dscal
DIGITAL SYSTEMS & COMPUTER ARCHITECTURE LABORATORY

Εργαστήριο Σχεδίασης Ψηφιακών Συστημάτων

6η Διάλεξη

Τύποι σημάτων – Δομή Process – Εντολές υπό συνθήκη

Βασιλόπουλος Διονύσης

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

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

VHDL – Conditional & Selected assignment

Πρόβλημα

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.

VHDL – Conditional assignment (Dataflow)

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 :integer;
begin

temp<= to_integer(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

ΥΠΑΡΧΕΙ ΠΡΟΤΕΡΑΙΟΤΗΤΑ

Λογική Συνθήκη

Μία εντολή: Ανάθεση τιμής σε ένα σήμα

VHDL – Selected assignment (Dataflow)

With Select

```
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, 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,
         "11" 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

ΔΕΝ ΥΠΑΡΧΕΙ ΠΡΟΤΕΡΑΙΟΤΗΤΑ

VHDL – Conditional assignment (Behavioral)

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 :integer;
```

ΥΠΑΡΧΕΙ ΠΡΟΤΕΡΑΙΟΤΗΤΑ

```
begin
temp<= to_integer(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 (Behavioral)

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;
```

Μπορεί να υπάρχουν πολλές εντολές ανά περίπτωση (when)

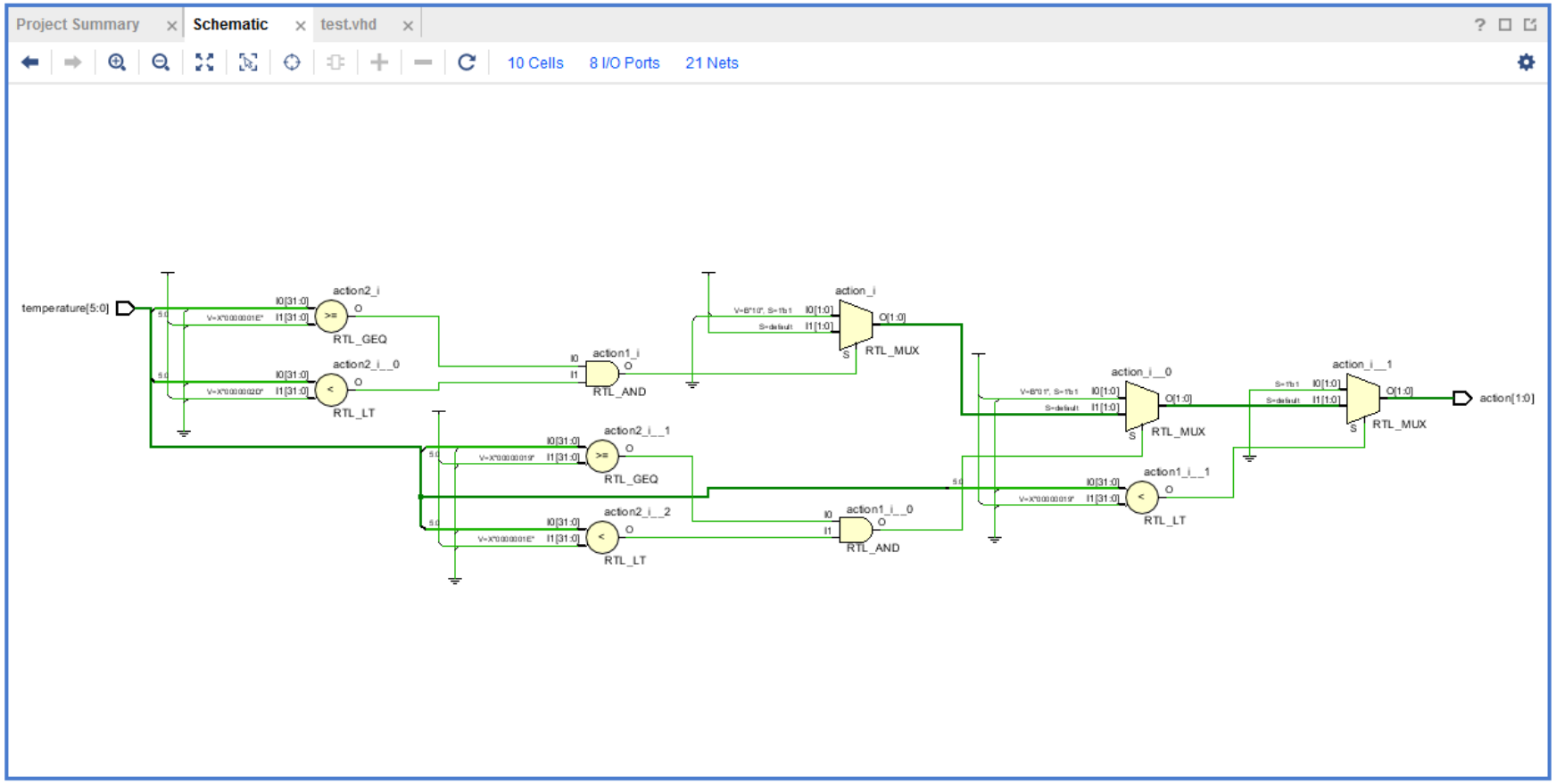
```
begin
temp<= to_integer(unsigned(temperature));

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<="XX";
end case;
end process action_temp;
end architecture Behavioral;
```

case
Διακριτές τιμές

ΔΕΝ ΥΠΑΡΧΕΙ ΠΡΟΤΕΡΑΙΟΤΗΤΑ

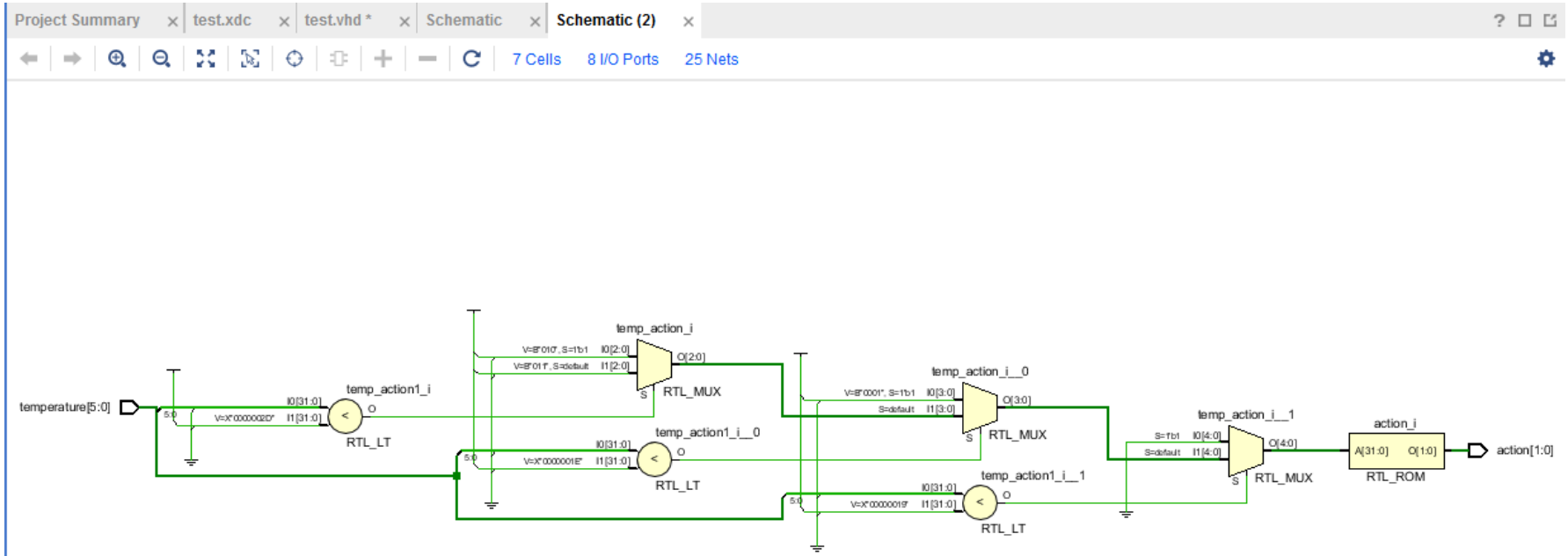
VHDL – Selected assignment

RTL Analysis



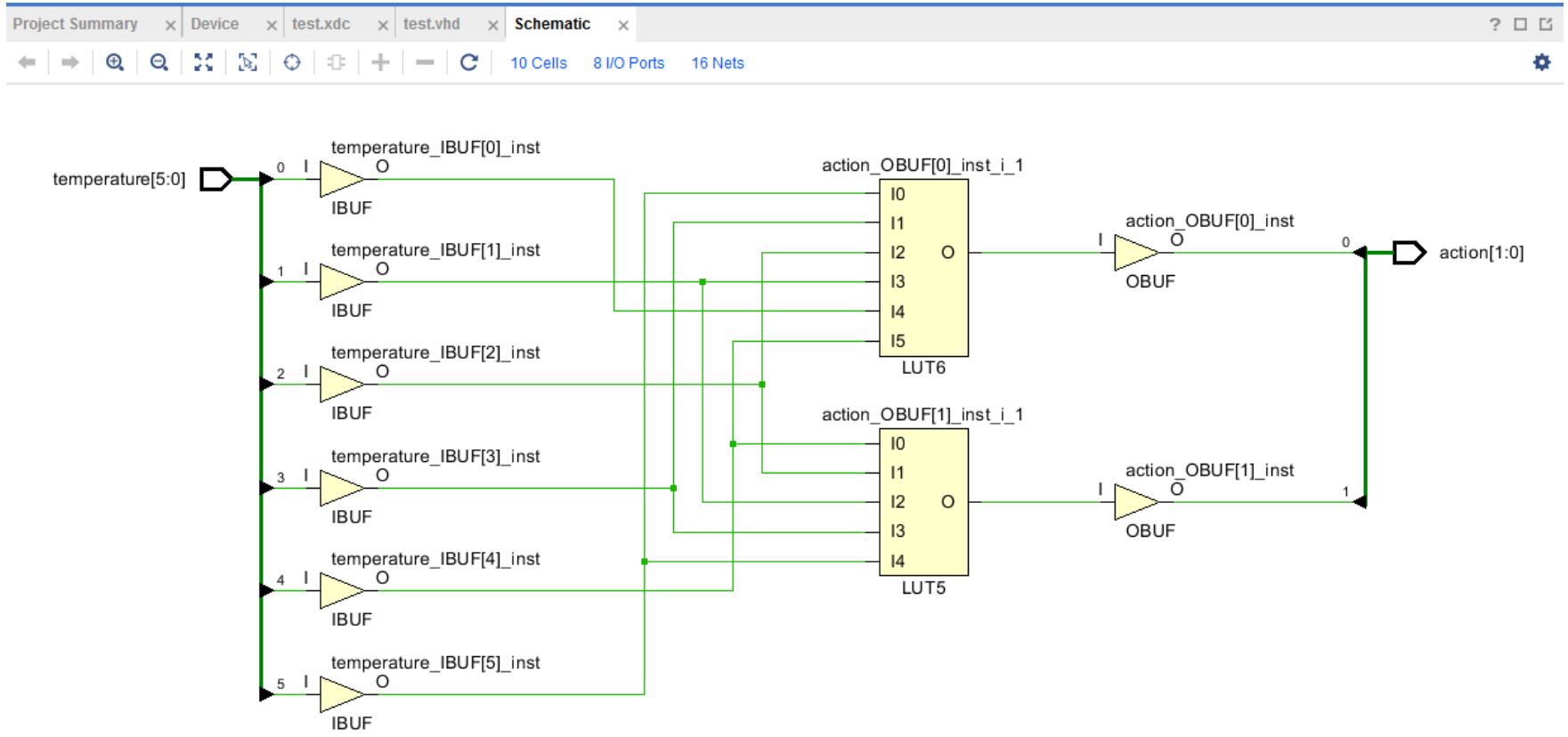
VHDL – Conditional assignment

RTL Analysis



VHDL – Selected & Conditional assignment

Synthesis



VHDL – Selected & Conditional assignment

Σύνοψη

ΠΡΟΣΟΧΗ: Εάν δεν ελέγξετε όλες τις περιπτώσεις τότε δημιουργούμε latches που είναι ανεπιθύμητο

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),
…

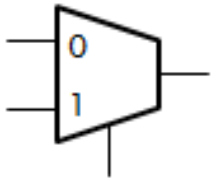
Και στις 2 περιπτώσεις είμαστε στο σώμα procedure ή process και αφορά την εκτέλεση μίας η περισσότερων εντολών ανάλογα τη συνθήκη, και μπορεί να αφορά πάνω από ένα σήματα.

http://www.pldworld.com/_hdl/2/_ref/acc-eda/language_overview/concurrent_statements/conditional_vs__selected_assignment.htm
https://insights.sigasi.com/tech/signal-assignments-vhdl-withselect-whenelse-and-case/
https://nandland.com/how-to-avoid-creating-a-latch/

VHDL – Multiplexer

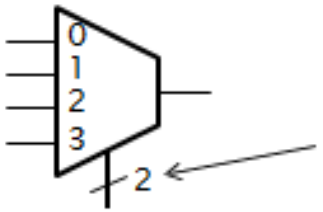
- Επιλέγει μεταξύ εισόδων δεδομένων με βάση μια είσοδο επιλογής

2-σε-1 πολυπλέκτης



sel	z
0	a_0
1	a_1

4-σε-1 πολυπλέκτης



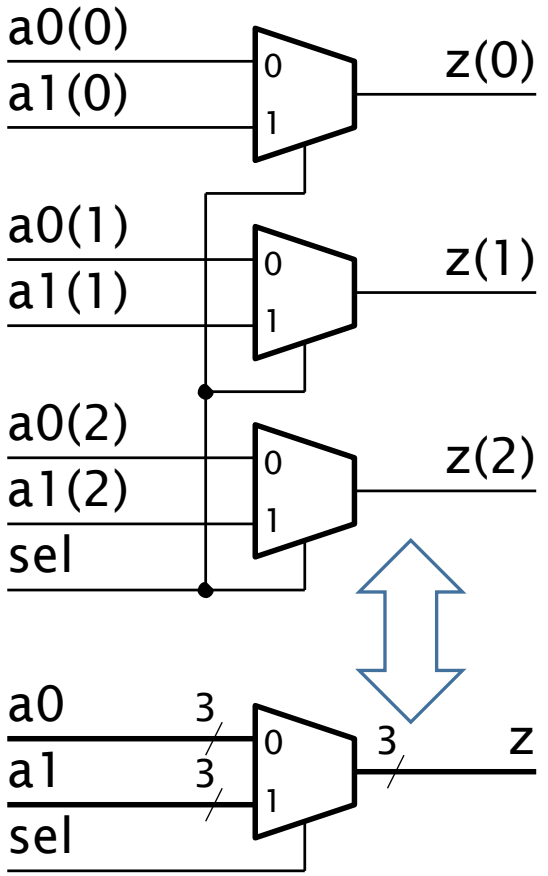
δύο bit επιλογής

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

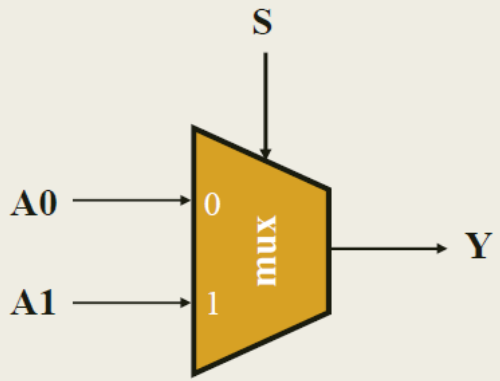
- Για να επιλέξουμε μεταξύ *N* κωδικών λέξεων των *m* bit
  - Συνδέουμε παράλληλα *m* πολυπλέκτες των *N* εισόδων



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)

Η αρχιτεκτονική του πολυπλέκτη 2 σε 1 στη VHDL
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;

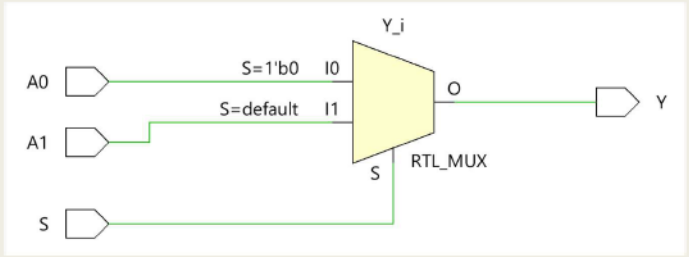
VHDL – Multiplexer

Architecture (2to1 - behavioral)

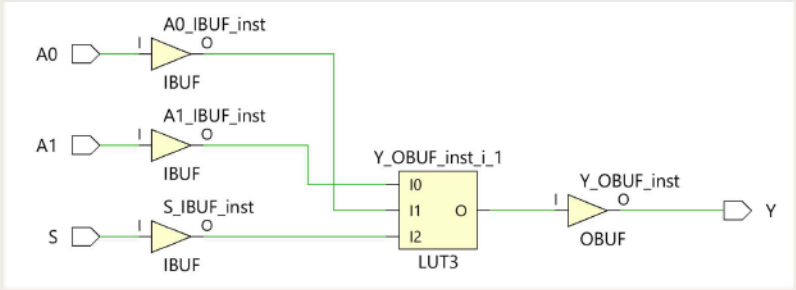
Η αρχιτεκτονική του πολυπλέκτη 2 σε 1 στη VHDL

Περιγραφή συμπεριφοράς

- Σχηματικό διάγραμμα RTL



- Σχηματικό διάγραμμα σε τεχνολογία FPGA

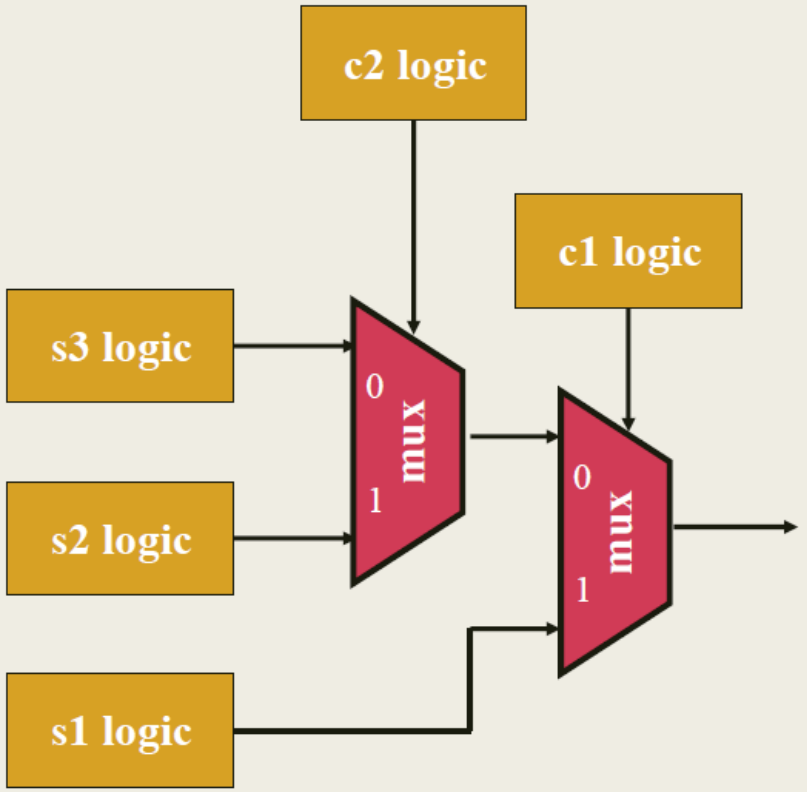


VHDL – Multiplexer

Υλοποίηση If

- Υλοποίηση εντολής IF με τη χρήση πολυπλεκτών 2 σε 1

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

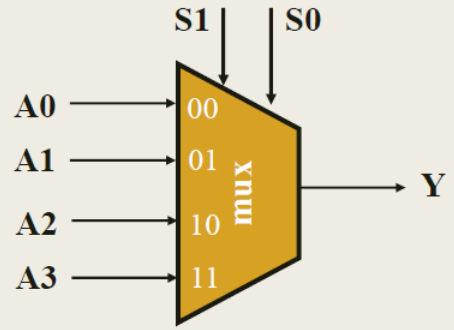


VHDL – Multiplexer

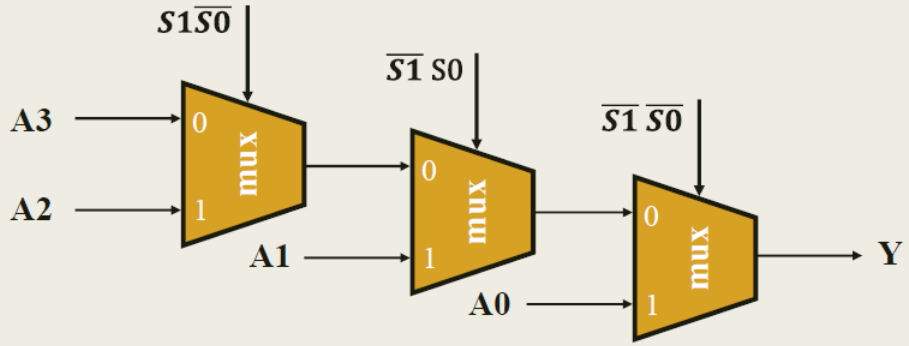
Entity (4to1)

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 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;
```

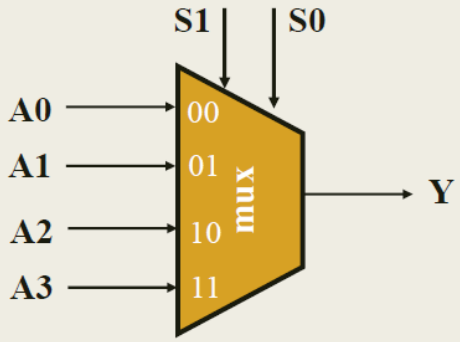
Στη λίστα ευαισθησίας συμπεριλαμβάνονται όλες
οι είσοδοι του συνδυαστικού κυκλώματος

VHDL – Multiplexer

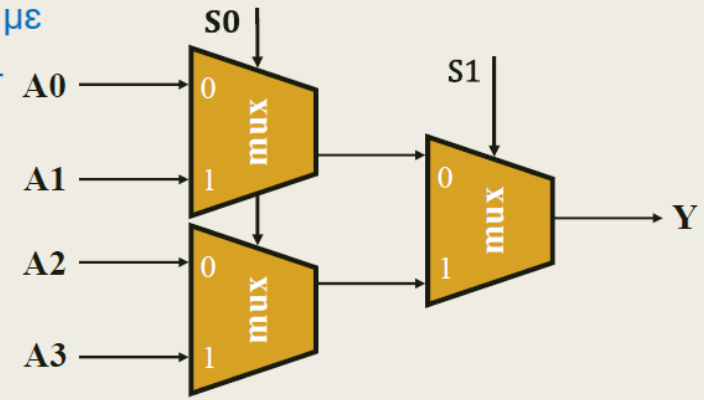
Entity (4to1)

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 – Τύποι (Types)

Declare new type

type identifier is (value1, value2, value3)
signal signal_name: identifier ;
type States is (S0,S1, S2); ← Μετατροπή εσωτερικά σε binary
signal FSM : States;

type indentifier is **scalar_type_definition**;
type own_small is range -5 to 5;

type indentifier is **array**(limit1 to/downto limit2) of other_type;
type new_array is array(0 to 5) of std_logic;

subtype indentifier is other_type_constraint;
subtype word is std_logic_vector(31 downto 0); ← **Subtype**

https://redirect.cs.umbc.edu/portal/help/VHDL/declare.html

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/ανά 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)

VHDL'93 Vivado 2019.2, **2022.2**

signed/unsigned

Για το std_logic_vector
ΔΕΝ υπάρχει κάποια συνάρτηση.

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

```
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)

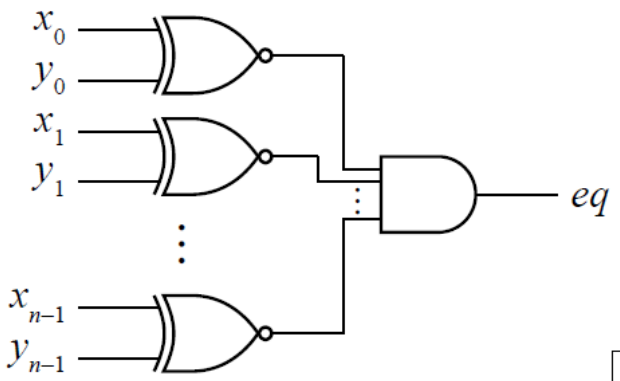
Προτεραιότητα

	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 – Τελεστές (operators)

Unsigned: Ισότητα/Σύγκριση

- Πύλη XNOR: Ισότητα δύο bit
  - Εφαρμογή σε κάθε bit δύο απρόσημων αριθμών



- Στην VHDL, το x = y δίνει boolean αποτέλεσμα
  - Ψευδές ή αληθές
  - Δεν μπορεί να γίνει ανάθεση σε σήμα std_logic

τελεστής

```
eq <= '1' when x = y else '0';
```

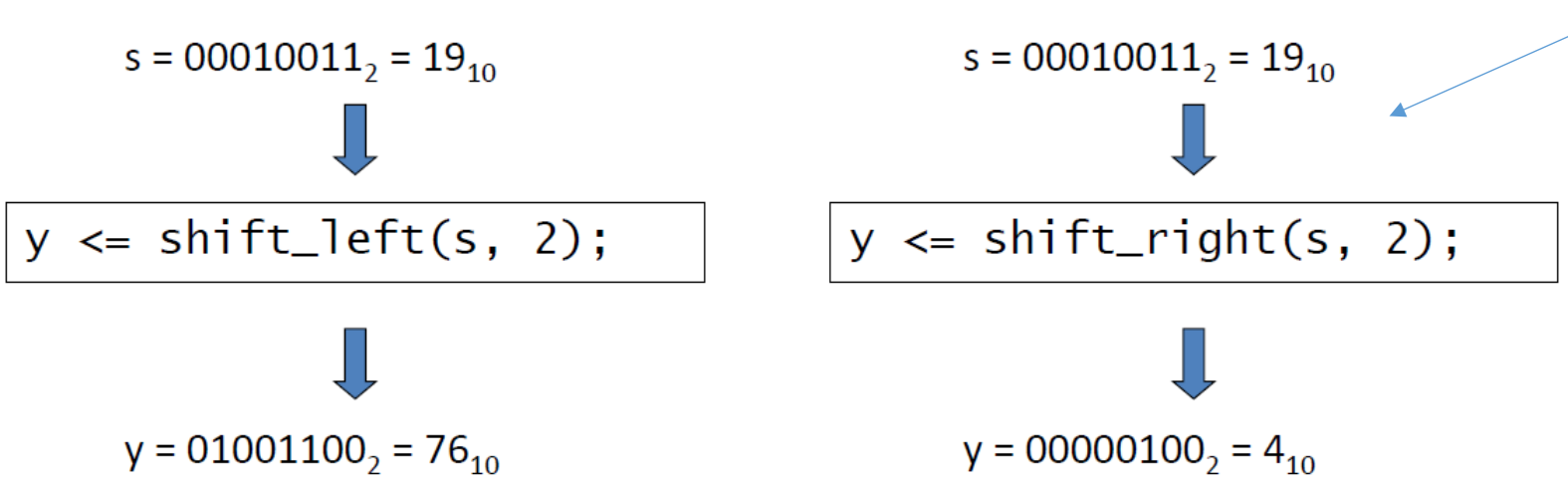
boolean

VHDL – Τελεστές (operators)

Unsigned: Ολίσθηση

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

Αποκοπή προς τα κάτω



VHDL – Τελεστές (operators)

Signed: Πολλαπλασιασμός-Διαίρεση

- 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)

Μπορείτε να διαβάσετε και:
https://www.nandland.com/vhdl/examples/example-shifts.html
https://redirect.cs.umbc.edu/portal/help/VHDL/operator.html
https://www.vhdl-online.de/vhdl_reference_93/shift_operators

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

Περίληψη

- Selected Assignment
- Conditional Statements
- Πολυπλέκτες
- Τελεστές
- Ολισθήσεις
- Διαβάζετε τις παραγράφους 2.3.1, 2.3.2, 3.1, 3.2 (θεωρία και VHDL) από Ashenden και 2.7, 4.2.1 - 4.2.6, 4.2.8, 4.2.9, 4.5.1, 4.5.2, 4.5.3 (ΟΧΙ το κομμάτι της VERILOG) από το βιβλίο των Harris.
- Συνοπτικός οδηγός VHDL: https://redirect.cs.umbc.edu/portal/help/VHDL/summary_one.html