ΕΙΣΑΓΩΓΗ ΣΤΗ ΓΛΩΣΣΑ VHDL

Η γλώσσα VHDL

- **VHDL**: **V**HSIC **H**ardware **D**escription **L**anguage
- VHSIC: **V**ery **H**igh-**S**peed **I**ntegrated **C**ircuits

- Ιστορική Αναδρομή: Ξεκίνησε το **1981 από το Υπουργείο Άμυνας των ΗΠΑ** ως γλώσσα περιγραφής ολοκληρωμένων κυκλωμάτων
- Οι εταιρείες IBM, Texas Instruments,καιIntermetrics ανάπτυξαν και κυκλοφόρησαν την 1η έκδοση το 1985

- Πρότυπο από τον οργανισμό IEEE
- **IEEE Standard 1076-1987 (VHDL-87)**
- **IEEE Standard 1076-1993 (VHDL-93)**
- **IEEE Standard 1076a (VHDL-2000**)

Ιεραρχική Σχεδίαση

- Τα κυκλώματα είναι αρκετά πολύπλοκα για να σχεδιάσουμε όλες τις λεπτομέρειες με τη μία
- Σχεδιάζουμε υποσυστήματα για απλές λειτουργίες
- Συνθέτουμε υποσυστήματα για να σχηματίσουμε το σύστημα
- Αντιμετωπίζουμε τα υποκυκλώματα ως «μαύρα κουτιά»
- Επαληθεύουμε ανεξάρτητα τη λειτουργία τους, και έπειτα επαληθεύουμε το σύνθετο σύστημα
- Σχεδίαση top-down (από πάνω προς τα κάτω) ή bottom-up (από κάτω προς τα πάνω)

Βασική μορφή κώδικα VHDL

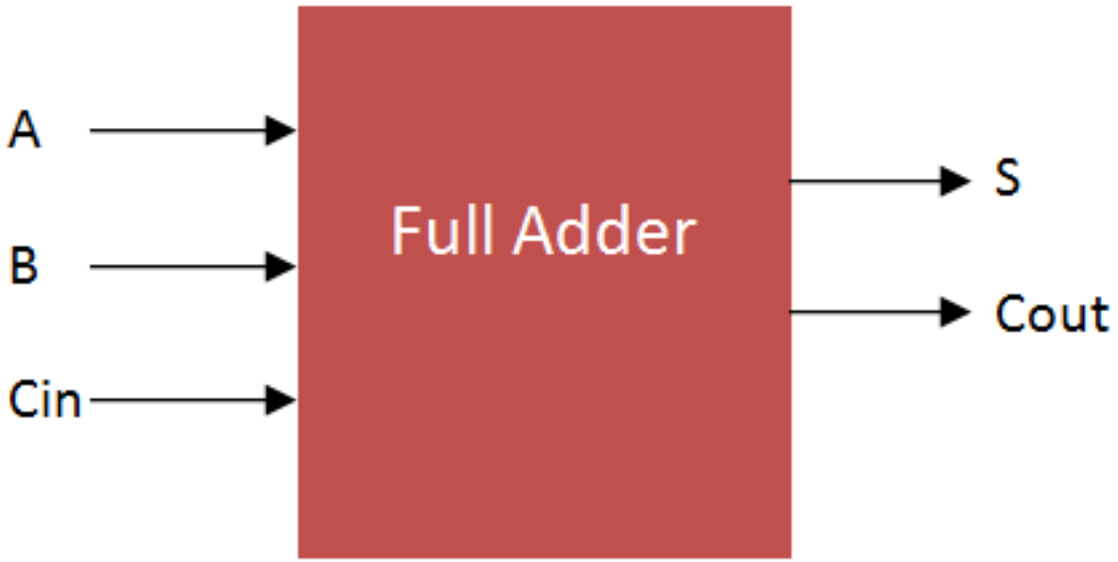
Δηλώσεις βιβλιοθηκών

Οντότητα (Entity)

Περιγραφή αρχιτεκτονικής (Architecture)

- Το τμήμα της οντότητας χρησιμοποιείται για να ορίσει τις θύρες Ε/Ε του κυκλώματος.
- Το τμήμα της αρχιτεκτονικής περιγράφει την συμπεριφορά του κυκλώματος.
- Ένα μοντέλο συμπεριφοράς είναι παρόμοιο με ένα “μαύρο κουτί”.
- Πρότυπες βιβλιοθήκες σχεδίασης περιλαμβάνονται πριν από τον ορισμό της οντότητας.

Πλήρης Αθροιστής



Πίνακας Αλήθειας Πλήρους Αθροιστή

A	B	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

Οντότητα Full Adder

```
entity full_adder  is
        port(A:   in  bit;
             B:   in  bit;
             Cin : in  bit;
             Sum : out bit;
             Cout : out  bit);
end full_adder;
```

in
out
input
buffer

Ορισμός εισόδων και εξόδων

```
architecture  behave of  full_adder  is
    Begin
     Εδώ περιγράφουμε την λειτουργία της οντότητας
    end;
```

Κανόνες Ονομάτων

- Όλα τα ονόματα πρέπει να αρχίζουν με έναν αλφαβητικό χαρακτήρα (a-z ή A-Z).
- Μόνο αλφαβητικοί χαρακτήρες (a-z ή A-Z) ψηφία (0-9) και υπογράμμιση (_).
- Οποιοσδήποτε ειδικός χαρακτήρας σε ένα όνομα (!,?,.,&,+,-,etc.) δεν επιτρέπεται
- Δύο ή περισσότερα διαδοχικά underscore (__) μέσα σε ένα όνομα είναι λάθος (π.χ. This__name).
- Όλα τα ονόματα μίας οντότητας ή αρχιτεκτονικής πρέπει να είναι μοναδικά.

Τύποι Δεδομένων

- **bit** (‘0’or‘1’).
- **Διάνυσμα bit** (πίνακας από bits).
- **Ακέραιος (integer) :** Ελάχιστο εύρος όπως ορίζεται από το πρότυπο: -2,147,483,647 έως 2,147,483,647
- **Πραγματικός (real):** Ελάχιστο εύρος όπως ορίζεται από το πρότυπο: -1.0E38 έως 1.0E38
- **Φυσικός:** Το εύρος ορίζεται από τον χρήστη
- **Χρόνος** (φυσικός τύπος δεδομένων).

Τύποι Δεδομένων

- **Πίνακας (array)**
  - Χρησιμοποιείται για να συλλέξει στοιχεία ίδιου τύπου σε μία δομή.
  - Τα στοιχεία μπορεί να είναι οποιουδήποτε τύπου δεδομένων VHDL.
- **Εγγραφή (record)**
  - Χρησιμοποιείται για να συλλέξει στοιχεία διαφορετικού τύπου σε μία δομή.
  - Τα στοιχεία μπορεί να είναι οποιουδήποτε τύπου της VHDL.
  - Τα στοιχεία προσπελάζονται μέσω του ονόματος του πεδίου.
- **Char και String**

Τύποι Δεδομένων

- STD_LOGIC

Value	Meaning
'X'	Forcing (Strong driven) Unknown
'0'	Forcing (Strong driven) 0
'1'	Forcing (Strong driven) 1
'Z'	High Impedance
'W'	Weak (Weakly driven) Unknown
'L'	Weak (Weakly driven) 0. Models a pull down.
'H'	Weak (Weakly driven) 1. Models a pull up.
'-'	Don't Care

Τύποι δεδομένων (κάποια παραδείγματα)

- Type mychar is array (1 to 6) of character
- Type distance is range 0 to 1E16
- Type bit_vector is array (Natural Range <>) of Bit
- Type String is array (Positive range <>) of Character
- Type Index is range 7 downto 0

Οντότητα Adder 8 ψηφίων

```
library ieee;
use ieee.std_logic_1164.all;

entity adder_8  is
  generic (size: integer := 8)
  port (A, B  :  in std_logic _vector (size-1 downto 0);
          Cin :    in bit;
          Sum : out   std_logic _vector (size-1 downto 0);
         Cout : out  bit);
end adder_8;

architecture  behave of adder_8  is
     Begin
          Εδώ περιγράφουμε την λειτουργία της οντότητας
     end;
```

Για την δημιουργία γενικευμένων οντοτήτων

Πρότυπες Βιβλιοθήκες

- Βιβλιοθήκη της ieee; πριν τον ορισμό της οντότητας.
- **ieee.std_logic_1164:** ορίζει ένα πρότυπο για τους σχεδιαστές για να χρησιμοποιηθεί στην περιγραφή των τύπων δεδομένων διασύνδεσης στην VHDL μοντελοποίηση.
- **ieee.std_logic_arith:** παρέχει ένα σύνολο συναρτήσεων αριθμητικής, μετατροπής, σύγκρισης για τύπους προσημασμένους -signed, μη -unsigned, std_ulogic, std_logic, std_logic_vector.
- **ieee.std_logic_unsigned:** παρέχει ένα σύνολο συναρτήσεων μη προσημασμένης αριθμητικής, μετατροπής και σύγκρισης για std_logic_vector.

Περιγραφή Συμπεριφοράς Πλήρους Αθροιστή

Inputs: A, B, Cin Bits

OutPuts: Sum, Cout Bits

Μετράμε πόσα από τα A,B και Cin εχουν τιμή 1.

Αν όλα είναι μηδέν τότε Sum = 0 , Cout =0;

Αν ένα μόνο είναι 1 τότε Sum = 1 , Cout =0;

Αν δύο είναι 1 τότε Sum = 0 , Cout =1;

Αν και τα τρία είναι 1 τότε Sum = 1 , Cout =1;

Behavioral Description of Full Adder

```
entity full_adder is
        port (A : in bit;
              B : in bit;
             Cin : in  bit;
             Sum : out  bit;
             Cout : out bit);
end full_adder;

architecture behavioral of full_adder is
  begin
```

Εδώ περιγράφουμε αλγοριθμικά την συμπεριφορά της οντότητας

Χρησιμοποιούνται **διαδικαστικές δηλώσεις** οι οποίες:

- μπορεί να είναι διαδοχικές δηλώσεις όπως σε γλώσσες **προγραμματισμού software.**
- χρησιμοποιούνται για να υπολογιστούν οι τιμές των εξόδων από τις τιμές των εισόδων.
- είναι **συχνά πιο ισχυρές**, αλλά πολλές φορές **ταιριάζουν άμεσα με** μία υλοποίηση hardware.

```
  end;
```

Behavioral Description of Full Adder

```
architecture behavioral of full_adder is
begin
    process (A,B,Cin);
    variable s: bit_vector(1 to 3);
    variable num: integer range 0 to 3;

    Begin
        s := A & B & Cin;
        num := 0;
        for i in 1 to 3 loop
            if s(i) ='1' then
                num= num+1;
            end if;
        end loop;

        case num is
            when 0=> Cout <='0'; Sum <='0';
            when 1=> Cout <='0'; Sum <='1';
            when 2=> Cout <='1'; Sum <='0';
            when 3=> Cout <='1'; Sum <='1';
        end case;
    end process;
end;
```

Σε μια διεργασία(process) ο κώδικας εκτελείται διαδοχικά όπως σε ένα πρόγραμμα

Περιγραφή Ροής Δεδομένων Πλήρους Αθροιστή

Inputs: A, B, Cin Bits

OutPuts: Sum, Cout Bits

S1 = A XOR B

Sum = s1 XOR Cin

S2 = (A AND B)

S3 = (A AND Cin)

S4 = (B AND Cin)

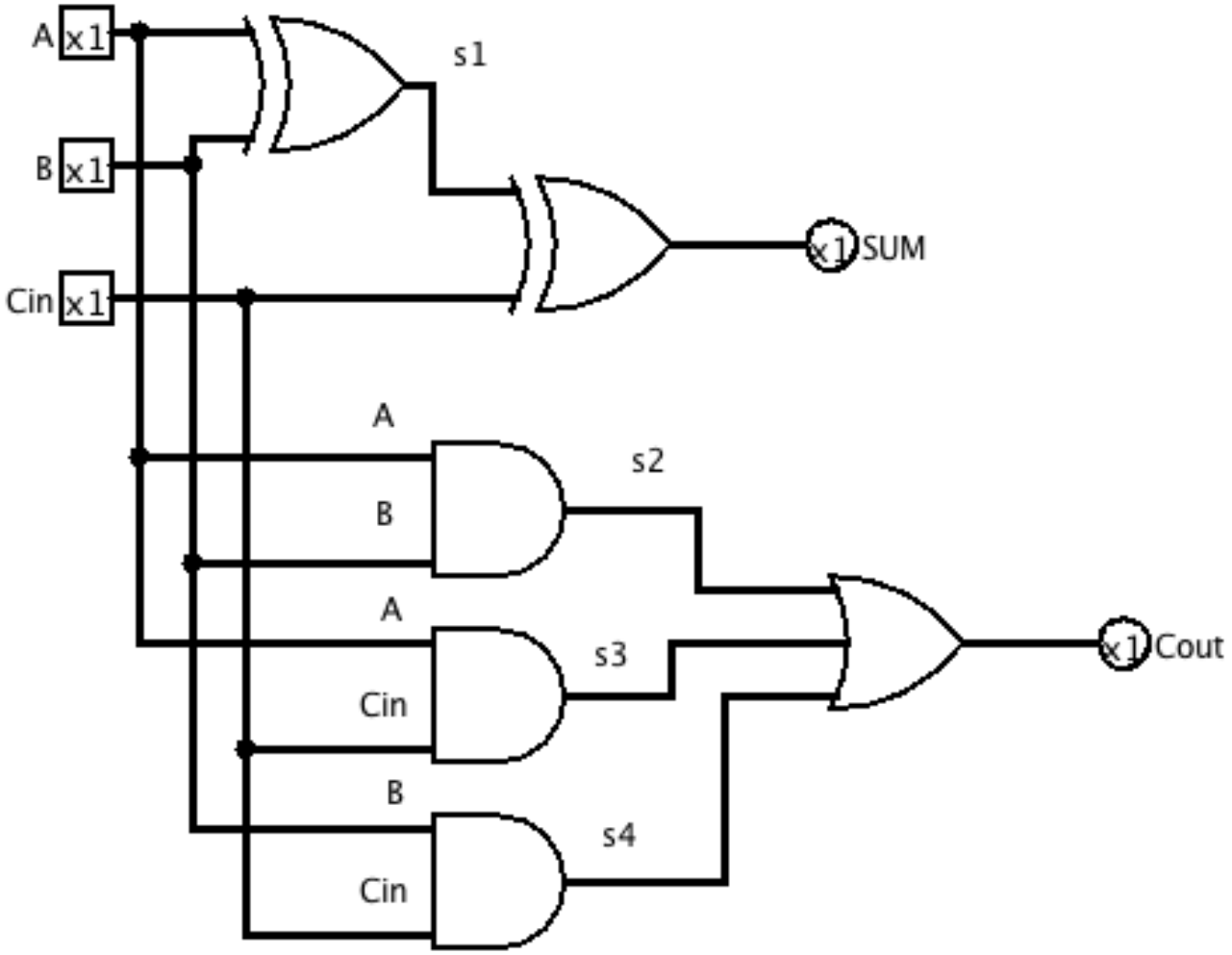
Cout = S2 OR S3 OR S4;

- Τα κυκλώματα περιγράφονται υποδεικνύοντας το πώς **οι είσοδοι και έξοδοι των αρχικών εξαρτημάτων (πχ μια πύλη and) συνδέονται μεταξύ τους.**
- Περιγράφουμε πως τα σήματα (δεδομένα) **ρέουν δια μέσου του κυκλώματος.**

Data flow Description of Full Adder

```
architecture data_flow of full_adder is
 signal s1, s2, s3, s4 : bit;

  begin;
        s1 <= A xor B;
        Sum <= s1 xor Cin;
        s2  <= A and B;
        s3 <=  A and Cin;
        s4 <=  B and Cin;
        Cout  <= s2 or s3 or s4;
  end;
```

Δομική Περιγραφή Πλήρους Αθροιστή



Δομική περιγραφή οντότητας

- Τυπικά αναλύεται σε πολλά **τμήματα-blocks:** Κάθε τμήμα ενός σχεδίου VHDL θεωρείται ως ένα block-οντότητα.
- Η οντότητα περιγράφει τη διεπαφή σε αυτό το block και ένα ξεχωριστό μέρος που σχετίζεται με την οντότητα περιγράφει πως λειτουργεί το block.
- Η περιγραφή της διεπαφής **είναι σαν μια περιγραφή για pin**,σε ένα τεχνικό φυλλάδιο.

- Τα blocks συνδέονται μεταξύ τους για να σχηματίσουν ένα ολοκληρωμένο σχέδιο.
- Ένα σχέδιο VHDL μπορεί να περιγράφεται εξολοκλήρου σε ένα μόνο block,ή μπορεί να αναλύεται σε πολλά blocks.

Structural Description of Full Adder

```
architecture  data_flow  of full_adder  is
      component and2
      port (I1,I2: in bit; O: out bit);
      end component;

      component  xor2
      port (I1,I2: in bit; O: out bit);
      end component;

      component or3
      port (I1,I2, I3: in bit; O: out bit);
      end component;

      signal  s1, s2, s3, s4 : bit;
begin;
        u0 : xor2 port map (A,B,  s1);
        u1 : xor2 port map (s1,Cin, Sum);
        u2 : and2 port map (A,B,  s2);
        u3 : and2 port map (A,Cin,  s3);
        u4 : and2 port map (B,Cin,  s4);
        u5 : or3  port map (s2,s3,  s4, Cout);
end;
```

Structural Description of Full Adder (2)

```
-- TWO INPUT XOR ENTITY
entity xor2 is port (
        I1,I2: in bit; O: out bit);
end  xor2;
architecture behave  of xor2  is
begin
        O <=I1 xor I2;
end behave;

-- TWO INPUT AND ENTITY
entity and2 is port (
        I1,I2: in bit; O: out bit);
end  and2;
architecture behave  of and2  is
begin
        O <=I1 and I2;
end behave;
```

Structural Description of Full Adder (3)

```
--THREE INPUT OR ENTITY
Entity or3 is port (
        I1,I2, I3: in bit; O: out bit);
end  or3;

architecture behave  of or3  is
begin
        O <=I1 or I2 or I3;
end behave;
```

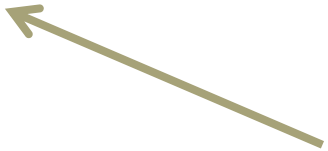
TEST BENCH

- Επαλήθευση της σχεδίασης με προσομοίωση.
- Ένα testbench είναι ένα VHDL μοντέλο χωρίς εισόδους/εξόδους που περιέχει ένα στιγμιότυπο της μονάδας υπό δοκιμή.
- Ορίζουμε ακολουθίες τιμών δοκιμής στις εισόδους της μονάδας
- Παρακολουθούμε τις τιμές στις εξόδους της μονάδας
- Σε ένα testbench είναι ενδεδειγμένη η χρήση διεργασιών για την διαδοχική αλλαγή των τιμών δοκιμής στις εισόδους.

Test Bench of Full Adder

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY Testbench_full_adder IS
END Testbench_full_adder;

ARCHITECTURE behavior OF Testbench_full_adder IS

 -- Component Declaration for the Unit Under Test (UUT)

 COMPONENT full_adder PORT(
            A : IN bit;
            B : IN bit;
            Cin : IN bit;
            Sum : OUT bit;
            Cout : OUT  bit
 );
 END COMPONENT;
```

Δήλωση οντότητας χωρίς εισόδους - εξόδους



Δήλωση στιγμιότυπου της μονάδας υπό δοκιμή

Test Bench of Full Adder

```
--Inputs
 signal SA : bit := '0';
 signal SB : bit := '0';
 signal SCin : bit := '0';
 --Outputs
 signal SSum : bit;
 signal SCout : bit;

BEGIN
 -- Instantiate the Unit Under Test (UUT)
 uut: full_adder PORT MAP (
 A => SA,
 B => SB,
 Cin => SCin,
 Sum => SSum,
 Cout => SCout
 );
```

Δήλωση σημάτων στα οποία δίνουμε τις δοκιμαστικές τιμές

Σύνδεση των σημάτων με τις εισόδους και εξόδους της μονάδας.

Test Bench of Full Adder (2)

```
-- Stimulus process
stim_proc: process
begin
-- hold reset state for 100 ns.
wait for 100 ns;

-- insert stimulus here
SA <= '1';
SB <= '0';
SCin <= '0';
wait for 10 ns;

SA <= '0';
SB <= '1';
SCin <= '0';
wait for 10 ns;

SA <= '1';
SB <= '1';
SCin <= '0';
wait for 10 ns;
```

Δημιουργία μιας διαδικασίας για την διαδοχική αλλαγή των τιμών δοκιμής

Test Bench of Full Adder (3)

```
SA <= '0';
SB <= '0';
SCin <= '1';
wait for 10 ns;

SA <= '1';
SB <= '0';
SCin <= '1';
wait for 10 ns;

SA <= '0';
SB <= '1';
SCin <= '1';
wait for 10 ns;

SA <= '1';
SB <= '1';
sCin <= '1';
wait for 10 ns;

end process;

END;
```

Full adder test bench waveform