1 Συστήματα Βασισμένα σε FPGA

Τα FPGAs στο Σχεδιασμό Συστημάτων

FPGAs vs ASICs

Μεθοδολογίες Σχεδιασμού

1.1 Εισαγωγή

Αυτό το κεφάλαιο θέτει το πλαίσιο αναφοράς ολόκληρου του βιβλίου. Στην επόμενη παράγραφο γίνεται αναφορά των βασικών αρχών της Boolean άλγεβρας και των συμβόλων των στοιχείων που χρησιμοποιεί. Στη παράγραφο 1.3 εισάγονται οι Δομές Πίνακα Πυλών Επαναδιατάξιμης Λογικής (Field Programmable Gate Arrays- FPGAs) και περιγράφεται η σπουδαιότητά τους και τέλος στη παράγραφο 1.4 περιγράφεται πως χρησιμοποιούνται τα FPGAs για το σχεδιασμό σύνθετων ψηφιακών συστημάτων.

1.2 Βασικές Αρχές

Αυτή η παράγραφος εισάγει τις βασικές έννοιες της λογικής σχεδίασης. Αν αυτές οι έννοιες αναθεωρηθούν από τον αναγνώστη θα τον βοηθήσουν στη καθιέρωση της ορολογίας που θα χρησιμοποιηθεί στο υπόλοιπο του βιβλίου.

1.2.1 Boolean Άλγεβρα

Η Boolean άλγεβρα χρησιμοποιείται για την αναπαράσταση των λογικών συναρτήσεων των ψηφιακών κυκλωμάτων. Ο Shannon [Sha38] έδειξε ότι ένα δίκτυο από διακόπτες μπορεί να αναπαρασταθεί από μια Boolean συνάρτηση. Σήμερα δε θεωρείται ότι οι λογικές πύλες αποτελούνται από διακόπτες, αλλά παρόλα αυτά θεωρείται ότι η Boolean άλγεβρα είναι μια θεμελιώδης αναπαράσταση. Η Boolean άλγεβρα χρησιμοποιείται για τη περιγραφή **συνδυαστικών** λογικών συναρτήσεων. Οι

Boolean συναρτήσεις περιγράφουν συνδυασμούς εισόδων και δεν χρησιμοποιούν εκφράσεις υπαρξιακής ($\exists x\ f(x)$) ή καθολικής ($\forall x\ g(x)$) ποσοτικοποίησης.

Για τις λογικές εκφράσεις χρησιμοποιούνται απλοί συμβολισμοί: Έαν οι a και b είναι δυαδικές μεταβλητές, τότε η a' (ή $\overline{a}$) είναι το συμπλήρωμα του a, $a \cdot b$ (ή ab) είναι το λογικό AND των δύο μεταβλητών, και $a + b$ είναι το λογικό OR των δύο μεταβλητών. Επίσης, για την NAND λογική συνάρτηση $(ab)'$ χρησιμοποιείται το σύμβολο[1] |, για τη NOR λογική συνάρτηση $(a + b)'$ χρησιμοποιείται η έκφραση $a\ NOR\ b$, και για τη λογική συνάρτηση του αποκλειστικού-Η (XOR) χρησιμοποιείται η έκφραση $a\ XOR\ b = ab' + a'b$ και το σύμβολο $\oplus$. Η Εικόνα 1-1 συνοψίζει τα ονόματα και τα σύμβολα των πιο συνηθισμένων λογικών συναρτήσεων.

Όνομα	**Σύμβολο**
NOT	**‘ , ~**
AND	**. , ^ , &**
NAND	**|**
OR	**+ , v**
NOR	**NOR**
XOR	$\oplus$
XNOR	**XNOR**

Εικόνα 1-1 *Τα σύμβολα των συναρτήσεων σε μορφή λογικών εκφράσεων*

Για την ευθεία έκφραση χρησιμοποιείται ο συμβολισμός (a) και για τη συμπληρωματική έκφραση χρησιμοποιείται ο συμβολισμός (a') μιας μεταβλητής. Για τη κατανόηση της σχέσης μεταξύ των λογικών εκφράσεων και των λογικών πυλών θα μελετηθούν κάποια απλά προβλήματα.

Παρακάτω θα γίνει μια ανασκόπηση των αξιωμάτων της άλγεβρας Boole που μπορούν να χρησιμοποιηθούν στο μετασχηματισμό εκφράσεων. Κάποια από αυτά είναι παρόμοια με αυτούς που ισχύουν στην αριθμητική άλγεβρα ενώ κάποια άλλα ισχύουν μόνο στη Boolean άλγεβρα:

- **Αυτοδυναμία:** $a \cdot a = a$, $a + a = a$.
- **Συμπλήρωμα**: $a + a' = 1$, $a \cdot a' = 0$.
- **Ουδέτερο στοιχείο**: $a + 0 = a$, $a \cdot 1 = a$.
- **Αντιμεταθετικότητα**: $a + b = b + a$, $a \cdot b = b \cdot a$.

[1] Ο Scheffer stroke συμβολισμός είναι μια τελεία με μια κάθετη γραμμή. Οι προγραμματιστές σε C γνωρίζουν ότι αυτό το σύμβολο χρησιμοποιείται για τη πύλη OR στη γλώσσα C.

- **Πράξεις με ουδέτερα στοιχεία**: $a \cdot 0 = 0$, $a + 1 = 1$.
- **Διπλή άρνηση**: $(a')' = a$.
- **Απορρόφηση**: $a + ab = a$.
- **Προσεταιριστικότητα**: $a + (b + c) = (a + b) + c$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.
- **Επιμεριστικότητα**: $a \cdot (b + c) = a \cdot b + a \cdot c$, $a + bc = (a + b)(a + c)$.
- **Κανόνες De Morgan**: $(a + b)' = a' \cdot b'$, $(a \cdot b)' = a' + b'$.

Αν και η Boolean άλγεβρα φαίνεται απλή, κάποια από τα μαθηματικά αποτελέσματα παρέχουν άμεση σχέση με τις φυσικές ιδιότητες των λογικών κυκλωμάτων. Τα δύο βασικά προβλήματα της άλγεβρας Boole στη λογική σχεδίαση είναι η **πληρότητα** (**completeness**) και ο **μη-πλεονασμός** (**irredundancy)**.

Ένα σύνολο λογικών συναρτήσεων είναι **πλήρες**, εάν είναι δυνατόν να δημιουργηθεί οποιοσδήποτε συνδυασμός Boolean εκφράσεων χρησιμοποιώντας αυτό το σύνολο των συναρτήσεων, ή αλλιώς για κάθε πιθανή συνάρτηση παραγόμενη από έναν τυχαίο συνδυασμό των λογικών πράξεων +, $\cdot$ και ‘ υπάρχει ένας ισοδύναμος τύπος με χρήση των λογικών συναρτήσεων. Γενικά για να αποδειχθεί ότι ένα σύνολο λογικών συναρτήσεων είναι πλήρες αρκεί να αποδειχθεί ότι οι πράξεις του κατάλληλα συνδυασμένες μπορούν να παράγουν όλους τους λογικούς τύπους. Εύκολα αποδεικνύεται ότι η συνάρτηση NAND είναι πλήρης, ξεκινώντας με τις πιο βασικούς τύπους:

- 1: $a \mid (a \mid a) = a \mid a' = 1$.
- 0: $\{a \mid (a \mid a)\} \mid \{a \mid (a \mid a)\} = 1 \mid 1 = 0$.
- *a*’: $a \mid a = a'$.
- *ab*: $(a \mid b) \mid (a \mid b) = ab$.
- *a+b:* $(a \mid a) \mid (b \mid b) = a' \mid b' = a + b$.

Από τους παραπάνω τύπους μπορούμε να δημιουργήσουμε όλους τους υπόλοιπους τύπους. Άρα η συνάρτηση $\{|\}$ μπορεί να χρησιμοποιηθεί για τη δημιουργία όλων των λογικών συναρτήσεων. Παρόμοια, οποιοσδήποτε τύπος μπορεί να δημιουργηθεί μόνο με συναρτήσεις NOR.

Ωστόσο, ο συνδυασμός AND και OR συναρτήσεων δεν είναι πλήρης και μπορεί να αποδειχθεί εύκολα: Δεν υπάρχει τρόπος να δημιουργηθεί είτε το λογικό 1 είτε το λογικό 0 απευθείας με χρήση οποιουδήποτε συνδυασμού AND και OR. Εάν προστεθεί μια NOT στο σύνολο των δύο παραπάνω συναρτήσεων, τότε μπορούν να δημιουργηθούν όλοι οι προηγούμενοι τύποι: $a + a' = 1$ κ.λ.π. Στη πραγματικότητα, τα σύνολα $\{',\cdot\}$ και $\{',+\}$ είναι πλήρης.

Οποιαδήποτε τεχνολογία επιλεχθεί για την υλοποίηση λογικών συναρτήσεων πρέπει να έχει τη δυνατότητα υλοποιήσεων πλήρων συνόλων συναρτήσεων. Στατικά, συμπληρωματικά κυκλώματα έχουν τη δυνατότητα υλοποιήσεων των NAND και NOR συναρτήσεων, αλλά υπάρχουν άλλες οικογένειες κυκλωμάτων που δεν έχουν δυνατότητα υλοποιήσεων πλήρων συνόλων συναρτήσεων. Λογικές οικογένειες που δεν είναι πλήρης απαιτούν επιπλέον επιβάρυνση στη λογική σχεδιασμού τους έτσι ώστε να διασφαλιστεί ότι η λογική συνάρτηση ορίζεται στη σωστή μορφή της.

Μια λογική έκφραση είναι **μη-πλεονάζουσα**, εάν δεν μπορεί να απλοποιηθεί χωρίς να αλλάξει η τιμή της. Για παράδειγμα, η έκφραση $ab + ab'$ είναι πλεονάζουσα, γιατί μπορεί να απλοποιηθεί στην έκφραση a. Ο μη-πλεονάζων τύπος και το αντίστοιχο λογικό δίκτυο έχουν κάποιες σημαντικές ιδιότητες: ο τύπος του είναι μικρότερος από τον αντίστοιχο λογικό πλεονάζοντα τύπο, και το λογικό δίκτυο του εγγυάται τη δυνατότητα ελέγχου για κατασκευαστικά σφάλματα. Ωστόσο, ο μη-πλεονάζων τύπος δεν είναι πανάκεια και δεν σημαίνει απαραίτητα **ελάσσων τύπος** (**minimality**) – υπάρχουν πολλοί μη-πλεονάζοντες τύποι σε μια έκφραση, κάποιος από τους οποίους είναι πιο μικρός από κάποιον άλλο, οπότε η εύρεση μιας μη-πλεονάζουσας έκφρασης δεν εγγυάται ότι παράγει το μικρότερο σχέδιο. Ο μη-πλεονασμός συνήθως σημαίνει πρόσθετη καθυστέρηση, η οποία είναι δύσκολο να μειωθεί, αν δε γίνει το λογικό δίκτυο πλεονάζον. Ωστόσο, η απλοποίηση της λογικής έκφρασης πριν το σχεδιασμό του δικτύου πυλών είναι σημαντική τόσο για τη μείωση της επικαλυπτόμενης επιφάνειας όσο και για τη μείωση της καθυστέρησης. Μερικές προφανείς απλοποιήσεις μπορούν να γίνουν με το χέρι ενώ τα CAD εργαλεία μπορούν να κάνουν δύσκολες απλοποιήσεις σε μεγαλύτερες εκφράσεις.

Επίσης ορίζονται οι καταστάσεις **on-set** και **off-set** μιας συνάρτησης. Η on-set κατάσταση είναι το σύνολο των τιμών στις εισόδους της συνάρτησης, κατά την οποία είναι αληθής ενώ η off-set κατάσταση είναι το σύνολο των τιμών στις εισόδους της συνάρτηση κατά την οποία είναι ψευδής. Συνήθως μια συνάρτηση καθορίζεται με την on-set κατάσταση, αλλά και η χρήση της off-set κατάστασης είναι ισοδύναμη (εφόσον είναι σαφές ποια από τις δύο καταστάσεις χρησιμοποιείται).

Χρήσιμη είναι η έννοια του αδιάφορου όρου (don't care) σε ένα συνδυαστικό κύκλωμα. Υπάρχουν δύο τύποι αδιάφορων όρων: Ο **αδιάφορος όρος εισόδου** (**input don't care**) και ο **αδιάφορος όρος εξόδου** (**output don't care**). Παρόλο που οι δύο τύποι έχουν παρόμοια ονόματα παράγουν πολύ διαφορετικές συναρτήσεις.

Ο αδιάφορος όρος εισόδου είναι ένας συμβολισμός διευκόλυνσης. Έστω η συνάρτηση που φαίνεται στην Εικόνα 1-2 . Σε αυτή τη συνάρτηση οι είσοδοι $a = 0$, $b = 0$ και $a = 0$, $b = 1$ δίνουν το ίδιο αποτέλεσμα. Όταν συμβολίζεται ο πίνακας αλήθειας αυτής της συνάρτησης είναι δυνατόν να συντμηθούν οι αντίστοιχες δύο γραμμές του πίνακα σε μια γραμμή με χρήση του συμβόλου του αδιάφορου όρου -. Σε άλλο παράδειγμα, είναι δυνατόν να απλοποιηθεί η παράσταση $ab + ab'$ σε a' με τη μεταβλητή b να έχει τη μορφή αδιάφορου όρου εισόδου (Η αρχική λογική

συνάρτηση είναι η $f = a'+ab'$. Αυτό μπορεί να επαληθευτεί από το χάρτη Karnaugh της συνάρτησης.)

Ο αδιάφορος όρος εξόδου έχει την έννοια του ελλιπούς προσδιορισμού μιας λογικής συνάρτησης. Ο αδιάφορος όρος εξόδου είναι ένας συνδυασμός εισόδων, οι οποίες δεν χρησιμοποιούνται.

a	b	f
0	0	1
0	1	1
1	0	0
1	1	1

Πλήρως καθορισμένη

a	b	f
0	-	1
1	0	0
1	1	1

Με αδιάφορο όρο εισόδου

Εικόνα 1-2 *Ένα παράδειγμα αδιάφορου όρου εισόδου*

Έστω η συνάρτηση που φαίνεται στην Εικόνα 1-3. Για το συνδυασμό εισόδων $a = 0$, $b = 1$ η τιμή της συνάρτησης είναι αδιάφορος όρος -. (Ο ίδιος συμβολισμός χρησιμοποιείται και για τη περίπτωση αδιάφορου όρου εισόδου.)

a	b	f
0	0	1
0	1	-
1	0	0
1	1	1

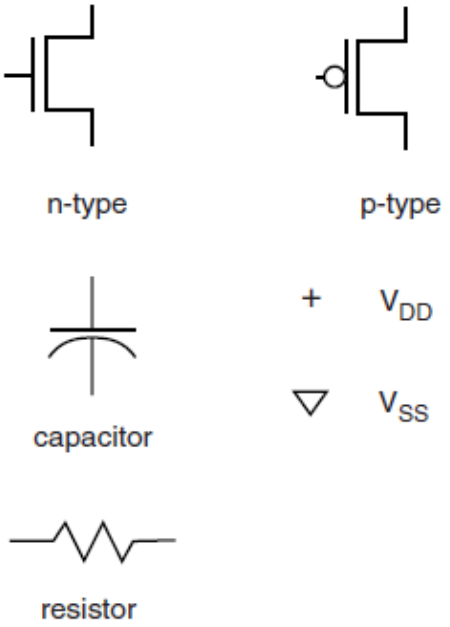
Εικόνα 1-3 *Ένα παράδειγμα αδιάφορου όρου εξόδου*

Αυτό σημαίνει ότι δεν έχει σημασία, αν η έξοδος έχει τιμή 0 ή 1 για αυτό το συνδυασμό εισόδων. Κατά την υλοποίηση αυτής της συνάρτησης σε επίπεδο πυλών, επιλέγεται η τιμή της εξόδου για αυτή τη περίπτωση που δίνει τη βέλτιστη υλοποίηση. Στη προηγούμενη παράγραφο φάνηκε ότι επιλέγοντας τιμή εξόδου της συνάρτησης ίση με 1 απλοποιείται σημαντικά η έκφραση της σε σχέση με την αντίθετη επιλογή.

Συνήθως οι Boolean εκφράσεις γράφονται σε μορφή **παραγόντων**. Για παράδειγμα η συνάρτηση $f = a'+ab'$ είτε μπορεί να χρησιμοποιηθεί στην αυτή μορφή είτε μπορεί να γραφεί με τη μορφή παραγόντων ως $f = a'+g$ όπου $g = ab'$. Αυτός ο μετασχηματισμός μπορεί σε μερικές περιπτώσεις να μην έχει καμιά σημασία, αλλά αν ο παράγοντας ab' χρησιμοποιείται σε κάποιο άλλο σημείο η συνάρτηση g

αποτελεί έναν κοινό παράγοντα. Η αντίθετη διαδικασία της παραγοντοποίησης είναι η διάσπαση Οπότε η συνάρτηση g διασπάται στη f εάν γραφτεί ως $f = a'+ab'$.

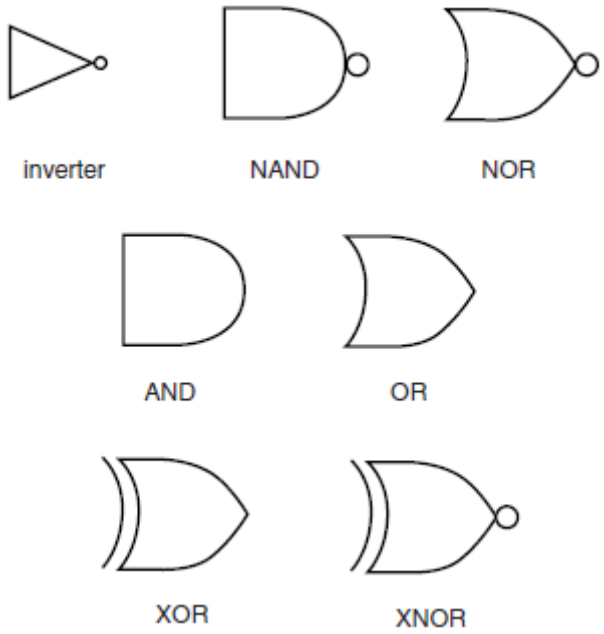
1.2.2 Σχηματικά και Λογικά Σύμβολα



Εικόνα 1-4 *Σχηματικά σύμβολα ηλεκτρικών στοιχείων*

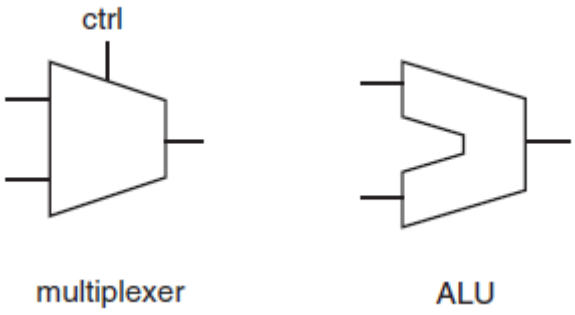
Παρακάτω θα γίνει μια σύντομη επανάληψη των σχηματικών συμβόλων που χρησιμοποιούνται. Στην Εικόνα 1-4 φαίνονται τα σύμβολα μερικών ηλεκτρικών στοιχείων, όπως: n-type και p-type τρανζίστορ, πυκνωτής (capacitor), αντίσταση (resistor), και ακροδέκτες τάσης (V_{DD} για τη θετική τάση και V_{SS} για την αρνητική τάση).

Η Εικόνα 1-5 απεικονίζει τα σύμβολα κάποιων βασικών λογικών πυλών όπως της NAND, NOR κ.λ.π.



Εικόνα 1-5 *Σχηματικά σύμβολα λογικών πυλών*

Τέλος η Εικόνα 1-6 δείχνει τα σύμβολα σε επίπεδο καταχωρητή του πολυπλέκτη (multiplexer - mux) και της αριθμητικής λογικής μονάδας (arithmetic logic unit – ALU).



Εικόνα 1-6 *Σχηματικά σύμβολα σε επίπεδο καταχωρητή*

1.3 Ψηφιακή Σχεδίαση και FPGAs

1.3.1 Ο Ρόλος των FPGAs

Οι **Συστοιχίες Πυλών Προγραμματιζόμενου Πεδίου** (**Field Programmable Gate Arrays- FPGAs**) καλύπτουν τις ανάγκες στο χώρο του σχεδιασμού ψηφιακών συστημάτων, συμπληρώνοντας το ρόλο που διαδραματίζουν οι μικροεπεξεργαστές. Οι μικροεπεξεργαστές μπορούν να χρησιμοποιηθούν σε μεγάλη ποικιλία από σχεδιαστικά περιβάλλοντα, αλλά επειδή βασίζονται σε λογισμικό, για να υλοποιήσουν συναρτήσεις, συνήθως είναι πιο αργοί και καταναλώνουν μεγαλύτερη ενέργεια σε σχέση με τα εξατομικευμένα ολοκληρωμένα. Παρόμοια, τα FPGAs δεν είναι πλήρως εξατομικευμένα στοιχεία σχεδιασμού και γι 'υτό δεν είναι τόσο βέλτιστα όσο θα ήταν ένα πλήρως εξατομικευμένο ολοκληρωμένο μιας συγκεκριμένης εφαρμογής. Τα FPGAs επιτυγχάνουν συνήθως μικρότερες ταχύτητες υπολογισμού και καταναλώνουν μεγαλύτερη ενέργεια σε σχέση με την εξατομικευμένη λογική. Επίσης είναι σχετικά πιο ακριβά σε σχέση με τα ολοκληρωμένα κυκλώματα εξατομικευμένου σχεδιασμού.

Εν τούτοις, παρουσιάζουν σημαντικά πλεονεκτήματα κυρίως εξαιτίας του γεγονότος ότι αποτελούν τυποποιημένα κυκλώματα.

- Δεν υπάρχει ανάγκη για αναμονή από τη στιγμή του σχεδιασμού του κυκλώματος μέχρι τη στιγμή του ελέγχου του ολοκληρωμένου. Το κύκλωμα μπορεί να προγραμματιστεί στο FPGA και να ελεγχθεί άμεσα.
- Τα FPGAs είναι τέλειο όχημα για τη πρωτοτυποποίηση. Εάν χρησιμοποιηθεί στο τελικό σχεδιασμό η μετάβαση από το πρωτότυπο σχέδιο στο τελικό προϊόν είναι βραχύχρονη και εύκολη διαδικασία.
- Το ίδιο FPGA μπορεί να χρησιμοποιηθεί στο σχεδιασμό πολλών κυκλωμάτων μειώνοντας αρκετά το κόστος.

Το πεδίο εφαρμογών των FPGAs αυξάνεται αλματωδώς τα τελευταία είκοσι χρόνια από τη στιγμή της εμφάνισής τους. Η **Προγραμματιζόμενη Λογική Συσκευών** (**Programmable Logic Devices – PLDs**) εμφανίστηκε περίπου του 1970. Αυτές οι συσκευές χρησιμοποιούσαν δύο επίπεδα λογικής για την υλοποίηση προγραμματιζόμενης λογικής. Το πρώτο επίπεδο λογικής αποτελούνταν από πύλες AND και ήταν συνήθως αμετάβλητο ενώ το δεύτερο επίπεδο λογικής αποτελούνταν από πύλες OR και ήταν προγραμματιζόμενο. Οι PLDs συνήθως προγραμματίζονταν με αντιασφάλειες (antifuses), στις οποίες εφαρμόζονταν μεγάλα επίπεδα τάσης με σκοπό να γίνουν οι συνδέσεις.

Κυρίως χρησιμοποιούνταν σαν **κύκλωμα συνδετικής λογικής (glue-logic)** – λογική η οποία απαιτούνταν για την διασύνδεση των κυρίως τμημάτων ενός συστήματος. Συνήθως χρησιμοποιούνταν στα πρωτότυπα σχέδια, επειδή μπορούσαν να προγραμματιστούν και να τοποθετηθούν στη πλακέτα σε σύντομο διάστημα, αλλά

δεν χρησιμοποιούνταν στα τελικά προϊόντα. Η προγραμματιζόμενη λογική συσκευών συνήθως δεν συναντάται στα κύρια τμήματα των συστημάτων στα οποία χρησιμοποιούνται. Καθώς τα ψηφιακά κυκλώματα γίνονται πιο περίπλοκα, απαιτούνται πιο πολυπληθείς προγραμματιζόμενες λογικές και ο περιορισμός των PLDs με τα δύο επίπεδα λογικής είναι αναπόφεκτος.

Τα δύο επίπεδα λογικής είναι χρήσιμα για απλές λογικές συναρτήσεις, αλλά καθώς τα επίπεδα ολοκλήρωσης αυξάνονται, κρίνονται αναποτελεσματικά. Τα FPGAs εξασφαλίζουν προγραμματιζόμενη λογική κάνοντας χρήση πολύ-επίπεδης λογικής μεταβλητού βάθους. Χρησιμοποιούν τόσο προγραμματιζόμενα λογικά στοιχεία όσο προγραμματιζόμενες διασυνδέσεις με σκοπό τη δημιουργία πολύ-επίπεδων λογικών συναρτήσεων.

Την εφεύρεση του FPGA τη πιστώνεται ο Ross Freeman. Το FPGA [Fre89] περιείχε προγραμματιζόμενα λογικά στοιχεία και λογική προγραμματιζόμενων διασυνδέσεων. Το FPGA του Freeman προγραμματίζονταν με χρήση στατικών μνημών SRAM και όχι αντιασφαλειών. Αυτό επέτρεπε τη κατασκευή του FPGA με τη συνήθη VLSI διεργασία, εξοικονομώντας χρήματα και δίνοντας τη δυνατότητα πολλών κατασκευαστικών επιλογών. Επιτρέπονταν επίσης να προγραμματιστεί εφόσον βρίσκονταν επί του κυκλώματος και αυτό ήταν πρωτοποριακό χαρακτηριστικό μια και η flash memory μέχρι τότε δεν χρησιμοποιούταν ευρέως.

Η Xilinx και η Altera είναι δύο εταιρίες που κατασκευάζουν και διανέμουν FPGAs βασισμένα σε μνήμες SRAMs. Μια εναλλακτική αρχιτεκτονική είχε προταθεί από την Actel με χρήση αρχιτεκτονικής αντιασφαλειών. Αυτή η αρχιτεκτονική δεν ήταν επαναπρογραμματιζόμενη στο πεδίο, στοιχείο με πλεονέκτημα σε περιπτώσεις που δεν απαιτούνταν επαναπρογραμματισμό. Τα FPGAs της Actel χρησιμοποιούσαν λογική βασισμένη σε πολυπλέκτες τοποθετημένοι πέριξ των καναλιών της καλωδίωσης.

Για πάρα πολλά χρόνια τα FPGAs χρησιμοποιούνταν κυρίως ως *συνδετική λογική* (*glue-logic)* και για τη προτυποποίηση τελικών κυκλωμάτων. Στη σύγχρονη εποχή χρησιμοποιούνται σε όλα τα είδη ψηφιακών συστημάτων:

- σαν τμήματα τηλεπικοινωνιακών συστημάτων υψηλών ταχυτήτων,
- σαν βίντεο επιταχυντές σε συστήματα εγγραφής βίντεο.

Τα FPGAs πλέον έχουν δεσπόζουσα θέση για υλοποιήσεις ψηφιακών συστημάτων.

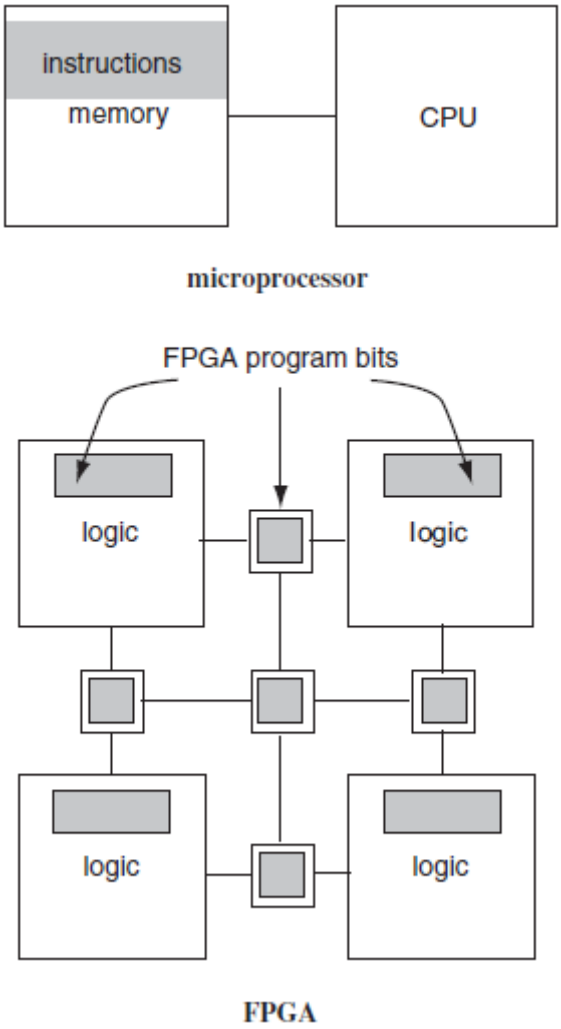
1.3.2 Τύποι των FPGAs

Μέχρι στιγμής δεν έχει καθοριστεί πλήρως ο ορισμός του FPGA. Ένας καλός ορισμός θα πρέπει να ξεχωρίζει το FPGA αφ' ενός από τις συσκευές μικρότερης προγραμματιζόμενης λογικής όπως τα PLDs και αφ' ετέρου από το εξατομικευμένο ολοκληρωμένο. Παρακάτω δίνονται κάποια ιδιαίτερα χαρακτηριστικά των FPGAs:

- **Είναι τυποποιημένα τμήματα λογικής**. Δεν έχουν σχεδιαστεί για οποιαδήποτε ιδιαίτερη συνάρτηση, αλλά μπορούν να προγραμματιστούν για ιδιαίτερο σκοπό.
- **Υλοποιούν πολλά επίπεδα λογικής.** Τα λογικά μπλοκς στο εσωτερικό των FPGAs μπορούν να συνδεθούν σα δίκτυα μεταβλητού βάθους. Αντίθετα τα PLDs χρησιμοποιούν δύο επίπεδα AND/OR πυλών για την υλοποίηση της λογικής τους.

Επειδή τα FPGAs υλοποιούν πολύ-επίπεδη λογική, σε γενικές γραμμές απαιτούν τόσο προγραμματιζόμενη λογική μπλοκ όσο και λογική προγραμματιζόμενων διασυνδέσεων. Οι PLDs συσκευές χρησιμοποιούν πάγιες διασυνδέσεις και απλά μετατρέπουν τις λογικές συναρτήσεις που επισυνάπτονται στα καλώδια. Αντίθετα, τα FPGAs απαιτούν προγραμματισμό των λογικών μπλοκς και διασύνδεση μεταξύ τους για την υλοποίηση συναρτήσεων. Ο συνδυασμός της λογικής και των διασυνδέσεων είναι γνωστός σα **δομή** (**fabric**) επειδή λαμβάνει χώρα μια διαδικασία κατά την οποία τα εργαλεία σχεδιασμού χαρτογραφούν την επιθυμητή λογική στο FPGA.

Ένα από τα σημαντικότερα χαρακτηριστικά των FPGAs είναι ο προγραμματισμός τους. Όπως φαίνεται στην Εικόνα 1-7, ο προγραμματισμός του FPGA είναι πολύ διαφορετικός από το προγραμματισμό ενός μικροεπεξεργαστή.



Εικόνα 1-7 *Microprocessor έναντι FPGA*

Ο μικροεπεξεργαστής είναι ένα υπολογιστικό σύστημα που έχει δυνατότητα αποθήκευσης του προγράμματος που εκτελεί. Το υπολογιστικό σύστημα περιέχει μια CPU και μια μνήμη στην οποία αποθηκεύονται εντολές και δεδομένα. Τα FPGA δεδομένα (που επίσης καλούνται **προσωπικότητα**) είναι συνυφασμένα τη λογική του FPGA. Ένα FPGA δεν αποθηκεύει εντολές – τα FPGA προγραμματίζουν απευθείας τη λογική συνάρτηση και τις διασυνδέσεις.

Για τον προγραμματισμό των FPGAs χρησιμοποιούνται διάφορες τεχνολογίες. Κάποια FPGAs προγραμματίζονται μόνιμα ενώ άλλα μπορούν να επαναπρογραμματιστούν. Τα επαναπρογραμματιζόμενα FPGAs είναι επίσης γνωστά σαν **επανα-**

διατάξιμες συσκευές. Τα επαναδιατάξιμα FPGAs προτιμούνται κατά τη διαδικασία της προτυποποίησης κυκλωμάτων γιατί δεν χρειάζεται να αντικατασταθούν κάθε φορά που γίνεται μια αλλαγή. Οι επαναδιατάξιμες συσκευές μπορούν επίσης να προγραμματιστούν on-the-fly κατά τη λειτουργία των συσκευών. Αυτό επιτρέπει σε ένα τμήμα υλικού να εκτελεί πολλές διαφορετικές συναρτήσεις. Βεβαίως αυτές οι συναρτήσεις δεν μπορούν να εκτελεστούν ταυτόχρονα, αλλά η δυνατότητα επαναδιάταξης είναι πολύ χρήσιμη, όταν η συσκευή λειτουργεί με διαφορετικούς τρόπους. Για παράδειγμα, η οθόνη του Radius υπολογιστή λειτουργεί τόσο κατά την οριζόντια όσο κατά την κάθετη τρόπο. Όταν ο χρήστης περιστρέφει την οθόνη, ένας διακόπτης υδραργύρου αναγκάζει το FPGA που υποστηρίζει την οθόνη να επαναπρογραμματιστεί στο νέο τρόπο λειτουργίας.

Τα FPGAs παραδοσιακά χρησιμοποιούν λογική λεπτής-υφής (fine-grained). Ένα συνδυαστικό λογικό στοιχείο σε ένα FPGA υλοποιεί τη συνάρτηση με ένα πλήθος λογικών πυλών μαζί με έναν καταχωρητή. Καθώς τα ολοκληρωμένα γίνονται μεγαλύτερα, χρησιμοποιούνται FPGAs με λογική χονδροειδούς-υφής (coarser-grained). Ένα λογικό στοιχείο σε αυτά τα ολοκληρωμένα μπορεί να υλοποιήσει μια ALU αρκετών ψηφίων (bit) και ένα καταχωρητή. Τα FPGAs με λογική χονδροειδούς-υφής μπορεί να κάνουν πιο αποδοτική χρήση στους πόρους του υλικού του ολοκληρωμένου για κάποιους τύπους συναρτήσεων.

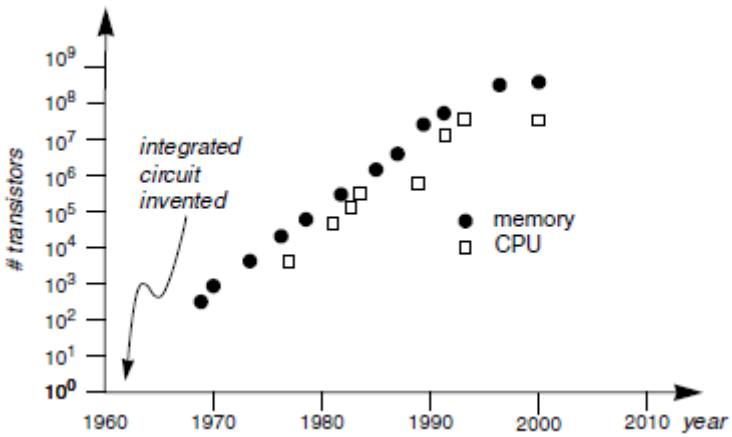
Οι νέες συσκευές των FPGAs περιλαμβάνουν κάτι περισσότερο από ένα FPGA αυτό καθεαυτό. Είναι ουσιαστικά ολοκληρωμένες **πλατφόρμες με FPGAs** που περιλαμβάνουν αρκετά περιφερειακά έτσι ώστε οποιοδήποτε μεγάλο σύστημα να μπορεί να υλοποιηθεί αποδοτικά. Τυπικά χαρακτηριστικά μιας πλατφόρμας FPGA είναι μια CPU έτσι ώστε να είναι δυνατός ο συν-σχεδιασμός υλικού-λογισμικού. Επίσης μπορεί να περιλαμβάνει λογικές διαύλου έτσι, ώστε για παράδειγμα ένας PCI δίαυλος να μπορεί να συμπεριληφθεί στο σχεδιασμό του υλοποιούμενου συστήματος.

1.3.3 FPGAs έναντι Εξατομικευμένης VLSI Λογικής

Η κυριότερη εναλλακτική λύση ενός FPGA είναι το **ολοκληρωμένο κύκλωμα ειδικού σκοπού** (**application specific IC-ASIC**). Σε αντίθεση με το FPGA ένα ASIC ολοκληρωμένο σχεδιάζεται έτσι ώστε να υλοποιεί μια συγκεκριμένη λογική συνάρτηση. Ο σχεδιασμός ενός ASIC κάνει χρήση των μασκών που χρησιμοποιούνται για τη κατασκευή των ολοκληρωμένων κυκλωμάτων. Το ASIC πρέπει να κατασκευαστεί σε μια γραμμή παραγωγής, διαδικασία που διαρκεί αρκετούς μήνες, έως ότου χρησιμοποιηθεί ή δοκιμαστεί. Πρέπει να γίνει διάκριση μεταξύ των ASICs ολοκληρωμένων και των πλήρως εξατομικευμένων σχεδιασμών: ένα πλήρως εξατομικευμένο σχέδιο έχει συγκεκριμένη διάταξη (layout), ενώ τα ASIC χρησιμοποιούν προσχεδιασμένες διατάξεις λογικών πυλών. Σήμερα, ένας μικρός αριθμός ψηφιακών ολοκληρωμένων εκτός από τους μικροεπεξεργαστές περιλαμβάνουν ένα μεγάλο αριθμό από εξατομικευμένα layout.

Όπως αναφέρθηκε πιο πριν, τα ASICs παρουσιάζουν κάποια σημαντικά πλεονεκτήματα, επειδή έχουν σχεδιαστεί για συγκεκριμένο σκοπό: είναι πιο γρήγορα στους υπολογισμούς τους και καταναλώνουν μικρότερη ενέργεια από τα ισοδύναμα FPGAs, και όταν κατασκευάζονται σε μεγάλες ποσότητες είναι φθηνότερα. Εν τούτοις δύο σημαντικοί λόγοι οδηγούν τους σχεδιαστές να χρησιμοποιούν FPGAs αντί ASICs για εξατομικευμένη λογική.

Αφενός ο **νόμος του Moore** παρέχει μια ευδιάκριτη εικόνα της αύξησης της επιφάνειας των ολοκληρωμένων κυκλωμάτων. Πιο συγκεκριμένα ο νόμος του Moore περιγράφει ότι ο αριθμός των τρανζίστορ που μπορούν να χρησιμοποιηθούν σε ένα ολοκληρωμένο θα διπλασιάζεται κάθε 18 μήνες. Όπως φαίνεται στην Εικόνα 1-8, η βιομηχανία των ημιαγωγών έχει κατορθώσει να κρατήσει αυτό το ρυθμό για αρκετές δεκαετίες.



Εικόνα 1-8 *Ο Νόμος του Moore*

Παρόλο, που ο νόμος του Moore δεν θα ισχύει για πάντα, ήδη υπάρχουν ολοκληρωμένα κυκλώματα με εκατομμύρια τρανζίστορ στο εσωτερικό τους. Με τόσο μεγάλο αριθμό τρανζίστορ σε ένα ολοκληρωμένο, είναι επιθυμητό και αναγκαίο η αποβολή και η απόρριψη αριθμού τρανζίστορ με σκοπό την απλούστευση του σχεδιασμού του ολοκληρωμένου. Τα FPGAs χρησιμοποιούν περισσότερα τρανζίστορ για μια δεδομένη συνάρτηση απ' ότι τα ASICs, αλλά τα FPGAs μπορούν να σχεδιαστούν σε μερικές μέρες σε σχέση με τον κύκλο εργασιών που απαιτείται για τα ASICs.

Αφετέρου, το κόστος για τη κατασκευή ενός ASIC είναι τεράστιο. Η βασική εικόνα της αξίας μιας γραμμής παραγωγής ολοκληρωμένων κυκλωμάτων είναι το πλάτος καναλιού του μικρότερου τρανζίστορ που μπορεί να κατασκευαστεί. Καθώς το πλάτος καναλιού μικραίνει το κόστος κατασκευής αυξάνεται και ένα σύγχρονο εργοστάσιο ημιαγωγών κοστίζει μερικά δισεκατομμύρια δολάρια. Εντούτοις, αυτό

το κόστος μπορεί να επεκτείνεται σε όλα τα τμήματα που απαρτίζουν τη γραμμή παραγωγής. Αντίθετα, οι μάσκες που ορίζουν τα τρανζίστορ και τις καλωδιώσεις ενός ολοκληρωμένου είναι χρήσιμα μόνο για αυτό το συγκεκριμένο ολοκληρωμένο. Το κόστος της μάσκας θα απορροφηθεί εξολοκλήρου από το κόστος του συγκεκριμένου ολοκληρωμένου. Στο Πίνακα 1-1 φαίνεται ότι το κόστος για το κάθε μέγεθος της μάσκας αυξάνεται εκθετικά.

τεχνολογία	**κόστος μάσκας**
0,09 μm	**1000000 $**
0,18 μm	**250000 $**
0,25 μm	**120000 $**
0,35 μm	**60000 $**

Πίνακας 1-1 *Κόστος μάσκας σα συνάρτηση του πλάτους γραμμής*

Καθώς το πλάτος του καναλιού ελαττώνεται, το κόστος της μάσκας επικαλύπτει το κόστος που απαιτείται από τους σχεδιαστές του ολοκληρωμένου. Καθώς το κόστος της μάσκας αυξάνεται, τα τυποποιημένα τμήματα υλικού από τα ολοκληρωμένα γίνονται ιδιαίτερα ελκυστικά. Επειδή τα FPGAs είναι τυποποιημένα τμήματα υλικού που μπορούν να προγραμματιστούν για πολλές διαφορετικές συναρτήσεις, είναι αναμενόμενο να πάρουν ένα μεγάλο μέρος της αγοράς ολοκληρωμένων σαν ολοκληρωμένα υψηλής ολοκλήρωσης.

1.4 Σχεδιασμός σε Συστήματα Βασισμένα σε FPGA

1.4.1 Σκοπός και Τεχνικές

Η υλοποίηση μιας λογικής συνάρτησης σε ένα FPGA ή ένα ψηφιακό σύστημα είναι μόνο ο αρχικός στόχος. Υπάρχουν συγκεκριμένες ιδιότητες που πρέπει να ικανοποιούνται για να θεωρείται επιτυχημένο ένα έργο:

- **Απόδοση (Performance)**. Η λογική του κυκλώματος πρέπει να εκτελείται σε συγκεκριμένο χρόνο. Η απόδοση μετριέται με διαφορετικούς τρόπους, ή με το ρυθμό μετάδοσης (throughput) και με τον λανθάνοντα χρόνο (latency). Επίσης η περίοδος ρολογιού συχνά χρησιμοποιείται σα μέτρο της απόδοσης.
- **Ισχύς / ενέργεια (Power / energy)**. Το ολοκληρωμένο κύκλωμα πρέπει να λειτουργεί σε συγκεκριμένα επίπεδα ενέργειας ή ισχύος. Η κατανάλωση ενέργειας είναι κρίσιμος παράγοντας σε συστήματα με μπαταρία. Ακόμα και εάν στο σύστημα έχει προβλεφθεί η απαγωγή ενέργειας, οι απώλειες λόγω θερμότητας κοστίζει σε χρήμα και πρέπει να ελεγχθεί.
- **Χρόνος σχεδιασμού (Design time)**. Είναι αδύνατος ο συνεχόμενος σχεδιασμός ενός συστήματος. Επειδή τα FPGA είναι τυποποιημένα τμήματα υλικού, παρουσιάζουν μερικά πλεονεκτήματα όσον αφορά το χρόνο σχεδιασμού. Μπορούν να χρησιμοποιηθούν σα πρότυπα συστήματα, μπορούν να

προγραμματιστούν γρήγορα και μπορούν να χρησιμοποιηθούν ως υποσύστημα στο τελικό σχέδιο.

- **Κόστος σχεδιασμού (Design cost)**. Ο χρόνος σχεδιασμού είναι σημαντική συνιστώσα του κόστους σχεδιασμού, μα άλλοι παράγοντες, όπως τα απαιτούμενα εργαλεία σχεδιασμού είναι επίσης σημαντικοί. Τα εργαλεία σχεδίασης των FPGA είναι συνήθως πιο φθηνά απ' ότι τα αντίστοιχα που χρησιμοποιούνται για το σχεδιασμό εξατομικευμένων VLSI συστημάτων.
- **Κόστος κατασκευής (Manufacturing cost)**. Το κόστος κατασκευής είναι το κόστος για τη κατασκευή πολλών αντιγράφων του συστήματος. Συνήθως τα FPGA είναι ακριβότερα από τα ASIC εξαιτίας της δυνατότητας προγραμματισμού. Εντούτοις, το γεγονός ότι είναι τυποποιημένα τμήματα υλικού βοηθάει στη μείωση του κόστους.

Ο σχεδιασμός συστημάτων είναι ιδιαίτερα δύσκολος, γιατί πρέπει να επιλυθούν μερικά προβλήματα, όπως:

- **Πολλαπλά σχεδιαστικά επίπεδα αφαίρεσης (Multiple levels of abstraction).** Ο σχεδιασμός ενός FPGA απαιτεί την επεξεργασία μιας ιδέας μέσω πολλών επιπέδων. Ξεκινώντας από τις προδιαγραφές λειτουργίας του ολοκληρωμένου, ο σχεδιαστής πρέπει να δημιουργήσει μια αρχιτεκτονική που να υλοποιεί την απαιτούμενη συνάρτηση και έπειτα να επεκτείνει την αρχιτεκτονική αυτή σε λογικό σχέδιο.
- **Πολλαπλά και αντικρουόμενα κόστη (Multiple and conflicting costs).** Το κόστος μπορεί να μετριέται σε χρήμα ως η δαπάνη ενός εργαλείου λογισμικού που απαιτείται για το σχεδιασμό του συνολικού σχεδίου. Το κόστος επίσης μπορεί να μετριέται ως προς την απόδοση ή κατανάλωση ισχύος του τελικού FPGA.
- **Βραχύς χρόνος σχεδιασμού (Short design time).** Η αγορά των ηλεκτρονικών εξελίσσεται και μεταβάλλεται ραγδαία. Η ταχύτερη κατασκευή ενός ολοκληρωμένου σημαίνει μείωση του κόστους και αύξηση των κερδών. Αντίθετα η βραδύτερη κατασκευή ενός ολοκληρωμένου σημαίνει πολύ μικρότερα κέρδη.

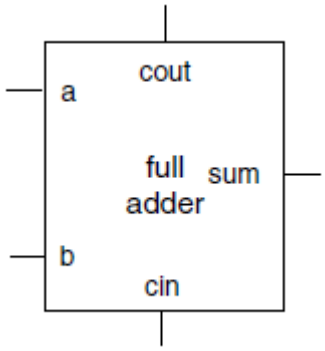
Ένα έργο σχεδιασμού ενός κυκλώματος μπορεί να ξεκινήσει με ένα εύρος πληροφοριών. Μερικά έργα αποτελούν αναθεώρηση κάποιων προηγούμενων σχεδίων, ενώ κάποια άλλα είναι υλοποιήσεις προτύπων. Σε αυτές τις περιπτώσεις η προς υλοποίηση συνάρτηση είναι πλήρως καθορισμένη και είναι ξεκάθαρες πολλές πτυχές της υλοποίησης. Άλλα έργα είναι λιγότερο καθορισμένα και είναι γνωστές μόνο βασικές πληροφορίες για το τι πρέπει να εκτελεί το ολοκληρωμένο. Χρησιμοποιείται ο όρος **απαιτήσεις (requirement)** για τη απλή περιγραφή του συστήματος, για παράδειγμα τη βασική συνάρτηση ενός ADSL modem. Χρησιμοποιείται ο όρος **προδιαγραφή (specification)** για την αναλυτική περιγραφή της συνάρτησης

όπως για παράδειγμα, το πρόγραμμα προσομοίωσης της βασικής συνάρτησης ενός ADSL modem. Χαρακτηριστικά όπως απόδοση, κατανάλωση ισχύος και κόστος ονομάζονται **μη-λειτουργικές απαιτήσεις (non-functional requirements)**.

1.4.2 Ιεραρχικός Σχεδιασμός

Ιεραρχικός σχεδιασμός (Hierarchical design) είναι η συνηθισμένη διαδικασία για την ανάλυση και το σχεδιασμό περίπλοκων σχεδίων. Χρησιμοποιείται ευρέως στο προγραμματισμό, καθώς μια διαδικασία δε γράφεται σα μια τεράστια λίστα από δηλώσεις αλλά τελεί απλούστερες διαδικασίες. Κάθε διαδικασία αποσυνθέτει τη λειτουργία της σε μικρότερες έως ότου κάθε λογικό βήμα να μπορεί να περιγραφεί από μια απλούστερη διαδικασία που μπορεί να περιγραφεί απευθείας. Αυτή η τεχνική είναι επίσης γνωστή σαν τεχνική του **διαίρει και βασίλευε (divide-and-conquer)** – η πολυπλοκότητα της διαδικασίας καταστέλλεται με επαναλαμβανόμενες αποσυνθέσεις, έως ότου επιτευχθούν απλούστερες διαδικασίες οι οποίες να μπορούν να περιγραφούν με εύκολο τρόπο.

Οι σχεδιαστές των ολοκληρωμένων διαιρούν το ολοκληρωμένο σε επίπεδα (ιεραρχίες) που αποτελούνται από μικρότερες λογικές μονάδες. Στην Εικόνα 1-9, φαίνεται μια λογική μονάδα (ένας πλήρης αθροιστής του $1-bit$) που αποτελείται από το **σώμα (body)** και ένα αριθμό από **ακροδέκτες (pins).** Οι ακροδέκτες του είναι οι a, b, cin, $cout$ και sum.

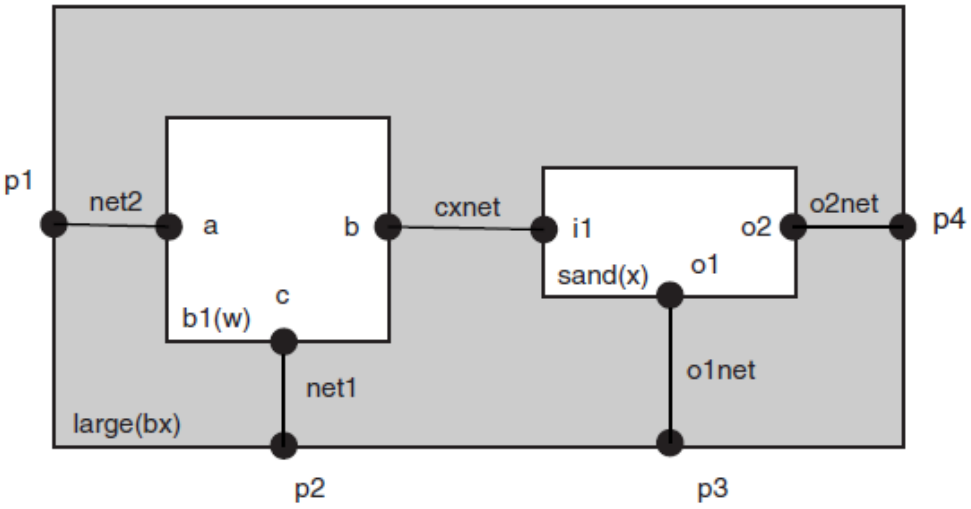


Εικόνα 1-9 *Η μονάδα του πλήρους αθροιστή και οι ακροδέκτες του*

Η επαναλαμβανόμενη χρήση (δομή) τέτοιων εξαρτημάτων είναι πολύ χρήσιμη, για παράδειγμα, στην υλοποίηση ενός $n-bit$ αθροιστή με χρήση n αθροιστών του $1-bit$. Συνήθως κάθε μονάδα (από την επαναλαμβανόμενη δομή) ξεχωρίζεται με ένα χαρακτηριστικό **όνομα (name)**. Καθώς όλες οι μονάδες του ίδιου τύπου έχουν τους ίδιους ακροδέκτες, κάθε ακροδέκτης μιας συγκεκριμένης μονάδας ονομάζεται με το όνομα της αντίστοιχης μονάδας και το όνομα του αντίστοιχου ακροδέκτη διαχωριζόμενο με μια τελεία. Για παράδειγμα εάν υπάρχουν δύο πλήρεις αθροιστές

$add1$ και $add2$, οι αντίστοιχοι ακροδέκτες των αθροισμάτων ονομάζονται $add1.sum$ και $add2.sum$.

Στην Εικόνα 1-10 απεικονίζεται μια πιο σύνθετη μονάδα η οποία συγκροτείται από δύο μικρότερες. Απεικονίζεται επίσης τόσο το όνομα όσο και ο τύπος της κάθε μονάδας, όπως για παράδειγμα *large(bx)* όπου *large* είναι το όνομα της μονάδας και *bx* είναι ο τύπος της. Τέλος, απεικονίζονται τα ονόματα όλων των ακροδεκτών των μονάδων καθώς επίσης και τα εσωτερικά καλώδια τα οποία συνδέουν τους ακροδέκτες.



Εικόνα 1-10 *Ιεραρχικός λογικός σχεδιασμός*

Οι λογικές μονάδες που συνθέτουν το κύκλωμα της Εικόνας 1-10 είναι δυνατό να περιγραφούν με δύο ισοδύναμους τρόπους (λίστες): τη **λίστα των συνδέσεων (net list)** ή τη **λίστα των εξαρτημάτων (component list).** Στη λίστα των συνδέσεων, για κάθε σύνδεση δίνονται τα άκρα των καλωδίων που συνδέονται σε αυτή τη σύνδεση. Παρακάτω δίνεται η λίστα των συνδέσεων για την Εικόνα 1-10:

net2: large.p1, b1.a;

net1: large.p2, b1.c;

cxnet: b1.b, sand.i1;

o1net: large.p3, sand.o1;

o2net: sand.o2, large.p4.

Η λίστα των στοιχείων, για κάθε μονάδα δίνει το αντίστοιχο καλώδιο που το συνδέει με το κάθε ακροδέκτη:

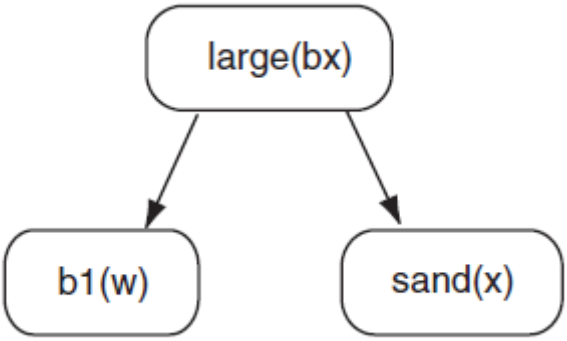
large: p1: net2, p2: net1, p3: o1net, p4: o2net;

b1: a: net2, b: cxnet, c: net1;

sand: i1: cxnet, o1: o1net, o2: o2net;

Δεδομένου ενός τρόπου περιγραφής, είναι δυνατή η μετατροπή στον άλλον τρόπο περιγραφής. Το ποια από τις δύο περιγραφές χρησιμοποιείται εξαρτάται από την εφαρμογή και από το σχεδιαστή. Κάθε αρχείο που περιέχει πληροφορίες σχετικές με τις ηλεκτρικές συνδέσεις των μονάδων ονομάζεται netlist αρχείο (netlist file) ακόμα και αν έχει συγγραφεί σε μορφή λίστας εξαρτημάτων.

Τα εξαρτήματα διαμορφώνουν μια ιεραρχία. Η ιεραρχία των εξαρτημάτων του παραδείγματος της Εικόνας 1-10 φαίνεται στην Εικόνα 1-11. Κάθε στρογγυλεμένο κουτί εκφράζει ένα εξάρτημα και το βέλος από το ένα εξάρτημα στο άλλο δείχνει ότι το εξάρτημα που καταλήγει το βέλος αποτελεί μέρος (είναι συστατικό) του εξαρτήματος που ξεκινάει το βέλος.



Εικόνα 1-11 *Ιεραρχία εξαρτημάτων*

Άλλες φορές απαιτείται διαφορετική ονοματολογία για τη διάκριση των εξαρτημάτων. Σε αυτή τη περίπτωση μπορεί να αναφερθεί είτε σαν *large/bw* ή *large/sand*, όπου αρχικά δηλώνεται το εξάρτημα με το υψηλότερο επίπεδο ιεραρχία και τα ονόματα των εξαρτημάτων που το αποτελούν διαχωρίζονται με τις καθέτους. (Η ομοιότητα αυτού του τρόπου ονοματολογίας με την αντίστοιχη του UNIX είναι σκόπιμη – πολλά σχεδιαστικά εργαλεία χρησιμοποιούν αρχεία και καταλόγους για την αναπαράσταση ιεραρχιών.)

Κάθε εξάρτημα χρησιμοποιείται σα μαύρο κουτί και για τη κατανόηση της λειτουργίας του συστήματος απαιτείται η γνώση της συμπεριφοράς των εισόδων και των εξόδων της μονάδας και όχι πως υλοποιείται στο εσωτερικό της. Για το σχεδιασμό κάθε μαύρου κουτιού απαιτείται ο σχεδιασμός άλλων απλούστερων κουτιών. Το εσωτερικό του κάθε κουτιού καθορίζει τη συμπεριφορά του ως προς τα εξαρτήματα που θα χρησιμοποιηθούν για το σχεδιασμό του. Εάν είναι γνωστή η συμπεριφορά των στοιχειωδών εξαρτημάτων, όπως για παράδειγμα του τρανζίστορ,

είναι δυνατή η πρόβλεψη της συμπεριφοράς οποιουδήποτε ιεραρχικά περιγραφόμενου εξαρτήματος.

Οι σχεδιαστές μπορούν εύκολα να κατανοήσουν ένα ιεραρχικό σχεδιασμό αποτελούμενο από 10.000.000 λογικές πύλες σε σχέση με έναν σχεδιασμό αποτελούμενο από τις ίδιες πύλες απλά συνδεδεμένες μεταξύ τους. Ο ιεραρχικός σχεδιασμός βοηθά στην οργάνωση της σκέψης, καθώς η ιεραρχία οργανώνει μια συνάρτηση ενός μεγάλου συνόλου τρανζίστορ σε μια απλούστερη συνάρτηση. Ο ιεραρχικός σχεδιασμός επίσης διευκολύνει την επαναχρησιμοποίηση τμημάτων του ολοκληρωμένου, είτε με τροποποίηση ενός παλιού σχεδίου για την εκτέλεση νέων λειτουργιών ή με τη χρήση ενός εξαρτήματος για νέους σκοπούς.

1.4.3 Σχεδιαστικά Επίπεδα Αφαίρεσης

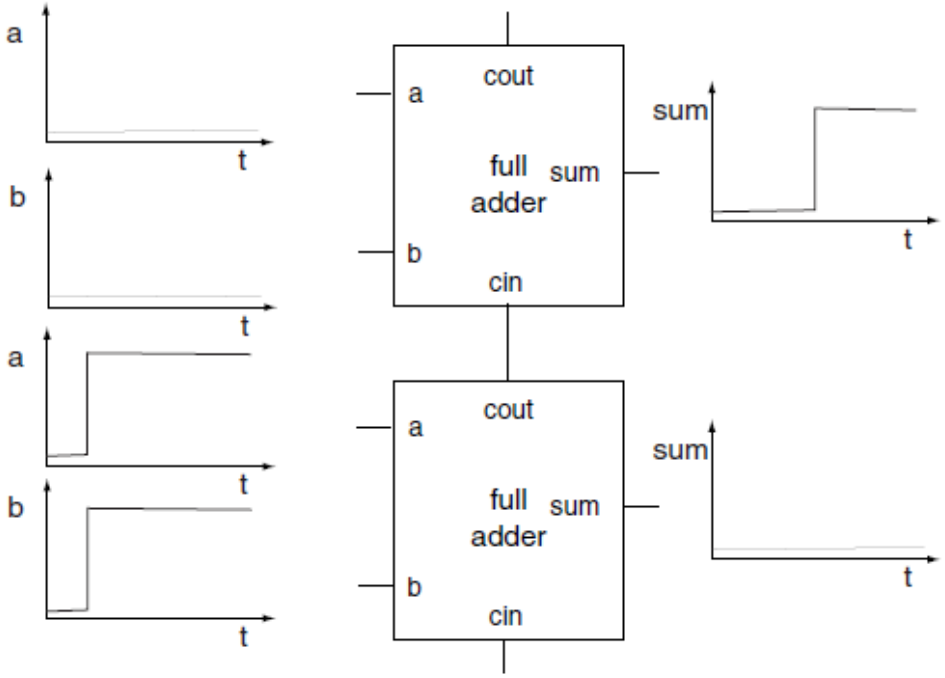
Τα σχεδιαστικά επίπεδα αφαίρεσης είναι σημαντικά για το σχεδιασμό συστημάτων υλικού. Οι σχεδιαστές συστημάτων υλικού χρησιμοποιούν πολλαπλά επίπεδα αφαίρεσης, για να διαχειριστούν το σχεδιασμό τους και να είναι σίγουροι ότι έχουν επιτύχει όλους τους στόχους τους, για παράδειγμα ως προς τη ταχύτητα και τη κατανάλωση ισχύος. Το πιο απλό παράδειγμα σχεδιαστικού επιπέδου αφαίρεσης είναι η λογική πύλη. Μια λογική πύλη είναι η απλούστευση ενός μη-γραμμικού κυκλώματος που χρησιμοποιείται για τη δημιουργία μιας πύλης: μια λογική πύλη δέχεται σαν εισόδους δυαδικές Boolean τιμές. Κάποιοι σχεδιαστικοί στόχοι, όπως για παράδειγμα ο ακριβής υπολογισμός καθυστέρησης είναι δύσκολο ή αδύνατο να γίνει, όταν γίνεται αναφορά σε λογικές πύλες. Ωστόσο, άλλοι σχεδιαστικοί στόχοι, όπως για παράδειγμα, η βελτιστοποίηση της λογικής είναι αρκετά χρονοβόρα διαδικασία, για να γίνει σε ένα κύκλωμα. Κάθε φορά επιλέγεται η καλύτερη προσέγγιση σχεδιαστικού επιπέδου αφαίρεσης ανάλογα με το σκοπό.

Είναι δυνατόν να γίνει χρήση μεγαλύτερου επιπέδου αφαίρεσης για να παρθούν οι πρώτες αποφάσεις και αργότερα να βελτιωθεί η προσέγγιση χρησιμοποιώντας πιο ακριβή μοντέλα: για παράδειγμα, συνήθως, γίνεται βελτιστοποίηση της λογικής χρησιμοποιώντας απλά μοντέλα καθυστέρησης, και έπειτα η λογική επαναπροσεγγίζεται χρησιμοποιώντας αναλυτικές πληροφορίες.Τα σχεδιαστικά επίπεδα αφαίρεσης δεν είναι το ίδιο πράγμα με τον ιεραρχικό σχεδιασμό. Ο σχεδιασμός σε ένα επίπεδο ιεραρχίας χρησιμοποιεί εξαρτήματα του ίδιου επιπέδου αφαίρεσης – για παράδειγμα, μια αρχιτεκτονική χρησιμοποιεί Boolean λογικές συναρτήσεις – και κάθε επίπεδο ιεραρχίας αυξάνει τη πολυπλοκότητα προσθέτοντας εξαρτήματα. Ο αριθμός των εξαρτημάτων μπορεί να μην αλλάξει δεδομένου ότι ανασχηματίζεται σε ένα χαμηλότερο επίπεδο αφαίρεσης – η επιπλέον πολυπλοκότητα επέρχεται χάρη στη περίπλοκη συμπεριφορά αυτών των εξαρτημάτων.

Το επόμενο παράδειγμα καταδεικνύει τον τρόπο με τον οποίο ένα κύκλωμα μπορεί να χρησιμοποιηθεί για τη δημιουργία ενός πολύπλοκου κυκλώματος σε υλικό.

Παράδειγμα 1-1 Σχεδιασμός ψηφιακού κυκλώματος με χρήση επιπέδων αφαίρεσης

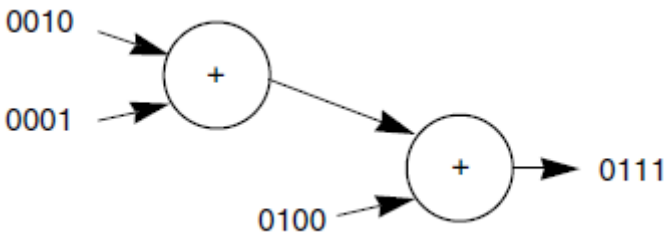
Το βασικό στοιχείο ενός FPGA είναι το τμήμα λογικής. Είναι δυνατή η υλοποίηση συναρτήσεων όπως για παράδειγμα ενός αθροιστή με χρήση αυτών των τμημάτων λογικής.



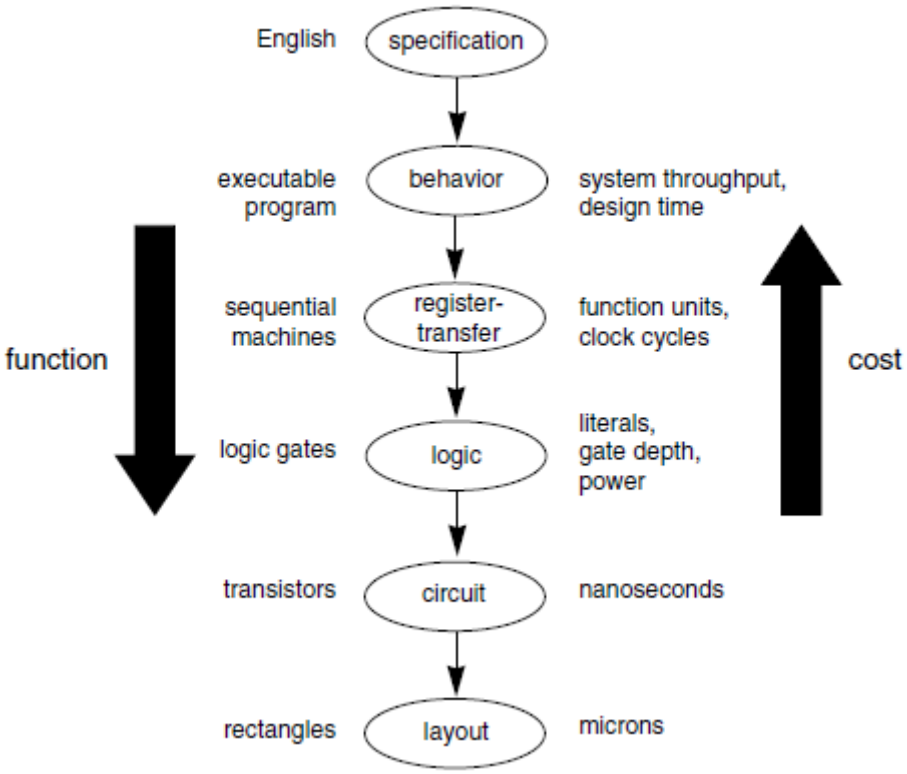
Ο καθορισμός όλων των κυματομορφών για όλα τα σήματα του FPGA δεν είναι τόσο πρακτική διαδικασία. Εντούτοις κάνοντας λογικές υποθέσεις είναι δυνατόν να καθοριστεί κατά προσέγγιση η καθυστέρηση στο λογικό δικτύωμα του FPGA.

Κατά το σχεδιασμό μεγάλων κυκλωμάτων σε επίπεδο καταχωρητή, όπως για παράδειγμα μονοπάτια επεξεργασίας δεδομένων, ίσως είναι απαραίτητος ο σχεδιασμός των αθροιστών κατά ένα ακόμα επίπεδο αφαίρεσης:

Από τη στιγμή που δεν είναι γνωστή η δομή των αθροιστών δεν μπορεί να εξαχθεί κανένα συμπέρασμα για τη καθυστέρηση του κυκλώματος. Αυτά τα κυκλώματα είναι συνδυαστικά – παράγουν μια έξοδο δεδομένης μιας τιμής εισόδου. Η σχεδιαστική αφαίρεση των αθροιστών βοηθά στη κατανόηση της λειτουργίας της συνάρτησης που χρησιμοποιούν πριν το προβληματισμό για την απόδοση του συστήματος.



Η Εικόνα 1-12 παρουσιάζει τα σχεδιαστικά επίπεδα αφαίρεσης των FPGAs:



Εικόνα 1-12 *Τα σχεδιαστικά επίπεδα αφαίρεσης του FPGA*

- **Συμπεριφορά (Behavior).** Είναι μια αναλυτική περιγραφή του τι πρέπει να εκτελεί το ολοκληρωμένο, αλλά όχι πως το εκτελεί. Για παράδειγμα ένα πρόγραμμα σε γλώσσα C, μπορεί να χρησιμοποιηθεί σαν περιγραφή συμπεριφοράς (behavioral description). Το πρόγραμμα στη C δεν μπορεί να μιμηθεί τη χρονική συμπεριφορά του ολοκληρωμένου, αλλά επιτρέπει τη περιγραφή με λεπτομέρειες του τι απαιτείται να υπολογιστεί, σφάλματα και οριακές συνθήκες.
- **Επίπεδο καταχωρητή (Register transfer).** Η χρονική συμπεριφορά του συστήματος είναι πλήρως καθορισμένη – είναι γνωστές οι επιτρεπτές είσοδοι και έξοδοι σε κάθε κύκλο ρολογιού – αλλά η λογική δεν είναι καθορισμένη ως λογικές πύλες. Το σύστημα είναι καθορισμένο ως Boolean συναρτήσεις αποθηκευμένες σε αφηρημένα στοιχεία μνήμης. Μόνο ασαφείς εκτιμήσεις της καθυστέρησης και της καλυπτόμενης επιφάνειας μπορούν να γίνουν από τις Boolean συναρτήσεις.
- **Λογική (Logic).** Το σύστημα είναι σχεδιασμένο με Boolean λογικές πύλες, μανδαλωτές, και flip-flops. Είναι γνωστά πάρα πολλά στοιχεία από τη δομή του συστήματος αλλά ακόμα είναι αδύνατος ο ακριβής υπολογισμός της καθυστέρησης του κυκλώματος.
- **Διαμόρφωση (Configuration).** Η λογική του συστήματος πρέπει να τοποθετηθεί στα λογικά στοιχεία του FPGA και να γίνουν οι απαραίτητες διασυνδέσεις μεταξύ αυτών. Η χωροθέτηση και η δρομολόγηση εκτελούν αυτά τα βήματα.

Ο σχεδιασμός απαιτεί εργασία κάτω από το υψηλότερο επίπεδο αφαίρεσης και πάνω από τη λιγότερο αφηρημένη μορφή. Προφανώς, η εργασία πρέπει να προσδίδει λεπτομέρεια στην αφαίρεση – ο **top-down** σχεδιασμός προσθέτει λειτουργικές λεπτομέρειες. Αλλά στο top-down τρόπο σχεδιασμού οι αποφάσεις παίρνονται έχοντας λίγες πληροφορίες: υπάρχουν αρκετοί εναλλακτικοί σχεδιασμοί σε κάθε επίπεδο αφαίρεσης και πρέπει να βρεθεί ο κατάλληλος που ικανοποιεί τις απαιτήσεις σε ταχύτητα, σε καλυπτόμενη επιφάνεια και σε καταναλισκόμενη ισχύ. Συνήθως είναι αδύνατος ο ακριβής υπολογισμός εάν δεν υπάρχει ένα αρχικό σχέδιο. Ο **Bottom-up** σχεδιασμός μεταφέρει πληροφορίες σε υψηλότερα επίπεδα αφαίρεσης, κατά τεκμήριο χρησιμοποιούνται καλύτερες πληροφορίες ως προς τη καθυστέρηση του κυκλώματος οπότε είναι πιο εύκολος ο επανασχεδιασμός αν απαιτείται. Η εμπειρία βοηθάει στην εκτίμηση του κόστους, αλλά στη πραγματικότητα ένας σχεδιασμός απαιτεί κύκλους από το top-down τρόπο ακολουθούμενο από το bottom-up τρόπο σχεδιασμού.

1.4.4 Μεθοδολογίες

Τα σύνθετα προβλήματα μπορούν να επιλυθούν χρησιμοποιώντας **μεθοδολογίες**. Είναι δυνατόν να χρησιμοποιηθεί η διδασκαλία από ένα κύκλωμα για το σχεδια-

σμό ενός άλλου κυκλώματος. Οι μεθοδολογίες σχεδιασμού των ψηφιακών κυκλωμάτων έχουν αναπτυχθεί μετά από εμπειρία μερικών χρόνων. Μια μεθοδολογία παρέχει οδηγίες για το τι πρέπει να κάνουμε, πότε πρέπει να το κάνουμε και γιατί. Κάποιες μεθοδολογίες είναι αποκλειστικές για χρήση σε FPGA, αλλά κάποιες άλλες είναι γενικές και μπορούν να εφαρμοστούν σε όλα τα ψηφιακά συστήματα.

Οι σχεδιαστές ψηφιακών κυκλωμάτων βασίζονται στις **γλώσσες περιγραφής υλικού (hardware description languages – HDL)** για τη περιγραφή των ψηφιακών συστημάτων. Τα σχηματικά κυκλώματα σπάνια χρησιμοποιούνται για τη περιγραφή λογικής, και τα διαγράμματα συχνά περιγράφουν υψηλότερα επίπεδα ιεραρχίας και συνήθως σε κείμενα. Οι HDL συνδέονται με **προσομοιωτές (simulators)** οι οποίοι προσομοιώνουν τη λειτουργία του συστήματος. Συνδέονται επίσης με **εργαλεία σύνθεσης (synthesis tools)** που δημιουργούν την υλοποίηση των συστημάτων.

Οι μεθοδολογίες έχουν τους εξής στόχους:

- **Ορθή Λειτουργικότητα (Correct functionality).** Με απλά λόγια το ολοκληρωμένο πρέπει να δουλεύει.
- **Ικανοποίηση των στόχων ως προς την απόδοση και τη κατανάλωση ισχύος (satisfaction of performance and power goals).** Μια διαφορετική πτυχή το ότι το ολοκληρωμένο δουλεύει σύμφωνα με τις προδιαγραφές που έχουν οριστεί.
- **Άμεση απασφαλμάτωση (Catching bugs early).** Όσο πιο αργότερα γίνεται η απασφαλμάτωση τόσο μεγαλύτερο είναι το κόστος της. Η σωστή μεθοδολογία ελέγχει τα τμήματα των σχεδίων, ώστε να είναι σίγουρο ότι είναι ορθά σχεδιασμένα.
- **Καλή τεκμηρίωση (Good documentation).** Ένα πρότυπο κείμενο που περιγράφει το σχέδιο επιτρέπει στου σχεδιαστές τον έλεγχο σε διαδοχικά σημεία του σχεδιασμού έτσι ώστε να είναι σίγουροι ότι όλα βαίνουν καλώς.

Η δημιουργία μιας μορφής λειτουργικού προτύπου είναι αρκετά χρήσιμο εργαλείο. Ένα τέτοιο πρότυπο μπορεί να συγγραφεί με μια γλώσσα προγραμματισμού υψηλού επιπέδου όπως για παράδειγμα η C. Αυτά τα πρότυπα είναι χρήσιμα επειδή μπορούν να συγγραφούν αδιαφορώντας για το τρόπο που δημιουργείται ένα σχέδιο. Οι έξοδοι ενός λειτουργικού προτύπου θα μπορούσε να χρησιμοποιηθεί για τον έλεγχο της ορθής λειτουργίας του σχεδίου σε επίπεδο καταχωρητή (register-transfer). Όταν ένα σχέδιο βασίζεται σε ήδη υπάρχοντα πρότυπα, αυτά περιλαμβάνουν κάποιες εκτελέσιμες προδιαγραφές που μπορούν να χρησιμοποιηθούν για τον έλεγχο της ορθής λειτουργίας της υλοποίησης.

Η μετατροπή της λειτουργικής περιγραφής σε επίπεδο-καταχωρητή απαιτεί πολλά στάδια. Πρέπει να σχεδιαστεί η αρχιτεκτονική του συστήματος με βάση τα βασικά συστατικά του. Τα συστατικά αυτά έπειτα διαιρούνται σε άλλα μικρότερα τμήματα

λογικής τα οποία σχεδιάζονται χωριστά. Κάθε τμήμα λογικής πρέπει να ελεγχθεί λειτουργικά. Καθώς τα τμήματα αυτά επανασυνδέονται, τα βασικά συστατικά πρέπει επίσης να ελεγχθούν λειτουργικά.

Στο επίπεδο καταχωρητή είναι δυνατόν να γίνει ανάλυση ως προς το χρόνο εκτέλεσης και τη κατανάλωση ισχύος. Ωστόσο, για την ακριβή ανάλυση απαιτούνται πληροφορίες για τη χωροθέτηση και τη διασύνδεση της λογικής στο ολοκληρωμένο. Τα εργαλεία που χρησιμοποιούνται για τη χωροθέτηση και τη δρομολόγηση απαιτείται να εκτελεστούν αρκετές φορές, για να δώσουν πληροφορίες για την αποσύνθεση του κυκλώματος σε λογικά στοιχεία καθώς και για τις διασυνδέσεις του ολοκληρωμένου.

1.5 Σύνοψη

Η βιομηχανία κατασκευής εφαρμογών VLSI ακολουθεί πιστά τον νόμο του Moore για πολλές δεκαετίες, παρέχοντας ολοκληρωμένα κυκλώματα με πολλά εκατομμύρια τρανζίστορ. Μια χρήση αυτών των τρανζίστορ είναι η απλοποίηση του σχεδιασμού των ολοκληρωμένων. Τα FPGAs αξιοποιούν τα περίπλοκα ολοκληρωμένα με σκοπό την βελτίωση του σχεδιασμού τους με διάφορους τρόπους: Μπορούν να προγραμματιστούν γρήγορα, τα χαρακτηριστικά απόδοσης τους είναι προβλέψιμα και το ίδιο τμήμα υλικού μπορεί να χρησιμοποιηθεί για πολλά διαφορετικά σχέδια λογικής. Στα επόμενα κεφάλαια του βιβλίου θα μελετηθούν τα χαρακτηριστικά των FPGAs και των τεχνικών για το σχεδιασμό ψηφιακών συστημάτων σε FPGAs.

1.6 Προβλήματα

1-1. Περιγράψτε τη λειτουργία των παρακάτω συναρτήσεων με τη βοήθεια πινάκων αλήθειας.

α. a & b.

β. a |~ b.

γ. (a & b) |~ c.

δ. w | (x &~ y).

ε. w | (~x & ~y) | z.

1-2. Ο ιεραρχικός σχεδιασμός βοηθάει την επαναχρησιμοποίηση τμημάτων ενός σχεδίου. Στην εποχή μας μια μεγάλη κεντρική μονάδα επεξεργασίας περιλαμβάνει περίπου 10 εκατομμύρια τρανζίστορ.

α) Πόσο χρόνο θα απαιτούσε ο σχεδιασμός μιας ΚΜΕ αν ο σχεδιασμός σε μια γλώσσα περιγραφής υλικού απαιτούσε χρόνο 1 δευτερόλεπτο για κάθε τρανζίστορ?

β) Πόσο χρόνο θα απαιτούσε αν χρησιμοποιούνταν ο μισός αριθμός τρανζίστορ για τη κρυφή μνήμη, η οποία έχει σχεδιαστεί από κύτταρα των 6 τρανζίστορ τα οποία επαναλαμβάνονται ώστε να σχεδιάσουν τη κρυφή μνήμη.

γ) Πόσο χρόνο θα απαιτούσε αν ο μισός αριθμός τρανζίστορ από αυτά που δεν χρησιμοποιούνται στη κρυφή μνήμη, είχαν υλοποιηθεί σε διάταξη 32 όμοιων ψηφιοφετών, *requiring one copy of the bit slice to be drawn, with the rest created by replication*?

1-3. Σχεδιάστε το λογικό διάγραμμα ενός πλήρους αθροιστή (συμβουλευτείτε το κεφάλαιο 4 εάν δεν γνωρίζετε τη λειτουργία του πλήρους αθροιστή). Ονοματίστε κάθε λογική πύλη. Σχεδιάστε έναν πλήρη αθροιστή των 4-bit με χρήση τεσσάρων πλήρων αθροιστών. Ονοματίστε το κάθε εξάρτημα του 4-bit αθροιστή και ορίστε τον 4-bit αθροιστή ως τύπο. Σχεδιάστε την ιεραρχία των εξαρτημάτων για τον 4-bit αθροιστή, τον πλήρη αθροιστή και τις λογικές πύλες. Όταν ένα εξάρτημα επαναλαμβάνεται, μπορείτε να σχεδιάσετε τα υπο-εξαρτήματά του μια φορά και αναφερθείτε σε αυτά οπουδήποτε στο διάγραμμα.

1-4. Να σχεδιάσετε ένα διάγραμμα του κόστους των FPGAs σε συνάρτηση με το μέγεθος του. Επιλέξτε μια οικογένεια FPGA, ανατρέξτε σε έναν προμηθευτή ολοκληρωμένων κυκλωμάτων (π.χ. σε κάποια ιστοσελίδα του) και αναζητήστε τις τιμές διαφόρων FPGAs. Να δώσετε το σχεδιάγραμμα. Είναι το κόστος γραμμική συνάρτηση σε σχέση με το μέγεθος του?