dscal
DIGITAL SYSTEMS & COMPUTER ARCHITECTURE LABORATORY

Εργαστήριο Λογικής Σχεδίασης

VHDL - Εισαγωγή

Βασιλόπουλος Διονύσης

Ε.ΔΙ.Π. Τμήματος Πληροφορικής & Τηλεπικοινωνιών - ΕΚΠΑ

ΤΜΗΜΑ
ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

VHDL – Signed

- Τύπος signed από το numeric_std

Απαραίτητη η χρήση της βιβλιοθήκης numeric_std

```
library ieee; use ieee.numeric_std.all;
...
s : signed(15 downto 0);
```

- Οι τύποι signed και unsigned είναι ξεχωριστοί

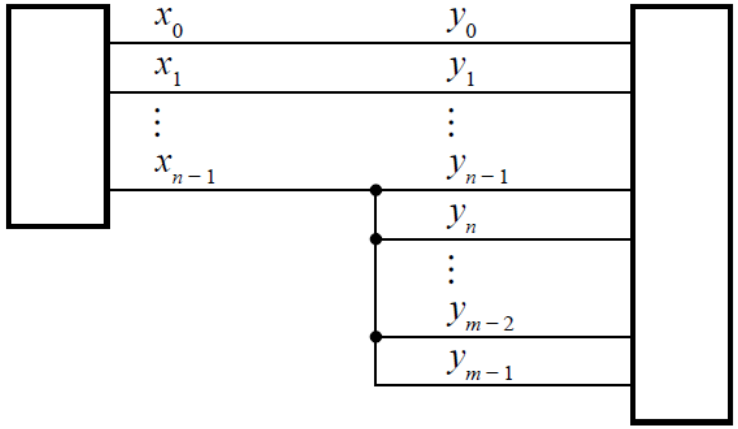
```
signal s1 : unsigned(11 downto 0);
signal s2 : signed(11 downto 0);
...
s1 <= s2; -- illegal
```

```
s1 <= unsigned(s2); -- s2 is known to be non-negative
...
s2 <= signed(s1); -- s1 is known to be less than 2**11
```

VHDL – Signed

Αλλαγή μεγέθους

- Γενικά, για ακεραίους σε συμπλήρωμα ως προς 2
  - Επέκταση με επανάληψη του bit προσήμου
    - *Επέκταση προσήμου*
  - Αποκοπή με απόρριψη αρχικών bit
    - Τα bit που απορρίπτονται πρέπει όλα να είναι τα ίδια, και ίδια με το bit προσήμου του αποτελέσματος




```
signal x : signed (7 downto 0);
signal y : signed (15 downto 0);
...
y <= resize(x, y'length);
...
x <= resize(y, x'length);
```

VHDL – Signed

Πρόσθεση

0 0 0 0 0 0 0 0	1 1 0 0 0 0 0 0	0 0 0 0 0 0 0 0
72: 0 1 0 0 1 0 0 0	–63: 1 1 0 0 0 0 0 1	–42: 1 1 0 1 0 1 1 0
49: 0 0 1 1 0 0 0 1	–32: 1 1 1 0 0 0 0 0	8: 0 0 0 0 1 0 0 0
121: 0 1 1 1 1 0 0 1	–95: 1 0 1 0 0 0 0 1	–34: 1 1 0 1 1 1 1 0
χωρίς υπερχείλιση	χωρίς υπερχείλιση	χωρίς υπερχείλιση
0 1 0 0 1 0 0 0	1 0 0 0 0 0 0 0	1 1 1 1 1 0 0 0
72: 0 1 0 0 1 0 0 0	–63: 1 1 0 0 0 0 0 1	42: 0 0 1 0 1 0 1 0
105: 0 1 1 0 1 0 0 1	–96: 1 0 1 0 0 0 0 0	–8: 1 1 1 1 1 0 0 0
1 0 1 1 0 0 0 1	0 1 1 0 0 0 0 1	34: 0 0 1 0 0 0 1 0
θετική υπερχείλιση	αρνητική υπερχείλιση	χωρίς υπερχείλιση

VHDL – Signed

Πρόσθεση

- Το αποτέλεσμα του + έχει ίδιο μέγεθος με τους τελεστέους

ισχύει και για unsigned

```
signal v1, v2 : signed(11 downto 0);
signal sum : signed(12 downto 0);
...
sum <= resize(v1, sum'length) + resize(v2,sum'length);
```

- Για έλεγχο υπερχείλισης, σύγκριση προσήμων

```
signal x, y, z: signed(7 downto 0);
signal ovf : std_logic;
...
z <= x + y;
ovf <= (not x(7) and not y(7) and z(7))
       or (x(7) and y(7) and not z(7));
```

VHDL – Τύποι σημάτων

Οι ακόλουθοι τύποι υποστηρίζονται εξ' ορισμού στην VHDL.

- bit : τιμές MONO '0' και '1' {0,1}
- Boolean: τιμές Αληθής/Ψευδής {true, false}
- Character: ASCII character (είναι 256 σε πλήθος}

Enumerated type: Συγκεκριμένες τιμές

- bit_vector: κατ' αναλογία με std_logic_vector
- integer: -2^{31} έως 2^{31} -1 (signed binary number range: $[-2^{n-1}, 2^{n-1}-1]$, 32bit word arch.)
  - Integer: range 0 to 1000
- natural: 0 έως 2^{31} -1
- positive: 1 έως 2^{31} -1

Subtypes: Υποτύποι του integer

Υπάρχουν και άλλα Type/Subtypes είτε στη standard βιβλιοθήκη είτε σε άλλες (ieee.numeric_std.all: signed/unsigned, ieee.std_logic_1164: std_logic, std_logic_vector)

VHDL – Τελεστές (operators)

Λογικοί Τελεστές

Operator	Operation
not	Logical negation
and	Logical AND
nand	Logical NAND
or	Logical OR
nor	Logical NOR
xor	Logical Exclusive-OR
xnor	Logical Exclusive-NOR

Εφαρμόζονται σε bit

VHDL – Τελεστές (operators)

Αριθμητικοί Τελεστές

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
mod	Modulus
rem	Remainder
abs	Absolute value
**	Exponential

Εφαρμόζονται σε σήματα τύπου integer/signed/unsigned

VHDL – Τελεστές (operators)

Τελεστές σύγκρισης (Relational Operators)

Operator	Returns true if the comparison is:
=	Equal
/=	Not equal
<	Less than
<=	Less than or equal
>	Greater than
>=	Greater than or equal

VHDL – Τελεστές (operators)

Τελεστές Ολίσθησης (Shift/Rotate Operators)

Operator	Operation
sll	Shift left logical
srl	Shift right logical
sla	Shift left arithmetic
sra	Shift right arithmetic
rol	Rotate left
ror	Rotate right

Εφαρμόζονται σε vector

VHDL – Τελεστές (operators)

Συναρτήσεις Ολίσθησης (Πακέτο numeric_std)

shift_left()
shift_right()
-

Αριθμητική Ολίσθηση

rotate_left()
rotate_right()

Εφαρμόζονται σε signed/unsigned

VHDL – Τελεστές (operators)

Unsigned: Ολίσθηση

- Λειτουργίες shift_left και shift_right
  - Αποτέλεσμα ίδιου μεγέθους με τον τελεστέο

Αποκοπή προς τα κάτω

$s = 00010011_2 = 19_{10}$

```
y <= shift_left(s, 2);
```

$y = 01001100_2 = 76_{10}$

$s = 00010011_2 = 19_{10}$

```
y <= shift_right(s, 2);
```

$y = 00000100_2 = 4_{10}$

VHDL – Τελεστές (operators)

VHDL'93 Vivado 2019.2, **2022.2**

Για το std_logic_vector
ΔΕΝ υπάρχει κάποια συνάρτηση.

Εάν όμως ενεργοποιήσουμε τη VHDL 2008 τότε μπορούμε να χρησιμοποιήσουμε τα sll, srl, rol, ror.

signed/unsigned

```
x<=a sll 2;
y<=a srl 2;
-----------------
r<=shift_left(a,2);
t<=shift_right(a,2);
-----------------
q<=a rol 2;
u<=a ror 2;
i<=rotate_left(a,2);
o<=rotate_right(a,2)
```

Εάν όμως ενεργοποιήσουμε τη VHDL 2008 τότε μπορούμε να χρησιμοποιήσουμε και τα sla, sra.

VHDL – Τελεστές (operators)

VHDL'93 Vivado 2022.1

std_logic_vector

```
x<=a sll 2;
y<=a srl 2;
-----------------
q<=a rol 2;
u<=a ror 2;
```

signed/unsigned

```
x<=a sll 2;
y<=a srl 2;
-----------------
z<=a sla 2;
w<=a sra 2;
r<=shift_left(a,2);
t<=shift_right(a,2);
-----------------
q<=a rol 2;
u<=a ror 2;
i<=rotate_left(a,2);
o<=rotate_right(a,2)
```

VHDL – Τελεστές (operators)

Signed: Πολλαπλασιασμός

- Πολλαπλασιασμός με 2^k
  - Αριστερή λογική ολίσθηση (όπως για απρόσημους)

Το πρόσημο πρέπει να παραμείνει

- Διαίρεση με 2^k
  - *Δεξιά αριθμητική ολίσθηση*
  - Απόρριψη των k λιγότερο σημαντικών bit, και εισαγωγή k αντιγράφων του bit προσήμου στο περισσότερο σημαντικό άκρο
  - π.χ., s = "11110011" -- –13
    shift_right(s, 2) = "11111100" -- –13 / 2^2

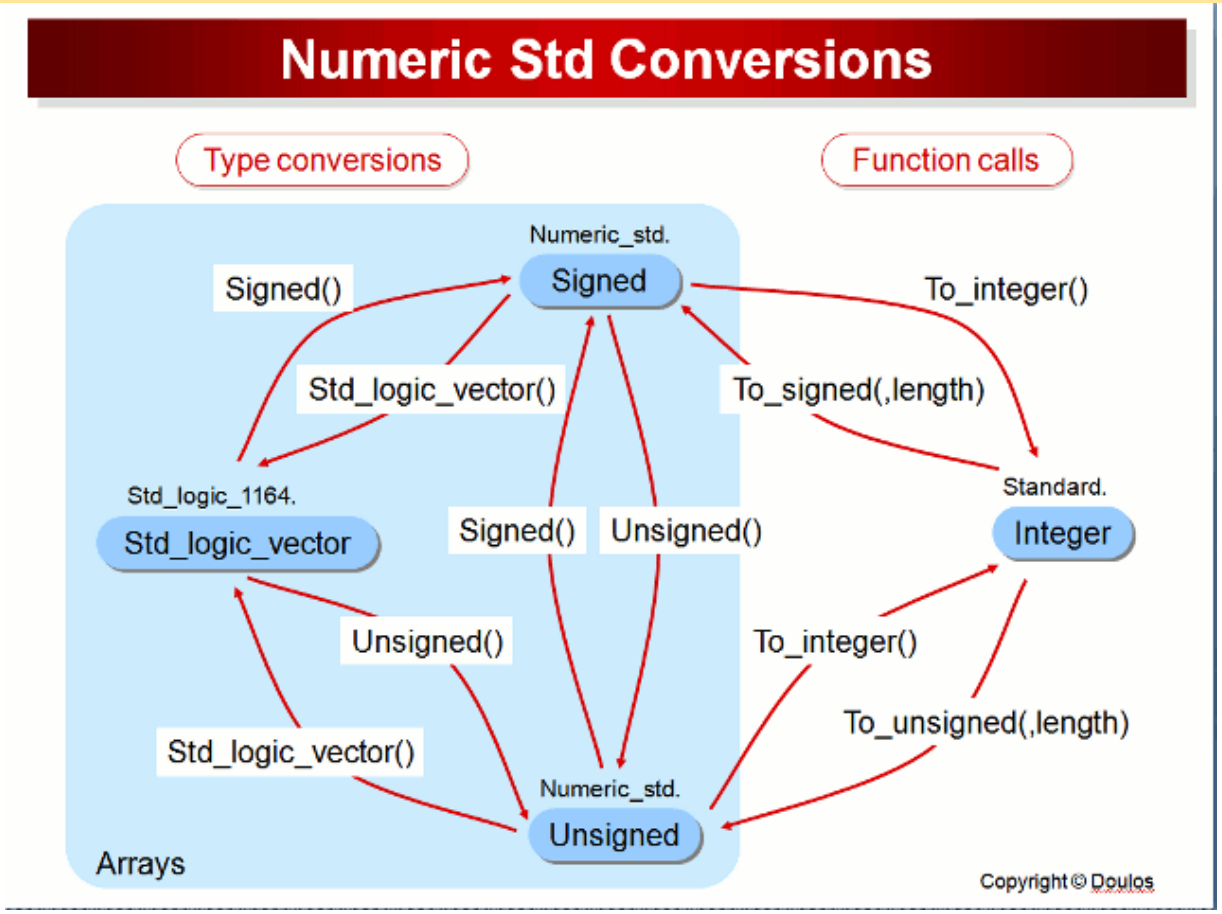
VHDL – Τελεστές (operators)

Προτεραιότητα

	Τελεστής	Σημασία
Υψηλότερη	`not`	NOT
	`*` `/` `mod` `rem`	MUL, DIV, MOD, REM
	`+` `-`	PLUS, MINUS
	`rol` `ror` `srl` `sll`	Περιστροφή, λογική ολίσθηση
	`<` `<=` `>` `>=`	Σχετική σύγκριση
	`=` `/=`	Σύγκριση ισότητας
Χαμηλότερη	`and` `or` `nand` `nor` `xor` `xnor`	Λογικές πράξεις (εκτελούνται από αριστερά προς τα δεξιά)

VHDL – Τύποι σημάτων

Μετατροπές



VHDL – Τελεστές (operators)

Μπορείτε να διαβάσετε και:
https://www.nandland.com/vhdl/examples/example-shifts.html

Μπορείτε να δείτε επίσης και τη συμπεριφορά τους σε bit_vector:
https://hdlworks.com/hdl_corner/vhdl_ref/VHDLContents/BitVector.htm

Συνοπτικός οδηγός VHDL
https://www.ics.uci.edu/~jmoorkan/vhdlref/vhdl.html

Για διαφορές mod/rem:
https://stackoverflow.com/questions/25848879/difference-between-mod-and-rem-operators-in-vhdl

VHDL – Conditional assignment

Example 1

Ένα σύστημα δέχεται στην είσοδο το σήμα a των 2 bit και στην έξοδο υπάρχει το σήμα b των 3 bit που αντιστοιχεί στη μετάδοση του σήματος a με επιπλέον bit άρτιας ισοτιμίας (b=parity_bit&a).

VHDL – Conditional assignment (Dataflow)

Example 2: With Select

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;USE ieee.numeric_std.ALL;



entity testing is
port (a  :   in   std_logic_vector(1 downto 0);
       b  :  out std_logic_vector(2 downto 0)
);

end entity testing;
```

```
architecture Dataflow of testing is

begin

with a select
b<="000" when "00",
     "101" when "01",
     "110" when "10",
     "011" when "11",
     "000" when others;

end architecture Dataflow ;
```

Λογική Συνθήκη

Ανάθεση με επιλογή
(διακριτές τιμές συγκεκριμένου σήματος)

Μία εντολή: Ανάθεση τιμής σε ένα σήμα

Πρέπει να ελέγχω ΟΛΕΣ τις τιμές

VHDL – Conditional assignment (Behavioral)

Example 2: Case

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;USE ieee.numeric_std.ALL;



entity testing is
port (a  :   in   std_logic_vector(1 downto 0);
        b  :  out std_logic_vector(2 downto 0)
);

end entity testing;
```

```
architecture Behavioral of testing is

begin

test: process (a) is

begin

case a is
when “00” => b<=“000";
when “01” => b<=“101";
when “10” => b<=“110";
when “11” => b<=“011";
when others => b<=“000";
end case;
end process test;
end architecture Behavioral;
```

Μπορεί και πολλαπλές εντολές ανά περίπτωση (when)

case
Διακριτές τιμές

VHDL –Selected & Conditional assignment

Example 2

Σε ένα Computer Room υπάρχουν 3 κλιματιστικά και ένας αισθητήρας θερμοκρασίας (θερμόμετρο). Εάν το θερμόμετρο δείξει θερμοκρασία κάτω των 25 βαθμών τότε δεν λειτουργεί το κλιματιστικό. Αν η θερμοκρασία είναι μέχρι 30 βαθμούς λειτουργεί το 1ο. Αν η θερμοκρασία είναι μέχρι 45 βαθμούς λειτουργεί και το 2ο. Εάν είναι πάνω από 45 λειτουργεί και το 3ο. Θεωρείστε ότι η σειρά λειτουργίας των κλιματιστικών είναι σαφώς ορισμένη και δεν μας απασχολεί για το πρόβλημά μας. Είσοδος του συστήματος το σήμα temperature και έξοδος το σήμα action (έχει 4 δυνατές τιμές).

VHDL – Selected assignment (Dataflow)

Example 2: When

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;USE ieee.numeric_std.ALL;

entity testing is
port (temperature:  in std_logic_vector(5 downto 0);
        action          :  out std_logic_vector(1 downto 0)
);
end entity testing;

architecture Dataflow of testing is
signal temp :unsigned(5 downto 0);
begin

temp<= unsigned(temperature);
action<="00" when temp<25 else
      "01" when (temp>=25 and temp<30) else
      "10"  when (temp>=30 and temp<45) else
      "11";


end architecture Dataflow ;
```

max=2^5-1=63

action:
00: Δεν δουλεύει τίποτα
01: Δουλεύει 1 AC
10: Δουλεύουν 2 AC
11: Δουλεύουν και τα 3 AC

Λογική Συνθήκη

Μία εντολή: Ανάθεση τιμής σε ένα σήμα

VHDL – Conditional assignment (Behavioral)

Example 2: If

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;USE ieee.numeric_std.ALL;

entity testing is
port (temperature:  in std_logic_vector(5 downto 0);
        action          :   out std_logic_vector(1 downto 0)
);
end entity testing;

architecture Behavioral of testing is
signal temp :unsigned(5 downto 0);
begin
temp<= unsigned(temperature);

action_temp: process(temp) is
begin
if (temp<25) then
   action<="00";
elsif (temp<30) then
   action<="01";
elsif (temp<45) then
   action<="10";
else
   action<="11";
end if;
end process action_temp;

end architecture Behavioral;
```

Μπορεί και πολλαπλές εντολές ανά περίπτωση (if/elsif/else)

if, **elsif**, else
Μόνο μέσα
σε process

Πρέπει να
ελέγχω ΟΛΕΣ
τις τιμές

VHDL – Selected assignment (Dataflow)

Example 2: With Select

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;USE ieee.numeric_std.ALL;



entity testing is
port (a       :   in std_logic_vector(4 downto 0);
      bcd     :  out std_logic_vector(3 downto 0)
);

end entity testing;
```

```
architecture Dataflow of testing is

signal temp, temp_action :integer;

begin

temp<= to_integer(unsigned(temperature));
temp_action<=0 when temp<25 else
             1 when (temp>=25 and temp<30) else
             2 when (temp>=30 and temp<45) else
             3;


with temp_action select
action<="00" when 0,
         "01" when 1,
         "10" when 2,
         "11" when 3,
         “00" when others;


end architecture Dataflow ;
```

Λογική Συνθήκη

Ανάθεση με επιλογή
(διακριτές τιμές συγκεκριμένου σήματος)

Μία εντολή: Ανάθεση τιμής σε ένα σήμα

Πρέπει να ελέγχω ΟΛΕΣ τις τιμές

VHDL – Conditional assignment (Behavioral)

Example 2: Case

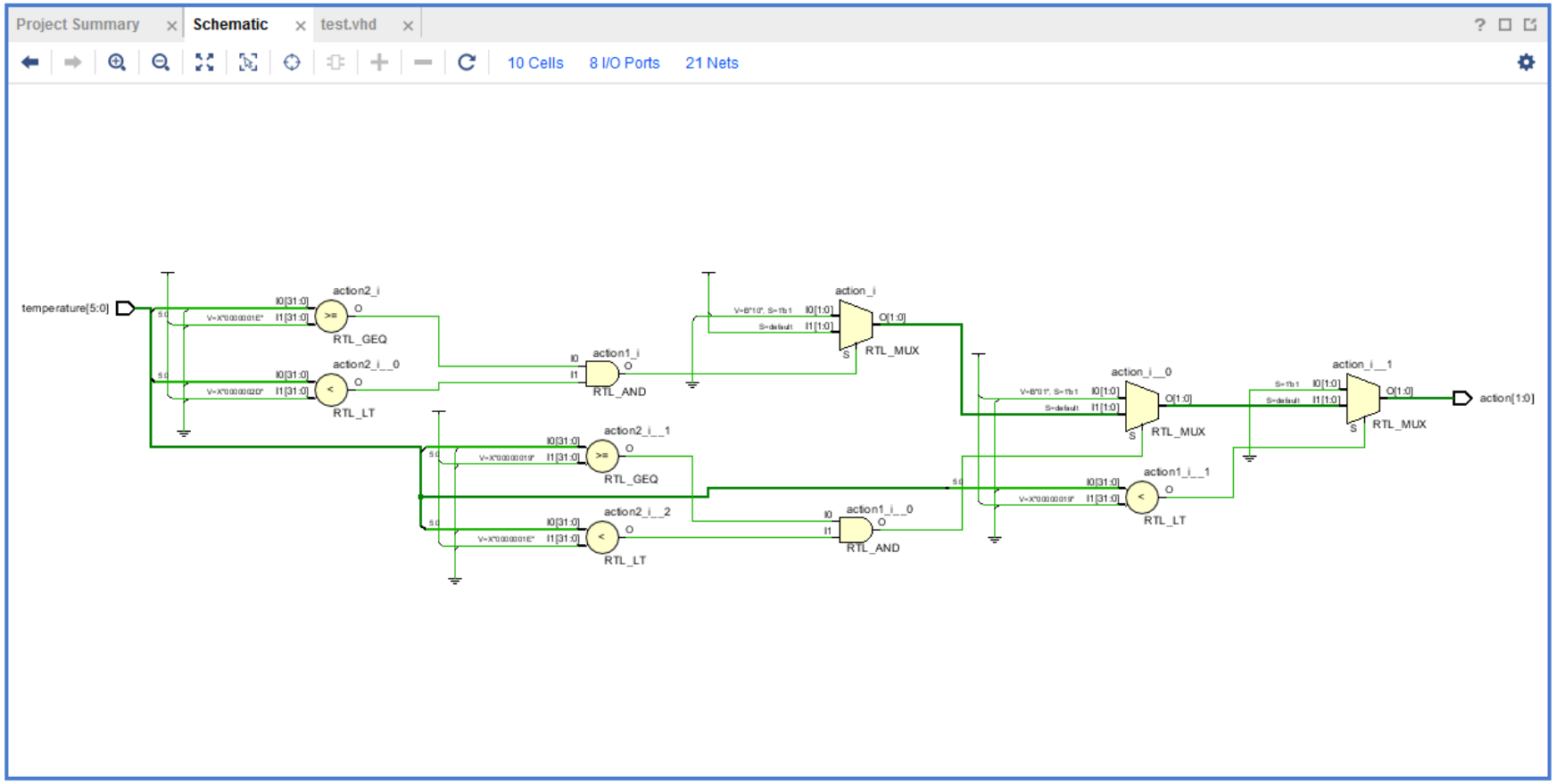
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;USE ieee.numeric_std.ALL;

entity testing is
port (temperature:  in std_logic_vector(5 downto 0);
        action          :  out std_logic_vector(1 downto 0)
);
end entity testing;

architecture Behavioral of testing is
signal temp :integer;

begin
temp<= to_integer(unsigned(temperature));

action_temp: process(temp) is
begin
if (temp<25) then       temp_action<=0;
elsif (temp<30) then    temp_action<=1;
elsif (temp<45) then    temp_action<=2;
else                    temp_action<=3;
end if;

case temp_action is
            when 0 => action<="00";
            when 1 => action<="01";
            when 2 => action<="10";
            when 3 => action<="11";
            when others => action<="00";
end case;
end process action_temp;
end architecture Behavioral;
```

Μπορεί και πολλαπλές εντολές ανά περίπτωση (when)

case
Διακριτές τιμές

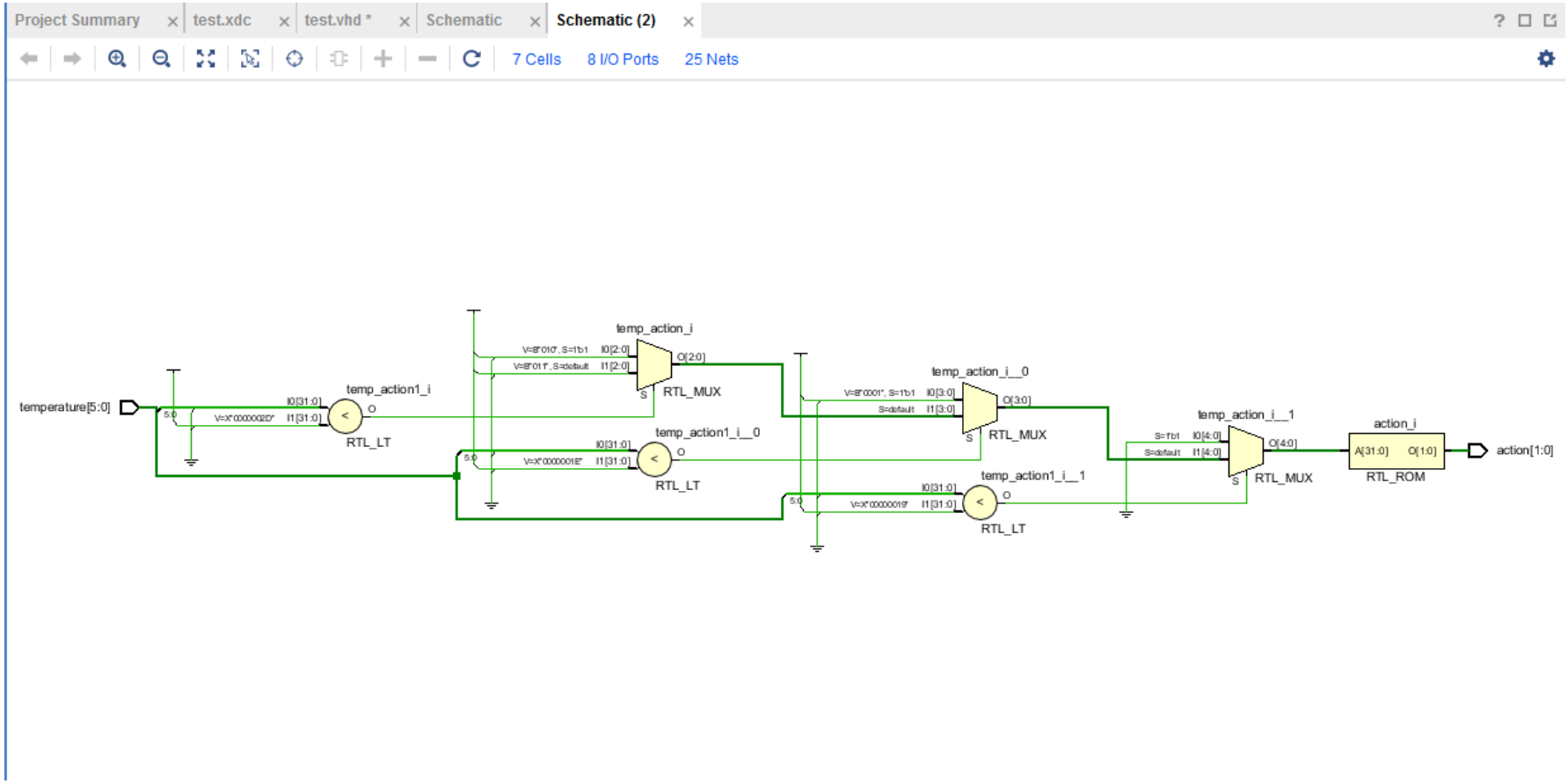
VHDL – Selected assignment

Example 2: RTL Analysis



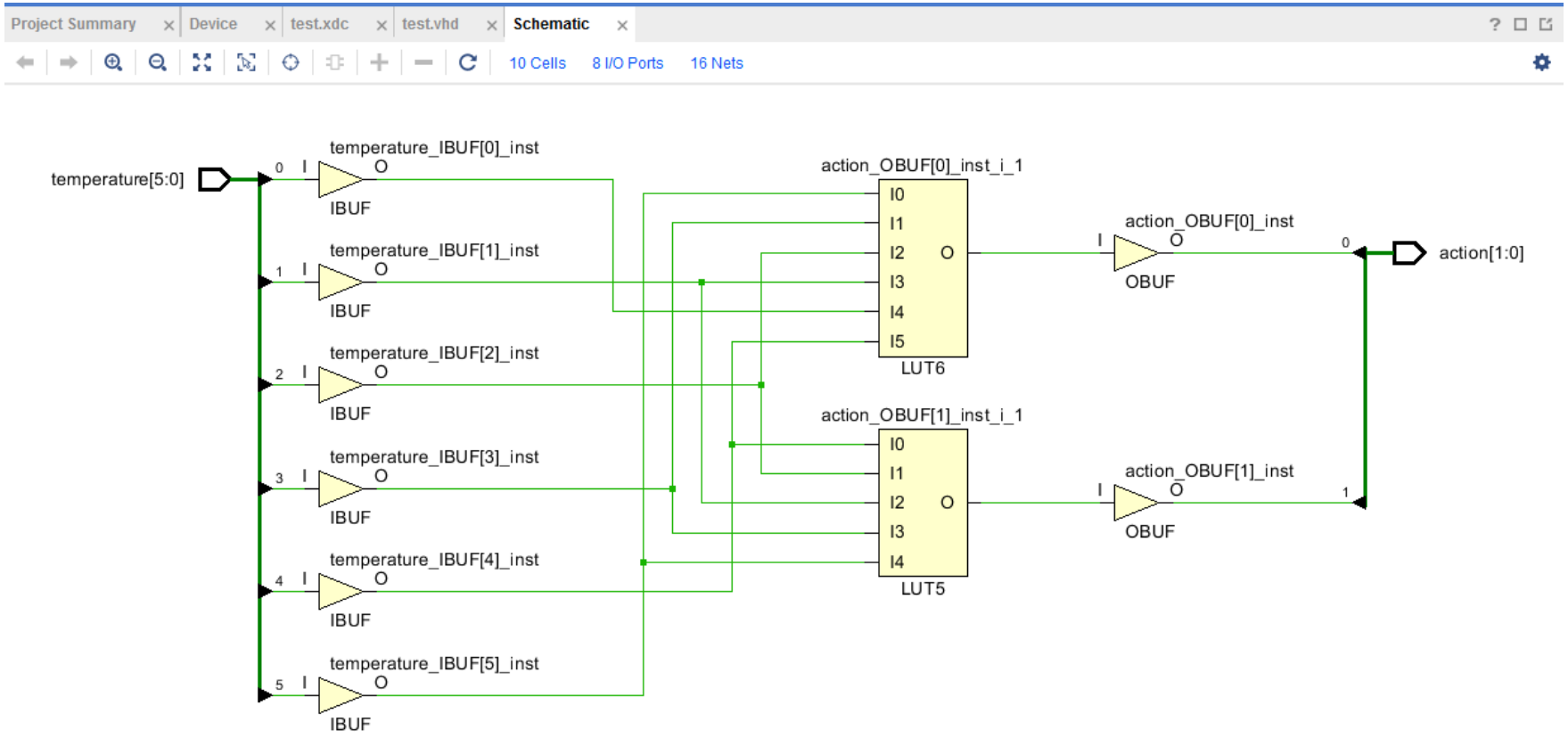
VHDL – Conditional assignment

Example 2: RTL Analysis



VHDL – Selected & Conditional assignment

Example 2: Synthesis



VHDL – Selected & Conditional assignment

Synthesis

1.
signal <= value when condition else …

2.
with signal_1 select
signal_2<=value when (discrete signal_1 value),
…

Και στις 2 περιπτώσεις είμαστε απευθείας στο σώμα της αρχιτεκτονικής και η εντολή αφορά απόδοσης τιμής σε ένα συγκεκριμένο σήμα

3.
If condition then action;elsif … else end if

4.
case signal is
when value => assignment (discrete signal value),
…

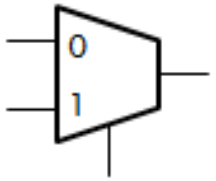
Και στις 2 περιπτώσεις είμαστε στο σώμα process (ή procedure) και αφορά την εκτέλεση μίας η περισσότερων εντολών ανάλογα τη συνθήκη, και μπορεί να αφορά πάνω από ένα σήματα.

http://www.pldworld.com/_hdl/2/_ref/acc-eda/language_overview/concurrent_statements/conditional_vs__selected_assignment.htm

VHDL – Multiplexer

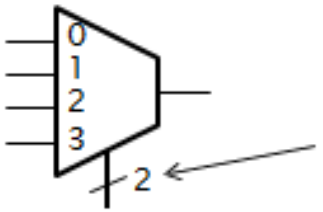
- Επιλέγει μεταξύ εισόδων δεδομένων με βάση μια είσοδο επιλογής

2-σε-1 πολυπλέκτης



sel	z
0	a_0
1	a_1

4-σε-1 πολυπλέκτης



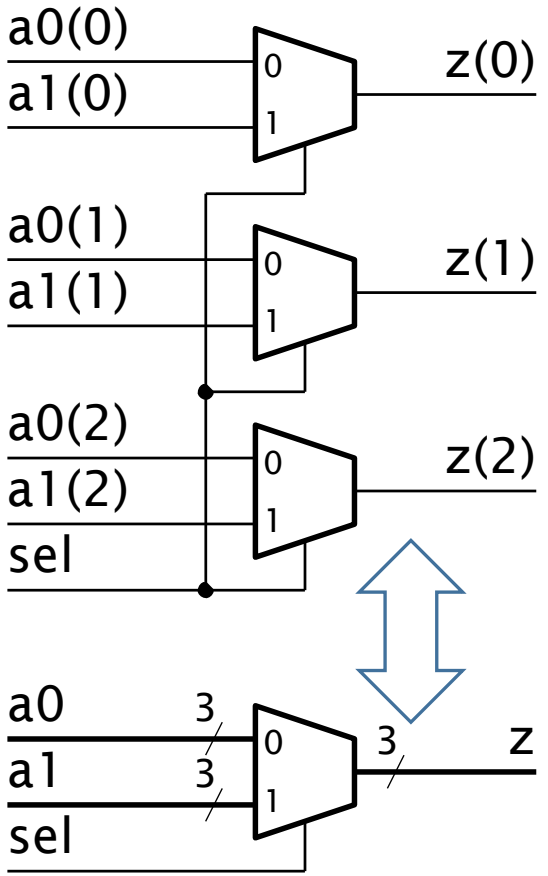
δύο bit επιλογής

sel	z
00	a_0
01	a_1
10	a_2
11	a_3

- Ένας N-σε-1 πολυπλέκτης απαιτεί $\lceil \log_2 N \rceil$ bit επιλογής

VHDL – Multiplexer

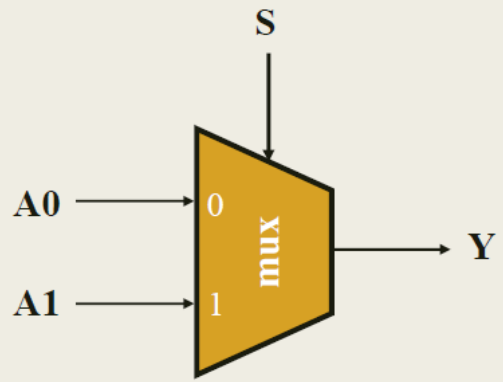
- Για να επιλέξουμε μεταξύ *N* κωδικών λέξεων των *m* bit
  - Συνδέουμε παράλληλα *m* πολυπλέκτες των *N* εισόδων



VHDL – Multiplexer

Entity (2to1)

```
entity MUX_2_to_1 is
    port (
      A0: in STD_LOGIC;
      A1: in STD_LOGIC;
      S: in STD_LOGIC;
      Y: out STD_LOGIC);
end MUX_2_to_1;
```



VHDL – Multiplexer

Architecture (2to1 - behavioral)

Περιγραφή συμπεριφοράς

```
architecture BEHAVIORAL of MUX_2_to_1 is
begin
  process (A0, A1, S)
  begin
    if (S = '0') then
      Y <= A0;
    else
      Y <= A1;
    end if;
  end process;
end BEHAVIORAL;
```

Στη λίστα ευαισθησίας συμπεριλαμβάνονται όλες οι είσοδοι του συνδυαστικού κυκλώματος

y<=A0 when s='0' else A1;

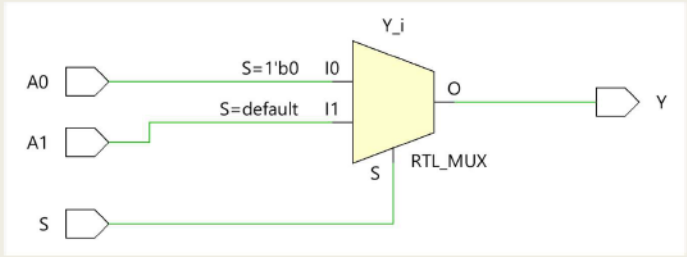
VHDL – Multiplexer

Architecture (2to1 - behavioral)

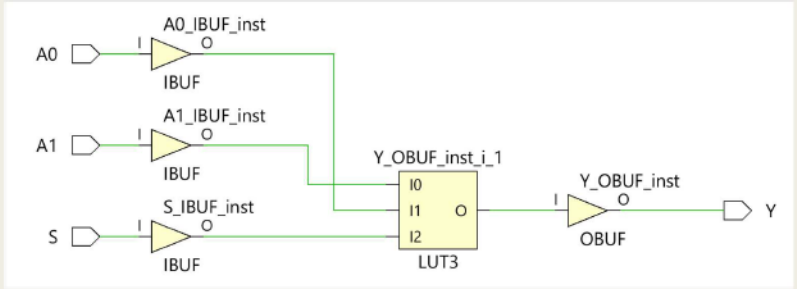
Η αρχιτεκτονική του πολυπλέκτη 2 σε 1 στη VHDL

Περιγραφή συμπεριφοράς

- Σχηματικό διάγραμμα RTL



- Σχηματικό διάγραμμα σε τεχνολογία FPGA

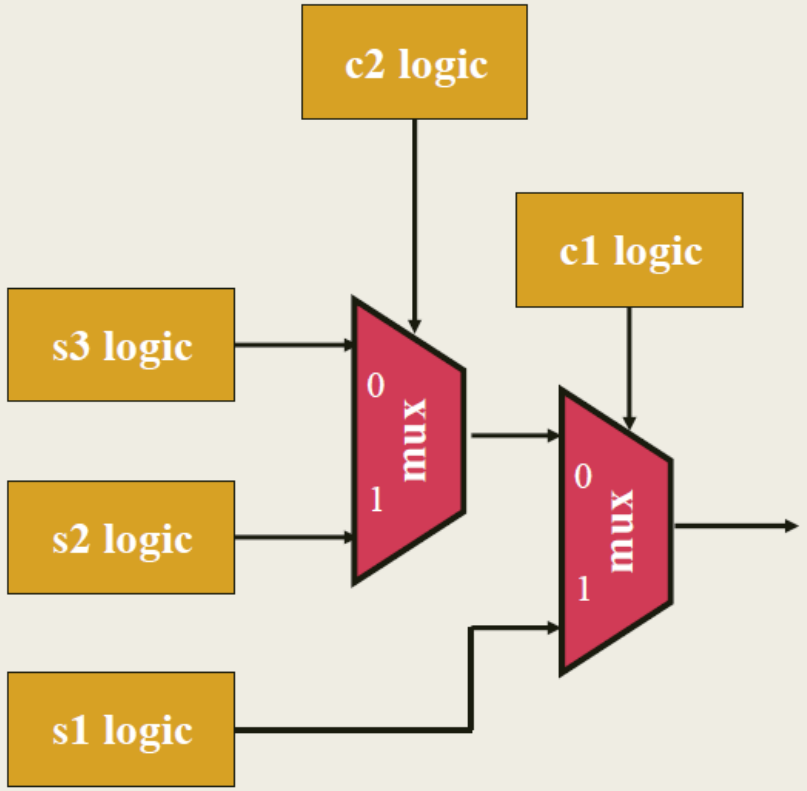


VHDL – Multiplexer

Υλοποίηση If

- Υλοποίηση εντολής IF με τη χρήση πολυπλεκτών 2 σε 1

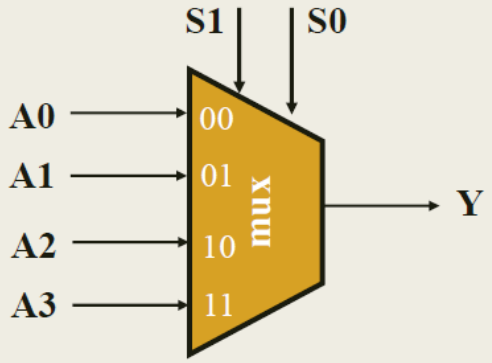
```
if c1 then
    s1;
elsif c2 then
    s2;
else
    s3;
end if;
```



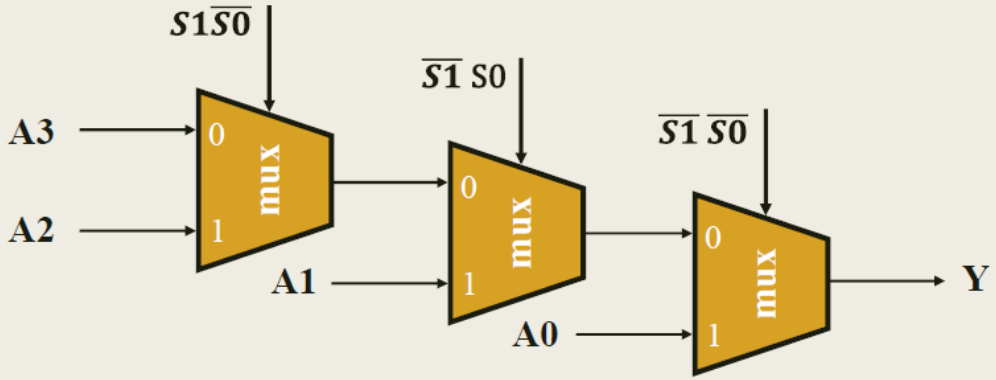
VHDL – Multiplexer

Entity (4to1)

```
entity MUX_4_to_1 is
    port (
      A0: in STD_LOGIC;
      A1: in STD_LOGIC;
      A2: in STD_LOGIC;
      A3: in STD_LOGIC;
      S0: in STD_LOGIC;
      S1: in STD_LOGIC;
      Y: out STD_LOGIC);
end MUX_4_to_1;
```



Λύση 1: Υλοποίηση με πολυπλέκτες 2 σε 1 σε δομή αλυσίδας



VHDL – Multiplexer

Architecture (4to1)

Η αρχιτεκτονική του πολυπλέκτη 4 σε 1 στη VHDL
Περιγραφή συμπεριφοράς – Λύση 1

```
architecture BEHAVIORAL of MUX_4_to_1 is
begin
  process (A0, A1, A2, A3, S0, S1)
  begin
    if    (S1 = '0' and S0 = '0') then Y <= A0;
    elsif (S1 = '0' and S0 = '1') then Y <= A1;
    elsif (S1 = '1' and S0 = '0') then Y <= A2;
    else                               Y <= A3;
    end if;
  end process;
end BEHAVIORAL;
```

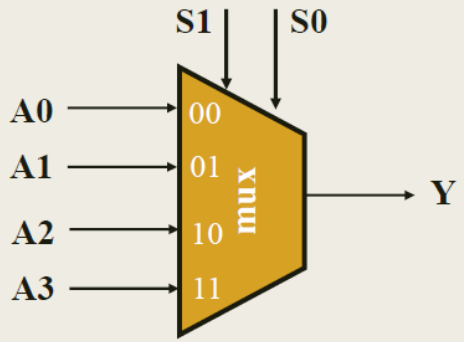
Στη λίστα ευαισθησίας συμπεριλαμβάνονται όλες οι είσοδοι του συνδυαστικού κυκλώματος

VHDL – Multiplexer

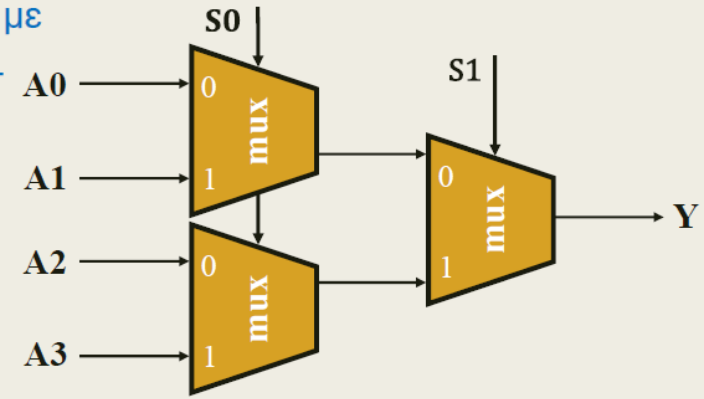
Entity (4to1)

Η οντότητα του πολυπλέκτη 4 σε 1 στη VHDL

```
entity MUX_4_to_1 is
    port (
      A0: in STD_LOGIC;
      A1: in STD_LOGIC;
      A2: in STD_LOGIC;
      A3: in STD_LOGIC;
      S0: in STD_LOGIC;
      S1: in STD_LOGIC;
      Y: out STD_LOGIC);
end MUX_4_to_1;
```



Λύση 2: Υλοποίηση με πολυπλέκτες 2 σε 1 σε δομή δένδρου



VHDL – Multiplexer

Architecture (4to1)

Η αρχιτεκτονική του πολυπλέκτη 4 σε 1 στη VHDL
Περιγραφή συμπεριφοράς – Λύση 2

```
architecture BEHAVIORAL of MUX_4_to_1 is
begin
  process (A0, A1, A2, A3, S0, S1)
  begin
    if (S1 = '0') then
      if (S0 = '0') then Y <= A0;
      else Y <= A1;
      end if;
    else
      if (S0 = '0') then Y <= A2;
      else Y <= A3;
      end if;
    end if;
  end process;
end BEHAVIORAL;
```

Στη λίστα ευαισθησίας συμπεριλαμβάνονται όλες
οι είσοδοι του συνδυαστικού κυκλώματος

VHDL - Περίληψη

- Τύπος Signed
- Άλλοι τύπο σημάτων
- Τελεστές
- Ολισθήσεις
- Selected Assignment
- Conditional Statements
- Διαβάζετε τις παραγράφους 2.4, 3.2 (θεωρία και VHDL) από Ashenden και 2.7, 2.8, 4.2.4, 4.2.6, 4.3, 4.5.1, 4.5.2, 4.5.3 (ΟΧΙ το κομμάτι της VERILOG) από το βιβλίο των Harris.