμοποιούνται από το ABS για τον προσδιορισμό του τρόπου μετα-

βολής της πίεσης του υδραυλικού υγρού για την αποφυγή της ολίσθησης των τροχών.

Ο ρόλος του συστήματος ASC+T είναι ο έλεγχος της ισχύος του κινητήρα και των φρένων για τη βελτίωση της σταθερότητας του αυτοκινήτου κατά τη διάρκεια των ελιγμών. Το σύστημα ASC+T ελέγχει τέσσερα διαφορετικά συστήματα: την ρυθμιστική βαλβίδα, τον χρονισμό της ανάφλεξης, τα διαφορικά φρένα, και (στα αυτοκίνητα με αυτόματη μετάδοση) την αλλαγή των ταχυτήτων. Το ASC+T μπορεί να απενεργοποιηθεί από τον οδηγό, κάτι που μπορεί να είναι σημαντικό κατά την κίνηση του αυτοκινήτου με αλυσίδες.

Τα συστήματα ABS και ASC+T πρέπει ξεκάθαρα να επικοινωνούν επειδή το ASC+T σύστημα αλληλεπιδρά με το σύστημα των φρένων. Εφόσον το ABS παρουσιάστηκε αρκετά χρόνια νωρίτερα από το σύστημα ASC+T, ήταν σημαντικό να υπάρχει δυνατότητα διασύνδεσης του με την υπάρχουσα μονάδα ABS, όπως και με τις υπόλοιπες ηλεκτρονικές μονάδες. Οι μονάδες διαχείρισης μηχανής και ελέγχου περιλαμβάνουν την ηλεκτρονικά ελεγχόμενη ρυθμιστική βαλβίδα, την ψηφιακή διαχείριση της μηχανής, και τον ηλεκτρονικό έλεγχο μετάδοσης. Η μονάδα ελέγχου του συστήματος ASC+T διαθέτει δύο μικροεπεξεργαστές σε δύο κάρτες τυπωμένων κυκλωμάτων (printed circuit boards - PCBs), ο ένας από τους οποίους επικεντρώνεται σε σχετιζόμενα με τη λογική συστατικά – και ο άλλος σε συστατικά που αφορούν την απόδοση.



1.2.2 Χαρακτηριστικά Εφαρμογών Ενσωματωμένης Υπολογιστικής

Η ενσωματωμένη υπολογιστική (embedded computing) είναι από πολλές απόψεις περισσότερο απαιτητική από τα προγράμματα που μπορεί να έχετε γράψει για προσωπικούς υπολογιστές ή για σταθμούς εργασίας (workstations). Η λειτουργικότητα είναι σημαντική τόσο στην υπολογιστική γενικού σκοπού όσο και στην ενσωματωμένη υπολογιστική, αλλά οι ενσωματωμένες εφαρμογές πρέπει να ικανοποιούν πολλούς άλλους περιορισμούς επίσης.

Από τη μία πλευρά, τα ενσωματωμένα υπολογιστικά συστήματα πρέπει να παρέχουν σύνθετες λειτουργικότητες:

- *Πολύπλοκοι αλγόριθμοι*: Οι λειτουργίες που εκτελούνται από το μικροεπεξεργαστή μπορεί να είναι πολύ σύνθετες. Για παράδειγμα, ο μικροεπεξεργαστής ο οποίος ελέγχει τη μηχανή του αυτοκινήτου πρέπει να πραγματοποιήσει πολύπλοκες λειτουργίες φιλτραρίσματος για τη βελτιστοποίηση της απόδοσης του αυτοκινήτου με ταυτόχρονη ελαχιστοποίηση της ρύπανσης και της χρήσης καυσίμου.
- *Διασύνδεση με τον χρήστη (user interface)*: Οι μικροεπεξεργαστές χρησιμοποιούνται συχνά για τον έλεγχο πολύπλοκων διασυνδέσεων με το χρήστη οι οποίες μπορούν να περιλαμβάνουν πολλά μενού και πολλές επιλογές. Οι κινούμενοι χάρτες στην πλοήγηση μέσω του παγκόσμιου συστήματος εντοπισμού θέσης (Global Positioning System - GPS) είναι καλά παραδείγματα σύνθετων διασυνδέσεων χρήστη.

Για να γίνουν τα πράγματα δυσκολότερα, οι λειτουργίες ενσωματωμένης υπολογιστικής πρέπει συχνά να πραγματοποιούνται μέσα σε συγκεκριμένες προθεσμίες (deadlines):

- *Πραγματικός χρόνος (real time)*: Πολλά ενσωματωμένα υπολογιστικά συστήματα πρέπει να λειτουργούν σε πραγματικό χρόνο – αν τα δεδομένα δεν είναι έτοιμα μέχρι μια συγκεκριμένη προθεσμία, το σύστημα καταρρέει. Σε ορισμένες περιπτώσεις, η αποτυχία ικανοποίησης μιας προθεσμίας είναι επικίνδυνη και μπορεί ακόμη να βάλει σε κίνδυνο ζωές. Σε άλλες περιπτώσεις, το χάσιμο μιας προθεσμίας δεν δημιουργεί προβλήματα ασφάλειας αλλά δυσαρεστημένους πελάτες – για παράδειγμα χαμένες προθεσμίες σε εκτυπωτές μπορούν να οδηγήσουν σε μπέρδεμα σελίδων.
- *Λειτουργίες πολλαπλών ρυθμών (multirate)*: Όχι μόνο πρέπει οι λειτουργίες να ολοκληρώνονται σε συγκεκριμένες προθεσμίες, αλλά πολλά ενσωματωμένα υπολογιστικά συστήματα έχουν αρκετές δραστηριότητες πραγματικού χρόνου οι οποίες εξελίσσονται ταυτό-

χρονα. Είναι πιθανό να ελέγχουν κάποιες λειτουργίες οι οποίες εκτελούνται με αργούς ρυθμούς και άλλες που εκτελούνται με γρήγορους ρυθμούς. Οι εφαρμογές πολυμέσων (multimedia) είναι κύρια παραδείγματα συμπεριφοράς **πολλαπλών ρυθμών** (**multirate** behavior). Τα τμήματα ήχου και βίντεο ενός ρεύματος πολυμέσων (multimedia stream) εκτελούνται με πολύ διαφορετικούς ρυθμούς, αλλά πρέπει να παραμείνουν στενά συγχρονισμένα. Η αποτυχία ικανοποίησης μιας προθεσμίας είτε στο κομμάτι του ήχου είτε στο κομμάτι του βίντεο καταστρέφει την αντίληψη της συνολικής παρουσίασης.

Τα διαφορετικών τύπων κόστη είναι πολύ σημαντικά:

- *Κόστος κατασκευής (manufacturing cost)*: Το συνολικό κόστος κατασκευής ενός συστήματος είναι πολύ σημαντικό σε πολλές περιπτώσεις. Το κόστος κατασκευής προσδιορίζεται από πολλούς παράγοντες, συμπεριλαμβανομένου του τύπου του μικροεπεξεργαστή που χρησιμοποιείται, της ποσότητας της απαιτούμενης μνήμης, και του τύπου των συσκευών εισόδου/εξόδου.
- *Ισχύς (power)*: Η κατανάλωση ισχύος (power consumption) επηρεάζει άμεσα το κόστος του υλικού, εφόσον μια μεγαλύτερη τροφοδοσία ισχύος (power supply) μπορεί να είναι απαραίτητη. Αλλά η κατανάλωση ισχύος επηρεάζει επίσης και τη διάρκεια ζωής της μπαταρίας, κάτι το οποίο είναι πολύ σημαντικό σε αρκετές εφαρμογές, όπως και την κατανάλωση θερμότητας, η οποία μπορεί να είναι σημαντική ακόμα και σε επιτραπέζιες εφαρμογές.

Τέλος, τα περισσότερα ενσωματωμένα υπολογιστικά συστήματα σχεδιάζονται από μικρές ομάδες σε αυστηρές προθεσμίες. Η χρήση μικρών ομάδων σχεδίασης για συστήματα βασισμένα σε μικροεπεξεργαστές είναι μια αυτοεκπληρούμενη προφητεία – το γεγονός ότι συστήματα μπορούν να κατασκευαστούν με μικροεπεξεργαστές μόνο από μερικούς ανθρώπους σταθερά ενθαρρύνει τη διοίκηση να υποθέτει ότι όλα τα βασισμένα σε μικροεπεξεργαστές συστήματα μπορούν να κατασκευαστούν από μικρές ομάδες. Οι αυστηρές προθεσμίες είναι γεγονότα της ζωής στο σημερινό διεθνές ανταγωνιστικό περιβάλλον. Ωστόσο, η κατασκευή ενός προϊόντος με χρήση ενσωματωμένου λογισμικού (embedded software) έχει νόημα: το υλικό και το λογισμικό μπορούν να αποσφαλματωθούν (debugged) αρκετά ανεξάρτητα και σχεδιαστικές επανεκδόσεις (design revisions) μπορούν να φτιαχτούν πολύ γρηγορότερα.

1.2.3 Γιατί Χρησιμοποιούμε Μικροεπεξεργαστές;

Υπάρχουν πολλοί τρόποι σχεδίασης ενός ψηφιακού συστήματος: προσαρμοσμένη λογική (custom logic), προγραμματιζόμενες στο πεδίο συ-

στοιχίες πυλών (Field Programmable Gate Arrays – FPGAs), και ούτω καθεξής. Για ποιο λόγο χρησιμοποιούμε μικροεπεξεργαστές; Υπάρχουν δύο απαντήσεις:

- Οι μικροεπεξεργαστές είναι ένας πολύ αποδοτικός τρόπος υλοποίησης ψηφιακών συστημάτων.
- Οι μικροεπεξεργαστές κάνουν ευκολότερη τη σχεδίαση οικογενειών προϊόντων, τα οποία μπορούν να κατασκευαστούν για να παρέχουν διαφορετικά σύνολα χαρακτηριστικών σε διαφορετικές τιμές και μπορούν να επεκταθούν για να παρέχουν νέα χαρακτηριστικά ώστε να συμβαδίζουν με τις ραγδαία μεταβαλλόμενες αγορές.

Το παράδοξο της ψηφιακής σχεδίασης είναι ότι η χρησιμοποίηση ενός προσχεδιασμένου (predesigned) επεξεργαστή συνόλου εντολών (instruction set processor) μπορεί στην πραγματικότητα να οδηγήσει σε μια γρηγορότερη υλοποίηση της εφαρμογής σας σε σχέση με τη σχεδίαση της δικής σας προσαρμοσμένης λογικής. Μπαίνει εύκολα κάποιος στον πειρασμό να σκεφθεί ότι η επιβάρυνση της προσκόμισης (fetching), αποκωδικοποίησης (decoding) και εκτέλεσης (executing) εντολών είναι τόσο υψηλή που δεν μπορεί να αντισταθμιστεί. Υπάρχουν όμως δύο παράγοντες οι οποίοι μαζί καθιστούν τις βασισμένες σε μικροεπεξεργαστή σχεδιάσεις γρήγορες. Κατ' αρχήν, οι μικροεπεξεργαστές εκτελούν προγράμματα πολύ αποδοτικά. Οι σύγχρονοι επεξεργαστές μειωμένου συνόλου εντολών (Reduced Instruction Set Computer - RISC) μπορούν να εκτελέσουν μια εντολή ανά κύκλο ρολογιού τις περισσότερες φορές, ενώ οι επεξεργαστές υψηλών επιδόσεων μπορούν να εκτελέσουν αρκετές εντολές ανά κύκλο ρολογιού. Μολονότι υπάρχει επιβάρυνση για την ερμηνεία (interpreting) των εντολών, μπορεί συχνά να κρυφθεί με έξυπνη χρησιμοποίηση του παραλληλισμού (parallelism) μέσα στη CPU. Δεύτερον, οι κατασκευαστές μικροεπεξεργαστών ξοδεύουν πολλά χρήματα για να κάνουν τις CPU τους να λειτουργούν πολύ γρήγορα. Προσλαμβάνουν μεγάλες ομάδες σχεδιαστών για να ρυθμίσουν κάθε πλευρά του μικροεπεξεργαστή, ώστε να τον κάνουν να λειτουργεί στη μέγιστη δυνατή ταχύτητα. Λίγα προϊόντα μπορούν να δικαιολογήσουν δεκάδες ή και εκατοντάδες αρχιτέκτονες υπολογιστών και σχεδιαστών VLSI οι οποίοι συνήθως απασχολούνται για τη σχεδίαση ενός μικροεπεξεργαστή· τα ολοκληρωμένα κυκλώματα/τσιπ τα οποία σχεδιάζονται από μικρές ομάδες σχεδίασης είναι λιγότερο πιθανό να είναι τόσο πολύ βελτιστοποιημένα ως προς την ταχύτητα (ή την κατανάλωση ισχύος), όπως οι μικροεπεξεργαστές. Χρησιμοποιούν επίσης την πιο πρόσφατη τεχνολογία κατασκευής. Η χρήση απλά της τελευταίας γενιάς VLSI τεχνολογίας κατασκευής, αντί για μια τεχνολογία παλιότερης γενιάς, μπορεί να δημιουργήσει μια τεράστια διαφορά στην απόδοση. Οι μικροεπεξερ-

γαστές κυριαρχούν γενικά στις νέες γραμμές παραγωγής επειδή μπορούν να κατασκευαστούν σε μεγάλες ποσότητες και είναι εγγυημένο ότι θα απαιτήσουν υψηλές τιμές. Οι πελάτες οι οποίοι επιθυμούν να κατασκευάσουν τη δική τους λογική πρέπει συχνά να περιμένουν για να χρησιμοποιήσουν τη VLSI τεχνολογία της τελευταίας γενιάς των μικροεπεξεργαστών. Συνεπώς, ακόμα κι αν η λογική που σχεδιάζετε αποφεύγει όλη την επιβάρυνση της εκτέλεσης εντολών, το γεγονός ότι κατασκευάζεται από πιο αργά κυκλώματα συχνά σημαίνει ότι το πλεονέκτημα απόδοσής της είναι μικρό και ίσως ανύπαρκτο.

Προκαλεί επίσης έκπληξη το γεγονός ότι οι μικροεπεξεργαστές χρησιμοποιούν πολύ αποδοτικά τη λογική. Η γενικότητα του μικροεπεξεργαστή και η ανάγκη για ξεχωριστή μνήμη μπορεί να υπονοούν ότι οι βασισμένες σε μικροεπεξεργαστές σχεδιάσεις είναι εγγενώς πολύ μεγαλύτερες από σχεδιάσεις προσαρμοσμένης λογικής. Ωστόσο, σε πολλές περιπτώσεις ο μικροεπεξεργαστής είναι μικρότερος όταν το μέγεθος μετριέται σε μονάδες λογικών πυλών. Όταν σχεδιάζεται λογική ειδικού σκοπού (special purpose logic) για μια συγκεκριμένη λειτουργία, δεν μπορεί να χρησιμοποιηθεί για άλλες λειτουργίες. Από την άλλη, ένας μικροεπεξεργαστής μπορεί να χρησιμοποιηθεί για πολλούς διαφορετικούς αλγορίθμους απλά αλλάζοντας το πρόγραμμα το οποίο εκτελεί. Δεδομένου ότι πολλά σύγχρονα συστήματα χρησιμοποιούν πολύπλοκους αλγορίθμους και διασυνδέσεις χρήστη, θα έπρεπε γενικά να σχεδιάσουμε πολλά διαφορετικά μπλοκ προσαρμοσμένης λογικής για την υλοποίηση όλης της απαραίτητης λειτουργικότητας. Πολλά από αυτά τα τμήματα συχνά μένουν αδρανή – για παράδειγμα, η λογική της επεξεργασίας μπορεί να μένει αδρανής όταν πραγματοποιούνται λειτουργίες διασύνδεσης χρήστη. Η υλοποίηση αρκετών λειτουργιών σε έναν επεξεργαστή κάνει πολύ καλύτερη χρήση του διαθέσιμου υλικού.

Λαμβάνοντας υπόψη τα μικρά ή ανύπαρκτα κέρδη που μπορεί να υπάρξουν με την αποφυγή της χρήσης μικροεπεξεργαστών, το γεγονός ότι οι μικροεπεξεργαστές παρέχουν ουσιαστικά πλεονεκτήματα τους καθιστά την καλύτερη επιλογή σε μια μεγάλη ποικιλία συστημάτων. Η δυνατότητα προγραμματισμού των μικροεπεξεργαστών μπορεί να είναι ένα ουσιαστικό όφελος κατά τη διάρκεια της διαδικασίας σχεδίασης. Επιτρέπει το διαχωρισμό της σχεδίασης του προγράμματος (τουλάχιστον ως έναν βαθμό) από τη σχεδίαση του υλικού στο οποίο θα εκτελεστούν τα προγράμματα. Ενώ μια ομάδα σχεδιάζει την κάρτα (board) που περιέχει τον μικροεπεξεργαστή, τις συσκευές εισόδου/εξόδου, τη μνήμη, και ούτω καθεξής, οι άλλοι μπορούν να γράφουν προγράμματα την ίδια στιγμή. Εξίσου σημαντικό είναι το γεγονός ότι η δυνατότητα προγραμματισμού κάνει ευκολότερη τη σχεδίαση οικογενειών προϊόντων. Σε πολλές περιπτώσεις, προϊόντα αιχμής μπορούν να δημιουργηθούν απλά

με την προσθήκη κώδικα χωρίς αλλαγή του υλικού. Αυτή η πρακτική μειώνει ουσιαστικά το κόστος κατασκευής. Ακόμα και όταν το υλικό πρέπει να ξανασχεδιαστεί για την επόμενη γενιά προϊόντων, μπορεί να είναι δυνατή η επαναχρησιμοποίηση του λογισμικού, μειώνοντας το χρόνο ανάπτυξης και το κόστος.

1.2.4 Προκλήσεις στη Σχεδίαση Συστημάτων Ενσωματωμένης Υπολογιστικής

Οι εξωτερικοί περιορισμοί (constraints) είναι μια σημαντική πηγή δυσκολίας στη σχεδίαση ενσωματωμένων συστημάτων. Ας εξετάσουμε ορισμένα σημαντικά προβλήματα που πρέπει να ληφθούν υπόψη κατά τη σχεδίαση ενσωματωμένων συστημάτων.

Πόσο υλικό χρειαζόμαστε; Έχουμε σημαντικό έλεγχο της ποσότητας της υπολογιστικής ισχύος που εφαρμόζουμε στο πρόβλημα μας. Όχι μόνο μπορούμε να επιλέξουμε τον τύπο του μικροεπεξεργαστή που χρησιμοποιείται, αλλά μπορούμε να επιλέξουμε και την ποσότητα της μνήμης, τις περιφερειακές συσκευές, και πολλά άλλα. Εφόσον πρέπει συχνά να ικανοποιούμε τόσο περιορισμούς απόδοσης όσο και κόστους κατασκευής, η επιλογή του υλικού είναι σημαντική – αν το υλικό είναι πολύ λίγο, το σύστημά μας αποτυγχάνει να ικανοποιήσει τις προθεσμίες του, αν το υλικό είναι πάρα πολύ, το σύστημά μας γίνεται πολύ ακριβό.

Πώς ικανοποιούμε τις προθεσμίες; Ο ωμός τρόπος ικανοποίησης μιας προθεσμίας είναι η επιτάχυνση του υλικού ώστε το πρόγραμμα να εκτελείται γρηγορότερα. Φυσικά, αυτό κάνει το σύστημα ακριβότερο. Είναι επίσης πολύ πιθανό η αύξηση του ρυθμού ρολογιού της CPU να μην κάνει μεγάλη διαφορά στο χρόνο εκτέλεσης (execution time), εφόσον η ταχύτητα του προγράμματος μπορεί να περιορίζεται από το σύστημα της μνήμης.

Πώς ελαχιστοποιούμε την κατανάλωση ισχύος; Σε εφαρμογές με μπαταρίες, η κατανάλωση ισχύος είναι εξαιρετικά σημαντική. Ακόμη και σε εφαρμογές χωρίς μπαταρίες, η υπερβολική κατανάλωση ισχύος μπορεί να αυξήσει την κατανάλωση θερμότητας. Ένας τρόπος να κάνουμε ένα ψηφιακό σύστημα να καταναλώνει λιγότερη ισχύ είναι να το κάνουμε να λειτουργεί πιο αργά, αλλά η αφελής επιβράδυνση του συστήματος μπορεί προφανώς να οδηγήσει σε χαμένες προθεσμίες. Απαιτείται προσεκτική σχεδίαση για την επιβράδυνση μη κρίσιμων τμημάτων της μηχανής για την κατανάλωση ισχύος ενώ ικανοποιούνται ακόμη οι απαραίτητοι στόχοι απόδοσης.

Πώς σχεδιάζουμε για δυνατότητα αναβάθμισης (upgradeability); Η πλατφόρμα υλικού μπορεί να χρησιμοποιηθεί για αρκετές γενιές προϊόντων, ή για αρκετές διαφορετικές εκδόσεις ενός προϊόντος της ίδιας γε-

νιάς, με λίγες ή καθόλου αλλαγές. Ωστόσο, θέλουμε να μπορούμε να προσθέτουμε χαρακτηριστικά με αλλαγή του λογισμικού. Πώς μπορούμε να σχεδιάσουμε μια μηχανή η οποία θα παρέχει την απαιτούμενη απόδοση για το λογισμικό το οποίο δεν έχουμε γράψει ακόμη;

Λειτουργεί πραγματικά; Η αξιοπιστία (reliability) είναι πάντοτε σημαντική κατά την πώληση προϊόντων – οι πελάτες ορθώς περιμένουν ότι τα προϊόντα που αγοράζουν θα λειτουργούν. Η αξιοπιστία είναι ιδιαίτερα σημαντική σε ορισμένες εφαρμογές, όπως για παράδειγμα τα κρίσιμα από πλευράς ασφάλειας συστήματα (safety–critical systems). Εάν περιμένουμε μέχρι να έχουμε ένα σύστημα που λειτουργεί και προσπαθήσουμε να εξαλείψουμε τα σφάλματα, θα αργήσουμε πολύ – δεν θα βρούμε αρκετά σφάλματα, η διόρθωσή τους θα έχει αυξημένο κόστος και επίσης θα διαρκέσει πολύ. Άλλο ένα σύνολο προκλήσεων προέρχεται από τα χαρακτηριστικά των συστατικών και των ίδιων των συστημάτων. Αν ο προγραμματισμός σε έναν σταθμό εργασίας είναι σαν τη συναρμολόγηση μιας μηχανής σ' έναν πάγκο, τότε η σχεδίαση ενσωματωμένων συστημάτων μοιάζει συχνά περισσότερο με την εργασία σε ένα αυτοκίνητο – περιορισμένη, λεπτή και δύσκολη.

Ας εξετάσουμε μερικούς τρόπους με τους οποίους η φύση των ενσωματωμένων υπολογιστικών μηχανών κάνει τη σχεδίαση τους περισσότερο δύσκολη.

- *Πολύπλοκη δοκιμή (testing):* Η δοκιμή ενός ενσωματωμένου συστήματος είναι γενικά πιο δύσκολη από την πληκτρολόγηση κάποιων δεδομένων. Μπορεί να πρέπει να λειτουργήσουμε μια πραγματική μηχανή προκειμένου να παραχθούν τα κατάλληλα δεδομένα. Ο χρονισμός (timing) των δεδομένων είναι συχνά σημαντικός, γεγονός που σημαίνει ότι δεν μπορούμε να διαχωρίσουμε τη δοκιμή ενός ενσωματωμένου υπολογιστή από τη μηχανή στην οποία είναι ενσωματωμένος.
- *Περιορισμένες δυνατότητες παρατήρησης (observability) και ελέγχου (controllability):* Τα ενσωματωμένα υπολογιστικά συστήματα συνήθως δεν συνοδεύονται από πληκτρολόγια και οθόνες. Αυτό καθιστά πιο δύσκολη την παρατήρηση του τι συμβαίνει και τον επηρεασμό της λειτουργίας του συστήματος. Για παράδειγμα, μπορεί να αναγκαστούμε να παρατηρήσουμε τις τιμές των ηλεκτρικών σημάτων στον δίαυλο του μικροεπεξεργαστή, για να μάθουμε τι συμβαίνει μέσα στο σύστημα. Επιπλέον, στις εφαρμογές πραγματικού χρόνου (real time applications) είναι πιθανό να μην μπορούμε να σταματήσουμε εύκολα το σύστημα για να δούμε τι συμβαίνει στο εσωτερικό.
- *Περιορισμένα περιβάλλοντα ανάπτυξης (development environments)*: Τα περιβάλλοντα ανάπτυξης για τα ενσωματωμένα συστήματα (τα ερ-

γαλεία που χρησιμοποιούνται για την ανάπτυξη λογισμικού και υλικού) είναι συχνά πολύ περισσότερο περιορισμένα από αυτά που είναι διαθέσιμα για τους προσωπικούς υπολογιστές και τους σταθμούς εργασίας. Γενικά, μεταγλωττίζουμε κώδικα σε έναν τύπο μηχανής, όπως ένας προσωπικός υπολογιστής, και τον «κατεβάζουμε» (download) στο ενσωματωμένο σύστημα. Για να αποσφαλματώσουμε τον κώδικα, πρέπει συνήθως να βασιστούμε σε προγράμματα που εκτελούνται στον προσωπικό υπολογιστή ή στον σταθμό εργασίας και στη συνέχεια να κοιτάξουμε μέσα στο ενσωματωμένο σύστημα.

1.3 Η Διαδικασία Σχεδίασης Ενσωματωμένων Συστημάτων

Αυτή η ενότητα παρέχει μια περίληψη της διαδικασίας σχεδίασης ενσωματωμένων συστημάτων με δύο στόχους. Πρώτον, θα μας δώσει μια εισαγωγή στα διάφορα βήματα της σχεδίασης ενσωματωμένων συστημάτων, πριν εμβαθύνουμε σε αυτά με περισσότερη λεπτομέρεια. Δεύτερον, θα μας επιτρέψει να εξετάσουμε την ίδια τη **μεθοδολογία** σχεδίασης. Μια μεθοδολογία σχεδίασης είναι σημαντική για τρεις λόγους. Πρώτον, μας επιτρέπει να καταγράφουμε τα στάδια μιας σχεδίασης για να εξασφαλίσουμε ότι έχουμε κάνει ότι είναι αναγκαίο, όπως η βελτιστοποίηση της **απόδοσης** (performance) ή η πραγματοποίηση λειτουργικών δοκιμών (functional tests). Δεύτερον, μας επιτρέπει την ανάπτυξη εργαλείων σχεδίασης με τη βοήθεια υπολογιστή (computer aided design tools). Η ανάπτυξη ενός προγράμματος το οποίο δέχεται σαν είσοδο μια έννοια για ένα ενσωματωμένο σύστημα και παράγει σαν έξοδο μια πλήρη σχεδίαση θα ήταν μια τρομακτική εργασία, αλλά σπάζοντας πρώτα τη διαδικασία σε διαχειρίσιμα βήματα, μπορούμε να δουλέψουμε για την αυτοματοποίηση (ή τουλάχιστον την ημιαυτοματοποίηση) των βημάτων ένα κάθε φορά. Τρίτον, μια μεθοδολογία σχεδίασης κάνει ευκολότερη την επικοινωνία των μελών μιας ομάδας σχεδίασης. Με τον καθορισμό της συνολικής διαδικασίας, τα μέλη της ομάδας μπορούν να κατανοήσουν ευκολότερα τι είναι υποχρεωμένοι να κάνουν, τι θα πρέπει να λάβουν από άλλα μέλη της ομάδας σε συγκεκριμένους χρόνους, και τι πρέπει να παραδώσουν όταν ολοκληρώσουν τα βήματα που τους έχουν ανατεθεί. Εφόσον τα περισσότερα ενσωματωμένα συστήματα σχεδιάζονται από ομάδες, ο συντονισμός είναι ίσως ο σημαντικότερος ρόλος μιας καλά ορισμένης μεθοδολογίας σχεδίασης.

Η Εικόνα 1–1 συνοψίζει τα κυριότερα στάδια στη διαδικασία σχεδίασης ενσωματωμένων συστημάτων. Σε αυτή την από πάνω–προς–τα–κά-

τω (top-down) θεώρηση, ξεκινάμε με τις **απαιτήσεις** (requirements) του συστήματος. Στο επόμενο βήμα, την **προδιαγραφή (specification)**, δημιουργούμε μια πιο λεπτομερή περιγραφή αυτού που θέλουμε. Αλλά η προδιαγραφή δηλώνει μόνο το πώς συμπεριφέρεται το σύστημα, και όχι τον τρόπο κατασκευής του. Οι εσωτερικές λεπτομέρειες του συστήματος αρχίζουν να παίρνουν μορφή όταν αναπτύσσουμε την αρχιτεκτονική, η οποία δίνει τη δομή του συστήματος με τη μορφή μεγάλων συστατικών. Από τη στιγμή που γνωρίζουμε τα συστατικά που χρειαζόμαστε, μπορούμε να σχεδιάσουμε αυτά τα συστατικά συμπεριλαμβανομένων τόσο των μονάδων λογισμικού όσο και οποιουδήποτε εξειδικευμένου υλικού χρειαζόμαστε. Με βάση αυτές τις μονάδες, μπορούμε τελικά να κατασκευάσουμε ένα πλήρες σύστημα.

Σε αυτή την ενότητα θα εξετάσουμε τη σχεδίαση από **πάνω–προς–τα–κάτω** θα ξεκινήσουμε με την πιο αφηρημένη περιγραφή (abstract description) του συστήματος και θα καταλήξουμε με συμπαγείς λεπτομέρειες. Η εναλλακτική προσέγγιση είναι η εξέταση από **κάτω–προς–τα–πάνω** (bottom up) στην οποία ξεκινάμε με συστατικά για την κατασκευή ενός συστήματος. Τα βήματα της σχεδίασης από κάτω–προς–τα–πάνω παρουσιάζονται στην Εικόνα 1–1 σαν διακεκομμένα βέλη. Χρειαζόμαστε την από κάτω–προς–τα–πάνω σχεδίαση επειδή δεν έχουμε τέλεια αίσθηση του τρόπου εξέλιξης των τελευταίων σταδίων της διαδικασίας σχεδίασης. Οι αποφάσεις σε ένα στάδιο της σχεδίασης βασίζονται σε εκτιμήσεις για το τι θα συμβεί αργότερα: Πόσο γρήγορη μπορούμε να κάνουμε την εκτέλεση μιας συγκεκριμένης λειτουργίας; Πόση μνήμη θα χρειαστούμε; Πόση χωρητικότητα διαύλου συστήματος θα χρειαστούμε; Αν οι εκτιμήσεις μας είναι ανεπαρκείς, θα πρέπει να ακολουθήσουμε αντίστροφη πορεία (backtracking) και να τροποποιήσουμε τις αρχικές μας αποφάσεις για να λάβουμε υπόψη μας τα νέα δεδομένα. Γενικά, όσο μικρότερη εμπειρία έχουμε στη σχεδίαση παρόμοιων συστημάτων, τόσο περισσότερο θα πρέπει να στηριζόμαστε στην πληροφορία της από κάτω–προς–τα–πάνω σχεδίασης για να μας βοηθήσει στην πρόσθεση επιπλέον λεπτομερειών στο σύστημά μας.

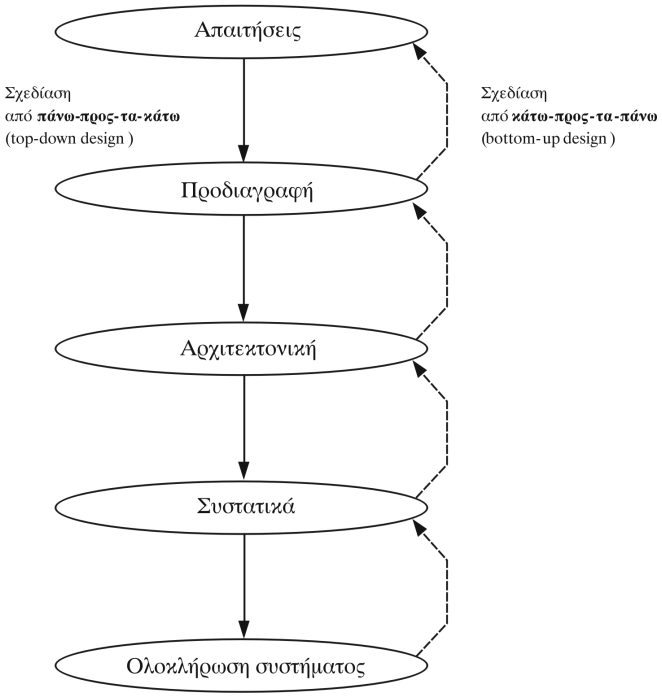
Αλλά τα επιμέρους στάδια της διαδικασίας σχεδίασης είναι μόνο ένας από τους άξονες ως προς τον οποίο μπορούμε να δούμε τη σχεδίαση ενσωματωμένων συστημάτων. Πρέπει επίσης να θεωρήσουμε τους βασικούς στόχους της σχεδίασης:

- το κόστος κατασκευής
- την απόδοση (τόσο από πλευράς συνολικής ταχύτητας όσο και από πλευράς προθεσμιών)
- την κατανάλωση ισχύος

- τη διασύνδεση με το χρήστη

Πρέπει επίσης να εξετάσουμε τις εργασίες (tasks) που εκτελούνται σε κάθε βήμα της διαδικασίας σχεδίασης. Σε κάθε βήμα της σχεδίασης, προσθέτουμε λεπτομέρειες:

- Πρέπει να *αναλύσουμε* τη σχεδίαση σε κάθε βήμα για να καθορίσουμε με ποιο τρόπο μπορούμε να ικανοποιήσουμε όλες τις προδιαγραφές.
- Πρέπει έπειτα να *εκλεπτύνουμε* (refine) τη σχεδίαση για να προσθέσουμε λεπτομέρειες.
- Και πρέπει να *επαληθεύσουμε* (verify) τη σχεδίαση για να εξασφαλίσουμε ότι ικανοποιεί όλους τους στόχους του συστήματος, όπως το κόστος, η ταχύτητα, και ούτω καθεξής.



Εικόνα 1–1 *Τα κύρια επίπεδα αφαίρεσης στη διαδικασία σχεδίασης.*

1.3.1 Απαιτήσεις

Είναι σαφές ότι προτού ξεκινήσουμε να σχεδιάσουμε ένα σύστημα, πρέπει να ξέρουμε τι σχεδιάζουμε. Η πληροφορία αυτή καταγράφεται στα αρχικά στάδια της διεργασίας σχεδίασης, και χρησιμοποιείται για τη δη-