dscal
DIGITAL SYSTEMS & COMPUTER ARCHITECTURE LABORATORY

Εργαστήριο Space Data Systems

VHDL

-

Εντολές υπό συνθήκη, Τελεστές, Εντολές επανάληψης, Procedure, Function, Generic

Βασιλόπουλος Διονύσης

ΕΔΙΠ 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. Είσοδος του συστήματος το σήμα temperature και έξοδος το σήμα action (έχει 4 δυνατές τιμές).

VHDL – Selected 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

Λογική Συνθήκη

Μία εντολή: Ανάθεση τιμής σε ένα σήμα

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 :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)

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,
         “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)

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 :unsigned(5 downto 0);
```

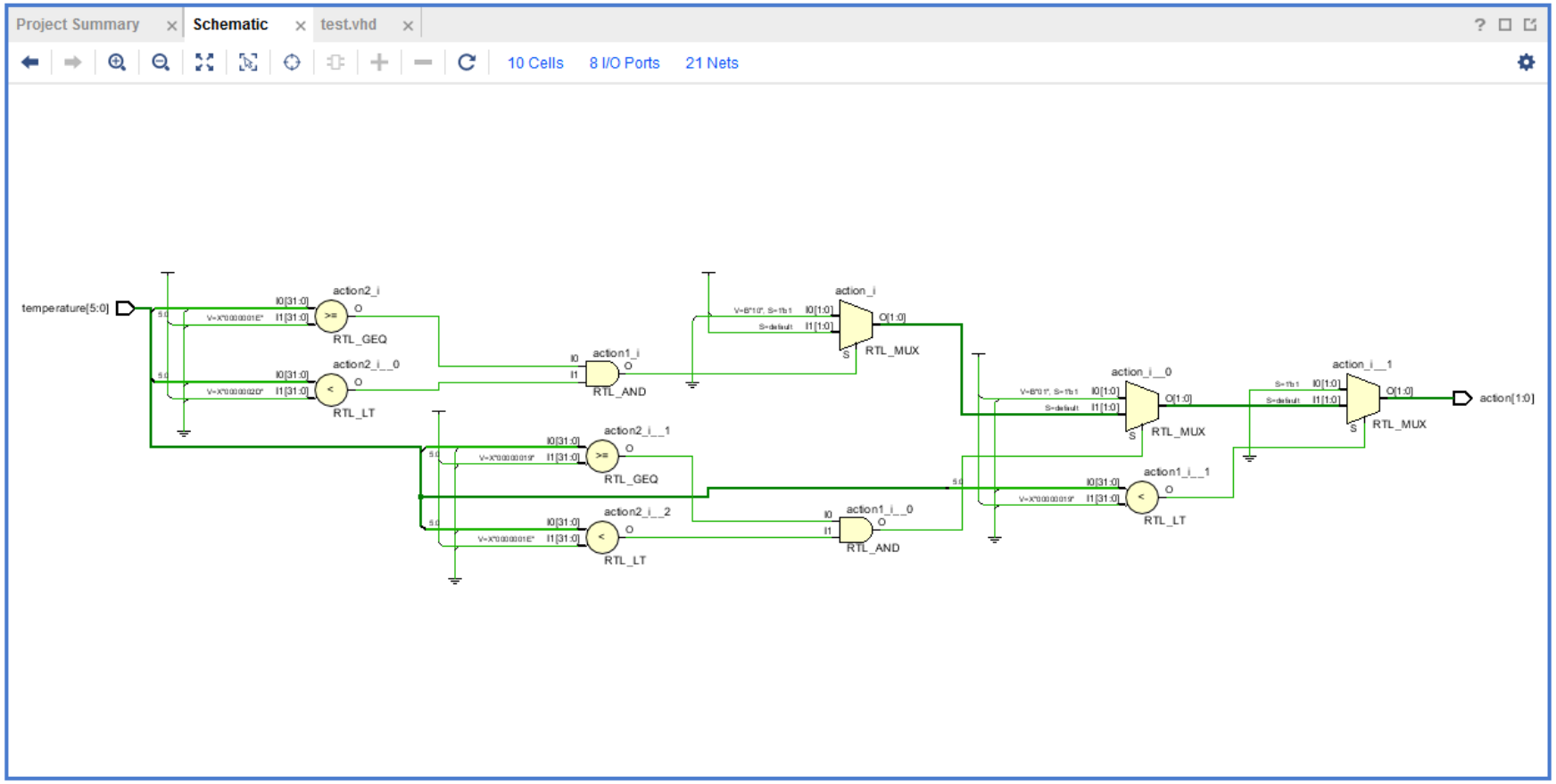
Μπορεί και πολλαπλές εντολές ανά περίπτωση (when)

```
begin
temp<= 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

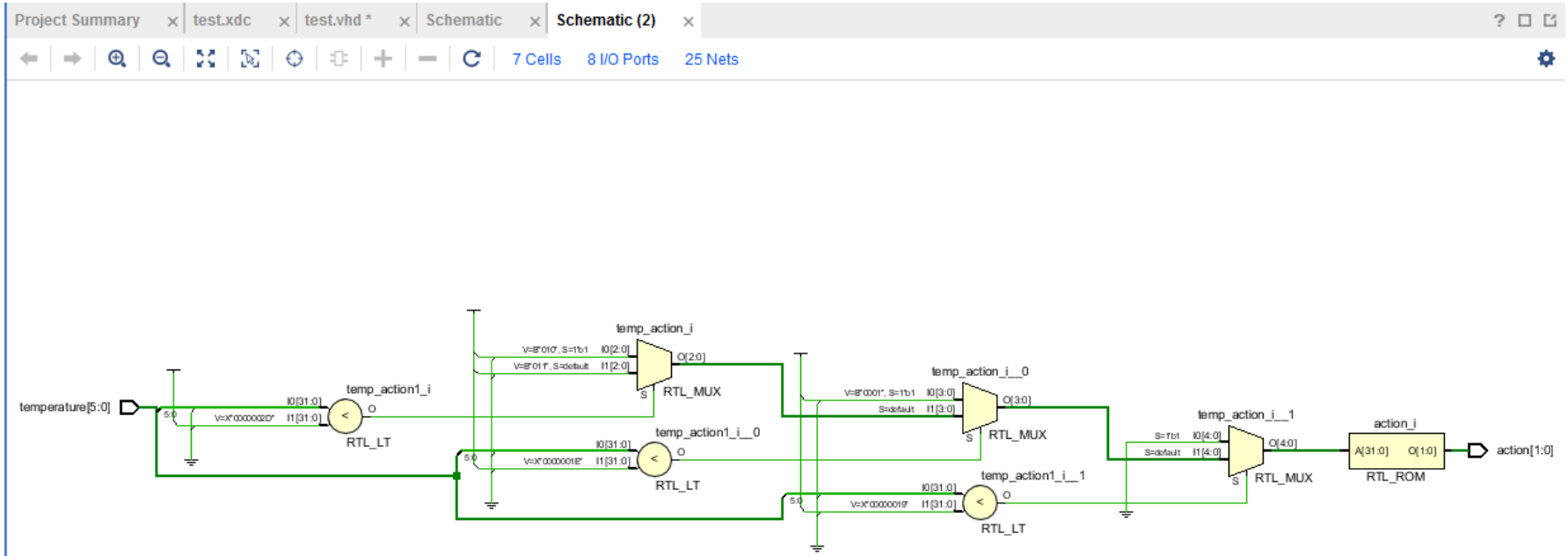
VHDL – Selected assignment

RTL Analysis



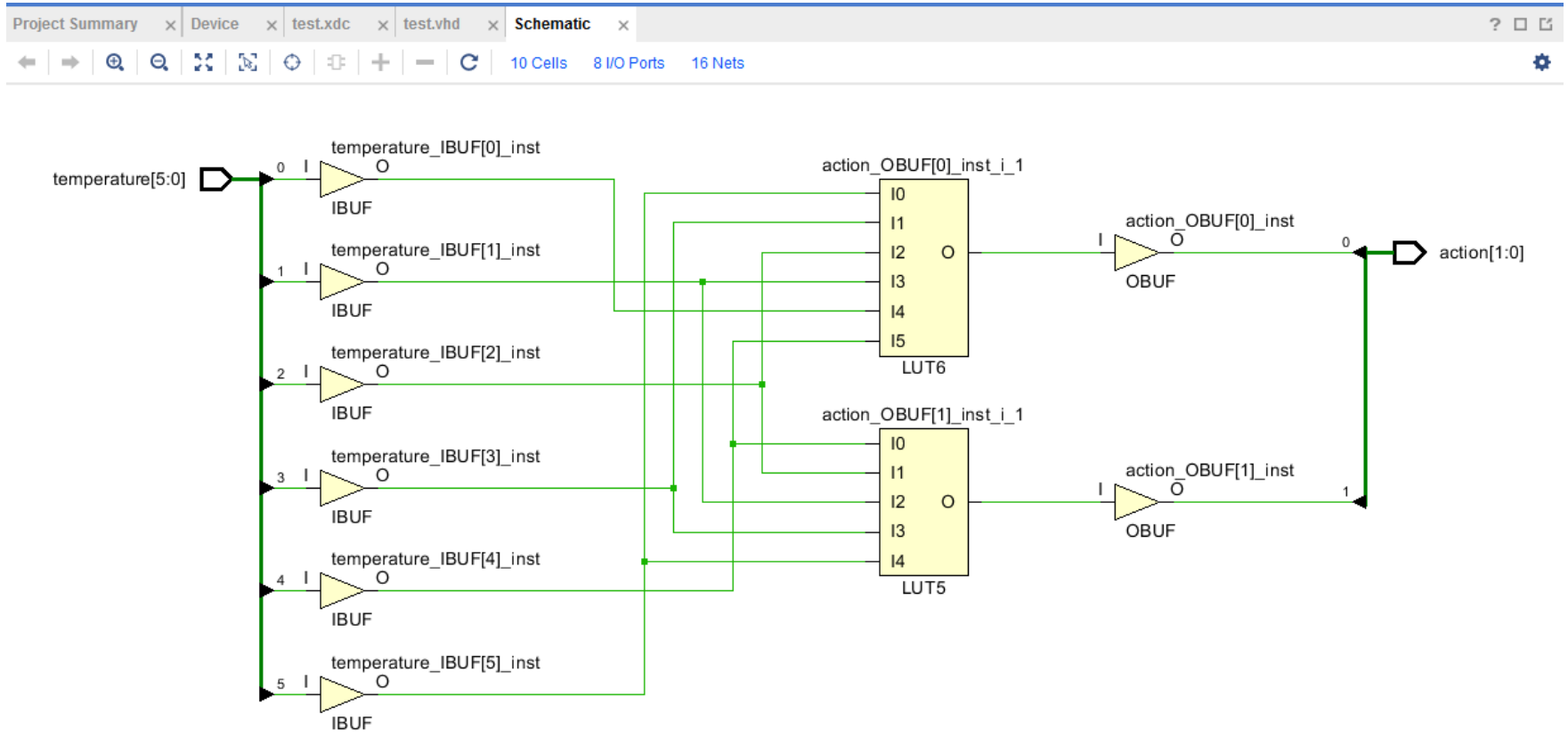
VHDL – Conditional assignment

RTL Analysis



VHDL – Selected & Conditional assignment

Synthesis



VHDL – Selected & Conditional assignment

Περίληψη

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(s);elsif … else end if

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

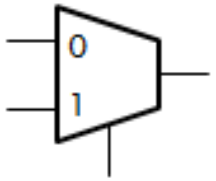
Και στις 2 περιπτώσεις είμαστε στο σώμα process (ή procedure) και αφορά την εκτέλεση μίας η περισσότερων εντολών ανάλογα τη συνθήκη, και μπορεί να αφορά πάνω από ένα σήματα.

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

VHDL – Multiplexer

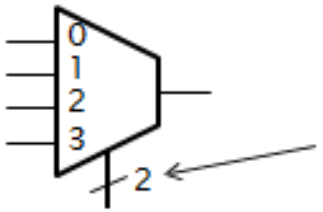
- Επιλέγει μεταξύ εισόδων δεδομένων με βάση μια είσοδο επιλογής

2-σε-1 πολυπλέκτης



sel	z
0	a_0
1	a_1

4-σε-1 πολυπλέκτης



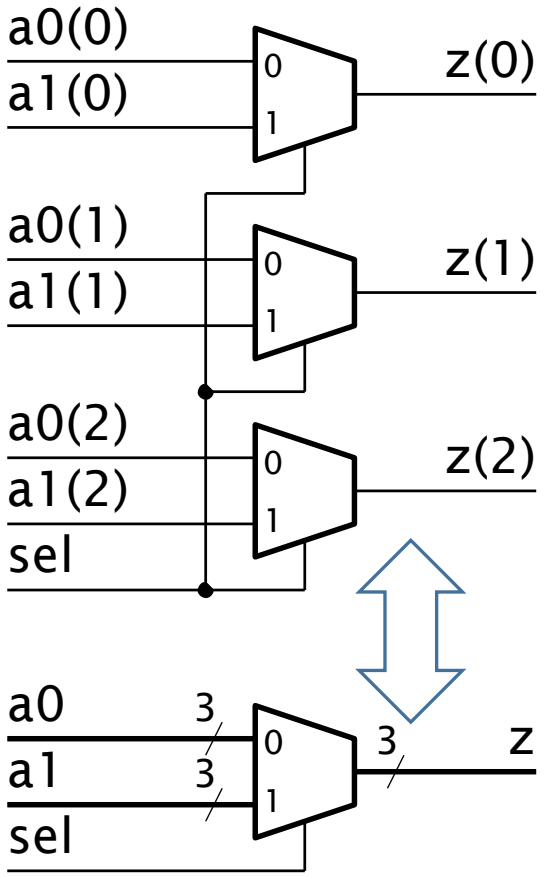
δύο bit επιλογής

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

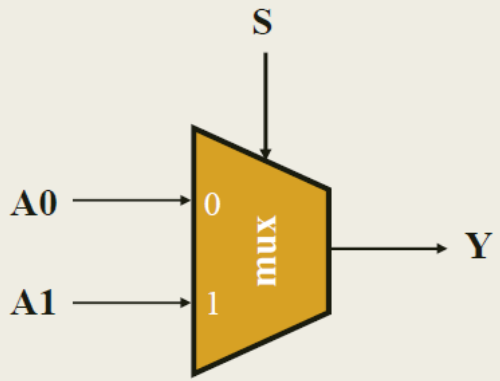
- Για να επιλέξουμε μεταξύ *N* κωδικών λέξεων των *m* bit
  - Συνδέουμε παράλληλα *m* πολυπλέκτες των *N* εισόδων



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)

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;

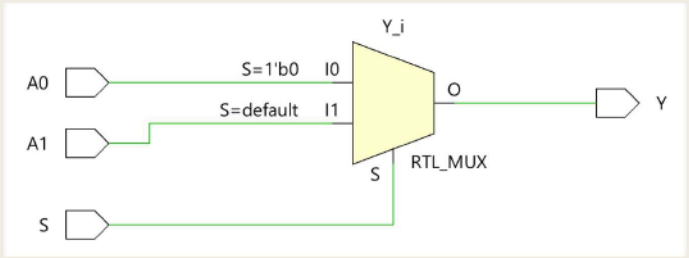
VHDL – Multiplexer

Architecture (2to1 - behavioral)

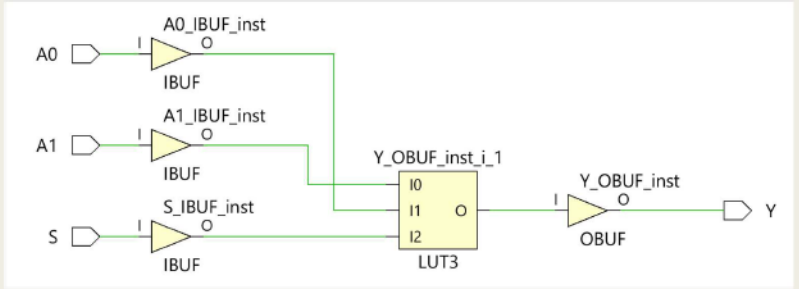
Η αρχιτεκτονική του πολυπλέκτη 2 σε 1 στη VHDL

Περιγραφή συμπεριφοράς

- Σχηματικό διάγραμμα RTL



- Σχηματικό διάγραμμα σε τεχνολογία FPGA

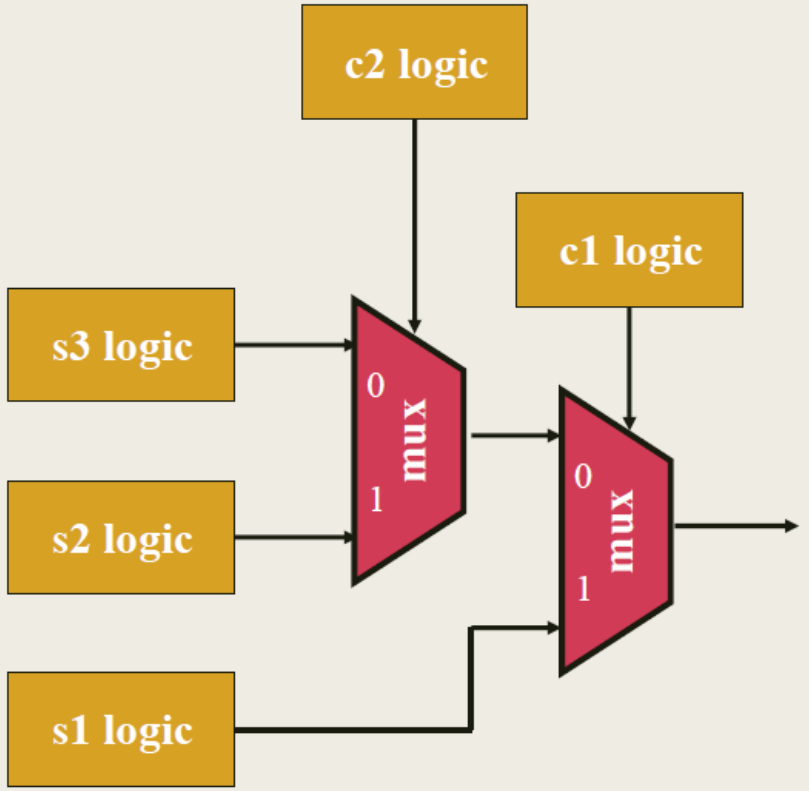


VHDL – Multiplexer

Υλοποίηση If

- Υλοποίηση εντολής IF με τη χρήση πολυπλεκτών 2 σε 1

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

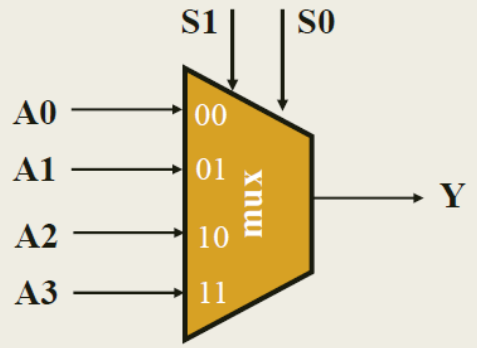


VHDL – Multiplexer

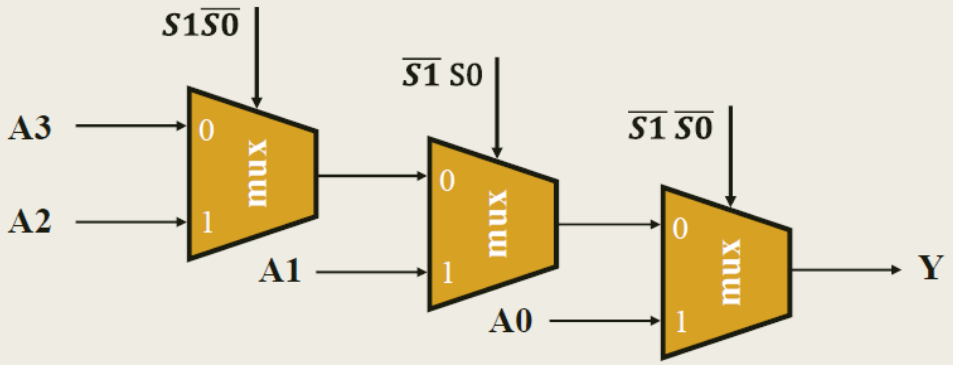
Entity (4to1)

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;
```

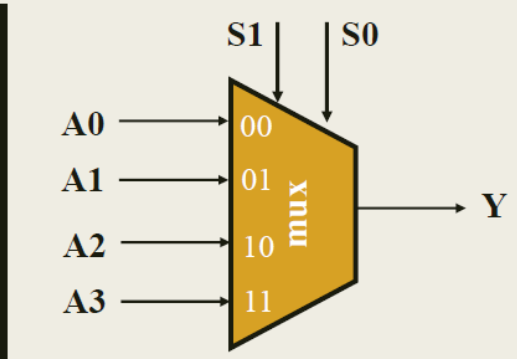
Στη λίστα ευαισθησίας συμπεριλαμβάνονται όλες
οι είσοδοι του συνδυαστικού κυκλώματος

VHDL – Multiplexer

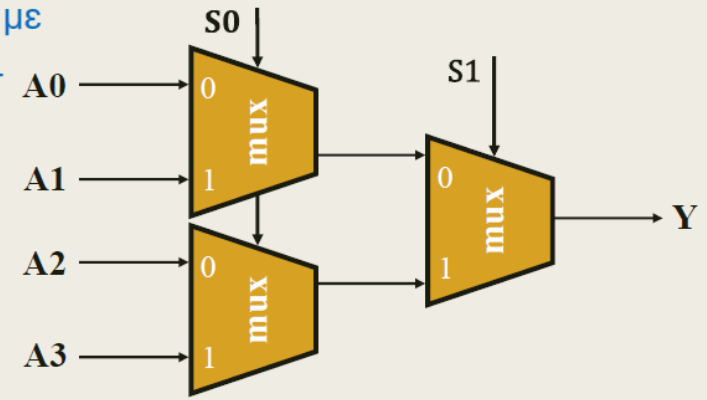
Entity (4to1)

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 – 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

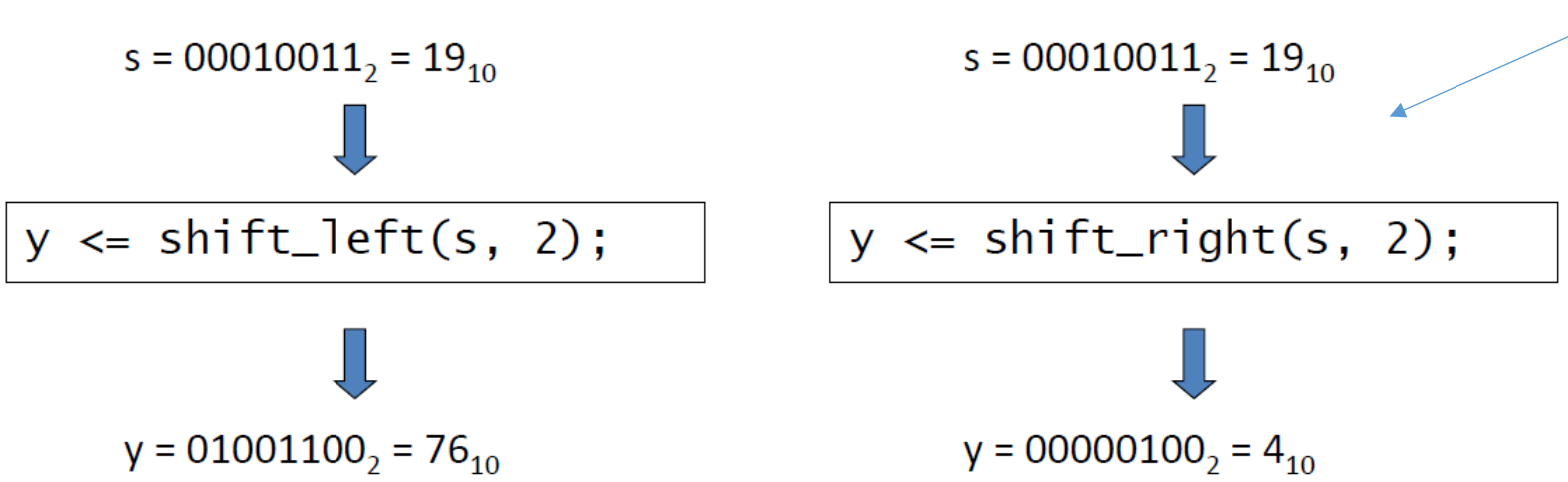
	Τελεστής	Σημασία
Υψηλότερη	`not`	NOT
	`*` `/` `mod` `rem`	MUL, DIV, MOD, REM
	`+` `-`	PLUS, MINUS
	`rol` `ror` `srl` `sll`	Περιστροφή, λογική ολίσθηση
	`<` `<=` `>` `>=`	Σχετική σύγκριση
	`=` `/=`	Σύγκριση ισότητας
Χαμηλότερη	`and` `or` `nand` `nor` `xor` `xnor`	Λογικές πράξεις (εκτελούνται από αριστερά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

VHDL – Τελεστές (operators)

Unsigned: 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

VHDL – Procedure

procedure procedure_name(input and output parameters) **is**

declarations

begin

sequential statement1;
sequential statement2;
.....

end procedure;

https://www.ics.uci.edu/~jmoorkan/vhdlref/procedur.html

VHDL – Procedure

```
Ctr_tb<='0';
for i in 0  to 2 loop
a_tb<=std_logic_vector(to_unsigned(i,a_tb'length));
   for j in 0 to 2 loop
      b_tb<=std_logic_vector(to_unsigned(j,a_tb'length));wait for 10ns;
   end loop;
end loop;

Ctr_tb<='1';
for i in 0  to 2 loop
   a_tb<=std_logic_vector(to_unsigned(i,a_tb'length));
   for j in 0 to 2 loop
      b_tb<=std_logic_vector(to_unsigned(j,a_tb'length));wait for 10ns;
   end loop ;
end loop;
```

```
procedure sim_test is
begin

for i in 0  to 2 loop

a_tb<=std_logic_vector(to_unsigned(i,a_tb'length));
   for j in 0 to 2 loop
      b_tb<=std_logic_vector(to_unsigned(j,a_tb'length));
      wait for 10ns;
end loop;

end loop;

end procedure;

Ctr_tb<='0';sim_test;
Ctr_tb<='1';sim_test;
```

VHDL – Procedure

```
procedure sim_test (min, max: in integer; step: in integer) is
variable a: integer;
begin
for i in min  to max loop
        a<=std_logic_vector(to_unsigned(i*step,a'length));
        for j in min to max loop
                b<=std_logic_vector(to_unsigned(j*step,a'length));wait for 10 ns;
        end loop;
end loop;

end procedure;

Ctr<='0';sim_test(0,2,5);
Ctr<='1';sim_test(0,2,5);
```

Όταν δεν αναφέρεται τύπος εννοείται variable. Σε αυτή την περίπτωση τα ορίσματα πρέπει να είναι σταθερές ή variables

a, b έχουν οριστεί εκτός procedure και εντός του process.

Όταν ο τύπος στα ορίσματα είναι signal τότε και στα ορίσματα στην κλήση της procedure θα πρέπει να υπάρχουν signal

VHDL – Function

Function function_name(input parameters) return_type **is**

declarations

begin

sequential statement1;
sequential statement2;
.....

end function_name;

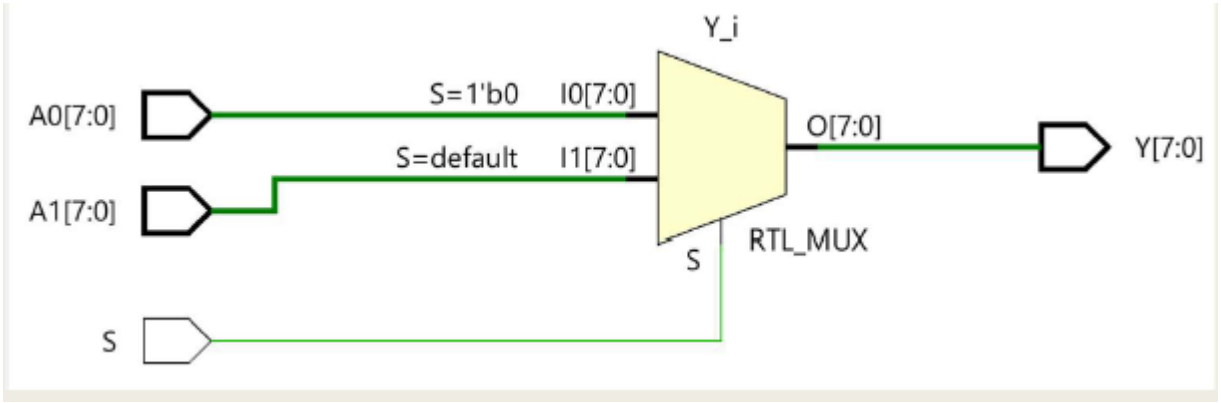
https://www.ics.uci.edu/~jmoorkan/vhdlref/function.html

VHDL – Function

```
function BOOL_TO_SL(X : boolean) return std_logic is

begin

if X then
   return '1';
else
   return '0';
end if;

end function BOOL_TO_SL;

std_logic_signal<=BOOL_TO_SL(X);
```

VHDL – Generic

```
entity MUX2in1_n is
generic (WIDTH : positive := 8); -- προεπιλεγμένη τιμή
port (      S: in STD_LOGIC;
            A0: in STD_LOGIC_VECTOR (WIDTH-1 downto 0);
            A1: in STD_LOGIC_VECTOR (WIDTH-1 downto 0);
            Y: out STD_LOGIC_VECTOR (WIDTH-1 downto 0));
end MUX2in1_n;
architecture BEHAVIORAL of MUX2in1_n is
begin
process (A0, A1, S)
begin
            if (S = '0') then
                        Y <= A0;
            else
                        Y <= A1;
            end if;
end process;
end BEHAVIORAL;
```



VHDL – Generic

Παραμετροποίηση του μεγέθους μίας αρτηρίας σε μία οντότητα με τη δήλωση της εντολής generic που ορίζει την σταθερά WIDTH

- Η δήλωση της εντολής generic γίνεται πριν από τη δήλωση των ports στην αρχή της οντότητας/component
- Η σταθερά WIDTH είναι θετικός ακέραιος (positive) και μπορεί να έχει προεπιλεγμένη τιμή
  - generic (WIDTH: positive := 8);
- Η σταθερά WIDTH μπορεί να χρησιμοποιηθεί κατά τη δήλωση των ports
  - STD_LOGIC_VECTOR (WIDTH-1 downto 0);
- Η τιμή της σταθεράς WIDTH μπορεί να παρακάμψει την προεπιλεγμένη τιμή με τη φράση generic map (συνδυάζεται με το port map)
  - generic map (WIDTH => 8)

https://electronics.stackexchange.com/questions/571759/when-to-set-defaults-for-vhdl-generics
https://fpgatutorial.com/vhdl-generic-generate/

VHDL – Constant

Μπορούν να δηλωθεί στην οντότητα, αρχιτεκτονική ή και σε process. Ανάλογα με το που δηλώνεται έχει και το ανάλογο visibilty

Δήλωση

constant constant_name **:** type **:=** value;

constant Size: Positive := 8;

Η τιμή της ΔΕΝ αλλάζει ποτέ

https://www.hdlworks.com/hdl_corner/vhdl_ref/VHDLContents/Constant.htm
https://peterfab.com/ref/vhdl/vhdl_renerta/source/vhd00022.htm

VHDL – Επαναλήψεις

Εντολή While

```
while condition loop
        sequential statements;
end loop;
```

Μόνο μέσα
σε Process, function,
…

```
process (A)

variable I : integer range 0 to 4
;
begin
Z <= "0000"; I := 0;
while (I <= 3) loop
if (A = I) then
        Z(I) <= '1';
end if;
I := I + 1;
end loop;

end process;
```

VHDL – Επαναλήψεις

Εντολή FOR

optional_label: for variable in range loop
sequential statements;
end loop;

Δεν χρειάζεται δήλωση.
Ο τύπος υπονοείται

```
for i in 0  to 2 loop
   a_tb<=std_logic_vector(to_signed(i,a_tb'length));
   for j in 0 to 2 loop
      b_tb<=std_logic_vector(to_signed(j,a_tb'length));wait for 10ns;
   end loop;
end loop;
```

Μόνο μέσα σε Process, function, …

wait μόνο σε simulation

Το for εκτελείται ακολουθιακά. Πρέπει να εκτελεστεί σε ένα κύκλο ρολογιού

VHDL – Επαναλήψεις

Εντολή FOR Generate

Εκτός Process

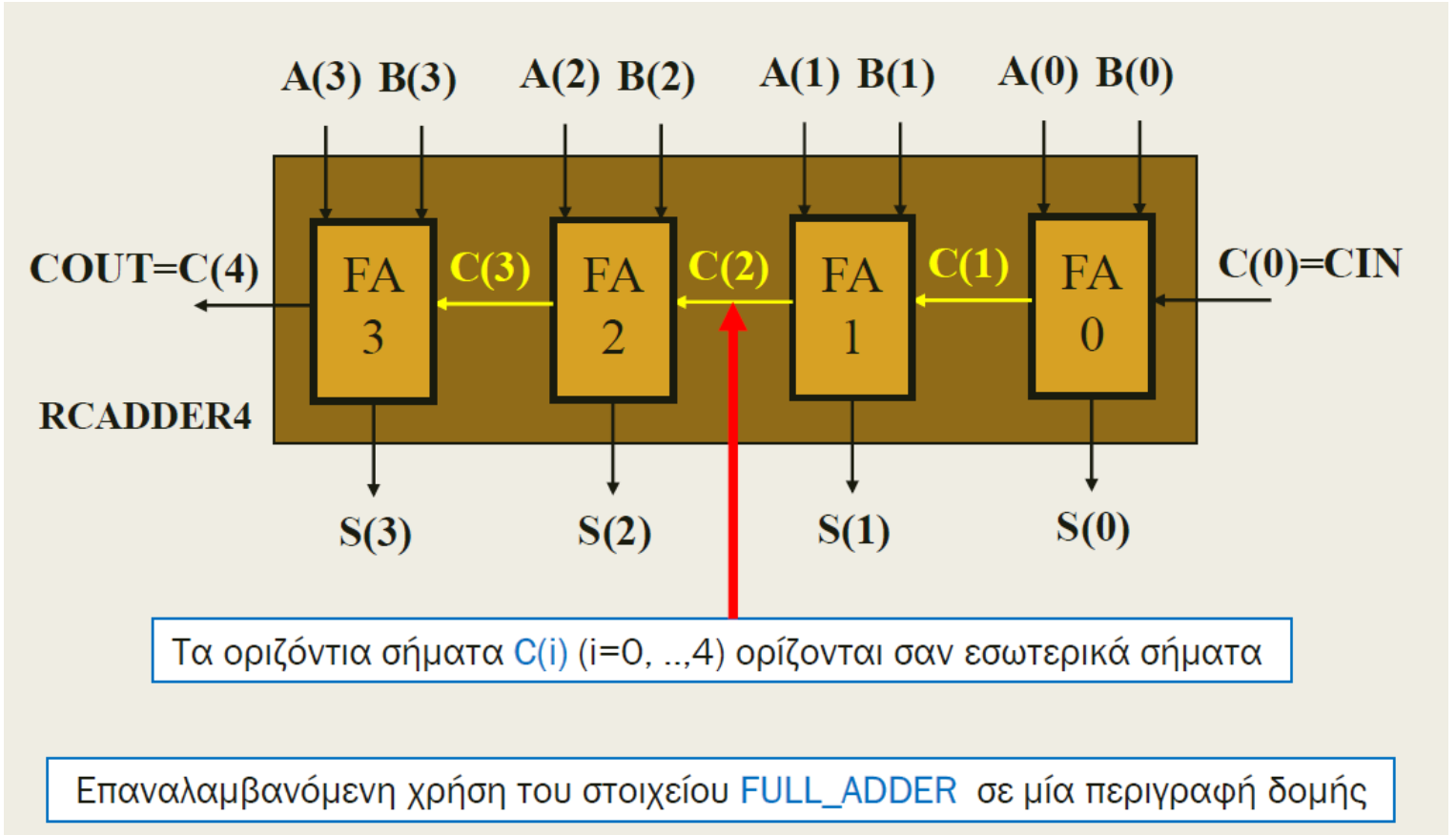
```
entity INV4 is
  port (
    X: in STD_LOGIC_VECTOR (0 to 3);
    Y: out STD_LOGIC_VECTOR (0 to 3));
end INV4;
architecture INV4_STR1 of INV4 is
  component INV
    port (O : out STD_LOGIC; I : in STD_LOGIC);
  end component;
begin
  U0: INV port map (Y(0), X(0));
  U1: INV port map (Y(1), X(1));
  U2: INV port map (Y(2), X(2));
  U3: INV port map (Y(3), X(3));
end INV4_STR1;
```

```
G1: for I in 0 to 3 generate
    U1: INV port map (Y(I), X(I));
  end generate G1;
```

optional_label: for variable in range generate
concurrent statements;
end loop;

VHDL – Structural architecture

Αθροιστής Ριπής Κρατουμένου των 4 bit (for generate)



VHDL – Structural architecture

Αθροιστής Ριπής Κρατουμένου των 4 bit

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FULL_ADDER is
    port (  A, B, CIN: in STD_LOGIC;
            SUM, CARRY: out STD_LOGIC);
end FULL_ADDER;

architecture FA_DATAFLOW2 of FULL_ADDER is
begin
    SUM <= A xor B xor CIN;
    CARRY <= (A and B) or (A and CIN) or (B and CIN);
end FA_DATAFLOW2;
```

1st Entity - Component

Στο ίδιο αρχείο δημιουργούνται 2 οντότητες
Οι επικεφαλίδες επαναλαμβάνονται.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity RCADDER4 is
    Port ( A : in  STD_LOGIC_VECTOR (3 downto 0);
           B : in  STD_LOGIC_VECTOR (3 downto 0);
           Cin : in  STD_LOGIC;
           S : out  STD_LOGIC_VECTOR (3 downto 0);
           Cout : out  STD_LOGIC);
end RCADDER4;

architecture Behavioral of RCADDER4 is

  signal C: STD_LOGIC_VECTOR (4 downto 0); -- οριζόντια σήματα
  component FULL_ADDER
    port (
    A, B, CIN: in STD_LOGIC;
    SUM, CARRY: out STD_LOGIC);
  end component;
begin
    C(0) <= CIN;     -- αρχικοποίηση
    G1: for I in 0 to 3 generate
        U1: FULL_ADDER port map (A(I),B(I),C(I),S(I),C(I+1));
        end generate G1;
    COUT <= C(4);    -- ανάθεση σε σήμα εξόδου

end Behavioral;
```

2nd Entity

VHDL – Structural architecture

ALU 8bit από 2 ALU των 4bit

Πρόσθεση

4 bit

```
  1001
 +1101
 -------
  0110   Carry=1
```

8 bit

1=Carry in στη 2η ALU

```
 0010 1001
+0101 1101
---------------
1000 0110   Carry_out=1 από 1η ALU
```

Carry_in='0' 1η ALU

Carry_out=0 2η ALU

Διπλασιασμός

4 bit

```
 1001 & '0'
 --------
  0010   Carry=1
```

8 bit

1=Carry in στη 2η ALU

```
0010 1001 & '0'
---------------
0101 0010     Carry_out=1 από 1η ALU
```

Carry_in=0 1η ALU

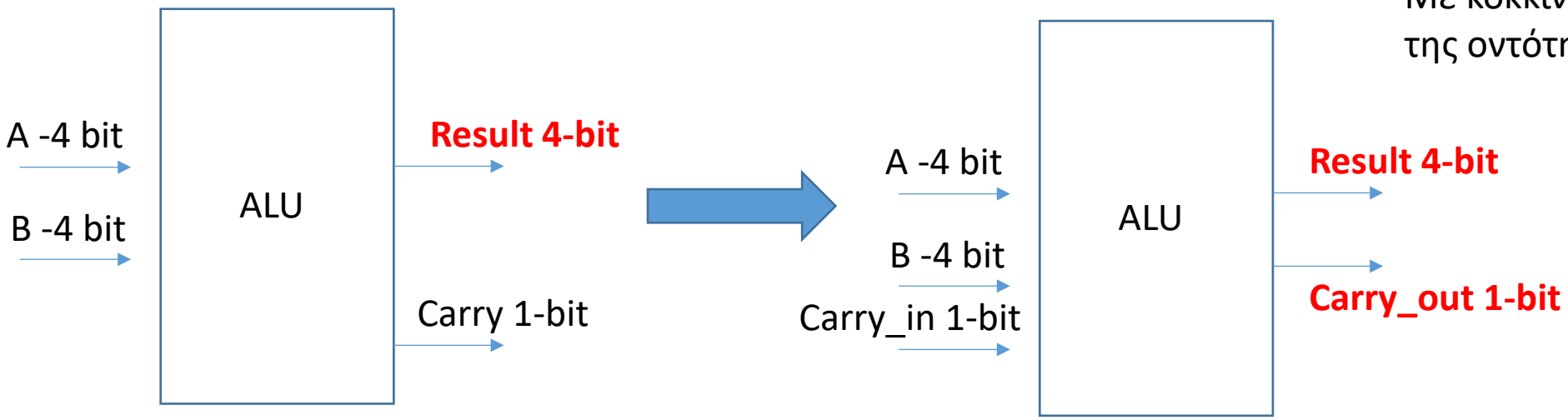
Carry_out=0 2η ALU

VHDL – Structural architecture

Αθροιστής

Για να μπορέσουμε να συνδυάσουμε 2 alu 4-bit θα πρέπει να μπορέσουμε να χειριστούμε το carry

Με κόκκινο οι έξοδοι της οντότητας

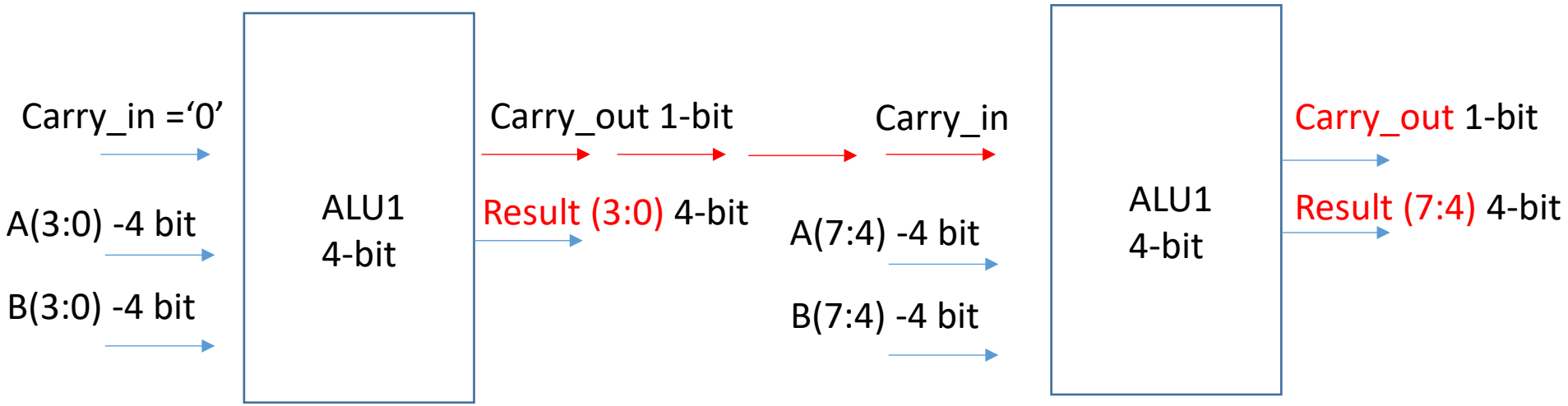


Πρώτα 4 bit (3 downto 0)

Επόμενα 4 bit (7 downto 4)

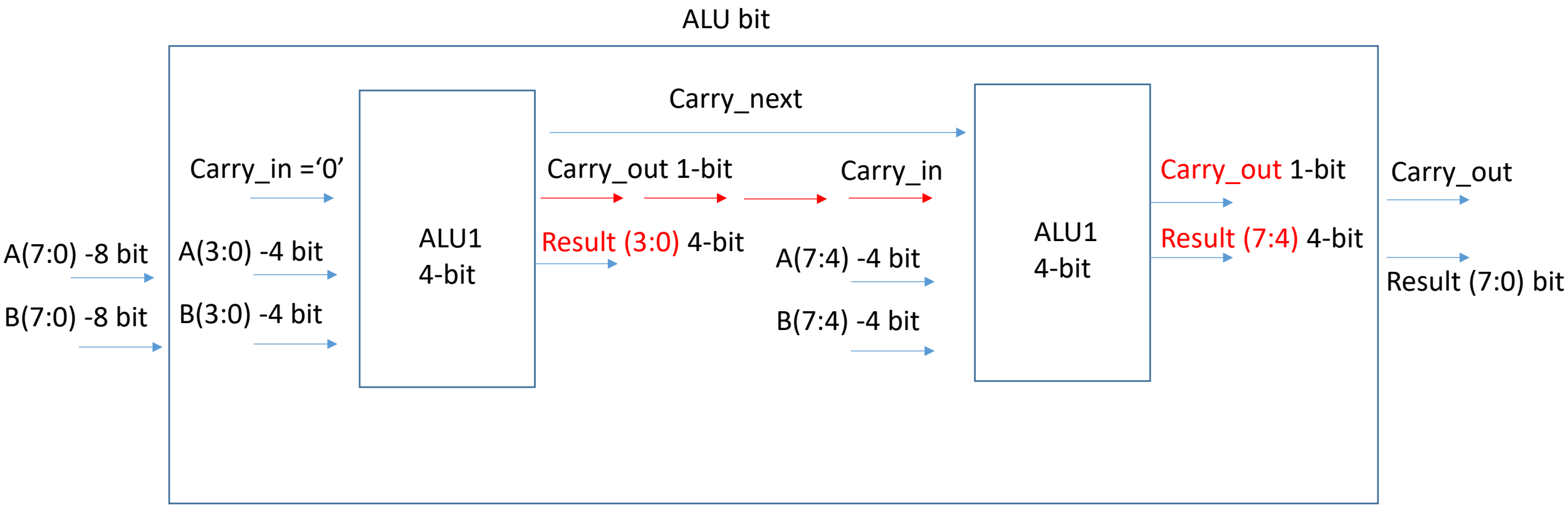
VHDL – Structural architecture

Αθροιστής



VHDL – Structural architecture

Αθροιστής 8bit από 2 αθροιστές των 4bit



VHDL – Structural architecture

Αθροιστής 8bit από 2 αθροιστές των 4bit (for generate)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;use IEEE.NUMERIC_STD.ALL;

entity ALU4bit is
   Port ( a       : in STD_LOGIC_VECTOR (3 downto 0);
          b       : in STD_LOGIC_VECTOR (3 downto 0);
          C_in  : in STD_LOGIC;
          Result   : out STD_LOGIC_VECTOR (3 downto 0);
          C_out    : out STD_LOGIC
       );
end entity ALU;

architecture Dataflow of ALU is
signal Result_temp  : unsigned (4 downto 0);
begin

Result_temp<=unsigned('0'&a)+unsigned('0'&b) +unsigned(“0000”&C_in);

Result<=std_logic_vector(Result_temp(3 downto 0));
C_out<=Result_temp(4);

end Dataflow;
```

```
library IEEE;use IEEE.STD_LOGIC_1164.ALL;use IEEE.NUMERIC_STD.ALL;

entity ALU8bit is
   Port ( a       : in STD_LOGIC_VECTOR (7 downto 0);
          b       : in STD_LOGIC_VECTOR (7 downto 0);
          C_in  : in STD_LOGIC;
          Result   : out STD_LOGIC_VECTOR (7 downto 0);
          C_out    : out STD_LOGIC
       );
end entity ALU;

architecture Dataflow of ALU is
signal Carry : unsigned (2 downto 0);
begin

Carry(0)<=C_in;
G1: for i=0 to 1 generate
       uut1: ALU4bit port map (a((i*4+3) downto (i*4)), b((i*4+3) downto (i*4), Carry(i),
             Result((i*4+3) downto (i*4)), Carry(I+1);
end generate G1;
C_out<=Carry(2);

end Dataflow;
```

Περίληψη

- Selected Assignment
- Conditional Statements
- Τελεστές
- Ολισθήσεις
- For/For Generate
- Procedure/Function
- Generic
- Διαβάζετε τις παραγράφους Appendix C από Ashenden και 4.2, 4.8, 4.9 (ΟΧΙ το κομμάτι της VERILOG) από το βιβλίο των Harris.