Concurrent VHDL

- Concurrency
- Signal
- Inertial και Transport delays
- Παράδειγμα σε Inertial και Transport delays
- Προσομοίωση παραδείγματος σε Inertial και Transport delays
- Καθυστέρηση Delta
- Η εντολή When-Else
- Η εντολή With-Select
- Objects, class και type
- Data types
- Operations
- std_ulogic και std_logic type
- Αρχικές τιμές

Concurrent VHDL

- Παράδειγμα 4-to-1 Multiplexer
  - 4-to-1 Multiplexer με When-Else
  - 4-to-1 Multiplexer με With-Select
  - προσομοίωση παραδείγματος 4-to-1 Multiplexer
- Παράδειγμα 4-bit Ripple Carry Adder
  - προσομοίωση 4-bit Ripple Carry Adder
- Ασκήσεις - Προβλήματα

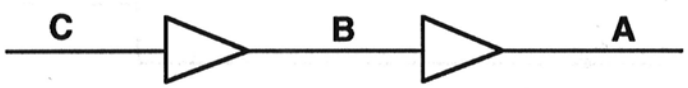
Concurrency

- Το Hardware από την φύση του είναι παράλληλο
  - η VHDL εκτελείται με ανάλογο τρόπο ο οποίος ονομάζεται **concurrent** (συντρέχων - παράλληλος)
  - αυτό επιτρέπει την όποια σειρά εκτέλεσης των εντολών στον κώδικα

- Τα ακόλουθα παραδείγματα είναι ισοδύναμα:

```
architecture example1 of ex is
begin
 a<=b;
 b<= c;
end example1;
```

```
architecture example2 of ex is
begin
  b<=c;
  a<= b;
end example2;
```



Signal

- Τα **signals** αποτελούν αντικείμενα τα οποία είναι **concurrent**
  - κατάλληλα για την περιγραφή του hardware
  - αντίθετα, οι **variables** αποτελούν **sequential** αντικείμενα

Syntax:

Signal assignment:
<target_identifier> '<=' <selected_expression>';'

- Παραδείγματα
  - **signal** a, b, c: std_logic;
  - c <= a **nor** b;

Inertial και Transport delays

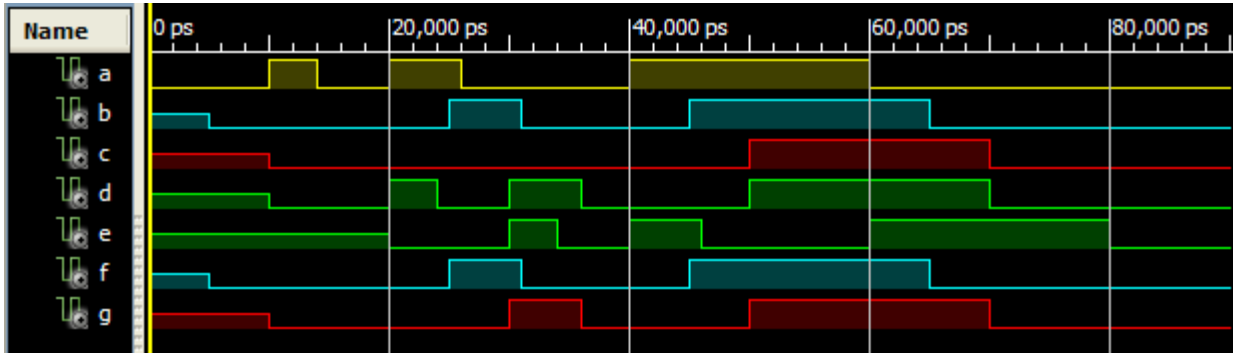
- Inertial delay
  - αποτελεί την κοινή καθυστέρηση στη VHDL
  - παλμοί εύρους μικρότερου του Inertial delay δεν διαδίδονται
  - επέκταση: *Reject Inertial Delay Model*
    παλμοί εύρους μικρότερου του Reject delay δεν διαδίδονται
- Transport delay
  - οι παλμοί διαδίδονται ανεξαρτήτως εύρους
- Παραδείγματα
  - b1 < = a **after** 5ns;
  - b1 < = **inertial** a **after** 5ns;
  - b1 < = a **after** 10ns, b **after** 20ns;
  - b1 < = **transport** a **after** 10ns;
  - b1 < = **reject** 4ns **inertial** a **after** 10ns;
- Χρησιμοποιούνται μόνο για προσομοιώσεις

Παράδειγμα σε Inertial και Transport delays

```
--  File: c:\my designs\delay_test1\SRC\delay_test1.VHD
--  created by Design Wizard: 01/25/01 19:54:57
library IEEE;
use IEEE.std_logic_1164.all;
entity delay_test1 is
    port (A : in STD_LOGIC;
            B, C, D, E, F, G: out STD_LOGIC);
end delay_test1;
architecture delay_test1_comp of delay_test1 is
begin
   B <= inertial A after 5ns;
   C <= inertial A after 10ns;
   D <= transport A after 10ns;
   E <= transport A after 20ns;
   F <= reject 4ns inertial A after 5ns;
   G <= reject 4ns inertial A after 10ns;
end delay_test1_comp;
```

Προσομοίωση παραδείγματος σε Inertial και Transport delays

Για το σήμα εισόδου Α χρησιμοποιήσαμε την ακολουθία:
L**10**ns H**4**ns L**6**ns H**6**ns L**14**ns H**20**ns L**40**ns



Καθυστέρηση Delta

- Delta delay: χρόνος ανάμεσα σε δύο ακολουθιακά (sequential) γεγονότα.
- δεν είναι πραγματικός χρόνος
- εκτελείται ενώ το ρολόι του προσομοιωτή είναι ακίνητο.

- Στον καθορισμό ενός σήματος αυτό παίρνει την τιμή του μετά από ένα χρονικό διάστημα ίσο με Delta delay

```
B <= a;  -- Signal B assign value of signal a after one delta delay.
```

- Η παρακάτω γραμμή θα προκαλέσει άπειρη ταλάντωση κατά την προσομοίωση και πρέπει να αποφεύγεται

```
a <=  not a;
```

Η εντολή When-Else

Syntax:

<target> <= <expression> [**after** <expression>] **when** <condition>
<expression> [**after** <expression>] .... ;

- Παράδειγμα

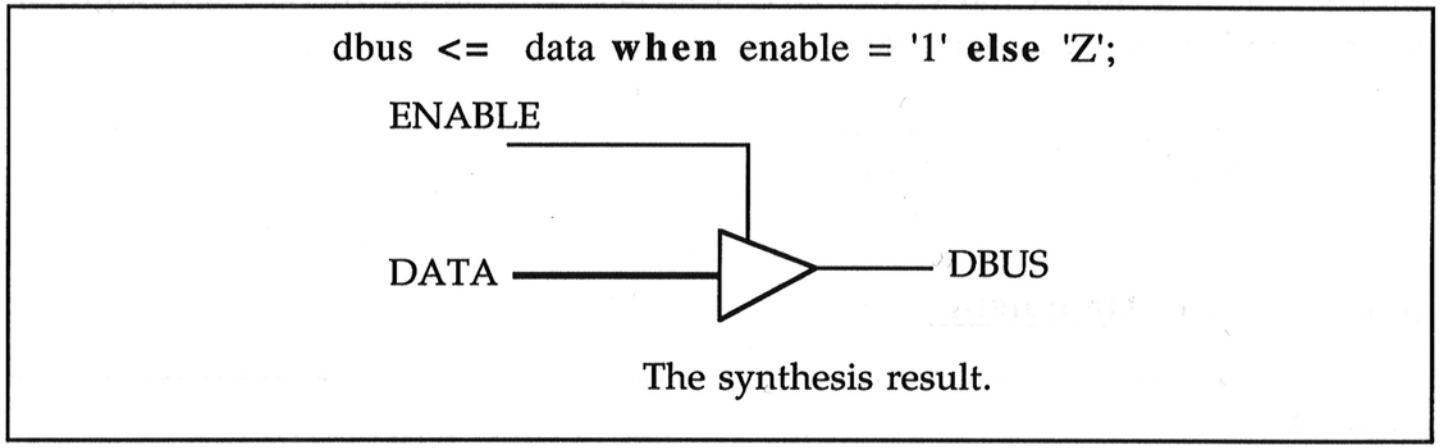
```
entity example is
port (a, b, c : in std_logic;
      data    : in std_logic_vector(1 downto 0);
      q       : out std_logic);
end example;
architecture rtl of example is
begin
  q <=  a when data="00" else
        b when data="11" else
        c;
end rtl;
```

Η εντολή When-Else

- Παράδειγμα VHDL-93

```
architecture rtl of example is
begin
  q <=  a when data="00" else
        b when data="11"       -- No else for VHDL-93
end rtl;
```

- Three state buffer



The synthesis result.

Η εντολή With-Select

Syntax:

with <expression> **select**
<target> **<=** <expression> **when** <chose> **;**

- Παράδειγμα

```
entity example is
port (a, b, c : in std_logic;
        data   : in std_logic_vector(1 downto 0);
        q      : out std_logic);
end example;
architecture rtl of example is
begin
  with data select
    q <=  a when “00”,
          b when “11”,
          c when others;
end rtl;
```

Objects, class και type

- Ένα **object** περιέχει μία τιμή ενός καθορισμένου τύπου (type)

Class	Object	Data type
Signal	a:	std_logic

- **Class**
  - **Constant**: δεν αλλάζει τιμή. Μπορεί να ορισθεί παντού και μπορεί να είναι οποιουδήποτε τύπου.
    - Παράδειγμα **constant** a: a_type := “1001”;
  - **Variable**: αλλάζει τιμές. Μπορεί να ορισθεί μέσα σε process και subprograms και μπορεί να είναι οποιουδήποτε τύπου.
    - Παράδειγμα a := John;
  - **Signal**: αλλάζει τιμές σε σχέση με τον χρόνο.
    - Παράδειγμα a <= b_in;

Data types

- **Boolean type**
  - Μπορεί να πάρει τις τιμές true και false.
    Constants, variables και signals μπορούν να ορισθούν ως Boolean.
  - Παράδειγμα:

```
Architecture rtl of example is
begin
   process(….)
   variable check: boolean;
   begin
      check := a < b;
      if check then ……
      …..
    end process;
end rtl;
```

- Κατά την σύνθεση το true γίνεται “1” και το false “0”.

Data types

- **Integer type**
  - Ακέραιος του οποίο το μήκος σε bits εξαρτάται από το εργαλείο που χρησιμοποιούμε. Συνήθως είναι 32 bits και άρα μπορεί να κυμανθεί από

    -2147483648 έως 2147483648.
  - Παραδείγματα :

```
constant loop_number: integer:=354;
signal my_int: integer range 0 to 255;
```

- **Bit type**
  - Μπορεί να πάρει τις τιμές ‘0’ ή ‘1’
  - Παράδειγμα:

```
signal my_sig: bit;
my_sig <=  '0';
```

Data types

- **Character type (VHDL-87)**

```
type character is (
    NUL, SOH, STX, ETX, EOT, ENQ, ACK, BEL,
     BS,  HT,  LF,  VT,  FF,  CR,  SO,  SI,
    DLE, DC1, DC2, DC3, DC4, NAK, SYN, ETB,
    CAN,  EM, SUB, ESC, FSP, GSP, RSP, USP,
     ' ',  '!',  '"',  '#',  '$',  '%',  '&',  ''',
     '(',  ')',  '*',  '+',  ',',  '-',  '.',  '/',
     '0',  '1',  '2',  '3',  '4',  '5',  '6',  '7',
     '8',  '9',  ':',  ';',  '<',  '=',  '>',  '?',
     '@',  'A',  'B',  'C',  'D',  'E',  'F',  'G',
     'H',  'I',  'J',  'K',  'L',  'M',  'N',  'O',
     'P',  'Q',  'R',  'S',  'T',  'U',  'V',  'W',
     'X',  'Y',  'Z',  '[',  '\',  ']',  '^',  '_',
     '`',  'a',  'b',  'c',  'd',  'e',  'f',  'g',
     'h',  'i',  'j',  'k',  'l',  'm',  'n',  'o',
     'p',  'q',  'r',  's',  't',  'u',  'v',  'w',
     'x',  'y',  'z',  '{',  '|',  '}',  '~',  DEL);
```

Data types

- **Vectors**
  - Παραδείγματα

```
entity example is
  port ( a : in std_logic_vector (3 downto 0);
         b : in std_logic_vector(0 to 3);
         c : out bit_vector (0 to 2);
end example;

architecture rtl of example is
signal int1: std_logic_vector (5 downto 0);
signal int2: std_logic_vector(0 to 2);
begin
   int1 <= “100100”
   int2 <= “110”
……..
end rtl;
```

Data types

- Για τα bit vectors ισχύουν τα ακόλουθα:

  Binary B”10001”

  Octal O”427”

  Hex X”FAE6”

```
a_vect<=B"1100_0011_0011_1100";--     Good
a_vect<="1100001100111100";    --     Identical to first
                               --     assignment
a_vect<=X"C33C";               --     Identical to first
                               --     assignment
a_vect<=X"C3_3C";              --     Identical to first
                               --     assignment
a_vect<="1100_0011_0011_1100"; --     Error, B missing
a_vect<="C33C";                --     Error, X missing
```

Data types

- Για τον ορισμό μεγάλων vectors και όταν θέλουμε να ορίσουμε πολλά στοιχεία του ανύσματος στην ίδια τιμή χρησιμοποιούμε το others
  - Παραδείγματα
    - **signal** a: std_logic_vector(31 **downto** 0);
      a <= “00000000000000000000000000000000”;
      a<=(**others**=>’0’);
    - **signal** b: std_logic_vector(5 **downto** 0);
      b <= “010010”;
      b <= (1=>’1’, 4=>’1’, **others**=>’0’);
- Concatenation (αλληλουχία) μέσω του τελεστή **&**
  - **signal** a: std_logic_vector(5 **downto** 0);
    signal b,c,d: std_logic_vector(2 **downto** 0);
    b <= ‘0’ & c(1) & d(2);
    a <= c & d;
  - Εάν c<=“011”, d<=“101” τότε το b είναι “011” και το a είναι “011101”.

Data types

- **Array type**
  - Μίας διάστασης

    type std_logic_vector **is array** (natural **range** <>) **of** std_logic;

    type my_array **is array** (0 **to** 5) **of** std_logic;
    type that_array **is array** (3 **downto** 0) **of** std_logic;

  - Πολλών διαστάσεων

    type data4x8 **is array** (0 **to** 3) **of** std_logic_vector (7 **downto** 0);
    type data3x4x8 **is array** (0 **to** 2) **of** data4x8;

Data types

- **Type declaration**
  - Είναι δυνατόν να δημιουργήσουμε τους δικούς μας τύπους ως
    type <identifier> **is** <type_definition>;
  - Παραδείγματα
    type A_type **is array** (3 **downto** 0) **of** std_logic;
    type my_int **is** integer **range** 0 **to** 63;
    signal int1 : A_type;
    signal int2 : my_int;
- **Time**
  - **constant** delay : time := 0ns;
    a <= b **after** delay;

Data types

- **Enumerated type**
  - **Type** state_type **is** (start, idle, waiting, run);
    signal state : state_type;

- **Record type**
  - Ένα record type μπορεί να περιέχει στοιχεία διαφορετικών τύπων

```
type <identifier> is record
      record definition
end record;
```

Data types

```
Architecture beh of ex is
type data_date is record
   year:      integer range 1996 to 2099;
   month:     integer range 1 to 12;
   date:      integer range 1 to 31;
   hour:      integer range 0 to 23;
   minute:    integer range 0 to 59;
   second:    integer range 0 to 59;
   data:      std_logic_vector(31 downto 0);
end record;

signal d:data_date;
begin
   d.year<=1997;
   d.month<=4;
   d.date<=8;
   d.hour<=11;
   d.minute<=57;
   d.second<=22;
   d.data<=data_in;
end;
```

Data types

- **Subtypes**
  - Χρησιμοποιείται για υποσύνολα
    - **type** counter **is** integer **range** 0 **to** 100;
    - **subtype** low_range **is** counter **range** 0 **to** 50;
- **Predefined Attributes**

  a **named entity** followed by an **apostrophe** and an **attribute name**.

  if clock'event then …… --sequential VHDL

  - S'**DELAYED(t)** is the signal value of S at time now - t .
  - S'**STABLE** is true if no event is occurring on signal S.
  - S'**STABLE(t)** is true if no even has occurred on signal S for t units of time.
  - S'**QUIET** is true if signal S is quiet. (no event this simulation cycle)
  - S'**QUIET(t)** is true if signal S has been quiet for t units of time.
  - S'**TRANSACTION** is a bit signal, the inverse of previous value each cycle S is active.
  - S'**EVENT** is true if signal S has had an event this simulation cycle.
  - S'**ACTIVE** is true if signal S is active during current simulation cycle.
  - S'**LAST_EVENT** is the time since the last event on signal S.
  - S'**LAST_ACTIVE** is the time since signal S was last active.
  - S'**LAST_VALUE** is the previous value of signal S.
  - S'**DRIVING** is false only if the current driver of S is a null transaction.
  - S'**DRIVING_VALUE** is the current driving value of signal S.

Operations

Relational Operators

Symbol	Operand
=	equal
/=	not equal
<	less than
>	greater than
<=	less than or equal
>=	greater than or equal

Arithmetic Operators

Symbol	Operator
+	addition
-	subtraction
*	multiplication
/	division
abs	absolute value
rem	remainder
mod	modulus
**	exponentiation

std_ulogic και std_logic type

- std_ulogic type

Std_ulogic is a type which is declared in the ieee package std_logic_1164. Std_ulogic can assume the following values:

1	'U'	-- Uninitialized
2	'X'	-- Forcing Unknown
3	'0'	-- Forcing 0
4	'1'	-- Forcing 1
5	'Z'	-- High Impedance
6	'W'	-- Weak Unknown
7	'L'	-- Weak 0
8	'H'	-- Weak 1
9	'-'	-- Don't care

- std_logic type

Std_ulogic is a type which is declared in the ieee package std_logic_1164. Std_ulogic can assume the same values as std_logic. The difference is that std_logic is defined as:

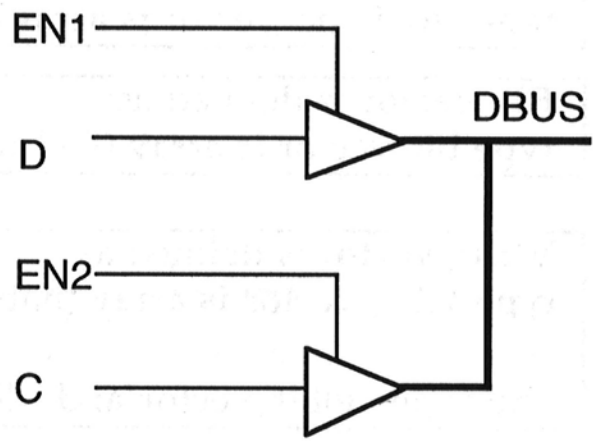
```
subtype std_logic is resolved std_ ulogic;
```

std_ulogic και std_logic type

- Η διαφορά ανάμεσα στα std_ulogic και std_logic είναι εμφανής σε σχέδια όπου ένα σήμα οδηγείται από περισσότερους του ενός drivers.
  - **std_ulogic**: το παραπάνω οδηγεί σε πρόβλημα και σε λάθος στο compilation
  - **std_logic**: όταν ένα σήμα οδηγείται από περισσότερους του ενός drivers, εκτελείται μία προκαθορισμένη συνάρτηση (resolution function) και αποφεύγεται το πρόβλημα.
    Για το λόγο αυτό, το std_logic χρησιμοποιείται σε three-state buses

std_ulogic και std_logic type

```
Entity ex is
  port( d,c,en1,en2:  in  std_logic;
        dbus:         out std_logic);
end;
Architecture rtl of ex is
begin
    dbus<= d when enable1='1' else 'Z';
    dbus<= c when enable2='1' else 'Z';
end ;
```




```
Entity ex is
  port( d,c,en1,en2:  in  std_ulogic;
        dbus:         out std_ulogic);
end;
Architecture rtl of ex is
begin
    dbus<= d when enable='1' else 'Z';
    dbus<= c when enable2='1' else 'Z';
end ;
```

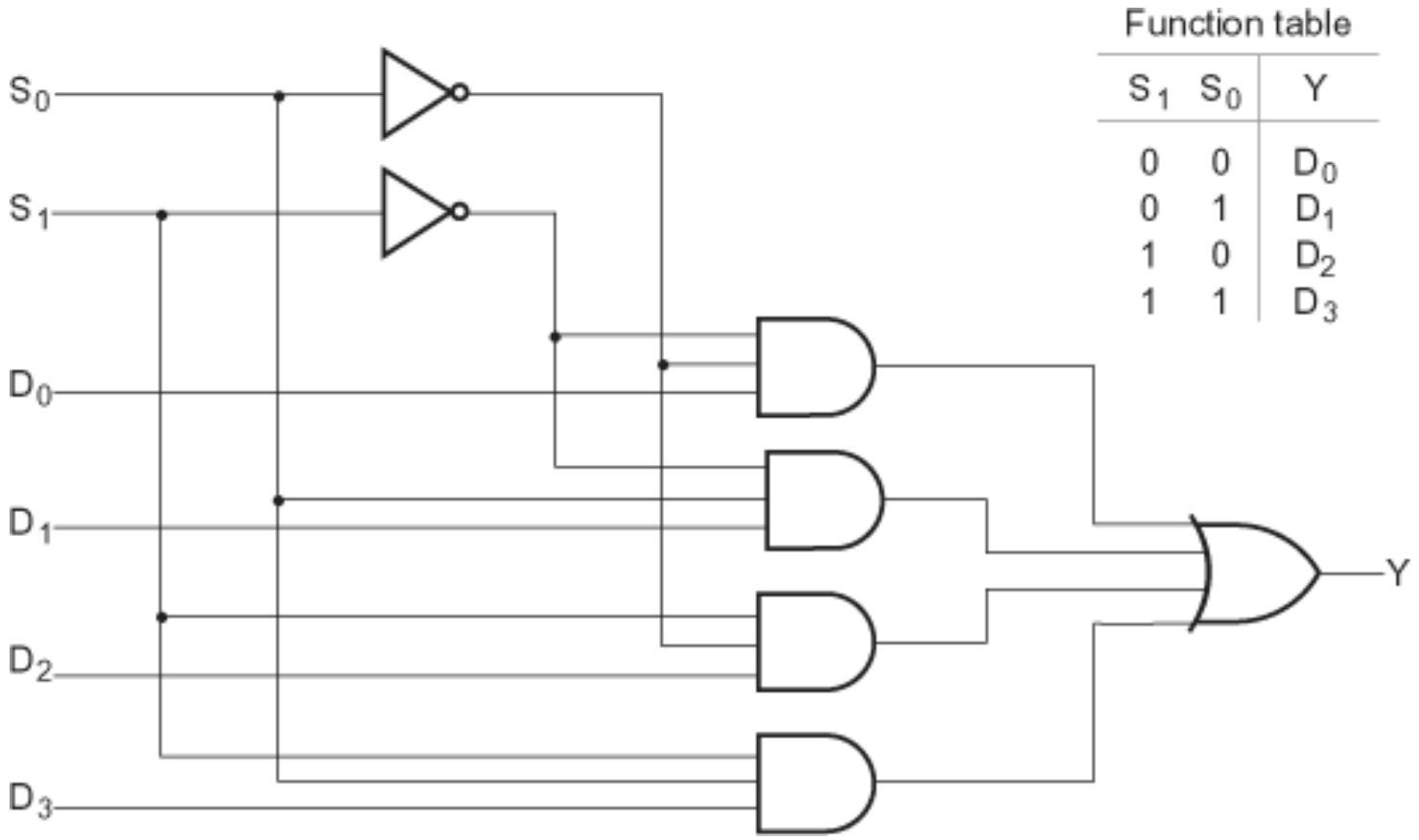
→ ERROR

Αρχικές τιμές

- Στο χρόνο μηδέν κατά την προσομοίωση όλα τα σήματα παίρνουν τις αρχικές τους τιμές. Η τιμή αυτή εξαρτάται από τον τύπο του σήματος και ισούται με την τιμή datatype'left.
  - **type** bit **is** ('0', '1');
  - **type** std_logic **is** ('U', 'X', '0', '1', 'Z', 'W', 'L', 'H', '-');
    - Bit'left='0'
    - Std_logic 'left='U'

- Μπορούμε να αλλάξουμε τις αρχικές τιμές ως εξής:
  - **signal** int1: std_logic:='1';
  - **signal** int2: std_logic:='Z';
  - **signal** my_vect: std_logic_vector(3 **downto** 0):="00LL";

Παράδειγμα 4-to-1 Multiplexer



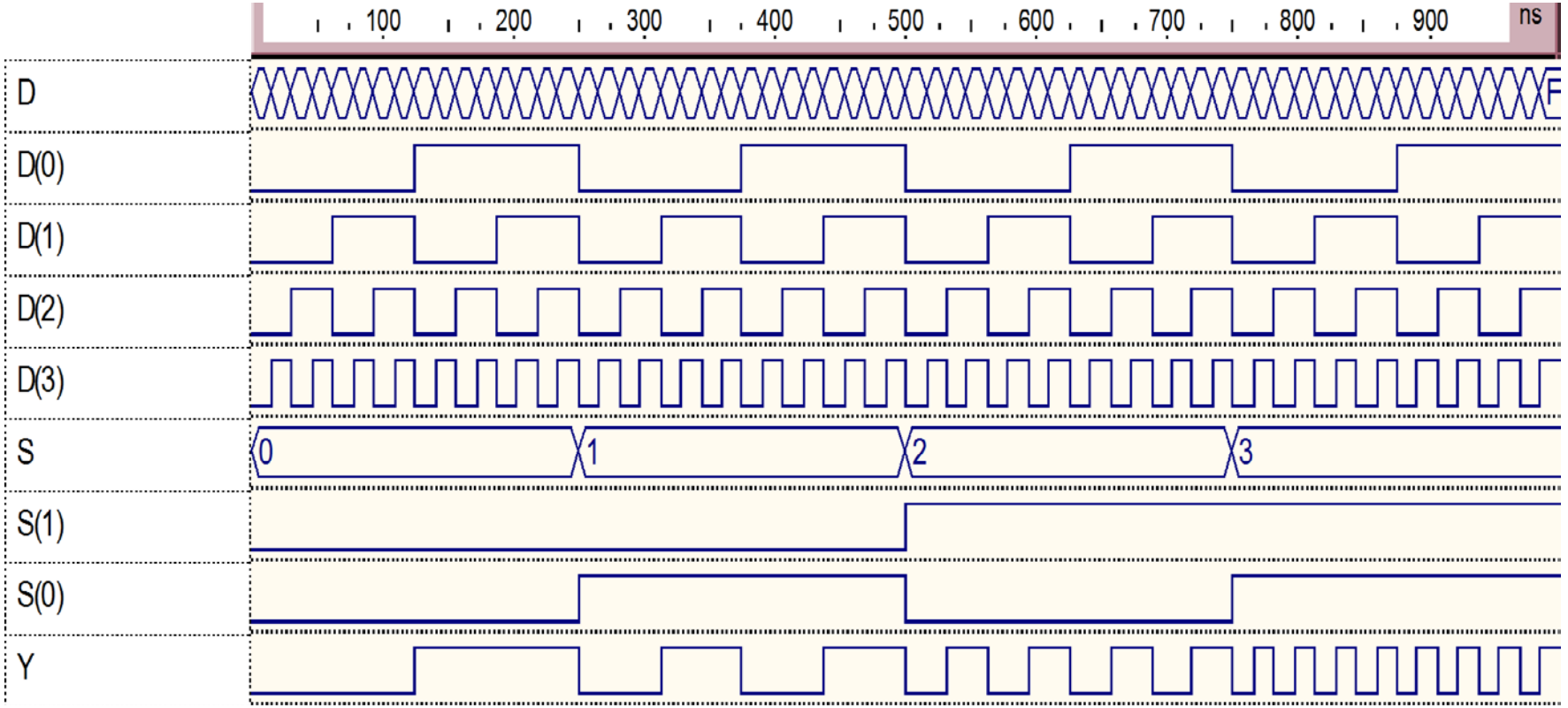
4-to-1 Multiplexer με When-Else

```
--  File: c:\my designs\mux_4_to_1_we\SRC\mux_4_to_1_we.VHD
--  Using When-Else. Created by Design Wizard: 01/26/01 17:20:27
library IEEE;
use IEEE.std_logic_1164.all;

entity mux_4_to_1_we is
    port ( S: in STD_LOGIC_VECTOR (1 downto 0);
           D: in STD_LOGIC_VECTOR (0 to 3);
           Y: out STD_LOGIC );
end mux_4_to_1_we;

architecture function_table of mux_4_to_1_we is
begin
 Y <=  D(0) when S = "00" else
           D(1) when S = "01" else
           D(2) when S = "10" else
           D(3) when S = "11" ;
end function_table;
```

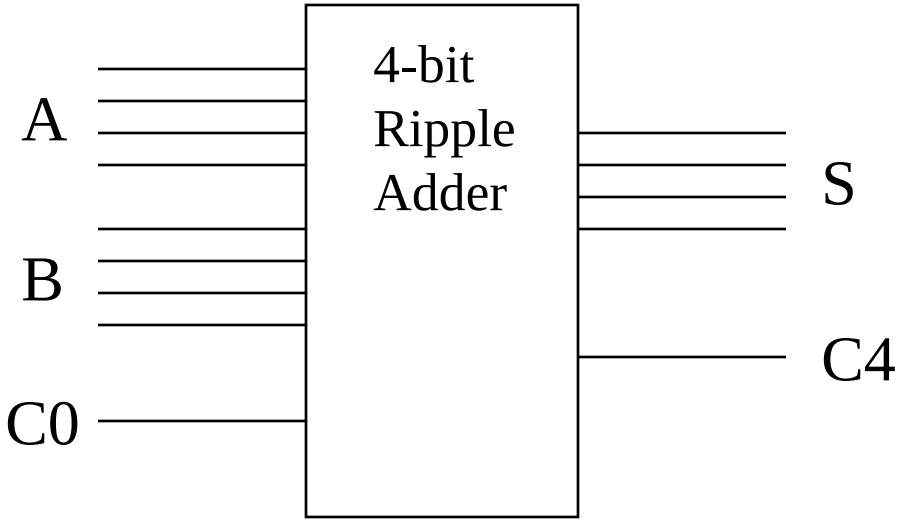
4-to-1 Multiplexer με With-Select

```
--  File: c:\my designs\mux_4_to_1_we\SRC\mux_4_to_1_we.VHD
--  Using With-Select. Created by Design Wizard: 01/26/01 17:20:27
library IEEE;
use IEEE.std_logic_1164.all;

entity mux_4_to_1_we is
   port ( S: in STD_LOGIC_VECTOR (1 downto 0);
          D: in STD_LOGIC_VECTOR (0 to 3);
          Y: out STD_LOGIC );
end mux_4_to_1_we;

architecture function_table of mux_4_to_1_we is
begin
with S select
 Y <= D(0) when "00",
      D(1) when "01",
      D(2) when "10",
      D(3) when others ;
end function_table;
```

Προσομοίωση παραδείγματος 4-to-1 Multiplexer



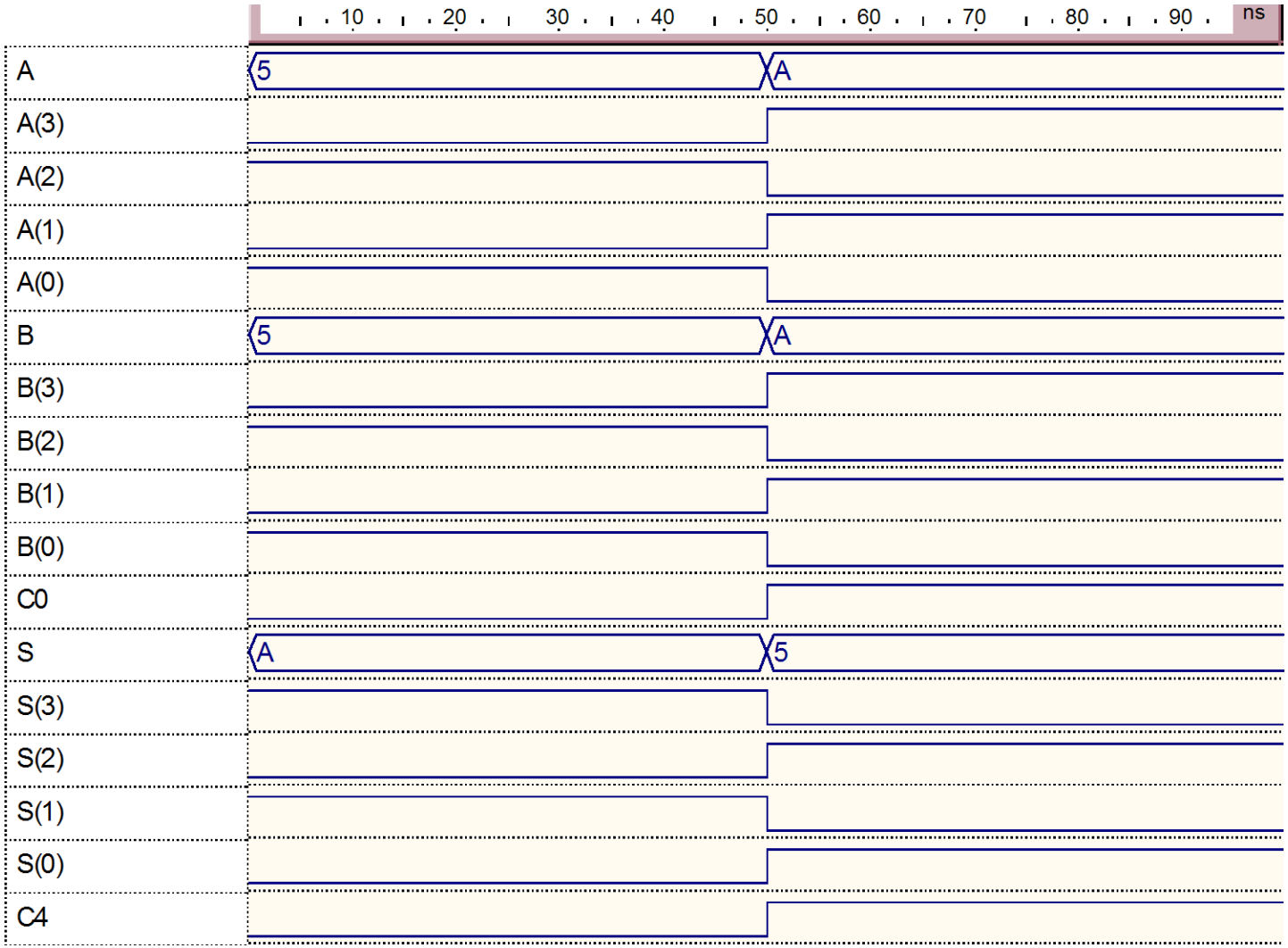
Παράδειγμα 4-bit Ripple Carry Adder



Παράδειγμα 4-bit Ripple Carry Adder

```
--  File: c:\my designs\adder_4_b_beh\SRC\adder_4_b_beh.VHD
--  Behavioral Description. Created by Design Wizard: 01/29/01 20:15:30
--
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity adder_4_b_beh is
   port ( A, B : in STD_LOGIC_VECTOR (3 downto 0);
          C0    : in STD_LOGIC;
          S      : out STD_LOGIC_VECTOR (3 downto 0);
          C4    : out STD_LOGIC);
end adder_4_b_beh;
architecture adder_beh of adder_4_b_beh is
signal sum : STD_LOGIC_VECTOR (4 downto 0);
begin
   sum <= ('0' & A) + ('0' & B) + ("0000" & C0);
   C4   <= sum(4);
   S     <= sum(3 downto 0);
end adder_beh;
```

Προσομοίωση 4-bit Ripple Carry Adder



Ασκήσεις - Προβλήματα

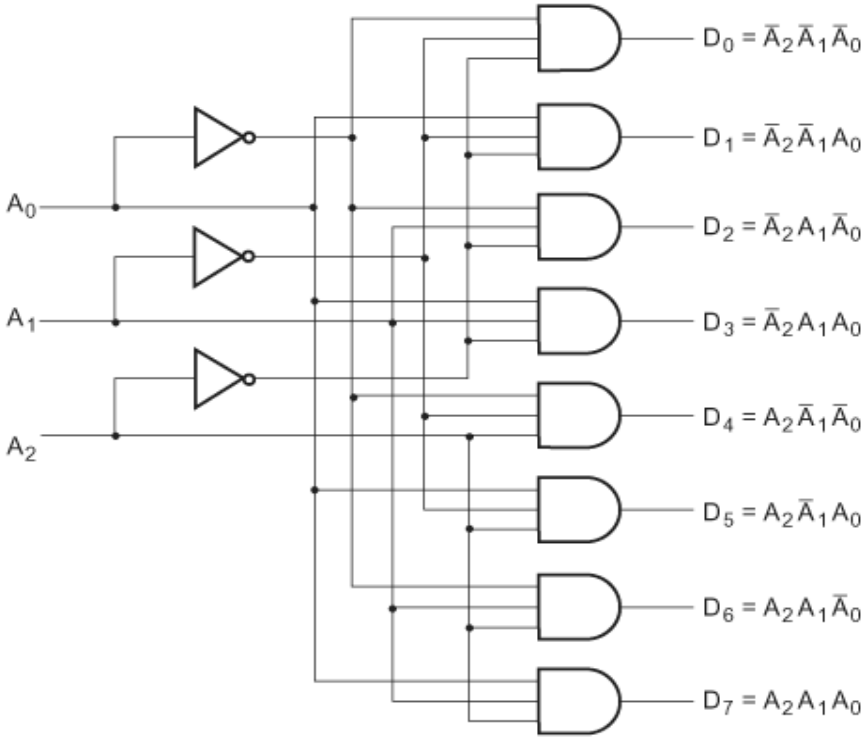
1. Περιγράψτε σε VHDL τον 4-to-1 Multiplexer σε επίπεδο πυλών. Ελέγξτε την λογική με προσομοίωση. Συγκρίνετε τον αριθμό σειρών του κώδικά σας με αυτόν στα λυμένα παραδείγματα.
2. Ο πίνακας αληθείας ενός 2-to-4 Line Decoder είναι ο ακόλουθος

A1	A0	D3	D2	D1	D0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Σχεδιάστε το κύκλωμα με πύλες, περιγράψτε το σε VHDL και ελέγξτε την λογική με προσομοίωση. Περιγράψτε ξανά το κύκλωμα σε VHDL χρησιμοποιώντας τις εντολές When-Else και With-Select.

Ασκήσεις - Προβλήματα

3. Περιγράψτε σε VHDL τον 3-to-8 Line Decoder χρησιμοποιώντας τις εντολές When-Else και With-Select. Ελέγξτε την λογική με προσομοίωση.



Ασκήσεις - Προβλήματα

4. Περιγράψτε σε VHDL και ελέγξτε την λογική με προσομοίωση για τα παρακάτω
   - Πολλαπλασιαστής 2 bits.
   - Πολλαπλασιαστής 4 bits.
   - Διαιρέτης 4 bits.