Μέρος Ι

ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ

Το πρώτο μέρος του παρόντος βιβλίου εισάγει σημαντικές έννοιες στη συσχεδίαση υλικού-λογισμικού. Συγκρίνουμε και αντιπαραβάλλουμε δύο σχολές σκέψης στον ηλεκτρονικό σχεδιασμό: τη νοοτροπία που χρησιμοποιείται από τον σχεδιαστή υλικού, σε αντίθεση με τη νοοτροπία που χρησιμοποιεί ο σχεδιαστής λογισμικού. Θα δείξουμε ότι η συσχεδίαση υλικού/λογισμικού δεν είναι απλά η συνένωση μεταξύ εξαρτημάτων υλικού και λογισμικού. Αντίθετα, πρόκειται για την εξεύρεση της σωστής ισορροπία μεταξύ ευελιξίας και απόδοσης κατά το σχεδιασμό.

Ο συμβιβασμός μεταξύ παράλληλων και ακολουθιακών υλοποιήσεων είναι ένα άλλο θεμελιώδες ζητούμενο για τον συσχεδιαστή υλικού/λογισμικού. Θα συζητήσουμε ένα μοντέλο ταυτόχρονου συστήματος (ροή δεδομένων), που μπορεί να μετατραπεί είτε μια υλοποίηση σε υλικό (παράλληλο) είτε σε υλοποίηση σε λογισμικό (ακολουθιακό).

Τέλος, θα δείξουμε πώς ένα πρόγραμμα στο C μπορεί να αναλυθεί και να αποσυντεθεί σε ροή ελέγχου και ροή δεδομένων. Αυτή η ανάλυση είναι ζωτικής σημασίας για να κατανοήσουμε πώς ένας πρόγραμμα σε C μπορεί να μεταφερθεί σε υλικό. Όπως θα συζητήσουμε, μια συνήθης προσέγγιση συσχεδίασης υλικού/λογισμικού είναι να μεταφέρουμε λειτουργικότητα από το λογισμικό στο υλικό, βελτιώνοντας έτσι τη συνολική απόδοση της εφαρμογής.

Κεφάλαιο 1

Η Φύση του Υλικού και του Λογισμικού

1.1 Εισαγωγή στη Συσχεδίαση Υλικού/Λογισμικού

Η Συσχεδίαση Υλικού/Λογισμικού είναι ένας ευρύς όρος που ενσωματώνει πολλά διαφορετικά στοιχεία του σχεδιασμού ηλεκτρονικού συστήματος. Ξεκινάμε παρέχοντας έναν απλό ορισμό του λογισμικού και του υλικού. Δεν είναι καθολικός ορισμός, αλλά θα βοηθήσει να τεθεί μια κοινή γραμμή για αναγνώστες με διαφορετικά υπόβαθρα. Αυτή η παράγραφος θα παρέχει επίσης ένα μικρό παράδειγμα συσχεδίασης υλικού/ λογισμικού και ολοκληρώνεται με ένα ορισμό της συσχεδίασης υλικού/λογισμικού.

1.1.1 Υλικό

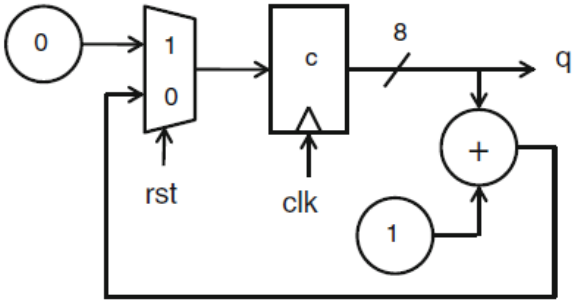
Σε αυτό το βιβλίο, θα μοντελοποιήσουμε το υλικό μέσω σύγχρονων ψηφιακών κυκλωμάτων μονού-ρολογιού, που δημιουργούνται χρησιμοποιώντας συνδυαστική λογική και flip-flops. Τέτοια κυκλώματα μπορούν να μοντελοποιηθούν με δομικά στοιχεία όπως για παράδειγμα καταχωρητές, αθροιστές και πολυπλέκτες. Η βασισμένη σε κύκλους μοντελοποίηση υλικού καλείται συχνά μοντελοποίηση επιπέδου καταχωρητή μεταφοράς- (register-transfer-level -RTL), επειδή μπορεί η συμπεριφορά ενός κυκλώματος να θεωρηθεί ως ακολουθία μεταφορών μεταξύ καταχωρητών, με λογικές και αριθμητικές πράξεις να εκτελούνται στα σήματα κατά τη διάρκεια των μεταφορών.

Το Εικόνα 1.1α δίνει ένα παράδειγμα μιας μονάδας υλικού που αποτυπώθηκε σε RTL. Ένας καταχωρητής μπορεί να αυξάνεται ή να καθαρίζεται ανάλογα με την τιμή του σήματος ελέγχου `rst`. Ο καταχωρητής ενημερώνεται στις ανοδικές ακμές ενός σήματος ρολογιού `clk`. Το μήκος λέξης του καταχωρητή είναι 8 bit. Παρόλο που οι συνδέσεις σε αυτή την εικόνα έχουν σχεδιαστεί ως μονές γραμμές, κάθε γραμμή αντιπροσωπεύει μια δέσμη οκτώ καλωδίων. Η Εικόνα 1.1α χρησι-

μοποιεί τα γραφικά για να αποτυπώσει το κύκλωμα. Σε αυτό το βιβλίο, θα χρησιμοποιήσουμε μια γλώσσα περιγραφής υλικού που ονομάζεται GEZEL. Η Εικόνα 1.1 1.1b δείχνει την ισοδύναμη περιγραφή αυτού του κυκλώματος στη Γλώσσα GEZEL. Το Κεφάλαιο 5 θα περιγράψει λεπτομερώς τη μοντελοποίηση σε GEZEL.

Η Εικόνα 1.11.1c απεικονίζει τη συμπεριφορά του κυκλώματος χρησιμοποιώντας ένα διάγραμμα χρονισμού. Σε ένα τέτοιο διάγραμμα, ο χρόνος τρέχει από αριστερά προς τα δεξιά και οι γραμμές του διαγράμματος αντιπροσωπεύουν διαφορετικά σήματα στο κύκλωμα.

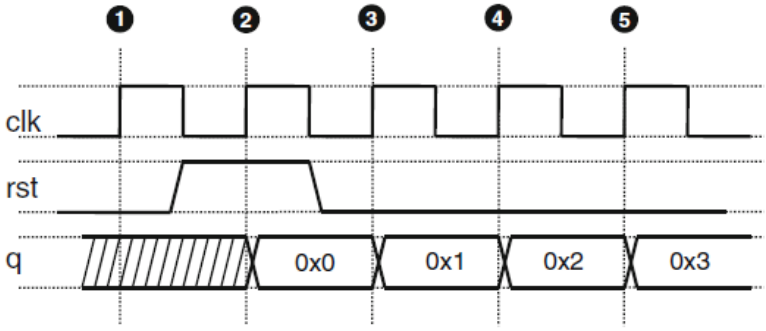
a



b

```
dp count(in  rst : ns(1);
         out q   : ns(8)) {
  reg c : ns(8);
  always {
    q = c;
    c = rst ? 0 : c + 1;
  }
}
```

c



Εικόνα 5-1 *Εξαρτήματα υλικού*

Σε αυτό το διάγραμμα, ο καταχωρητής καθαρίζεται στην ακμή ρολογιού 2, και αυξάνεται στην ακμή ρολογιού 3, 4 και 5. Πριν από την ακμή ρολογιού 2, η τιμή του καταχωρητή είναι άγνωστη και το διάγραμμα χρονισμού υποδεικνύει την τιμή του q ως σκιασμένη περιοχή. Θα χρησιμοποιήσουμε διαγράμματα χρονισμού για να περιγράψουμε τη χαμηλού επιπέδου συμπεριφορά των διασυνδέσεων υλικού-λογισμικού και για να περιγράψουμε γεγονότα στους επί-τουολοκληρωμένου διαύλους (on-chip buses).

Το μοντέλο μονού-ρολογιού είναι μια πολύ βολική αφαίρεση για έναν σχεδιαστή που απεικονίζει συμπεριφορές (π.χ. κάποιο αλγόριθμο) σε διακριτά βήματα του ενός κύκλου ρολογιού. Αυτό επιτρέπει στο σχεδιαστή να οραματιστεί πώς θα πρέπει να μοιάζει η υλοποίηση σε υλικό ενός συγκεκριμένου αλγορίθμου. Το σύγχρονο μοντέλο μονού-ρολογιού δεν μπορεί να περιγράψει όλα τα δυνατά κύκλωμα υλικού. Για παράδειγμα, δεν μπορεί να μοντελοποιήσει γεγονότα σε χρονική ανάλυση μικρότερη από έναν κύκλο ρολογιού. Ως αποτέλεσμα, ορισμένα στυλ του σχεδιασμού υλικού δεν μπορούν να αποτυπωθούν με ένα σύγχρονο μοντέλο μονού-ρολογιού, συμπεριλαμβάνοντας το ασύγχρονο υλικό, δυναμική λογική, υλικό με ρολόι πολλών-φάσεων, και υλικό με μανταλωτές. Ωστόσο, το σύγχρονο υλικό μονού-ρολογιού επαρκεί για να εξηγήσει τις βασικές έννοιες της συσχεδίασης υλικού-λογισμικού σε αυτό το βιβλίο.

Λίστα 1.1 Παράδειγμα C

```
1  int   max;
2
3  int   findmax(int a[10]) {
4     unsigned i;
5     max = a[0];
6      for (i=1;  i<10;  i++)
7        if (a[i]  >  max)  max  =  a[i];
8     }
```

1.1.2 Λογισμικό

Η συσχεδίαση υλικού / λογισμικού ασχολείται με διασυνδέσεις υλικού / λογισμικού. Οι λεπτομέρειες κατασκευής χαμηλού επιπέδου του λογισμικού είναι σημαντικές, επειδή επηρεάζουν άμεσα την απόδοση και το κόστος υλοποίησης της διασύνδεσης υλικού/λογισμικού. Αυτό το βιβλίο θα εξετάσει σημαντικές πτυχές υλοποίησης του λογισμικού, όπως η οργάνωση των μεταβλητών στη μνήμη και τις τεχνικές για τον έλεγχό της μέσα από μια γλώσσα προγραμματισμού υψηλού-επιπέδου, όπως η C.

Θα μοντελοποιήσουμε το λογισμικό ως ακολουθιακά προγράμματα μονού-νήματος, γραμμένα σε C ή συμβολική γλώσσα. Τα προγράμματα θα απεικονίζονται χρησιμοποιώντας λίστες, όπως για παράδειγμα οι Λίστες 1.1 και 1.2. Οι περισσότερες συζητήσεις σε αυτό το βιβλίο θα είναι ανεξάρτητες του επεξεργαστή. Σε ορισμένες περιπτώσεις, θα υποθέτουμε μια αρχιτεκτονική 32-bit (π.χ. ARM) ή μια αρχιτεκτονική 8-bit (π.χ. 8051).

Ένα ακολουθιακό πρόγραμμα C μονού-νήματος έχει εκπληκτικά καλή αντιστοιχία με την πραγματική εκτέλεση του προγράμματος αυτού σε έναν τυπικό μικροεπεξεργαστή. Για παράδειγμα, η ακολουθιακή εκτέλεση των προγραμμάτων C αντιστοιχεί στον κύκλο ανάκλησης-και-εκτέλεσης ακολουθιακής εντολής των μικροεπεξεργαστών. Οι μεταβλητές της C αποθηκεύονται σε ένα ενιαίο, κοινόχρηστο χώρο μνήμης, που αντιστοιχεί στη μνήμη που είναι συνδεδεμένη στον μικροεπεξεργαστή. Υπάρχει μια στενή αντιστοιχία μεταξύ των εννοιών αποθήκευσης ενός μικροεπεξεργαστή (καταχωρητές, στοίβα) και στους τύπους αποθήκευσης που υποστηρίζονται στην C (`register int`, τοπικές μεταβλητές). Επιπλέον, οι συνήθεις τύποι δεδομένων στην C (`char`, `int`) απεικονίζονται απευθείας σε μονάδες αποθήκευσης μικροεπεξεργαστή (byte, λέξη). Κατά συνέπεια, μια λεπτομερής κατανόηση της εκτέλεσης σε C, συνδέεται στενά με μια λεπτομερή κατανόηση της δραστηριότητας μικροεπεξεργαστή σε χαμηλότερο επίπεδο αφαίρεσης.

Φυσικά, υπάρχουν πολλές μορφές λογισμικού που δεν ταιριάζουν με το μοντέλο του μονού-νήματος ακολουθιακού προγράμματος C. Το λογισμικό πολλαπλών νημάτων, για παράδειγμα, δημιουργεί την ψευδαίσθηση του ταυτοχρονισμού (concurrency) και επιτρέπει στους χρήστες να εκτελούν πολλά προγράμματα ταυτόχρονα. Άλλες μορφές λογισμικού, όπως το αντικειμενοστρεφές λογισμικό και ο λειτουργικός προγραμματισμός, αντικαθιστούν το απλό μοντέλο μηχανής του μικροεπεξεργαστή με ένα πιο εξελιγμένο. Τέτοιες πιο προηγμένες μορφές λογισμικού είναι ζωτικής σημασίας για τον έλεγχο της πολυπλοκότητας των μεγάλων εφαρμογών λογισμικού. Ωστόσο, κάνουν αφαίρεση (δηλ. απόκρυψη) των δραστηριοτήτων εντός του μικροεπεξεργαστή. Για το λόγο αυτό, θα επικεντρωθούμε στην απλή, μονού-νήματος C.

Λίστα 1.2: Παράδειγμα συμβολικού κώδικα ARM

```
            .text
findmax:
            ldr   r2, .L10
            ldr   r3, [r0, #0]
            str   r3, [r2, #0]
            mov   ip, #1
```

```
.L7:
            ldr   r1, [r0, ip, asl #2]
            ldr   r3, [r2, #0]
            add   ip, ip, #1
            cmp   r1, r3
            strgt       r1, [r2, #0]
            cmp   ip, #9
            movhi       pc, lr
            b     .L7
.L11:
            .align      2
.L10:
            .word       max
```

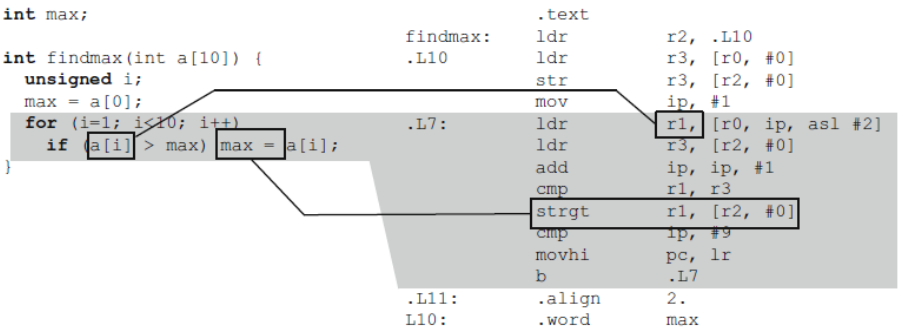
Το υλικό του βιβλίου δεν ακολουθεί κάποιο συγκεκριμένο μικροεπεξεργαστή και δεν γνωρίζει (agnostic) οποιοδήποτε συγκεκριμένο τύπο συμβολικής γλώσσας (assembly language). Το βιβλίο δίνει έμφαση στη σχέση μεταξύ της C και του συμβολικού κώδικα και υποθέτει ότι ο αναγνώστης είναι εξοικειωμένος με την έννοια του συμβολικού κώδικα. Ορισμένα προβλήματα βελτιστοποίησης στη συσχεδίαση υλικού/λογισμικού μπορεί να αντιμετωπιστούν μόνο σε επίπεδο συμβολικού κώδικα. Σε αυτή η περίπτωση, ο σχεδιαστής πρέπει να είναι σε θέση να συνδέσει το λογισμικό, όπως αποτυπώνεται σε C, με το πρόγραμμα που εκτελείται στον επεξεργαστή, όπως αναπαριστάται από τον συμβολικό κώδικα. Οι περισσότεροι μεταγλωττιστές C προσφέρουν τη δυνατότητα παραγωγής λίστας συμβολικής γλώσσας για τον παραγόμενο κώδικα, και θα κάνουμε χρήση αυτού του χαρακτηριστικού. Η Λίστα 1.2 για παράδειγμα, δημιουργήθηκε από Λίστα 1.1.

Η σύνδεση των εντολών ενός προγράμματος C με τις εντολές συμβολικού προγράμματος είναι ευκολότερη από ό, τι νομίζετε, ακόμα κι αν δεν γνωρίζετε τον μικροεπεξεργαστή στον οποίο στοχεύει το συμβολικό πρόγραμμα. Για παράδειγμα, συγκρίνετε τις Λίστες 1.1 και 1.2. Μια ιδανική εκκίνηση όταν αντιστοιχίζετε ένα πρόγραμμα C σε ένα συμβολικό πρόγραμμα, είναι να αναζητήσετε παρόμοια δομές: οι βρόχοι της C θα απεικονίζονται μέσα από τις εντολές άλματος στη συμβολική γλώσσα. Οι if-then-else εντολές της C θα απεικονίζονται, στη συμβολική γλώσσα, ως διακλαδώσεις υπό συνθήκη και ετικέτες. Ακόμα κι αν δεν είστε εξοικειωμένοι με τη μορφή συμβολικού προγράμματος ενός συγκεκριμένου μικροεπεξεργαστή, μπορείτε εύκολα να αντλήσετε τέτοιες δομές.

Η Εικόνα 1.2 δίνει ένα παράδειγμα για τα προγράμματα στις Λίστες 1.1 και 1.2. Ο βρόχος `for` στη C σημειώνεται με μια ετικέτα και μια εντολή διακλάδωσης. Όλες

οι συμβολικές εντολές μεταξύ της διακλάδωσης και της ετικέτας είναι μέρος του σώματος του βρόχου. Μόλις εντοπιστεί η δομή βρόχου, είναι εύκολο να εξαχθεί το υπόλοιπο του κώδικα, όπως δείχνουν τα ακόλουθα παραδείγματα.

- Η εντολή if στη C απαιτεί τον υπολογισμό μιας συνθήκης μεγαλύτερο από. Στη συμβολική γλώσσα, μπορεί να βρεθεί μια ισοδύναμη εντολή cmp (σύγκριση).



Εικόνα 1.2 *Απεικόνιση C σε συμβολική γλώσσα*

Αυτό δείχνει ότι οι τελεστές `r1` και `r3` της εντολής σύγκρισης πρέπει να περιέχουν το `a[i]` και το `max` του προγράμματος C. Και οι δύο μεταβλητές είναι αποθηκευμένες στη μνήμη. Το `a[i]` επειδή αποτελεί μια δεικτοδοτημένη μεταβλητή, και το `max` επειδή είναι καθολική μεταβλητή. Πράγματι, εξετάζοντας την προηγούμενη εντολή στο πρόγραμμα C, μπορείτε να δείτε ότι τόσο η `r1` και η `r3` ορίζονται από τις εντολές `ldr` (register-load – φορτώματος καταχωρητή) που απαιτούν μια διεύθυνση.

- Η διεύθυνση για την φόρτωση του `r1` ισούται με `[r0, ip, asl # 2]`, που ισοδυναμεί με την έκφραση `r0 + (ip << 2)`. Αυτό μπορεί να μην είναι προφανές αν είναι η πρώτη φορά που βλέπετε τη συμβολική γλώσσα του ARM, αλλά είναι κάτι που γρήγορα θα θυμόσαστε. Στην πραγματικότητα, η μορφή της έκφρασης είναι εύκολο να εξηγηθεί. Ο καταχωρητής `ip` περιέχει τον μετρητή βρόχου, δεδομένου ότι το `ip` αυξάνεται μία φορά εντός του σώματος του βρόχου, και η τιμή του `ip` συγκρίνεται με την τιμή ορίου βρόχου 9. Ο καταχωρητής `r0` είναι η *διεύθυνση βάσης (base address)* του πίνακα `a[]`, η θέση στη μνήμη όπου αποθηκεύεται το `a[0]`. Η ολίσθηση κατά 2 είναι απαραίτητη επειδή ο `a []` είναι ένας πίνακας ακεραίων. Οι μικροεπεξεργαστές χρησιμοποιούν μνήμη διευθυνσιο-

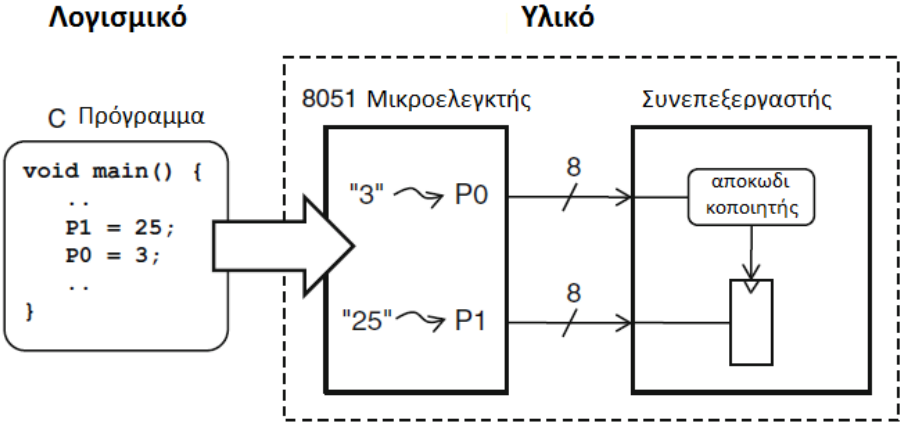
δοτημένη σε επίπεδο byte (byte-addressable), και οι ακέραιοι αποθηκεύονται σε τμήμα με τοποθεσίες 4 byte.

- Τέλος, η εκχώρηση υπό όρους της μεταβλητής `max` στη C δεν υλοποιείται με τη χρήση εντολών διακλάδωσης υπό συνθήκη στη συμβολική γλώσσα. Αντ ' αυτού, μια εντολή `strgt` (store-if-greater – αποθήκευσε εάν μεγαλύτερο) χρησιμοποιείται. Αυτή είναι μια *εκτελούμενη υπό συνθήκη (predicated)* εντολή, μια εντολή που εκτελείται μόνο όταν ισχύει μια δεδομένη σημαία συνθήκης είναι αληθής.

Στο τέλος αυτής της ανάλυσης προκύπτει ότι, με ελάχιστη προσπάθεια, θα μπορείτε να κατανοήσετε πολλά για τη συμπεριφορά ενός μικροεπεξεργαστή απλά συγκρίνοντας προγράμματα C με ισοδύναμα συμβολικά προγράμματα. Στο Κεφ. 7, θα χρησιμοποιήσετε την ίδια προσέγγιση για να αναλύσετε την ποιότητα του συμβολικού κώδικα που παράγεται από ένα μεταγλωττιστή από τον κώδικα σε C.

1.1.3 Υλικό και Λογισμικό

Ο στόχος αυτού του βιβλίου είναι να συζητηθεί ο συνδυασμός του σχεδιασμού υλικού και του σχεδιασμό λογισμικού σε όλες τις μορφές του. Το υλικό και το λογισμικό μπορούν να μοντελοποιηθούν χρησιμοποιώντας RTL και προγράμματα C αντίστοιχα. Ο όρος *μοντέλο* δείχνει απλώς ότι αυτά δεν αποτελούν την πραγματική υλοποίηση, αλλά μόνο μια αναπαράστασή του. Ένα πρόγραμμα RTL είναι ένα *μοντέλο* ενός δικτυώματος λογικών πυλών. Ένα πρόγραμμα C είναι ένα *μοντέλο* μιας δυαδικής εικόνας των εντολών μικροεπεξεργαστή. Δεν είναι σύνηθες να αναφερόμαστε σε προγράμματα C ως μοντέλα. Στην πραγματικότητα, οι σχεδιαστές λογισμικού σκέφτονται τα προγράμματα C ως πραγματικές υλοποιήσεις. Στο βιβλίο αυτό, θα αναφερόμαστε επομένως, σε *μοντέλα* υλικού και σε *προγράμματα* σε C ή συμβολική γλώσσα.



Εικόνα 1.3 *Ένα μοντέλο συσχεδίασης*

Τα μοντέλα αποτελούν ουσιαστικό μέρος της διαδικασίας σχεδιασμού. Αποτελούν μια επίσημη αναπαράσταση της πρόθεσης των σχεδιαστών και χρησιμοποιούνται ως είσοδος για εργαλεία προσομοίωσης και εργαλεία υλοποίησης. Στη συσχεδίαση υλικού/λογισμικού, εργαζόμαστε με μοντέλα που είναι εν μέρει γραμμένα ως προγράμματα C, και εν μέρει ως προγράμματα RTL. Θα συζητήσουμε αυτή την ιδέα μέσω ενός απλού παραδείγματος.

Στην Εικόνα 1.3 παρουσιάζεται ένας μικροελεγκτής 8051 και ένας συνδεδεμένος συνεπεξεργαστής. Ο συνεπεξεργαστής είναι συνδεδεμένος στον μικροελεγκτή 8051 μέσω δύο θυρών 8-bit P0 και `P1`. Ένα πρόγραμμα C εκτελείται στον μικροελεγκτή 8051 και αυτό το πρόγραμμα περιέχει εντολές για την εγγραφή δεδομένων σε αυτές τις δύο θύρες. Όταν εμφανίζεται μια δεδομένη, προκαθορισμένη τιμή στη θύρα P0, ο συνεπεξεργαστής θα κάνει ένα αντίγραφο της τιμής που βρίσκεται στη θύρα P1 σε ένα εσωτερικό καταχωρητή.

Αυτός ο πολύ απλός σχεδιασμός μπορεί να αντιμετωπιστεί χρησιμοποιώντας συσχεδίαση υλικού/λογισμικού. Περιλαμβάνει το σχεδιασμό ενός μοντέλου υλικού και το σχεδιασμό ενός προγράμματος C. Το μοντέλο υλικού περιέχει τον επεξεργαστή 8051, τον συνεπεξεργαστή και τις συνδέσεις μεταξύ τους. Κατά τη διάρκεια της εκτέλεσης, ο επεξεργαστής 8051 θα εκτελέσει ένα πρόγραμμα λογισμικού γραμμένο σε C. Η Λίστα 1.3 δείχνει αυτό το πρόγραμμα C. Η Λίστα 1.4 δείχνει ένα μοντέλο υλικού RTL για αυτό το σχεδιασμό, γραμμένο στη γλώσσα GEZEL.

Ο οδηγός C στέλνει τρεις τιμές στη θύρα `P1`, καλώντας μια συνάρτηση `sayhello`. Αυτή η συνάρτηση επίσης μεταφέρει κυκλικά την τιμή στη θύρα P0

μεταξύ των `ins_hello` και `ins_idle`, οι οποίες κωδικοποιούνται ως τιμές 1 και 0 αντίστοιχα.

Το μοντέλο υλικού περιλαμβάνει τόσο το μικροελεγκτή όσο και το συνεπεξεργαστή. Η συνολική συμπεριφορά του συνεπεξεργαστή στην Λίστα 1.4 είναι όπως αυτή: όταν αλλάζει η είσοδος `ins` από 0 έως 1, τότε η είσοδος `din` θα εκτυπωθεί στον επόμενο κύκλο ρολογιού.

Λίστα 1.3 Πρόγραμμα οδηγού 8051

```
1    #include <8051.h>
2
3    enum {ins_idle, ins_hello};
4
5    void sayhello(char d)  {
6      P1 = d;
7      P0 = ins_hello;
8      P0 = ins_idle;
9    }
10
11   void terminate()  {
12     // special command to stop simulator
13     P3 = 0x55;
14   }
15
16   void main()   {
17     sayhello(3);
18     sayhello(2);
19     sayhello(1);
20     terminate();
21   }
```

Ο συνεπεξεργαστής βρίσκεται στις γραμμές 1-19. Αυτό το συγκεκριμένο μοντέλο υλικού είναι ένας συνδυασμός μιας μηχανής πεπερασμένων καταστάσεων (γραμμές 10-18) και ενός διαδρόμου δεδομένων (γραμμές 1-9), μια μέθοδος μοντελοποίησης γνωστή ως FSMD (μηχανή πεπερασμένων καταστάσεων με διάδρομο δεδομένων). Θα συζητήσουμε την FSMD λεπτομερώς στο Κεφάλαιο 5. Η FSMD είναι αρκετά εύκολο να κατανοηθεί. Ο διάδρομος δεδομένων περιέχει διάφορες εντολές: `decode` και `hello`. Ο ελεγκτής FSM επιλέγει, σε κάθε κύκλο ρολογιού, ποια από αυτές τις εντολές θα εκτελεστεί. Για παράδειγμα, οι γραμμές 15-16 εμφανίζουν την ακόλουθη εντολή ελέγχου.

```
@s1 if (insreg == 1)  then (hello, decode)  -> s2;
                      else (decode)         -> s1;
```