Γλώσσες Περιγραφής

Μοντέλα

Ένα μοντέλο ενός κυκλώματος είναι μία αναπαράσταση που παρουσιάζει χαρακτηριστικά χωρίς να συνοδεύεται από λεπτομέρειες.

- ✓ Τα τυπικά μοντέλα έχουν καλά ορισμένη σύνταξη.
- ✓ Τα αυτόματα εργαλεία σχεδιασμού επεξεργάζονται τις πληροφορίες τους.
- ✓ Παράμετροι μοντελοποίησης κυκλώματος:

  1. επίπεδο αφαίρεσης (αρχιτεκτονικής, λογικό, γεωμετρικό),

  2. περιγραφή (συμπεριφοράς, κατασκευής, φυσική),

  3. μέσα (γλώσσα, διαγράμματα, μαθηματικά μοντέλα).
- ✓ Οι HDLs μοιάζουν με τις κοινές γλώσσες προγραμματισμού.
- ✓ Τα αφηρημένα μοντέλα είναι είναι μαθηματικά μοντέλα που βασίζονται σε γράφους και Boolean algebra.

Μοντέλα

- ✓ **Αρχιτεκτονική**: η συμπεριφορά περιγράφεται με λειτουργίες (operations) και εξαρτήσεις (dependencies).
- ✓ Συμπεριφορά ακολουθιακών κυκλωμάτων: μηχανή πεπερασμένων καταστάσεων (FSM).
- ✓ **Περιγραφή κατασκευής**: διασυνδέσεις λογικών blocks ή πυλών (επίπεδο λογικής) ή resources (επίπεδο αρχιτεκτονικής).

Γλώσσες Μοντελοποίησης

Οι HDLs έχουν μερικές διαφορές από τις κοινές γλώσσες προγραμματισμού:

- ✓ (Κυκλώματα = παράλληλες λειτουργίες) ≠ (Προγράμματα = σειριακά).
- ✓ Οι προδιαγραφές κυκλωμάτων περιέχουν και κατασκευαστικές πληροφορίες (structural) όπως θύρες I/O.
- ✓ Επιπλέον μπορεί να περιέχουν και τυχόν περιορισμούς.
- ✓ Ο χρονισμός των λειτουργιών είναι πολύ απαραίτητος στο Hardware.

Στο επίπεδο **αρχιτεκτονικής** και το **λογικό** επίπεδο χρησιμοποιούνται οι περιγραφές **κατασκευής**, **συμπεριφοράς** και συνδυασμοί.

Τα εργαλεία σύνθεσης παρέχουν την δυνατότητα μετατροπής από την περιγραφή **συμπεριφοράς** στην περιγραφή **κατασκευής**

Χαρακτηριστικά των HDLs

Οι HDLs έχουν τα ακόλουθα χαρακτηριστικά:

- ✓ Χαρακτηρίζονται από:

  (α) την *σύνταξή* τους (δομή γλώσσας, γραμματική),

  (β) τα *semantics* (έννοιες της γλώσσας),

  (γ) *pragmatics* (θέματα υλοποίησης).

- ✓ Χωρίζονται σε γλώσσες *procedural* και *declarative*:

  (α) *Procedural*: μία λειτουργία = ακολουθία βημάτων με ορισμένη σειρά.

  (β) *Declarative*: σύνολο δηλώσεων χωρίς λεπτομέρειες λύσης.

Χαρακτηριστικά των HDLs

- ✓ Μαζί με κάθε γλώσσα υπάρχει και ο *εξομοιωτής* (event-driven).
- ✓ Η εξομοίωση είναι η εκτίμηση τιμών σημάτων σε ένα παράθυρο χρόνου:

  1) Τα σήματα υπολογίζονται και διαδίδονται.

  2) Οι processes ενεργοποιούνται από τα σήματα, εκτελούνται και σταματούν.
- ✓ Υποστηρίζονται σύγχρονα και ασύγχρονα μοντέλα κυκλωμάτων.
- ✓ Προβλήματα παρουσιάζονται στην εξομοίωση όταν δεν είναι γνωστές οι καθυστερήσεις (δεν έχει γίνει η σύνθεση) και δεν μπορούν να εκτιμηθούν.

Structural HDLs

Μοιάζουν με σχηματικά κυκλωμάτων αφού περιγράφουν διασυνδέσεις σε components. Η ιεραρχία χρησιμοποιείται για την *τμηματοποίηση* και *συμπύκνωση* των περιγραφών.

```
architecture STRUCTURE of HALF_ADDER is
        component AND2
                port (x, y: in bit; o: out bit);
        component EXOR2
                port (x, y: in bit; o: out bit);
        begin
                G1: AND2
                        port map (a, b, carry);
                G2: EXOR2
                        port map (a, b, sum  );
end STRUCTURE;
```

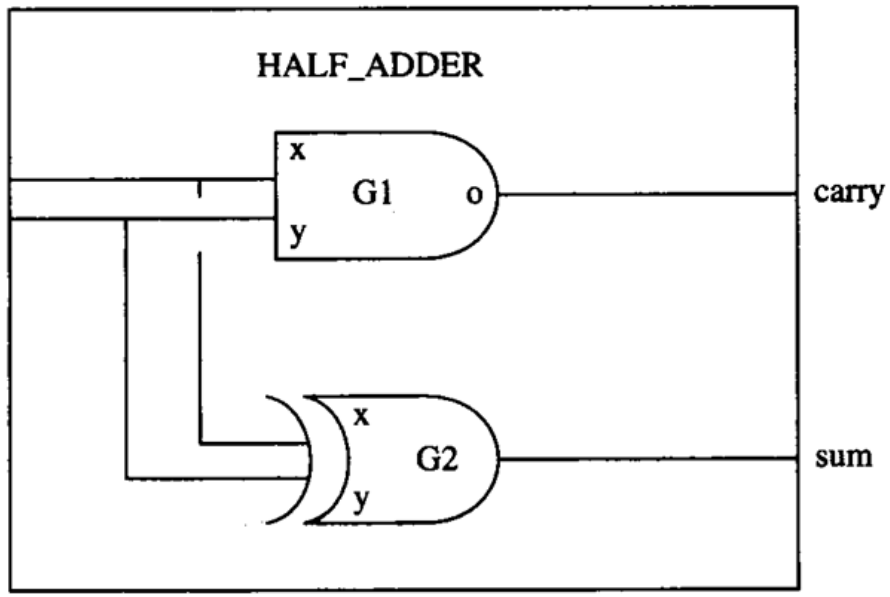


FIGURE 3.2
Structural representation of a half-adder circuit.

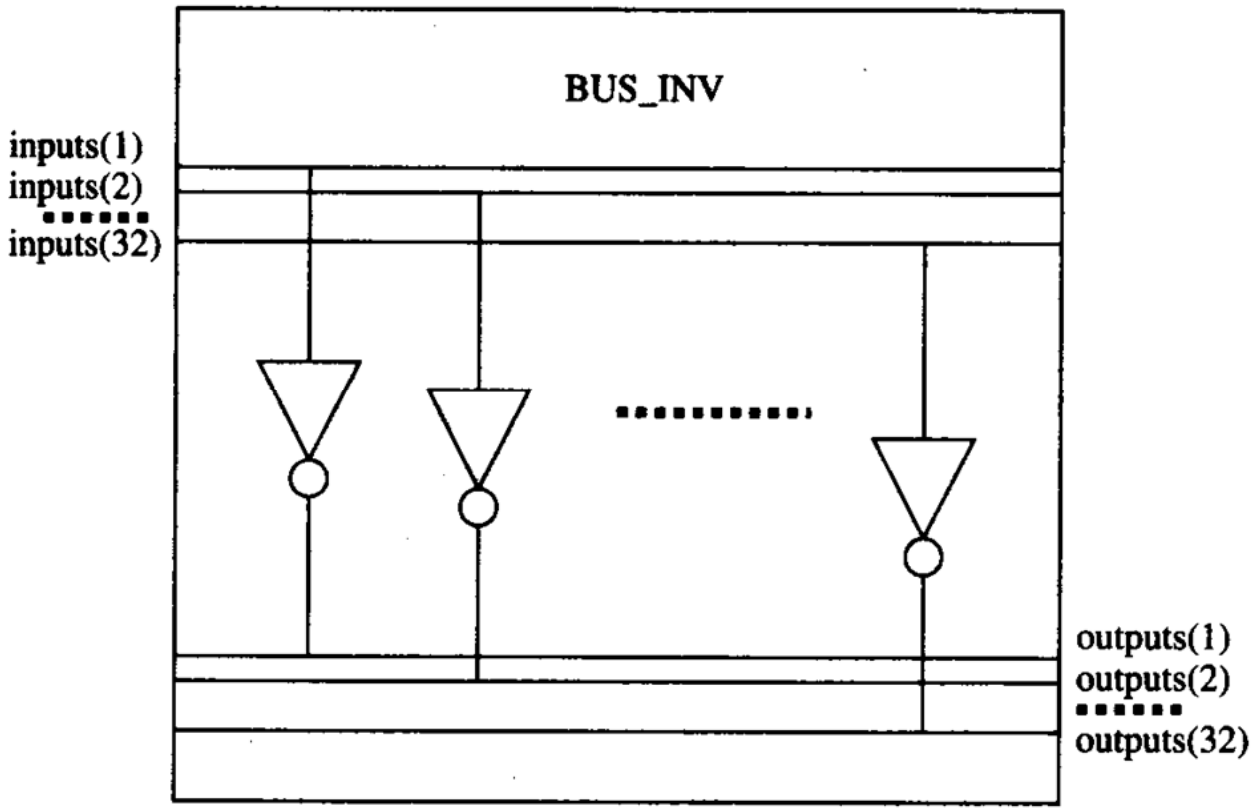
Structural HDLs

Οι μετα-μεταβλητές (metavariables) χρησιμοποιούνται για να κάνουν τα μοντέλα κυκλωμάτων πιο συμπαγή. Δεν αναπαριστούν οντότητες υλικού και απορρίπτονται από τα πρώτα βήματα του Compilation. (πχ ο δείκτης ενός πίνακα)

```
architecture STRUCTURE of BUS_INV is
     component INVERTER
             port (i1: in bit; o1: out bit);
     end component;
     begin
             G: for i in 1 to 32 generate
                     INV: INVERTER port map (inputs(i), outputs(i)
             end generate;
end STRUCTURE;
```

metavariable

Structural HDLs



Behavioral HDLs

- ✓ Μοντελοποίηση συμπεριφοράς: κυκλώματα υψηλής πολυπλοκότητας.
- ✓ Συνδυαστικά λογικά κυκλώματα:

  (α) σύνολο θυρών εισόδου/εξόδου

  (β) σύνολο εξισώσεων που σχετίζουν μεταβλητές με λογικές εκφράσεις.
- ✓ Διαφορά από κατασκευαστικά μοντέλα: δεν υπάρχει ένα-προς-ένα αντιστοιχία μεταξύ εκφράσεων και λογικών πυλών.
- ✓ Οι *Procedural* γλώσσες μπορούν να χρησιμοποιηθούν για την περιγραφή συνδυαστικών λογικών κυκλωμάτων

```
architecture BEHAVIOR of HALF_ADDER is
process
        begin
                carry <= ( a and b ) ;
                sum   <= ( a xor b ) ;
        end process;
end BEHAVIOR;
```

Behavioral HDLs

Η μοντελοποίηση συμπεριφοράς σύγχρονων λογικών κυκλωμάτων εξαρτάται από τα semantics χρονισμού της γλώσσας. Συνήθως χρησιμοποιείται ένα ρολόι αναφοράς και συγχρονισμού

Finite State Machines

```
architecture BEHAVIOR of REC is
        type STATE_TYPE is (STATE_ZERO, STATE_ONE);
        signal STATE : STATE_TYPE := STATE_ZERO;
        process
                begin
                wait until clock'event and clock='1';
                if ( in = '1' ) then
                        case STATE is
                                when => STATE_ZERO
                                        STATE <= STATE_ONE;
                                        out <= '0';
                                when => STATE_ONE
                                        STATE <= STATE_ONE;
                                        out <= '1';
                        end case;
                else
                                        STATE <= STATE_ZERO;
                                        out <= '0';
                end if;
        end process;
end BEHAVIOR;
```

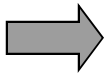
Behavioral HDLs

- ✓ Μοντελοποίηση συμπεριφοράς (επίπεδο αρχιτεκτονικής): ακολουθία αναθέσεων σε μεταβλητές μέσω δομών ελέγχου (διακλάδωση, επανάληψη κλπ).
- ✓ Σαφής διαχωρισμός σε λειτουργίες (αναθέσεις) και εξαρτήσεις (control flow).

```
...
ir <= fetch(pc);
case ir is
        when => AND
                acc <= rega and regb;
        when => OR
                acc <= rega or regb;
        when => XOR
                acc <= rega xor regb;
        when => ADD
                acc <= rega + regb;
end case;
pc <= pc + 1;
...
```

Οι εντολές ελέγχου καθορίζουν τις εξαρτήσεις (dependencies) των λειτουργιών (operations)

Η μοντελοποίηση συμπεριφοράς αφήνει πολλούς βαθμούς ελευθερίας στην σύνθεση και τα εργαλεία βελτιστοποίησης.



Μπορεί να διαφέρει το κύκλωμα της σύνθεσης από το αρχικό μοντέλο.

Behavioral HDLs

Εάν θέλουμε να διασφαλίσουμε την χρονική λειτουργία του κυκλώματος πρέπει να ακολουθήσουμε ακριβή πολιτική χρονισμού.

Παράδειγμα:

Καθυστέρηση του Decoding κατά ένα κύκλο για μείωση χρόνου κύκλου ρολογιού.

Behavioral HDLs

Εάν θέλουμε να διασφαλίσουμε την χρονική λειτουργία του κυκλώματος πρέπει να ακολουθήσουμε ακριβή πολιτική χρονισμού.

```
...
wait until clock'event and clock='1';
ir <= fetch(pc);
```

Κύκλος 1ος

```
wait until clock'event and clock='1';
case ir is
        when => AND
                acc <= rega and regb;
        when => OR
                acc <= rega or regb;
        when => XOR
                acc <= rega xor regb;
        when => ADD
                acc <= rega + regb;
end case;

pc <= pc + 1
...
```

Κύκλος 2ος

Καθυστέρηση του Decoding κατά ένα κύκλο για μείωση χρόνου κύκλου ρολογιού.

Παράδειγμα Περιγραφής Συμπεριφοράς

Πλήρης Αθροιστής

X	Y	CIN	COUT	SUM
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Παράδειγμα Περιγραφής Συμπεριφοράς

Πλήρης Αθροιστής

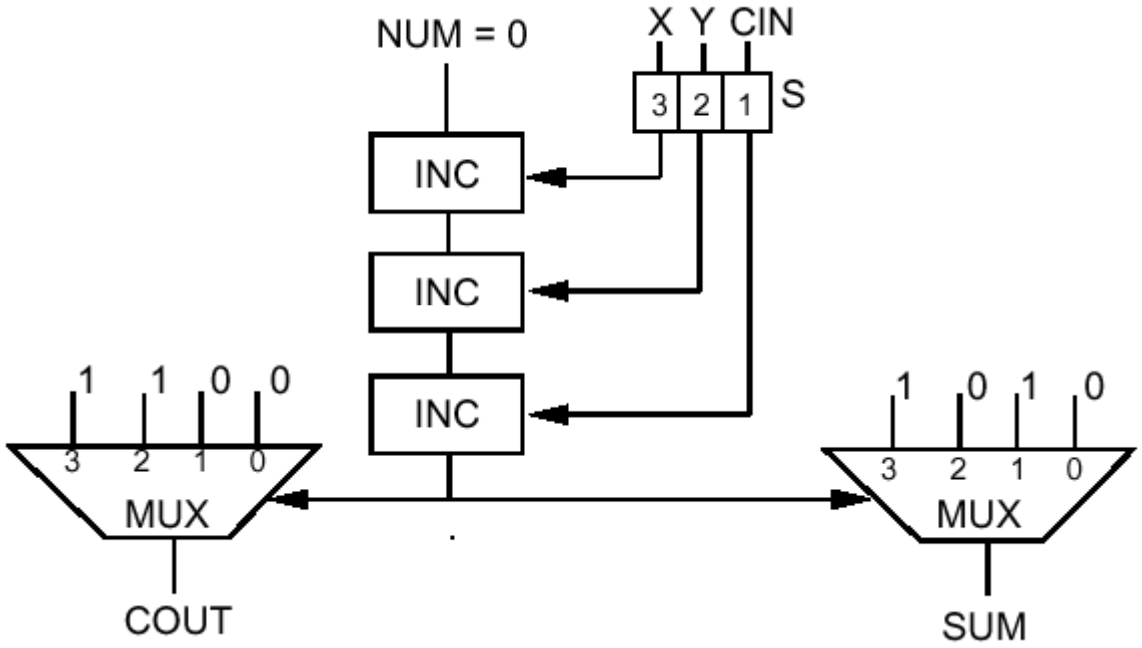
```
architecture FA_BEHAV  of
                FULL_ADDER is
begin
   process(X,Y,CIN)
     variable BV: BIT_VECTOR(1 to 3);
     variable NUM,I: INTEGER;
     variable Stemp, Ctemp: BIT;

    begin
      NUM := 0;
      BV := X & Y & CIN;
      for I := 1 to 3 loop
           if (BV(I) = '1') then
                NUM := NUM + 1;
           end if;
      end loop;

      case NUM is
           when 0 => Ctemp:='0'; Stemp:='0';
           when 1 => Ctemp:='0'; Stemp:='1';
           when 2 => Ctemp:='1'; Stemp:='0';
           when 3 => Ctemp:='1'; Stemp:='1';
      end case;
      SUM <= Stemp after 3 ns;
      COUT <= Ctemp after 5 ns;
   end process;
end FA_BEHAV;
```

Παράδειγμα Περιγραφής Συμπεριφοράς

Πλήρης Αθροιστής μετά από σύνθεση



VHDL

- ✓Δημιουργήθηκε σαν τμήμα του VHSIC project. Έγινε IEEE πρότυπο το ‘87.
- ✓Είχε σαν στόχο την περιγραφή μεγάλων κυκλωμάτων.
- ✓Δόθηκε έμφαση στην αναγνωσιμότητα (πολύλογη γλώσσα).
- ✓Το μοντέλο ενός κυκλώματος ονομάζεται σχεδιαστική οντότητα (entity) και αποτελείται από το interface και το σώμα αρχιτεκτονικής.

```
entity HALF_ADDER is
        port (a, b: in bit; sum, carry: out bit);
end HALF_ADDER
```

- ✓Το σώμα αρχιτεκτονικής μπορεί να περιγραφεί σε επίπεδο συμπεριφοράς, κατασκευής ή ροής δεδομένων.
- ✓Είναι γλώσσα ισχυρών τύπων.
- ✓Παρέχει σταθερές, μεταβλητές και σήματα.

VHDL

- ✓Οι μεταβλητές δεν σχετίζονται με hardware.
- ✓Τα σήματα έχουν ηλεκτρική υλοποίηση (με καθυστέρηση).
- ✓Η δήλωση wait χρησιμοποιείται για τον συγχρονισμό εκτέλεσης γεγονότων.
- ✓Η VHDL είναι πολύ τμηματοποιημένη γλώσσα. Κάθε block ξεκινάει με την δεσμευμένη λέξη begin και ολοκληρώνεται με την λέξη end.
- ✓Παρέχονται λογικοί, σχεσιακοί και αριθμητικοί τελεστές ενώ επιτρέπεται η υπερφόρτωση των τελεστών.
- ✓Παρέχεται μηχανισμός πακεταρίσματος (packaging) συναρτήσεων και δηλώσεων.
- ✓Παρέχεται η δυνατότητα δημιουργίας και χρήσης βιβλιοθηκών.
- ✓Δίνεται η δυνατότητα τμηματοποίησης και επαναχρησιμοποίησης μοντέλων.

```
package mypack is
  subtype bit8 is integer range 0 to 255;
end mypack;

use work.mypack.all;
entity DIFFEQ is
port(
       dx_port, a_port, x_port,  u_port : in  bit8;
       y_port : inout  bit8;
       clock, start   : in bit );
end diffeq;

architecture BEHAVIOR of DIFFEQ is
begin
process
        variable x, a, y, u, dx, xl, ul, yl:  bit8;
begin
        wait until start'event and start = '1';
        x := x_port; y := y_port; a := a_port;
        u := u_port; dx := dx_port;
        DIFFEQ_LOOP:
        while ( x < a ) loop
                wait until clock'event and clock = '1';
                xl := x + dx;
                ul := u - (3 * x * u * dx) - (3 * y * dx);
                yl := y + (u * dx);
                x  := xl; u := ul ; y := yl;
        end loop DIFFEQ_LOOP;
        y_port <=  y;
end process;
end BEHAVIOR;
```

Verilog

- ✓Η Verilog χρησιμοποιήθηκε για την μοντελοποίηση και εξομοίωση λογικών κυκλωμάτων σε λειτουργικό επίπεδο (αρχιτεκτονικής), λογικό, και επίπεδο διακοπτών (switch).
- ✓Είναι πιο συμπαγής από την VHDL και δεν απαιτεί διαχωρισμό δηλώσεων και αρχιτεκτονικής.

Structural

```
module  HALF_ADDER (a , b , carry , sum);
        input   a , b;
        output  carry, sum;
        and
                G1 (carry, a , b);
        xor
                G2 (sum,   a , b);
endmodule
```

Behavioral

```
module  HALF_ADDER (a , b , carry , sum);
        input   a , b;
        output  carry, sum;

                assign carry = a & b ;
                assign sum   = a ^ b ;
endmodule
```

Verilog

- ✓Παρέχει δύο είδη αναθέσεων: continuous (assign) και procedural. Το continuous γίνεται άμεσα, ενώ το procedural όταν εκτελεστεί το statement.
- ✓Έχει δύο είδη τύπων δεδομένων: register και net.
- ✓Υποστηρίζει εντολές ελέγχου ροής δεδομένων και κλήσεις σε υπορουτίνες και συναρτήσεις.
- ✓Ο συγχρονισμός εκτέλεσης λειτουργιών γίνεται με την εντολή wait.
- ✓Τα μοντέλα συμπεριφοράς χωρίζονται σε αρχικοποίηση (initial) και άπειρο βρόγχο (always).

Verilog

```
module DIFFEQ (x, y, u , dx, a, clock, start);
input [7:0] a, dx;
inout [7:0] x, y, u;
input       clock, start;
reg   [7:0] xl, ul, yl;
always
begin
wait ( start);
        while ( x < a )
                begin
                xl = x + dx;
                ul = u - (3 * x * u * dx) - (3 * y * dx);
                yl = y + (u * dx);
                @(posedge clock);
                x  = xl; u = ul ; y = yl;
                end
endmodule
```

It is instructive to contrast the model to the VHDL model of Example 3.2.10.