



dscal
DIGITAL SYSTEMS & COMPUTER ARCHITECTURE LABORATORY

Εργαστήριο Space Data Systems

**6ο Εργαστηριακό Μάθημα
2η Άσκηση (Εργαστήριο Αναγνώστηριου)**

Βασιλόπουλος Διονύσης

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

VHDL – Παράδειγμα

Άσκηση

Σχεδιάστε μία αριθμητική και λογική μονάδα (ALU). Στην είσοδο δέχεται 2 σήματα **μη προσημασμένων** αριθμών a και b , των 3 bit το καθένα, καθώς και ένα σήμα $Ctrl$ ενός bit. Η ALU για τιμή $Ctrl = '0'$ κάνει πρόσθεση ($a+b$) ενώ για τιμή $Ctrl = '1'$ κάνει διπλασιασμό του a ($a*2$). Στην έξοδο υπάρχει το σήμα $Result$ των 3 bit με το αποτέλεσμα της πράξης και ένα σήμα $Carry$ που έχει τιμή '1' σε περίπτωση που υπάρχει κρατούμενο/υπερχείλιση. Αναπτύξτε το αντίστοιχο κύκλωμα με VHDL και προγραμματίστε την FPGA κάρτα. Σας δίδεται ο ορισμός της οντότητας:

```
entity ALU is  
Port (  
  a      : in STD_LOGIC_VECTOR (2 downto 0);  
  b      : in STD_LOGIC_VECTOR (2 downto 0);  
  Ctrl   : in STD_LOGIC;  
  Result : out STD_LOGIC_VECTOR (2 downto 0);  
  Carry  : out STD_LOGIC      );  
end entity ALU;
```

Τι πόρους χρησιμοποιεί το κύκλωμα στο implementation. Ποια η καθυστέρηση διάδοσης και ποια η καθυστέρηση μόλυνσης του συνδυαστικού σας κυκλώματος.

VHDL – Παράδειγμα

Συσχέτιση port με FPGA

Είσοδοι	DIP Switch
Ctr	SW7
B[2]	SW5
B[1]	SW4
B[0]	SW3
A[2]	SW2
A[1]	SW1
A[0]	SW0

Έξοδοι	LED
Carry	LED7
Result[2]	LED2
Result[1]	LED1
Result[0]	LED0

VHDL – Παράδειγμα

Συσχέτιση port με FPGA-2 (Αρχείο constraints: ZedBoard.xdc)

```
# ZedBoard Pin Assignments
#####
# On-board Slide Switches #
#####
set_property -dict { PACKAGE_PIN F22  IOSTANDARD LVCMOS33 } [get_ports { a[0] }];
set_property -dict { PACKAGE_PIN G22  IOSTANDARD LVCMOS33 } [get_ports { a[1] }];
set_property -dict { PACKAGE_PIN H22  IOSTANDARD LVCMOS33 } [get_ports { a[2] }];
set_property -dict { PACKAGE_PIN F21  IOSTANDARD LVCMOS33 } [get_ports { b[0] }];
set_property -dict { PACKAGE_PIN H19  IOSTANDARD LVCMOS33 } [get_ports { b[1] }];
set_property -dict { PACKAGE_PIN H18  IOSTANDARD LVCMOS33 } [get_ports { b[2] }];
set_property -dict { PACKAGE_PIN M15  IOSTANDARD LVCMOS33 } [get_ports { Ctr }];

#####
# On-board led          #
#####
set_property -dict { PACKAGE_PIN T22  IOSTANDARD LVCMOS33 } [get_ports { Result[0] }];
set_property -dict { PACKAGE_PIN T21  IOSTANDARD LVCMOS33 } [get_ports { Result[1] }];
set_property -dict { PACKAGE_PIN U22  IOSTANDARD LVCMOS33 } [get_ports { Result[2] }];
set_property -dict { PACKAGE_PIN U14  IOSTANDARD LVCMOS33 } [get_ports { Carry }];
```

ΠΡΟΣΟΧΗ στις **διαφορές** με τον Κώδικα VHDL

1. Τα **σχόλια** εδώ είναι με **#**
2. Τα bit του vector εδώ είναι με **[]** αντί **()**.
3. Τα ονόματα των σημάτων, πρέπει να είναι **ΑΚΡΙΒΩΣ** ίδια με τη δήλωση στην οντότητα (**case sensitive**)
4. Κάρτα Zynq 7(000) ZC702 – Evaluation Board

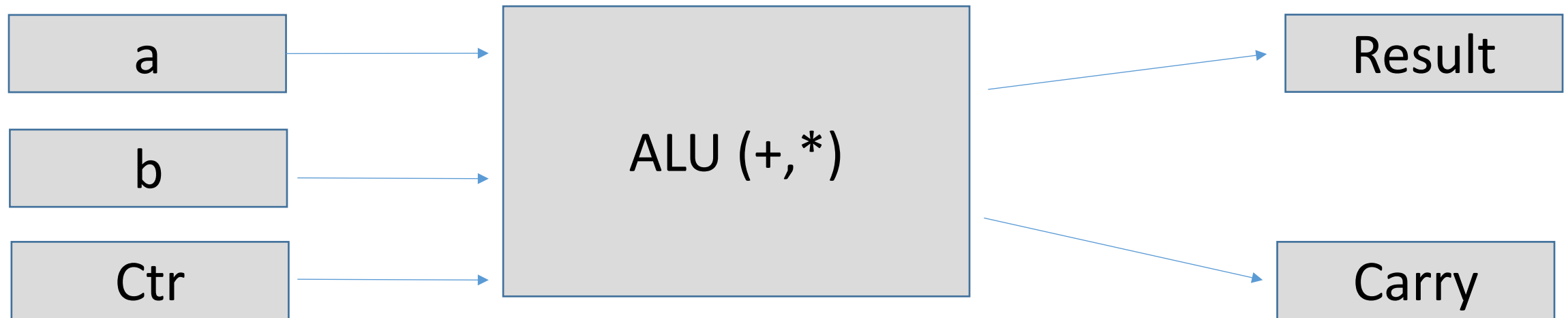
VHDL – Παράδειγμα

Βήματα Επίλυσης

1. Δημιουργία νέου project
2. Δημιουργία Entity – Εντοπισμός Input/Output του συστήματος
3. Εύρεση πίνακα αληθείας για κάθε έξοδο του συστήματος (αν χρειάζεται)
4. Δημιουργία Architecture – Θα έχετε τουλάχιστον τόσες εντολές όσες είναι και οι έξοδοι του συστήματος. Κάθε μία εντολή αντιστοιχεί σε μία έξοδο.
5. Δημιουργία RTL αναπαράστασης
6. Σύνθεση
7. Υλοποίηση
8. Προγραμματισμός κάρτας (Έγινε μόνο στο Εργαστήριο)
9. Προσομοίωση (Παρουσιάζεται μόνο στις διαφάνειες)

VHDL – Παράδειγμα

Απλοποιημένη μορφή – Είσοδοι/Εξοδοι



VHDL – Παράδειγμα

Βήμα 2. Περιγραφή Οντότητας

```
entity ALU is
  Port (
    a      : in STD_LOGIC_VECTOR (2 downto 0);
    b      : in STD_LOGIC_VECTOR (2 downto 0);
    Ctr    : in STD_LOGIC;
    Result : out STD_LOGIC_VECTOR (2 downto 0);
    Carry  : out STD_LOGIC
  );
end entity ALU;
```

VHDL – Παράδειγμα

Βήμα 3. Πίνακας Αληθείας κυκλώματος

ΔΕΝ ΧΡΕΙΑΖΕΤΑΙ ΓΙΑ ΤΗΝ ΠΕΡΙΠΤΩΣΗ ΜΑΣ

VHDL – Παράδειγμα

Βήμα 4. Περιγραφή Αρχιτεκτονικής – χωρίς process (α)

```
Result_temp<=unsigned('0'&a)+unsigned('0'&b) when Ctr='0'  
           else  
           unsigned(a&'0');
```

```
Result<=std_logic_vector(Result_temp(2 downto 0));
```

```
Carry<=Result_temp(3);
```

VHDL – Παράδειγμα

Βήμα 4. Περιγραφή Αρχιτεκτονικής – χωρίς process (β)

```
Result_temp<=resize(unsigned(a),Result_temp'length)+resize(unsigned(b),Result_temp'length) when Ctr='0'  
else  
resize(unsigned(a)*2,Result_temp'length);
```

```
Result<=std_logic_vector(Result_temp(2 downto 0));
```

```
Carry<=Result_temp(3);
```

VHDL – Παράδειγμα

Βήμα 4. Περιγραφή Αρχιτεκτονικής – Μια λύση με process (γ)

```
solution: process (a,b,Ctr) is
variable Result_var    : unsigned (3 downto 0);
begin

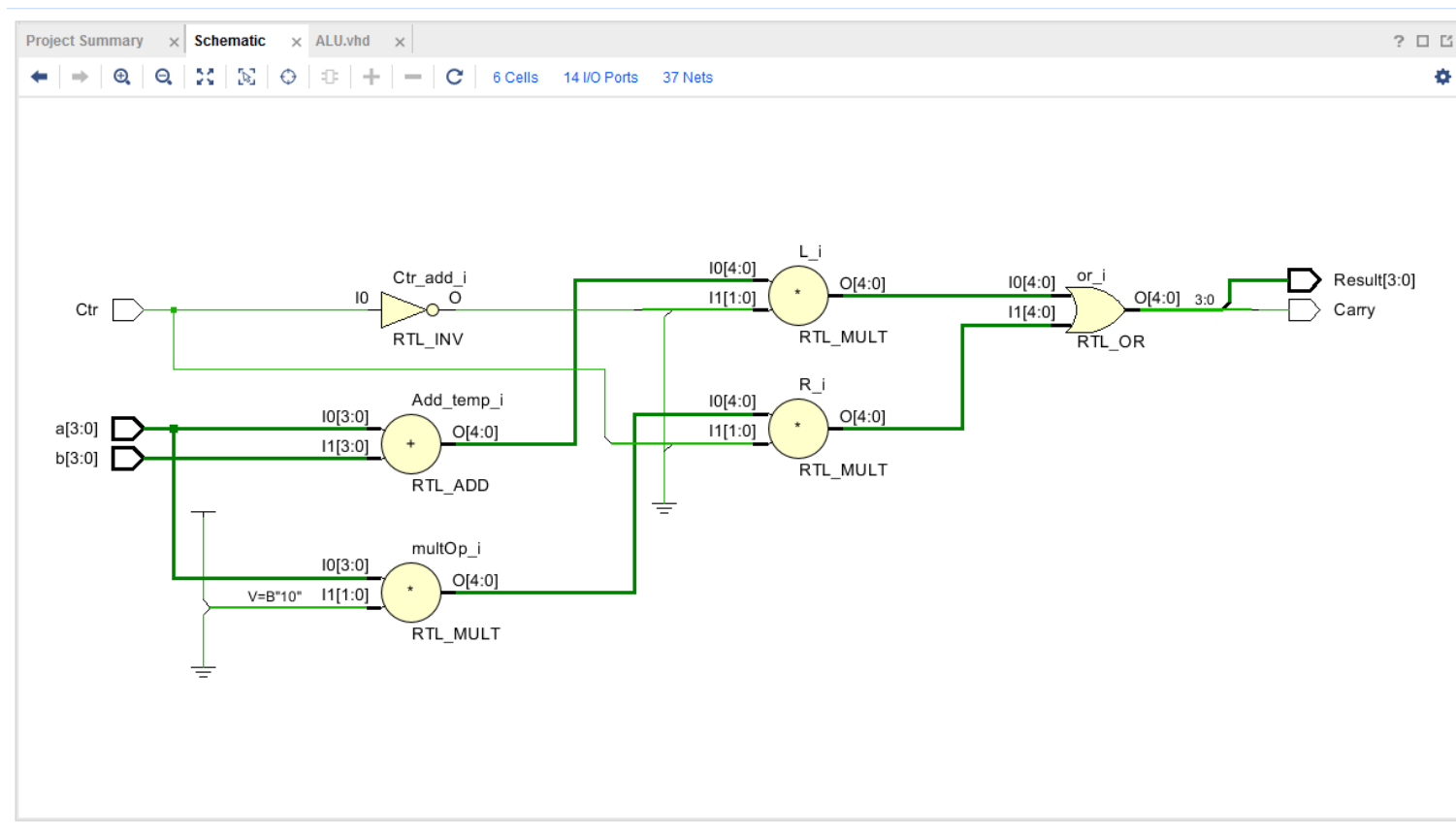
if Ctr='0' then
    Result_var:=unsigned('0'&a)+unsigned('0'&b);
elsif Ctr='1' then
    Result_var:=unsigned(a&'0');
else
    Result_var:=resize("0",Result_var'length);
end if; -- Ctr

Result<=std_logic_vector(Result_var(2 downto 0));
Carry<=Result_var(3);

end process solution;
```

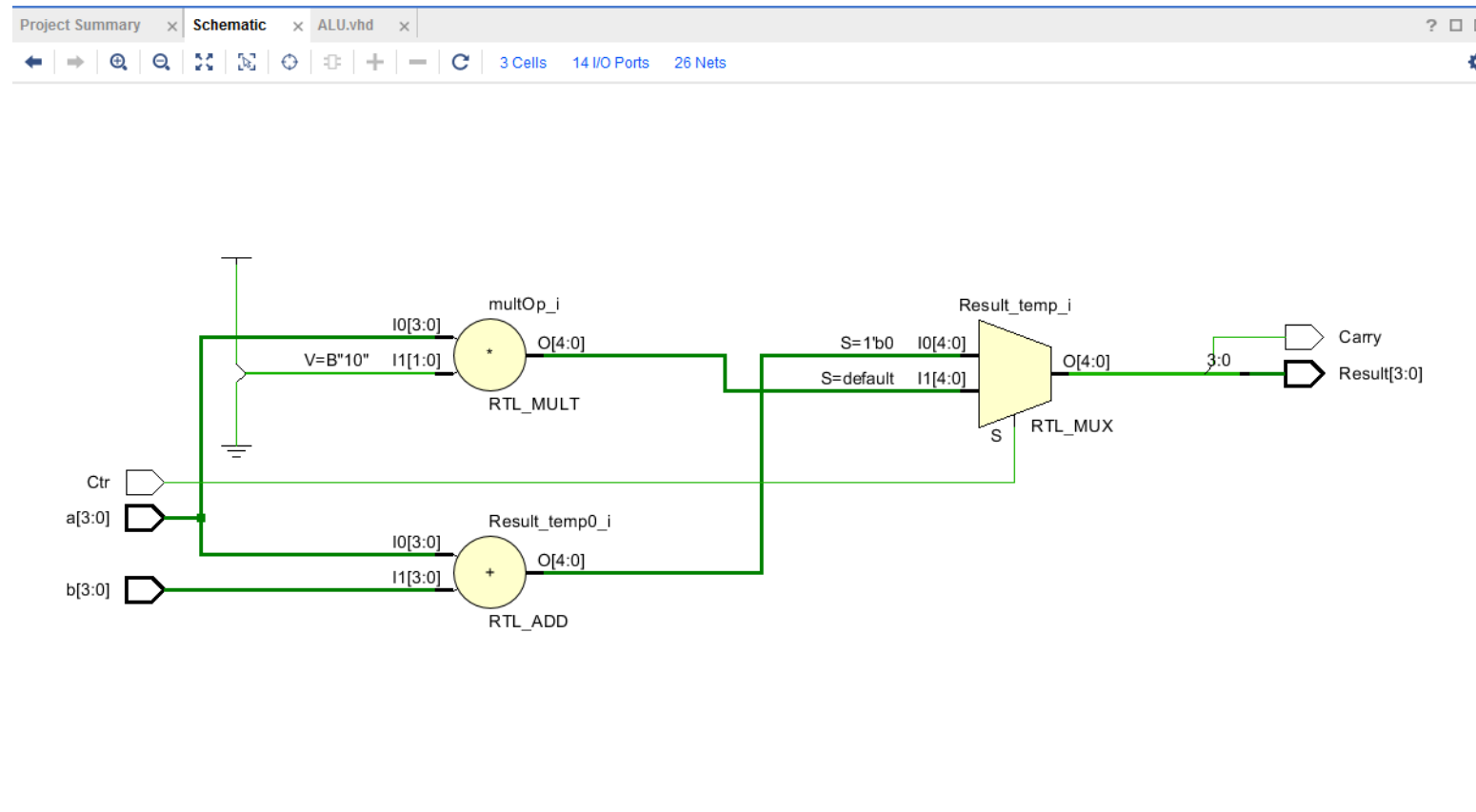
VHDL – Παράδειγμα

Βήμα 5. Λογικό κύκλωμα: Αναπαράσταση RTL (α)



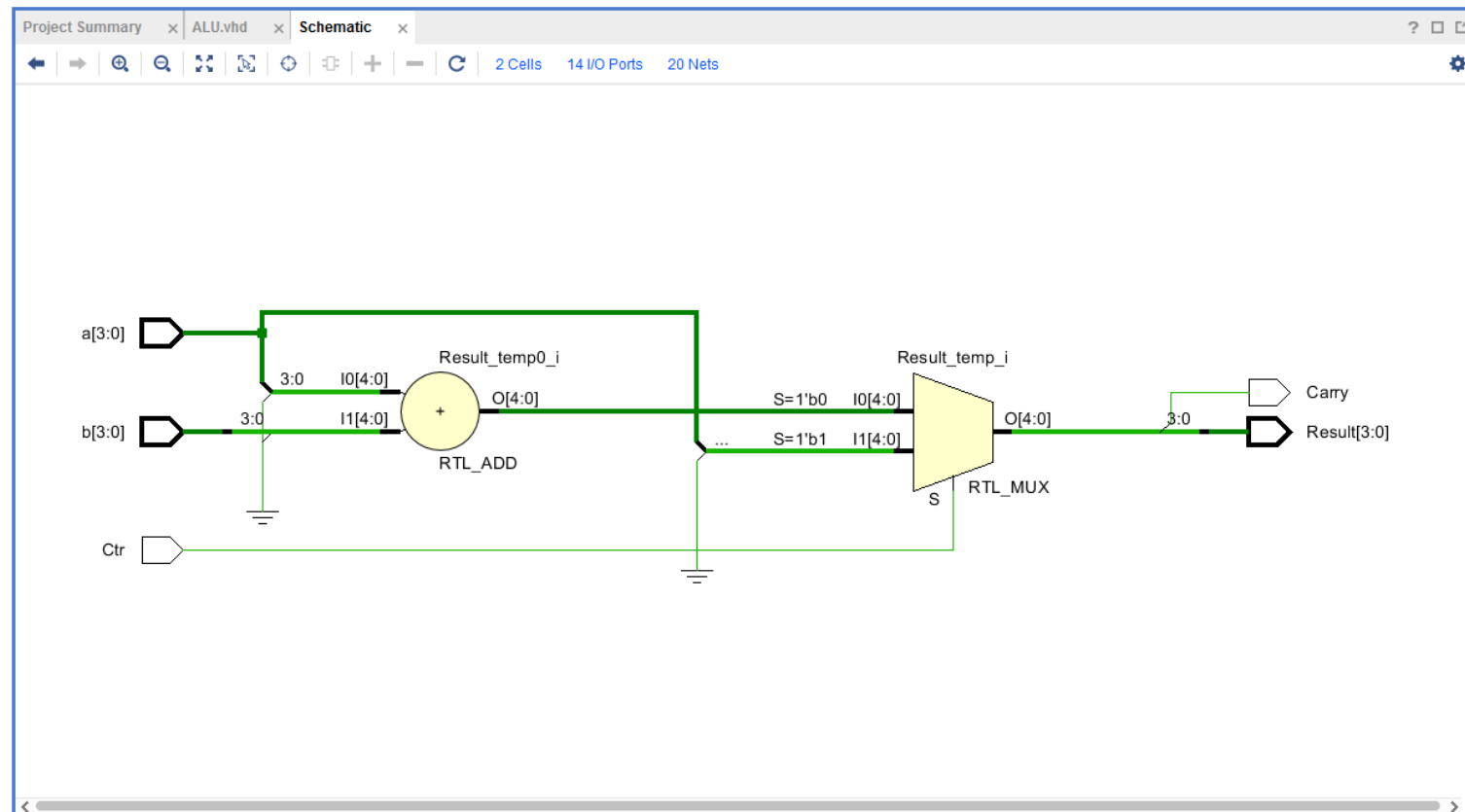
VHDL – Παράδειγμα

Βήμα 5. Λογικό κύκλωμα: Αναπαράσταση RTL (β)



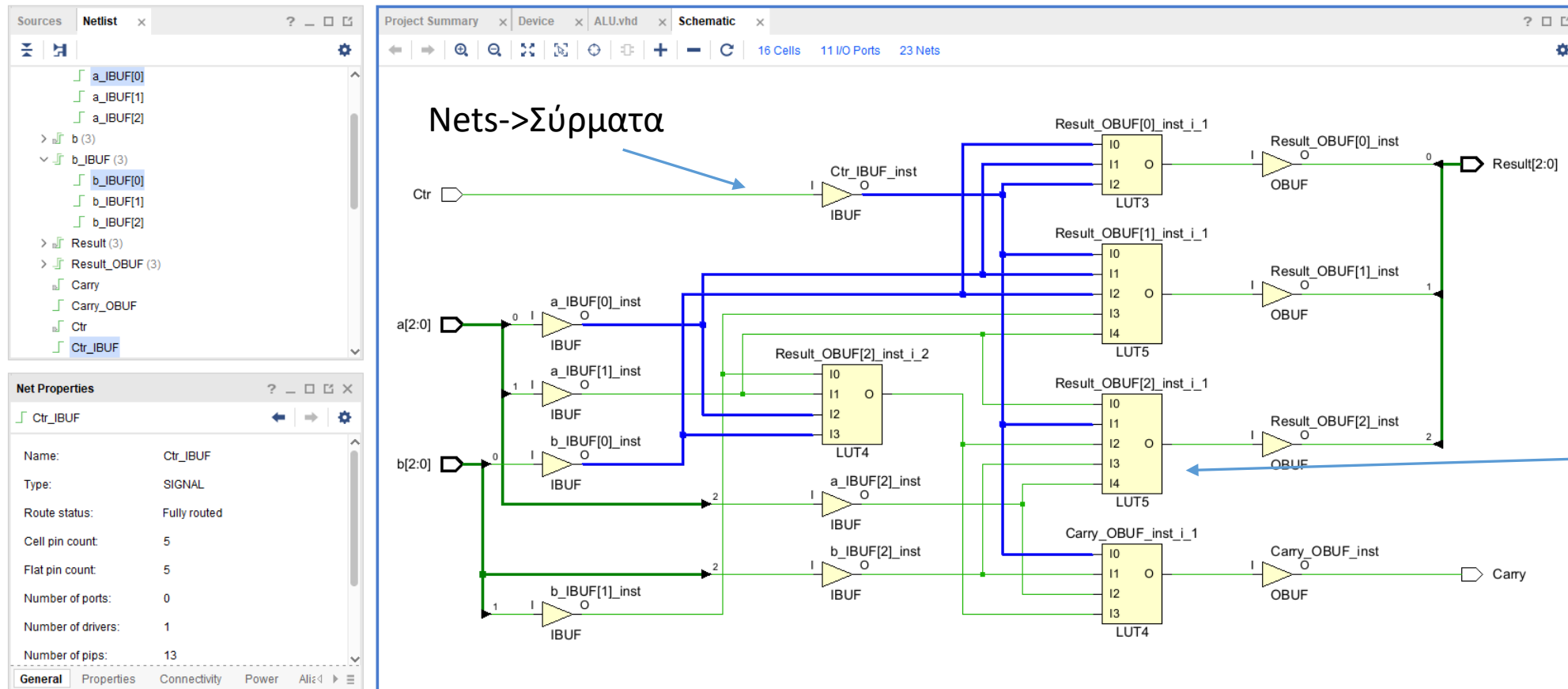
VHDL – Παράδειγμα

Βήμα 5. Λογικό κύκλωμα: Αναπαράσταση RTL (γ)



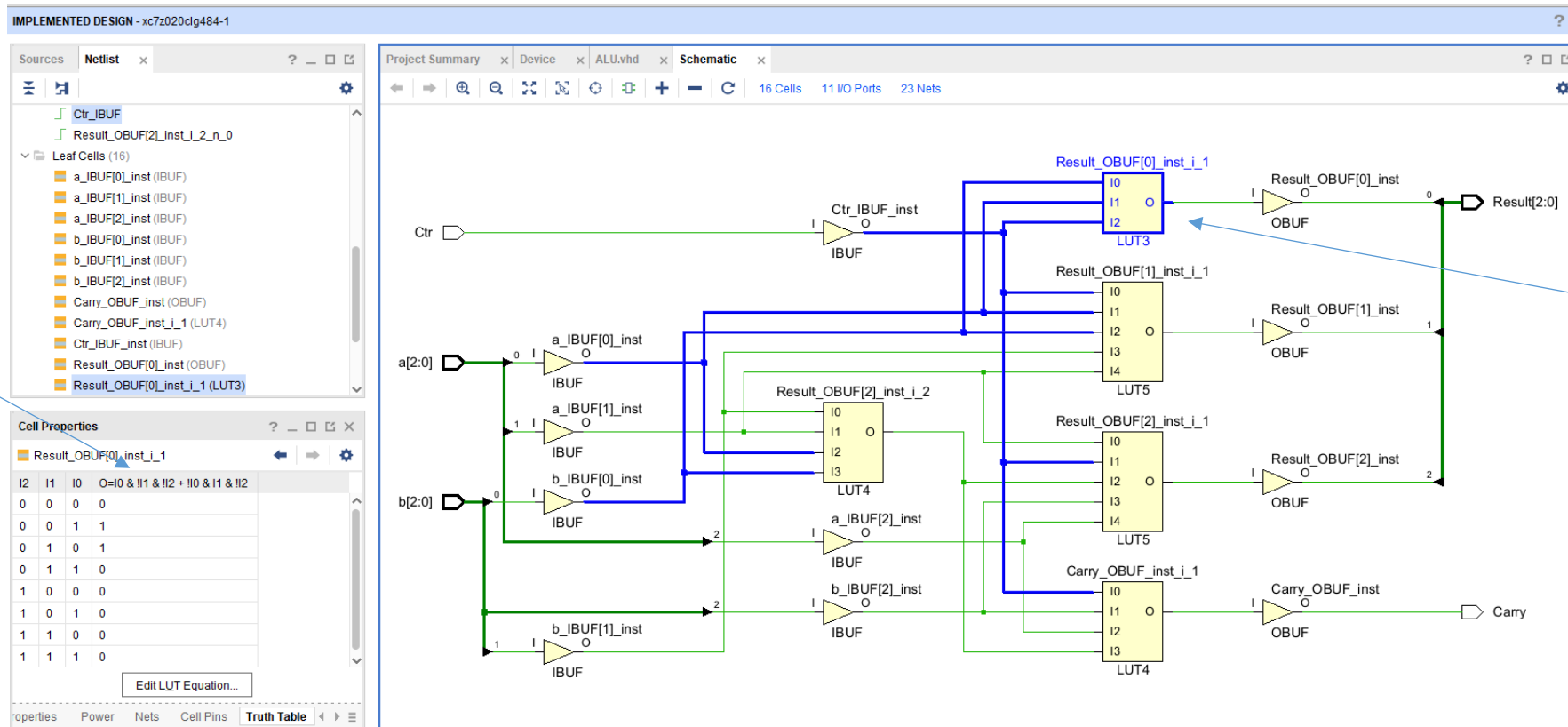
VHDL – Παράδειγμα

Βήμα 6. Λογικό κύκλωμα: Φάση Σύνθεσης – LUT(LookUp Table), υλοποιούν Πίνακες Αληθείας (προγραμματιζόμενα μέρη της κάρτας FPGA)



VHDL – Παράδειγμα

Βήμα 7α. Λογικό κύκλωμα: Φάση Υλοποίησης



Συνάρτηση LUT

Πίνακας Αληθείας

Στο LUT είσοδοι είναι $a(0)$, $b(0)$, $Ctrl$ και υπολογίζεται το ψηφίο $Result(0)$.

$Ctrl \Rightarrow I2$
 $a(0) \Rightarrow I1$
 $b(0) \Rightarrow I0$

VHDL – Παράδειγμα

Βήμα 7b. Λογικό κύκλωμα: Φάση Υλοποίησης – Modified Truth Table (αντιστοιχίες a(0), b(0), Ctr \leftrightarrow I του LUT

Πίνακας Αληθείας
True Table
για Result(0)



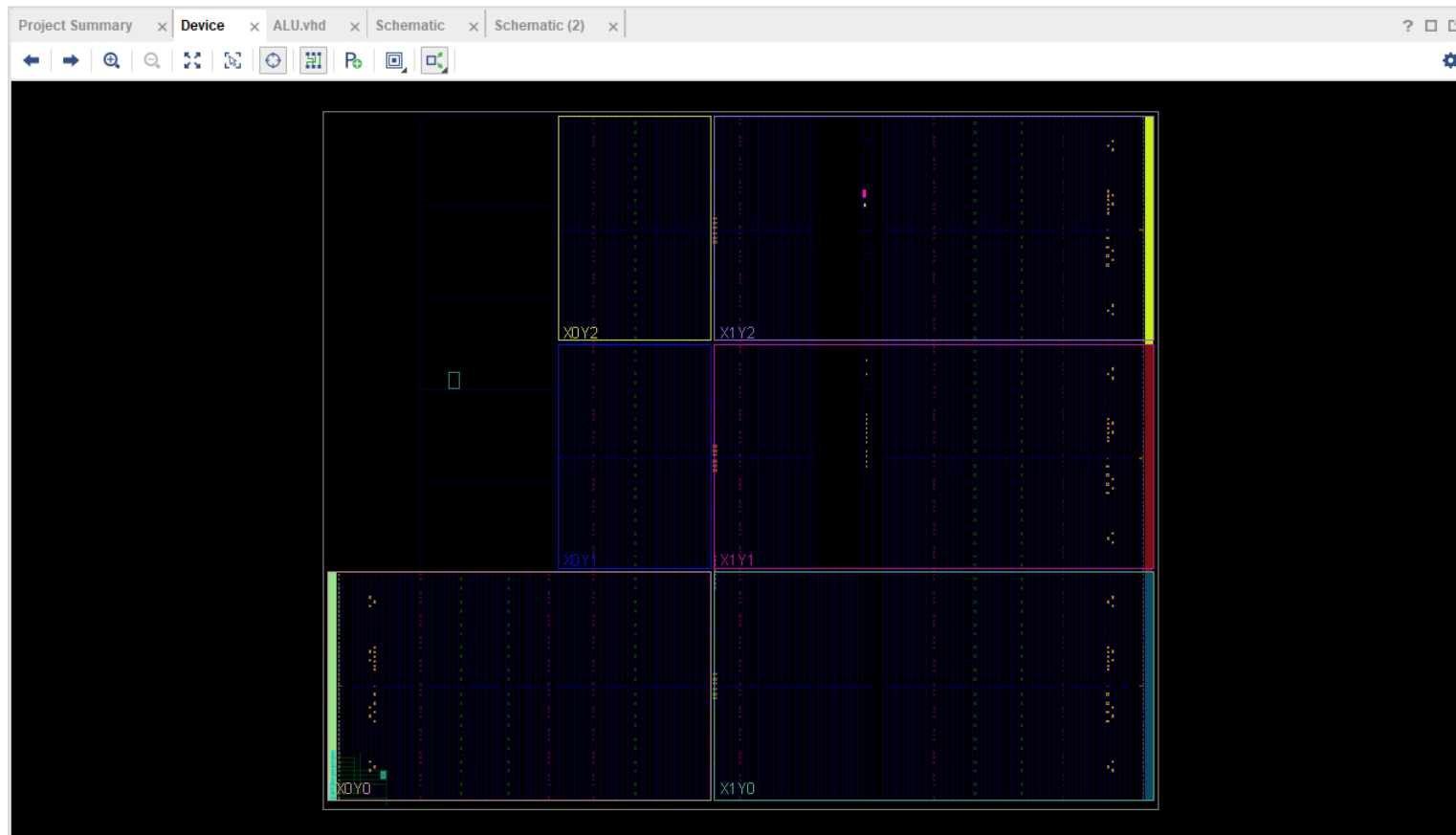
Είσοδοι			Έξοδοι
Ctr/I2	a(0)/I1	b(0)/I0	Result(0)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Ο αρχικός Πίνακας Αληθείας και ο αντίστοιχος του LUT είναι ίδιοι

Result(0) <= (a(0) xor b(0)) and not Ctr
ή στο LUT
 $O = I0 \& !I1 \& !I2 + !I0 \& I1 \& !I2$

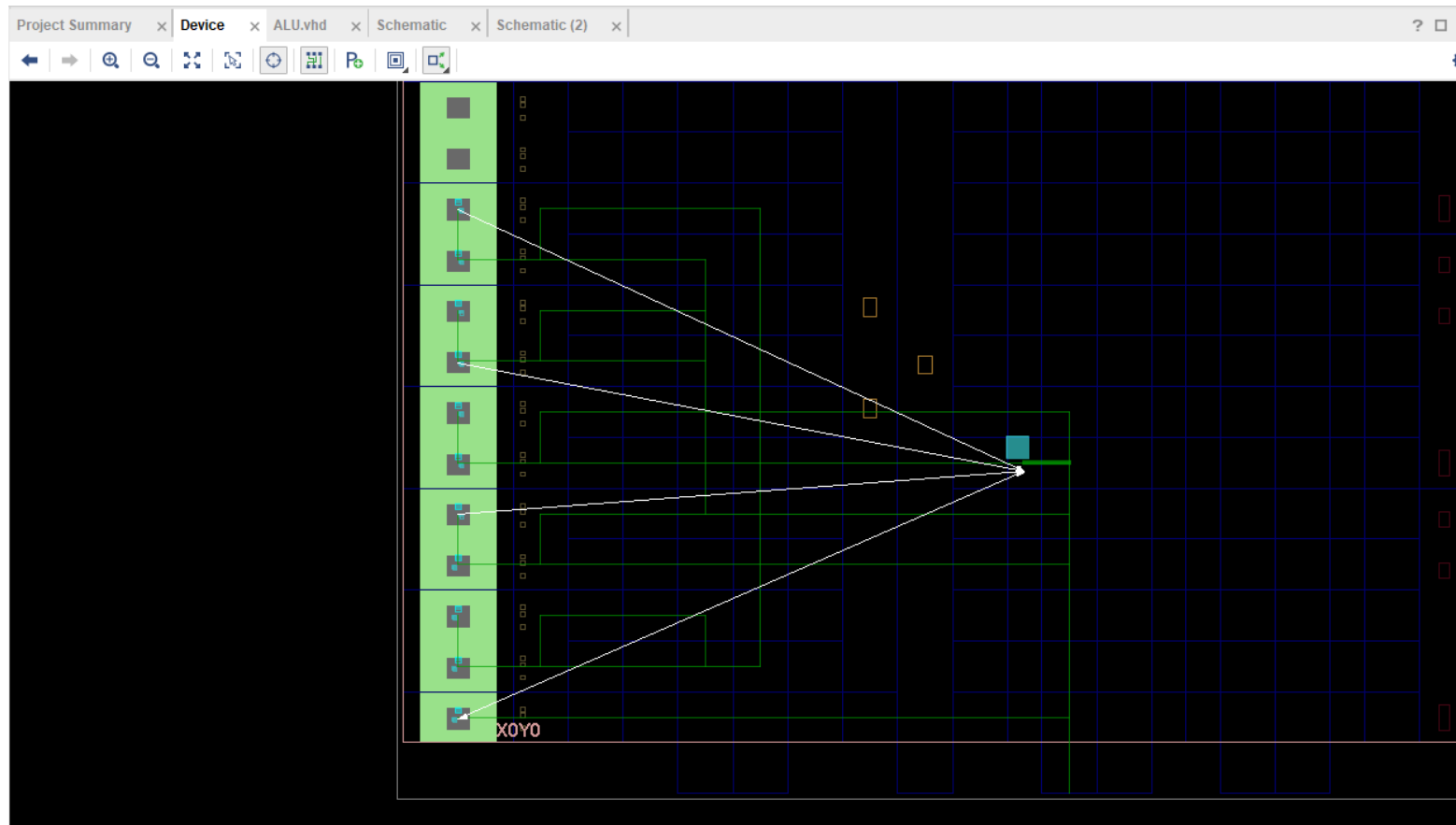
VHDL – Παράδειγμα

Βήμα 7c. Λογικό κύκλωμα: Φάση Υλοποίησης – Design



VHDL – Παράδειγμα

Βήμα 7d. Λογικό κύκλωμα: Φάση Υλοποίησης – Design: LUT on Board (LUT3 – Result(0))



Click on Leaf Cell
Result_OBUF[0]_inst_i_1

VHDL – Παράδειγμα

Βήμα 7ε. Λογικό κύκλωμα: Φάση Υλοποίησης – Design: LUT on Board (LUT3 – Result(0))

Netlist

- Ctrl_IBUF
- Result_OBUF[2]_inst_i_2_n_0
- Leaf Cells (16)
 - a_IBUF[0]_inst (IBUF)
 - a_IBUF[1]_inst (IBUF)
 - a_IBUF[2]_inst (IBUF)
 - b_IBUF[0]_inst (IBUF)
 - b_IBUF[1]_inst (IBUF)
 - b_IBUF[2]_inst (IBUF)
 - Carry_OBUF_inst (OBUF)
 - Carry_OBUF_inst_L_1 (LUT4)
 - Ctrl_IBUF_inst (IBUF)
 - Result_OBUF[0]_inst (OBUF)
 - Result_OBUF[0]_inst_i_1 (LUT3)

Cell Properties

Result_OBUF[0]_inst_i_1

I2	I1	I0	O=I0 & I1 & I1 & I2 + I0 & I1 & I2
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Edit LUT Equation...

Schematic (2)

Result_OBUF[0]_inst_i_1

WA8 WA7 WA6 WA5 WA4 WA3 WA2 WA1
DI2 DI1
CLK CLK
WE WE
A6 A5 A4 A3 A2 A1

O6 O5 O5

DMCS31
DI
CI

C6 C5 C4 C3 C2 C1
CX
CLK_B
CLK

VHDL – Παράδειγμα

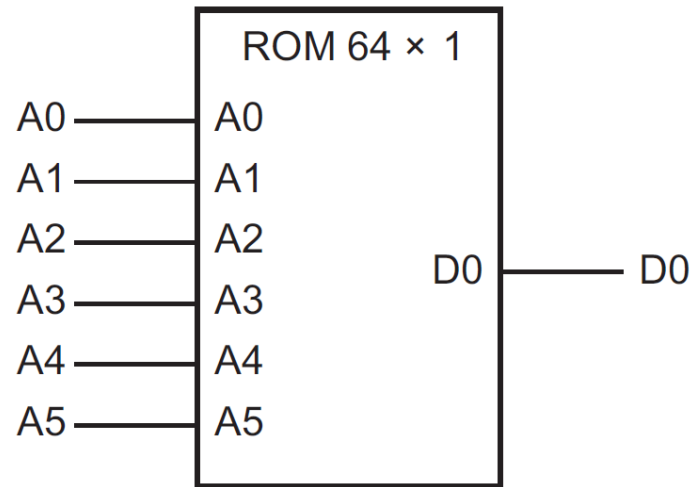
LUT

- **Πίνακες Αναζήτησης (Lookup Tables – LUTs):** μικρές μνήμες μόνο για ανάγνωση
 - Χρησιμοποιούνται για την υλοποίηση λογικών συναρτήσεων σε κυκλώματα FPGA
- Ένα LUT $2^n \times 1$ μπορεί να υλοποιήσει οποιαδήποτε λογική συνάρτηση μίας εξόδου η οποία έχει έως και n εισόδους
 - Το n συνήθως παίρνει μια τιμή από το 4 έως το 6
- Τα εργαλεία σύνθεσης διασπούν λογικές συναρτήσεις που έχουν περισσότερες από 4–6 εισόδους σε ένα σύνολο μικρότερων συναρτήσεων
- Κάθε συνάρτηση υλοποιείται σε έναν διαθέσιμο πίνακα αναζήτησης

VHDL – Παράδειγμα

LUT

- Στη σειρά 7 της εταιρείας Xilinx, έχουν 6 εισόδους
- Άρα, μπορούμε να θεωρήσουμε ότι κάθε πίνακας αναζήτησης είναι μια μνήμη ROM διαστάσεων 64×1 bit



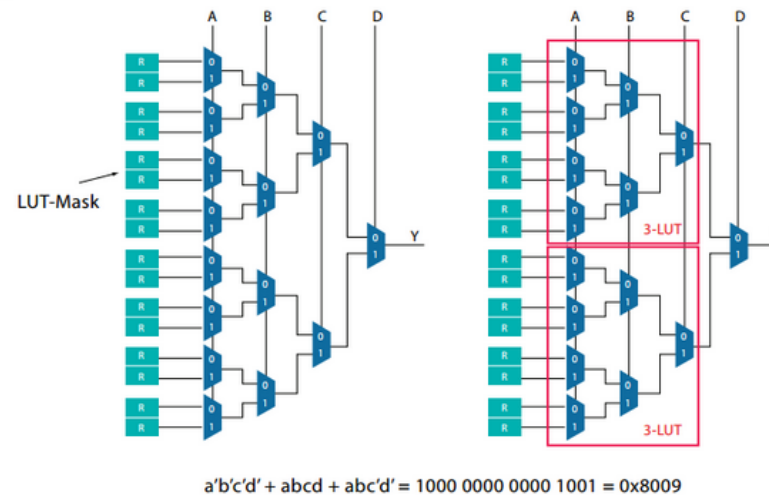
VHDL – Παράδειγμα

LUT

Building Look-up Tables (LUTs)

An overview of how LUTs are built helps describe the key innovations in the ALM. A LUT is typically built out of SRAM bits to hold the configuration memory (CRAM) LUT-mask and a set of multiplexers to select the bit of CRAM that is to drive the output. To implement a k-input LUT (k-LUT)—a LUT that can implement any function of k inputs— 2^k SRAM bits and a $2^k:1$ multiplexer are needed. Figure 2 shows a 4-LUT, which consists of 16 bits of SRAM and a 16:1 multiplexer implemented as a tree of 2:1 multiplexers. The 4-LUT can implement any function of 4 inputs (A, B, C, D) by setting the appropriate value in the LUT-mask. To simplify the 4-LUT in Figure 2, it can also be built from two 3-LUTs connected by a 2:1 multiplexer.


Figure 2. Building a LUT



Υλοποίηση LUT

Share Cite Follow

answered May 8, 2015 at 8:04

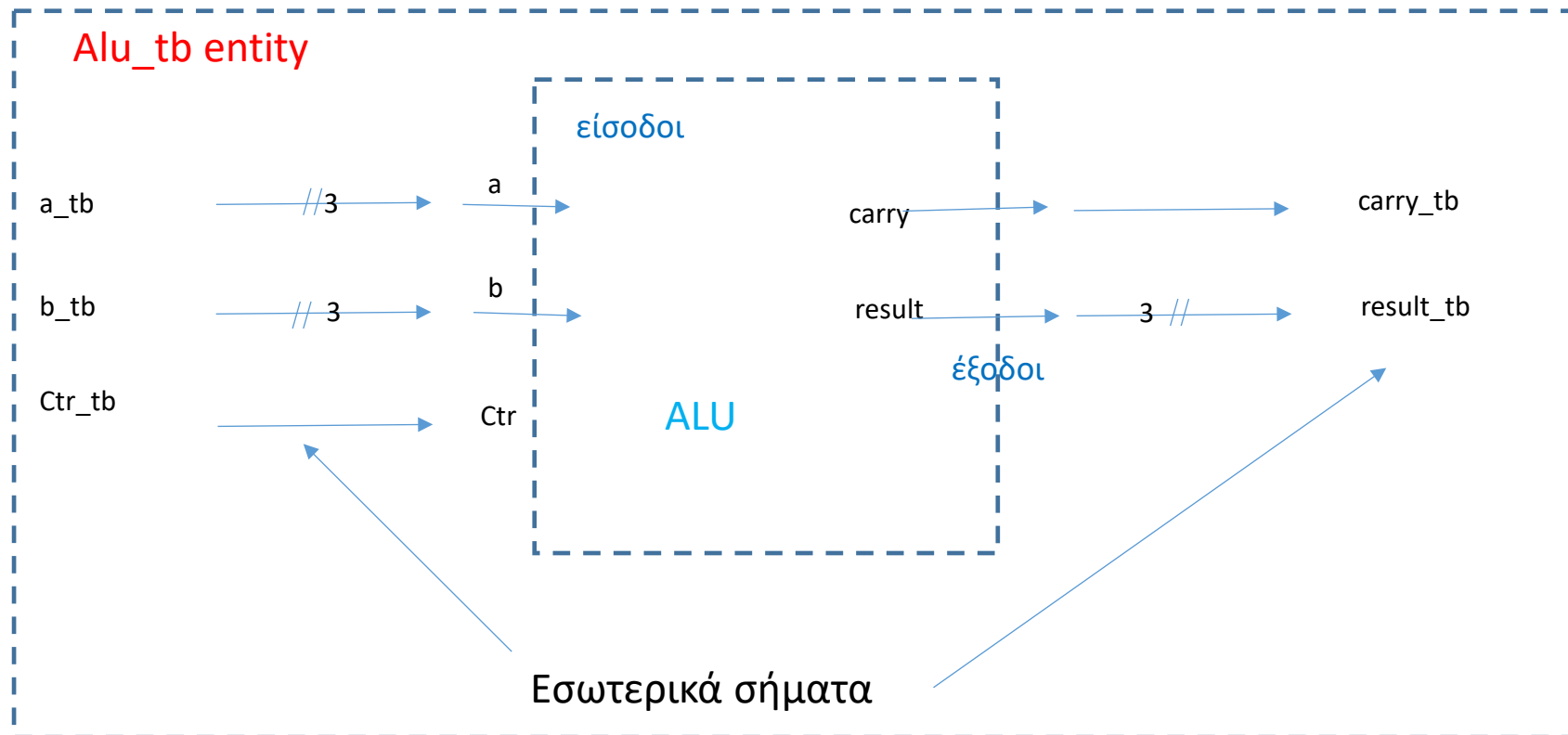
 apalorohapa
8,329 ● 2 ● 28 ● 39

Add a comment

Εργαστήριο Space Data Systems 2025-26 Δ.Βασιλόπουλος

VHDL – Παράδειγμα

Simulation



VHDL – Παράδειγμα

Simulation - κώδικας

```
entity ALU_tb is
-- Port ( );
end ALU_tb;

architecture Behavioral of ALU_tb is
component ALU is
Port ( a      : in STD_LOGIC_VECTOR (2 downto 0);
      b      : in STD_LOGIC_VECTOR (2 downto 0);
      Ctr    : in STD_LOGIC;
      Result  : out STD_LOGIC_VECTOR (2 downto 0);
      Carry  : out STD_LOGIC
);
end component ALU;

signal a_tb      : STD_LOGIC_VECTOR (2 downto 0);
signal b_tb      : STD_LOGIC_VECTOR (2 downto 0);
signal Ctr_tb    : STD_LOGIC;
signal Result_tb : STD_LOGIC_VECTOR (2 downto 0);
signal Carry_tb  : STD_LOGIC;

Begin
 uut: ALU port map (a_tb,b_tb,Ctr_tb,Result_tb, Carry_tb);

test: process is
Begin

Ctr_tb<='0';
for i in 0 to 7 loop
  a_tb<=std_logic_vector(to_unsigned(i,a_tb'length));
  for j in 0 to 7 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 7 loop
  a_tb<=std_logic_vector(to_unsigned(i,a_tb'length));
  for j in 0 to 7 loop
    b_tb<=std_logic_vector(to_unsigned(j,a_tb'length));wait for 10ns;
  end loop;
end loop;

end process test;
end architecture Behavioral;
```

VHDL – Παράδειγμα

Simulation – κώδικας με procedure

```
entity ALU_tb is
-- Port ( );
end ALU_tb;

architecture Behavioral of ALU_tb is
component ALU is
Port ( a      : in STD_LOGIC_VECTOR (3 downto 0);
      b      : in STD_LOGIC_VECTOR (3 downto 0);
      Ctr    : in STD_LOGIC;
      Result : out STD_LOGIC_VECTOR (3 downto 0);
      Carry  : out STD_LOGIC
);
end component ALU;

signal a_tb      : STD_LOGIC_VECTOR (3 downto 0);
signal b_tb      : STD_LOGIC_VECTOR (3 downto 0);
signal Ctr_tb    : STD_LOGIC;
signal Result_tb : STD_LOGIC_VECTOR (3 downto 0);
signal Carry_tb  : STD_LOGIC;

Begin
 uut: ALU port map (a_tb,b_tb,Ctr_tb,Result_tb, Carry_tb);

test: process is

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;

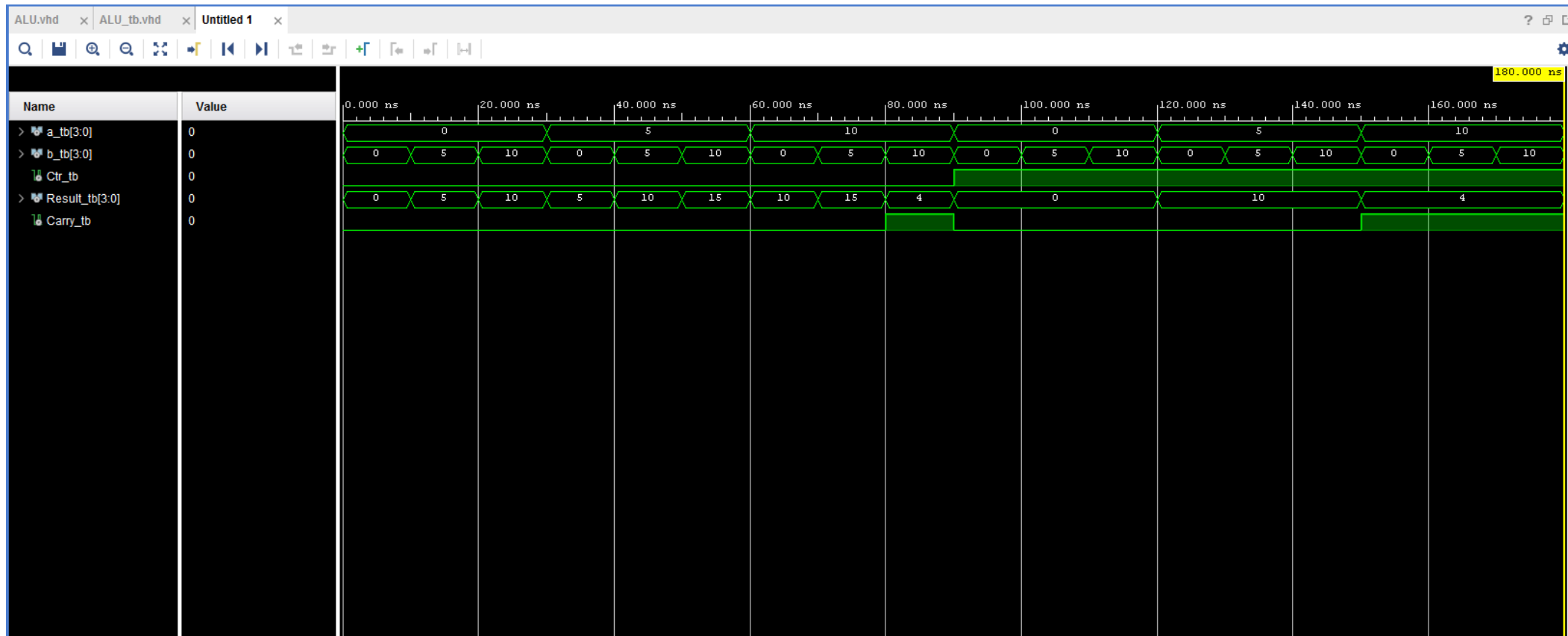
begin

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

end process test;
end architecture Behavioral;
```

VHDL – Παράδειγμα

Simulation – Χρονοσειρά RTL



VHDL – Παράδειγμα

Simulation – Χρονοσειρά post Synthesis



VHDL – Παράδειγμα

Περίληψη

Ανάπτυξη βήμα-βήμα μιας απλής εφαρμογής στο Vivado

RTL->Synthesis->Implementation

Προσομοίωση

LUT

Υλοποίηση LUT

<https://electronics.stackexchange.com/questions/169532/what-is-an-lut-in-fpga>

<https://hardwarebee.com/overview-of-lookup-tables-in-fpga-design/>

https://www.xilinx.com/htmldocs/xilinx2017_4/sdaccel_doc/yeo1504034293627.html