ΣΧΕΔΙΑΣΗ ΚΥΚΛΩΜΑΤΩΝ ΜΕ VHDL

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 4 ψηφίων

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

entity adder_4  is
      port (A, B  :  in bit_vector (3  downto 0);
            Cin :    in bit;
            S    :    out   bit _vector (3 downto 0);
            Cout : out  bit);
end adder_4;

architecture  behave of adder_4 is
    Begin
        Εδώ περιγράφουμε την λειτουργία της οντότητας
    end;
```

Χρησιμοποιούμε vectors για να ομαδοποιήσουμε bits σε μια ενιαία δομή.

Μπορούμε να ορισουμε την αρχική και τελική τιμή και την κατεύθυνση αρίθμησης των στοιχείων σε ένα πίνακα.

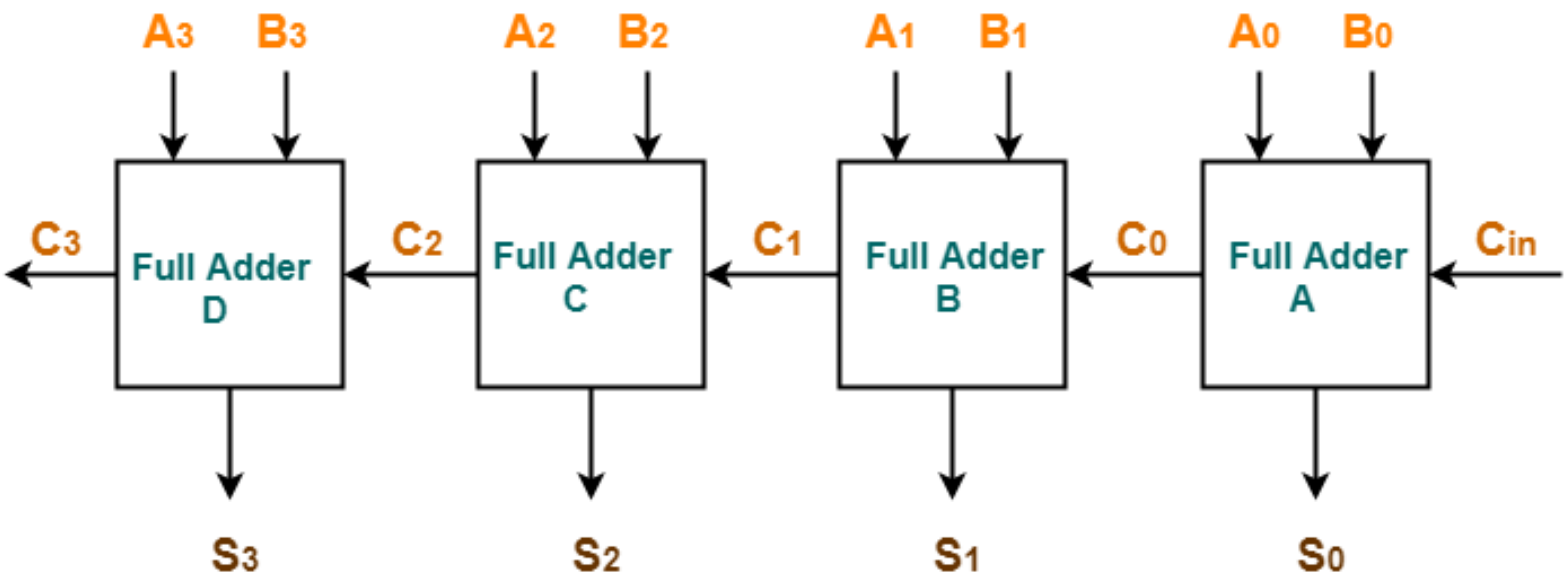
```
ARCHITECTURE circuit OF adder_4 IS

SIGNAL C: BIT_VECTOR (4 DOWNTO 0);

BEGIN
   C(0)<=Cin;

   S <=A XOR B XOR C(3 DOWNTO 0);

   C(4 DOWNTO 1)<=(A AND B) OR ((A XOR B) AND C(3 DOWNTO 0));

   Cout<=C(4);

END circuit;
```

Signals είναι σήματα εσωτερικά στην οντότητα και αντιστοιχούν σε συνδέσεις μεταξύ αντικειμένων.

Οι λογικές πράξεις αναφέρονται σε όλα τα ψηφία ενός vector
A3 XOR B3, A2 XOR B2 ….

Μπορούμε να χρησιμοποιήσουμε μερος ενός διανύσματος.

Διάγραμμα αθροιστή 4 ψηφιών



4-bit Ripple Carry Adder

Εναλλακτική αρχιτεκτονική adder_4

```
ARCHITECTURE version3 OF adder_4c IS
SIGNAL C: BIT_VECTOR (0 TO 4);
COMPONENT full_adder
   PORT        (A, B, Cin  :IN BIT;
                Sum , Cout :OUT BIT);
END COMPONENT;
   BEGIN
      C(0)<=Cin;
      repetition:
      FOR i IN 0 TO 3 GENERATE
      bit: full_adder PORT MAP (A(i), B(i), C(i), S(i), C(i+1));
      END GENERATE;
      Cout<=C(4);
END version3;
```

Χρήση επανάληψης για την δημιουργία τεσσάρων στιγμιότυπων του full_adder

Γενικευμένη Οντότητα adder n ψηφίων

```
entity adder_4  is
    generic (n: integer :=4);

    port (A, B  :  in bit_vector (n-1 downto 0);
          Cin :    in bit;
          S   :    out   bit _vector (n-1  downto 0);
          Cout : out  bit);
end adder_4;
```

Η δήλωση generic χρησιμοποιείται για τον ορισμό γενικευμένης οντότητας με μεταβλητό αριθμό ψηφίων.

Γενικευμένη Οντότητα adder n ψηφίων

```
ARCHITECTURE version4 OF adder_4 IS
SIGNAL C: BIT_VECTOR (0 TO N);
COMPONENT full_adder
   PORT          (A, B, Cin  :IN BIT;
                  Sum , Cout :OUT BIT);
END COMPONENT;
   BEGIN
        C(0)<=Cin;
        repetition:
        FOR i IN 0 TO N GENERATE
        bit: full_adder PORT MAP (A(i), B(i), C(i), S(i), C(i+1));
        END GENERATE;
        Cout<=C(N);
END version3;
```

Χρήση επανάληψης N φορές για την δημιουργία N στιγμιότυπων του full_adder

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_adder_4 IS
END Testbench_adder_4;

ARCHITECTURE behavior OF Testbench_adder_4 IS

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

 COMPONENT adder_4  PORT(
            A      : IN bit_vector (3 DOWNTO  0);
            B      : IN bit _vector (3 DOWNTO  0);
            Cin    : IN bit;
            S      : OUT bit _vector (3 DOWNTO  0);
            Cout : OUT  bit
 );
 END COMPONENT;
```

Δήλωση οντότητας χωρίς εισόδους - εξόδους

Δήλωση στιγμιότυπου της μονάδας υπό δοκιμή

Test Bench of Full Adder

```
--Inputs
 signal A : bit_vector (3 downto 0) := “0000”;
 signal B : bit _vector (3 downto 0) := “0000”';
 signal Cin : bit := '0';
 --Outputs
 signal Sum : bit_vector (3 downto 0) :=“0000”;
 signal Cout : bit;

BEGIN
 -- Instantiate the Unit Under Test (UUT)
 uut: adder_4  PORT MAP (
 A => A,
 B => B,
 Cin => Cin,
 S => Sum,
 Cout => Cout
 );
```

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
A <= “0011”;
B <=  “0101”;
Cin <= '0';
wait for 10 ns;

A <= “1000”;
B <= “1110”;
Cin <= '0';
wait for 10 ns;


end process;

END;
```

Δημιουργία μιας διαδικασίας για την διαδοχική αλλαγή των τιμών δοκιμής

Α και Β είναι vectors οπότε οι τιμές δινονται ως strings (“XXXX”)