

4^ο Εργαστήριο Σχεδίασης Ψηφιακών Συστημάτων

1^ο Εργαστηριακό Μάθημα

Βασιλόπουλος Διονύσης

ΕΔΙΠ Τμήματος Πληροφορικής & Τηλεπικοινωνιών - ΕΚΠΑ

1^ο Εργαστηριακό μάθημα

Περιβάλλον Linux

USER=guest

PASSWORD=linux!

Για να εκτελεστεί το Vivado πρέπει να γράψετε τις ακόλουθες εντολές:
(υπάρχει και στο eclass, κάνετε copy/paste κάθε γραμμή χωριστά στο **terminal**)

```
source /opt/Xilinx/Vivado/2022.2/settings64.sh
```

```
cd $home
```

```
cd VIVADO-users/
```

```
mkdir sdi2400XXX
```

```
cd sdi2400XXX
```

```
vivado
```

← Όπου sdi2400XXX είναι ο ΑΜ σας

1^ο Εργαστηριακό μάθημα

Άσκηση

Σχεδιάστε μία αριθμητική και λογική μονάδα (ALU). Στην είσοδο δέχεται ένα σήμα a των 3 bit, καθώς και ένα σήμα $Ctrl$ ενός bit. Η ALU για τιμή $Ctrl = '0'$ κάνει διαίρεση ($a/2$: ακέραιο μέρος της διαίρεσης) 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 ($Carry$). Σχεδιάστε το κύκλωμα. Σας δίδεται ο ορισμός της οντότητας:

entity ALU is

Port (

a : 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;

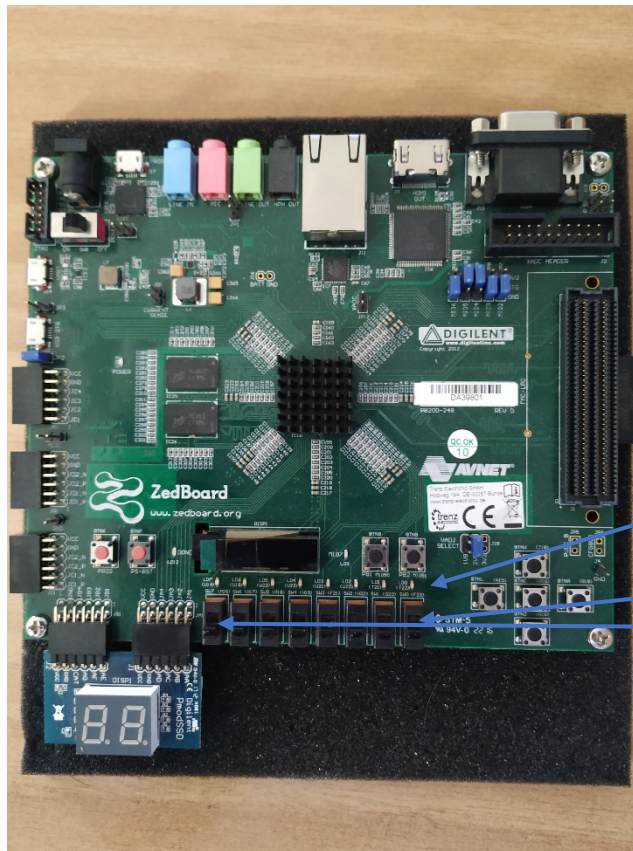
1^ο Εργαστηριακό μάθημα

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)

1^ο Εργαστηριακό μάθημα

Γνωριμία με την κάρτα



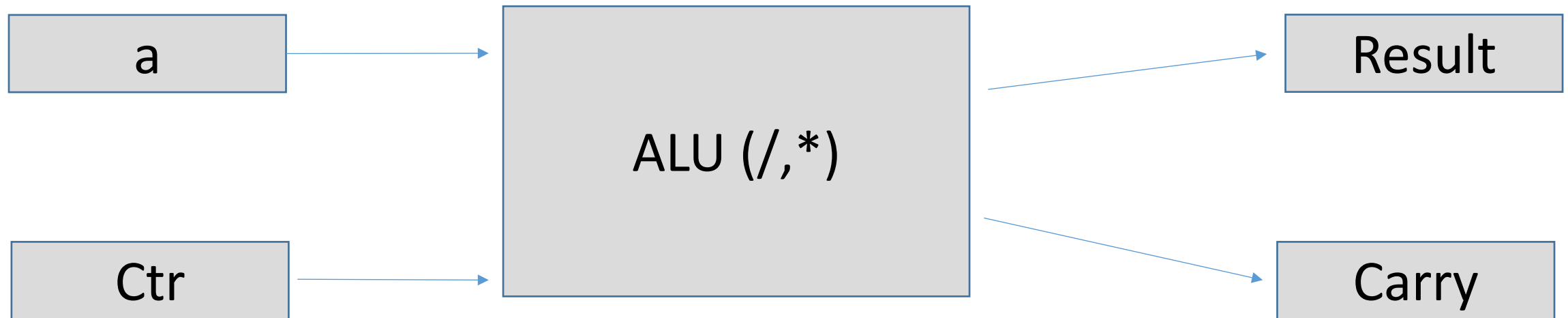
Led: ld0-ld2 (Result)
: ld7 (Carry)

SW: sw0-sw2 (a)
: sw7 (Ctr)

Το πραγματικό όνομα (pin) των led και switch
είναι σε παρένθεση

1^ο Εργαστηριακό μάθημα

Απλοποιημένη μορφή – Είσοδοι/Εξοδοι



1^ο Εργαστηριακό μάθημα

Βήμα 2. Περιγραφή Οντότητας

```
entity ALU is
  Port (
    a          : 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;
```

1^ο Εργαστηριακό μάθημα

Συσχέτιση port με FPGA

Είσοδοι	DIP Switch
Ctr	SW7
A[2]	SW2
A[1]	SW1
A[0]	SW0

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

1^ο Εργαστηριακό μάθημα

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 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

1^ο Εργαστηριακό μάθημα

Βήμα 3. Πίνακας Αληθείας κυκλώματος - Result

A2	A1	A0	Ctr	Result2	Result1	Result0	A2	A1	A0	Ctr	Result2	Result1	Result0
0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	0	0	1	1	0	1	0
0	1	0	0	0	0	1	0	1	0	1	1	0	0
0	1	1	0	0	0	1	0	1	1	1	1	1	0
1	0	0	0	0	1	0	1	0	0	1	0	0	0
1	0	1	0	0	1	0	1	0	1	1	0	1	0
1	1	0	0	0	1	1	1	1	0	1	1	0	0
1	1	1	0	0	1	1	1	1	1	1	1	1	0

1^ο Εργαστηριακό μάθημα

Βήμα 3. Πίνακας Αληθείας κυκλώματος - Carry

A2	A1	A0	Ctr	Carry
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

A2	A1	A0	Ctr	Carry
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

1^ο Εργαστηριακό μάθημα

Βήμα 4. Περιγραφή Αρχιτεκτονικής

Result(2) <= Ctr and a(1);

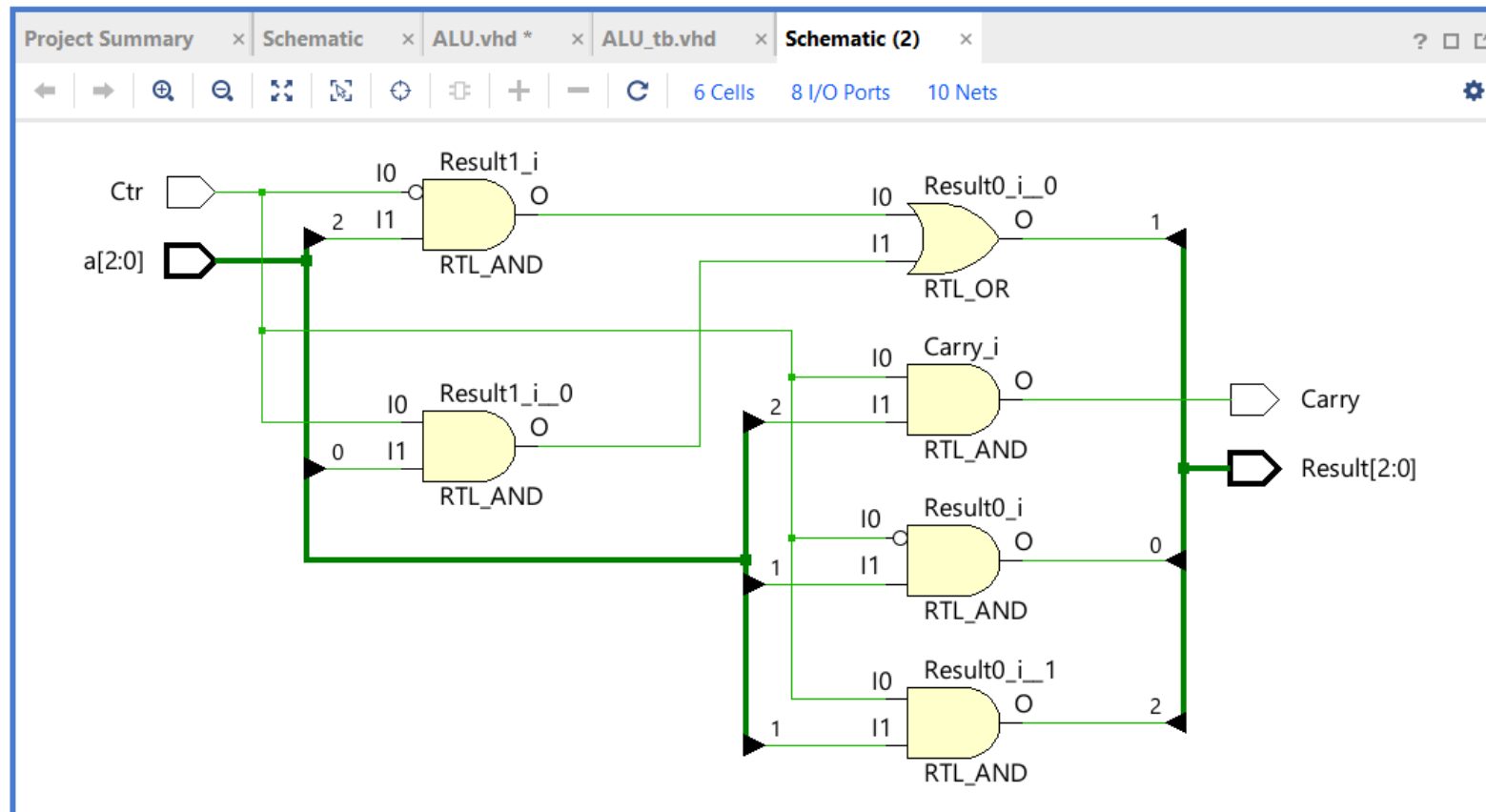
Result(0) <= not Ctr and a(1);

Result(1) <= (not Ctr and a(2)) or (Ctr and a(0));

Carry <= Ctr and a(2);

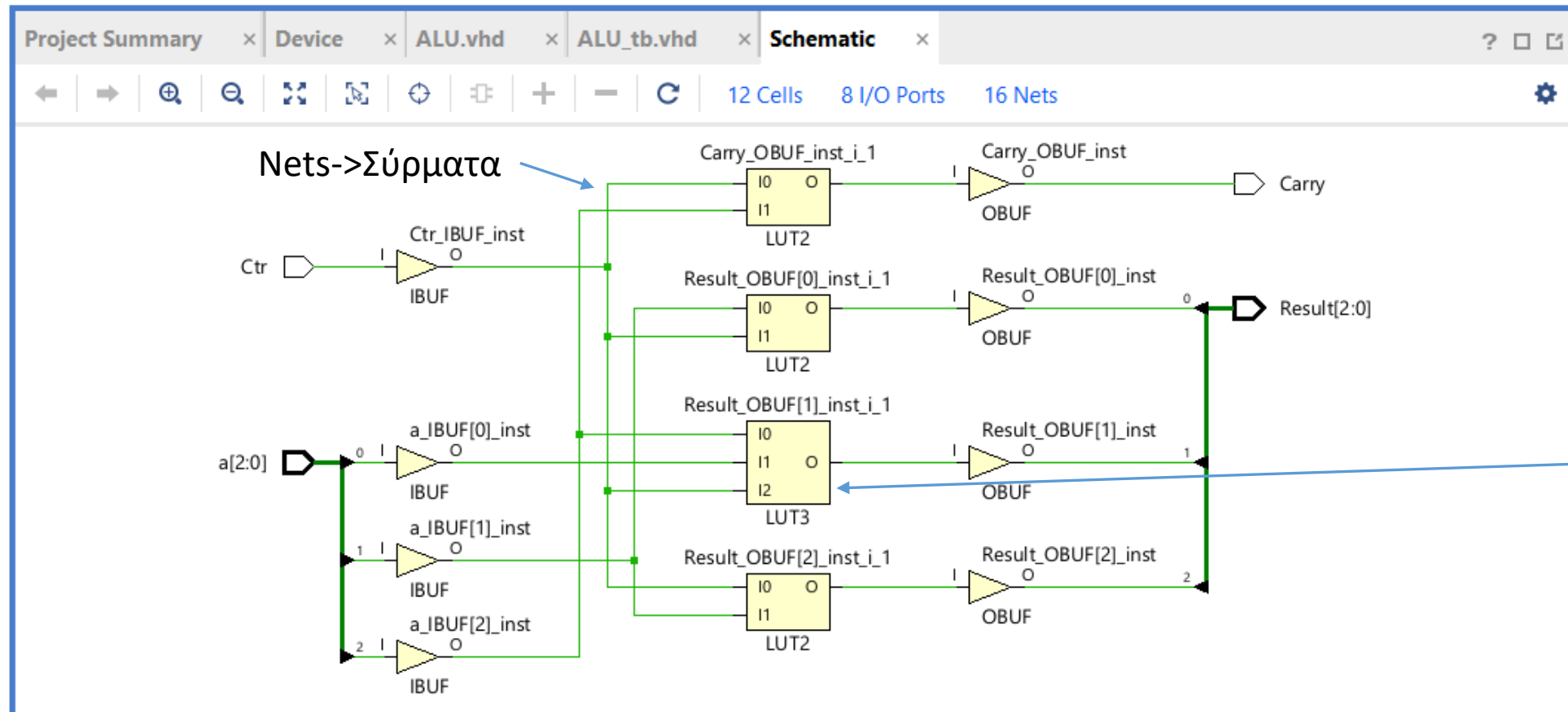
1^ο Εργαστηριακό μάθημα

Βήμα 5. Λογικό κύκλωμα: Αναπαράσταση RTL (α)



1^ο Εργαστηριακό μάθημα

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

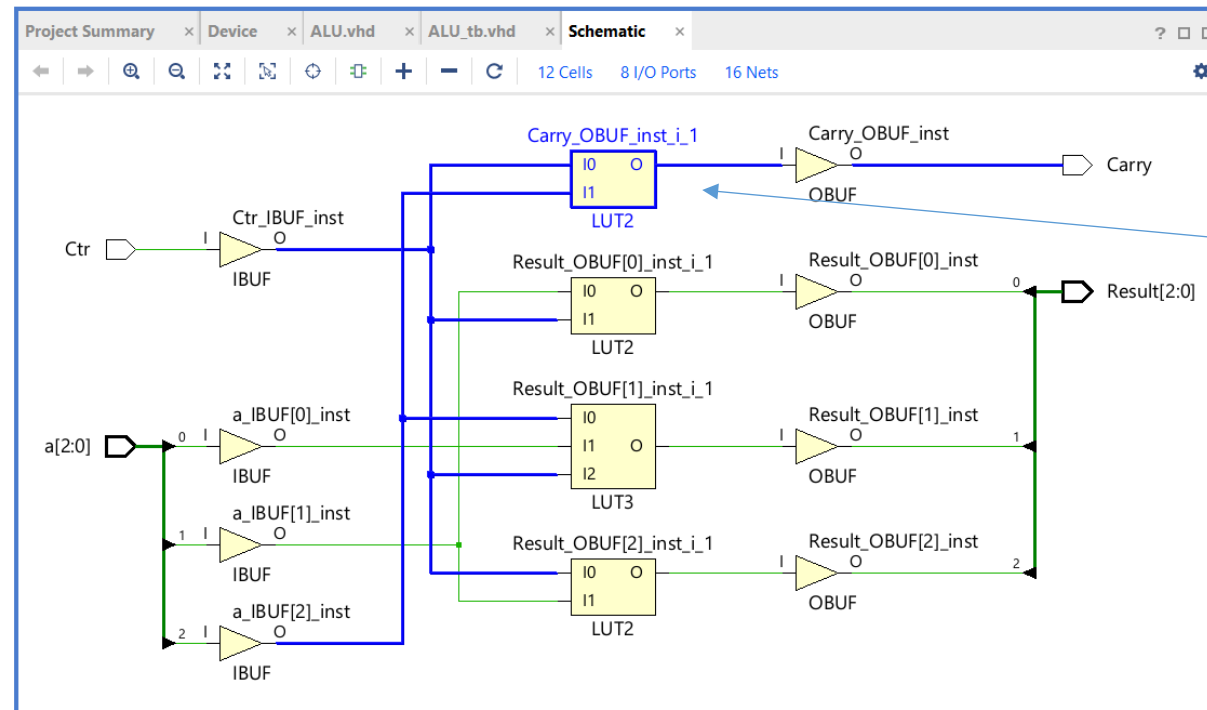
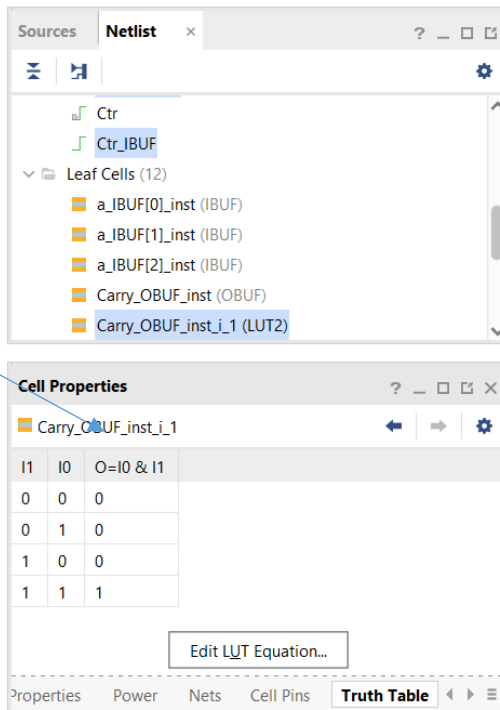


1^ο Εργαστηριακό μάθημα

Βήμα 7α. Λογικό κύκλωμα: Φάση Υλοποίησης

Συνάρτηση
LUT

Πίνακας Αληθείας



Στο LUT είσοδοι
είναι a(2), Ctr και
υπολογίζεται το
ψηφίο Carry.

Ctr => I0
a(2) => I1

1^ο Εργαστηριακό μάθημα

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

Πίνακας Αληθείας
Truth Table
για Carry



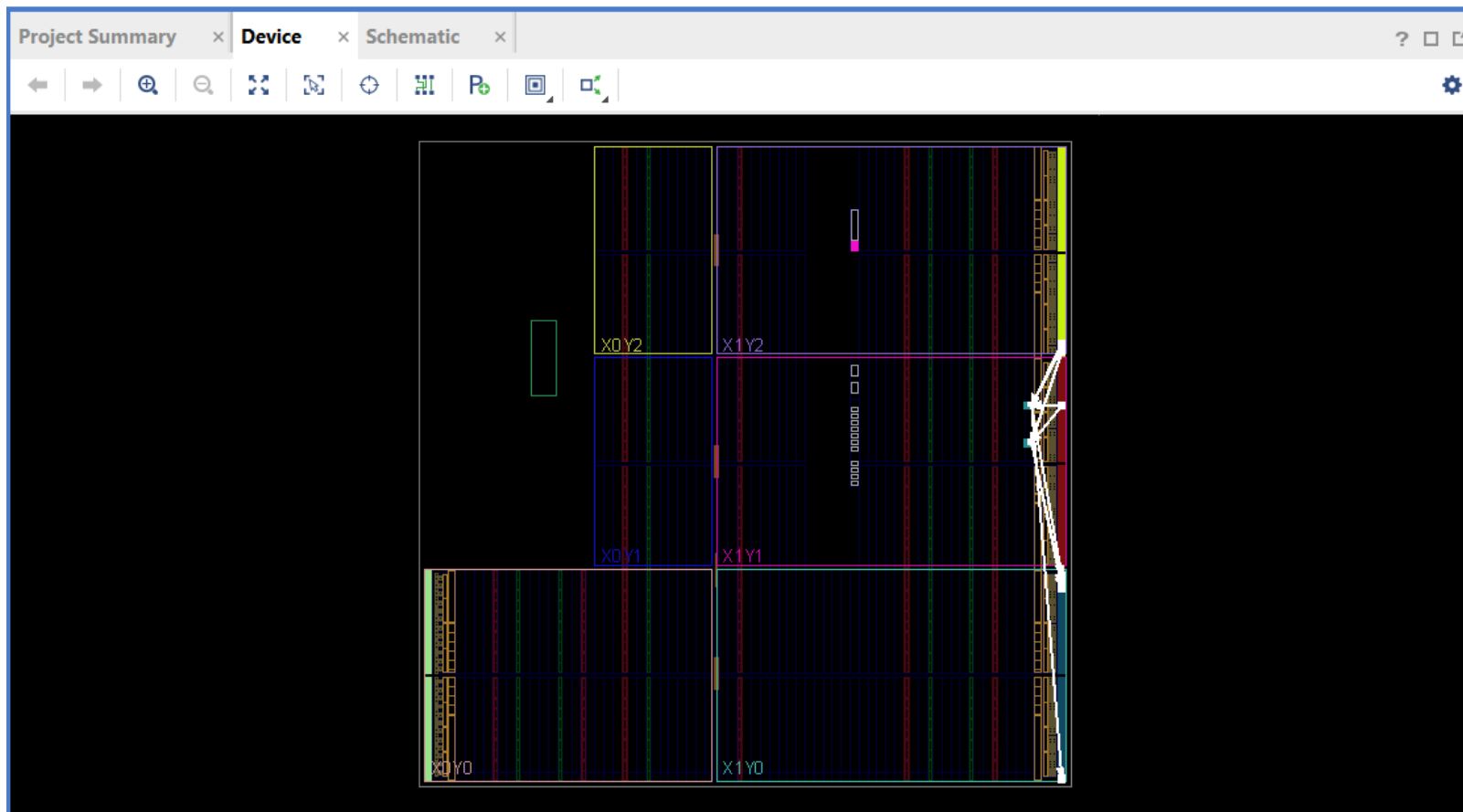
Είσοδοι		Έξοδοι
Ctrl/ I_0	$a(2)/I_1$	Carry
0	0	0
0	1	0
1	0	0
1	1	1

Ο αρχικός Πίνακας Αληθείας και ο αντίστοιχος του LUT είναι ίδιοι

$Carry \leq (a(2) \text{ and } Ctrl)$
 $O = I_0 \text{ \& } I_1$

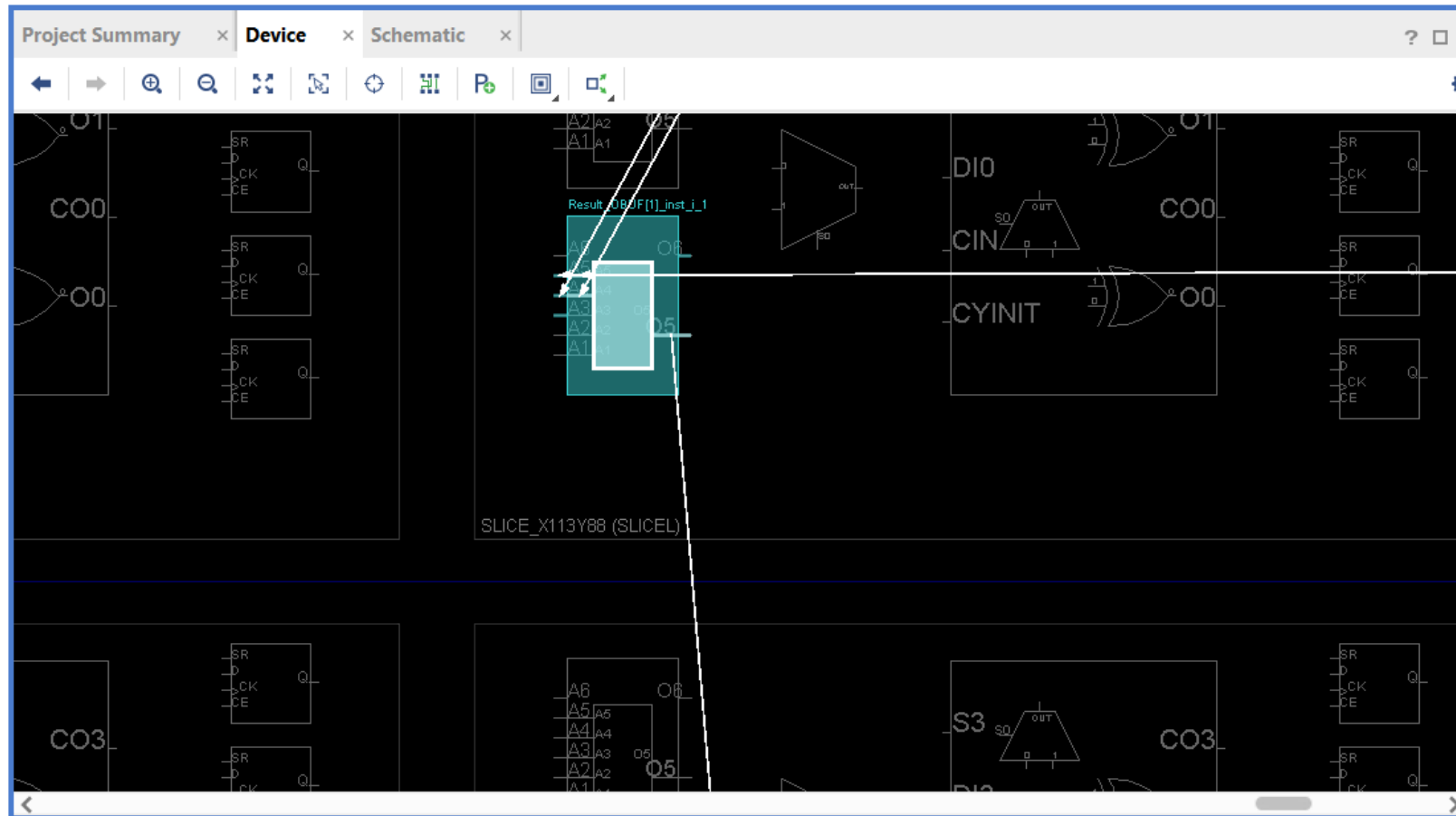
1^ο Εργαστηριακό μάθημα

Βήμα 7c. Λογικό κύκλωμα: Φάση Υλοποίησης – Design



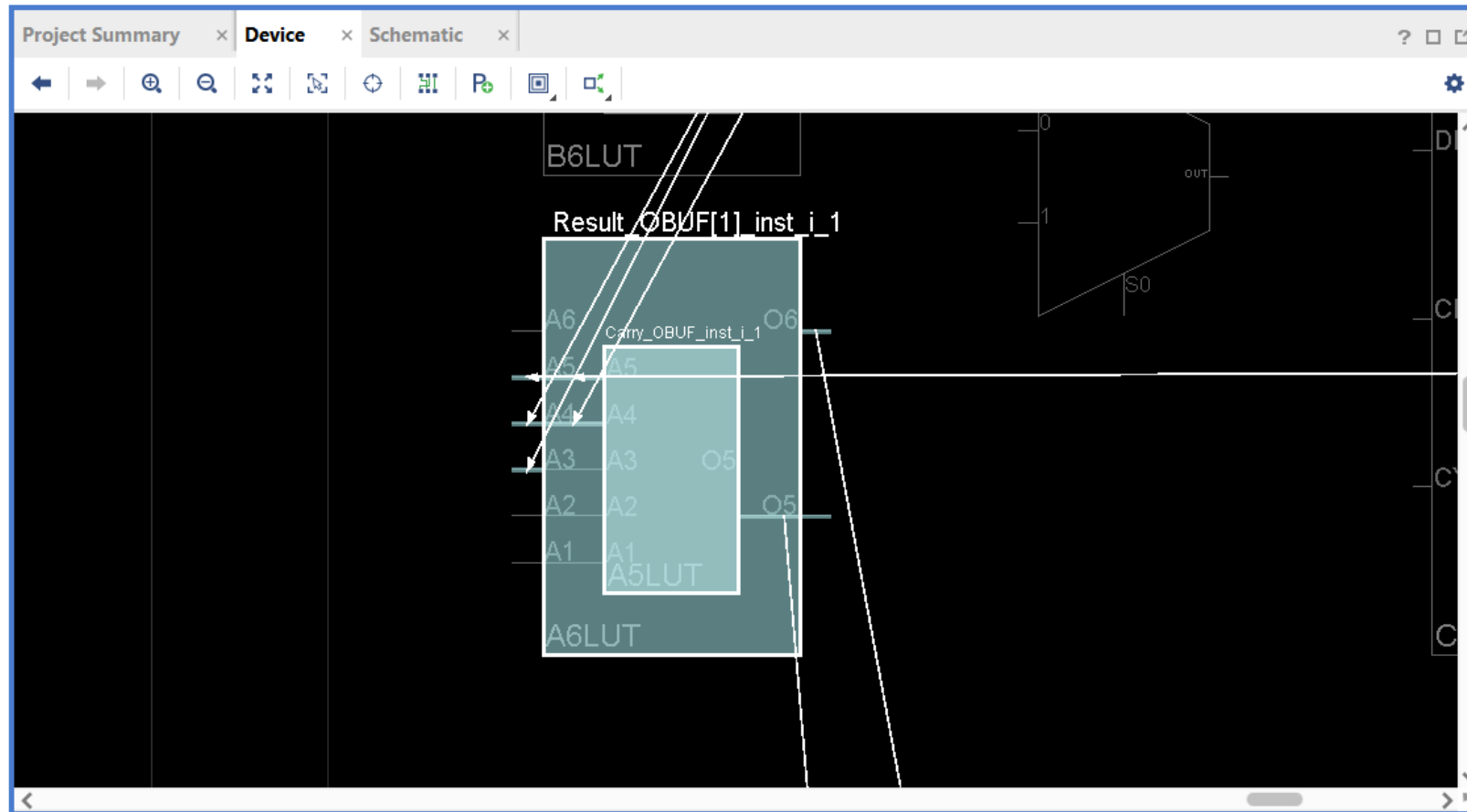
1^ο Εργαστηριακό μάθημα

Βήμα 7d. Λογικό κύκλωμα: Φάση Υλοποίησης – Design: LUT on Board (LUT2 – Carry)



1^ο Εργαστηριακό μάθημα

Βήμα 7d. Λογικό κύκλωμα: Φάση Υλοποίησης – Design: LUT on Board (LUT2 – Carry+Result bit)



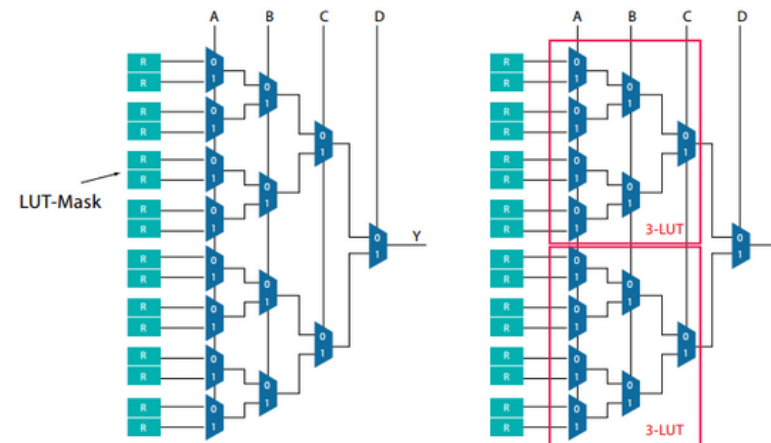
1^ο Εργαστηριακό μάθημα

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



$$a'b'c'd' + abcd + abc'd' = 1000\ 0000\ 0000\ 1001 = 0x8009$$

Share Cite Follow

answered May 8, 2015 at 8:04

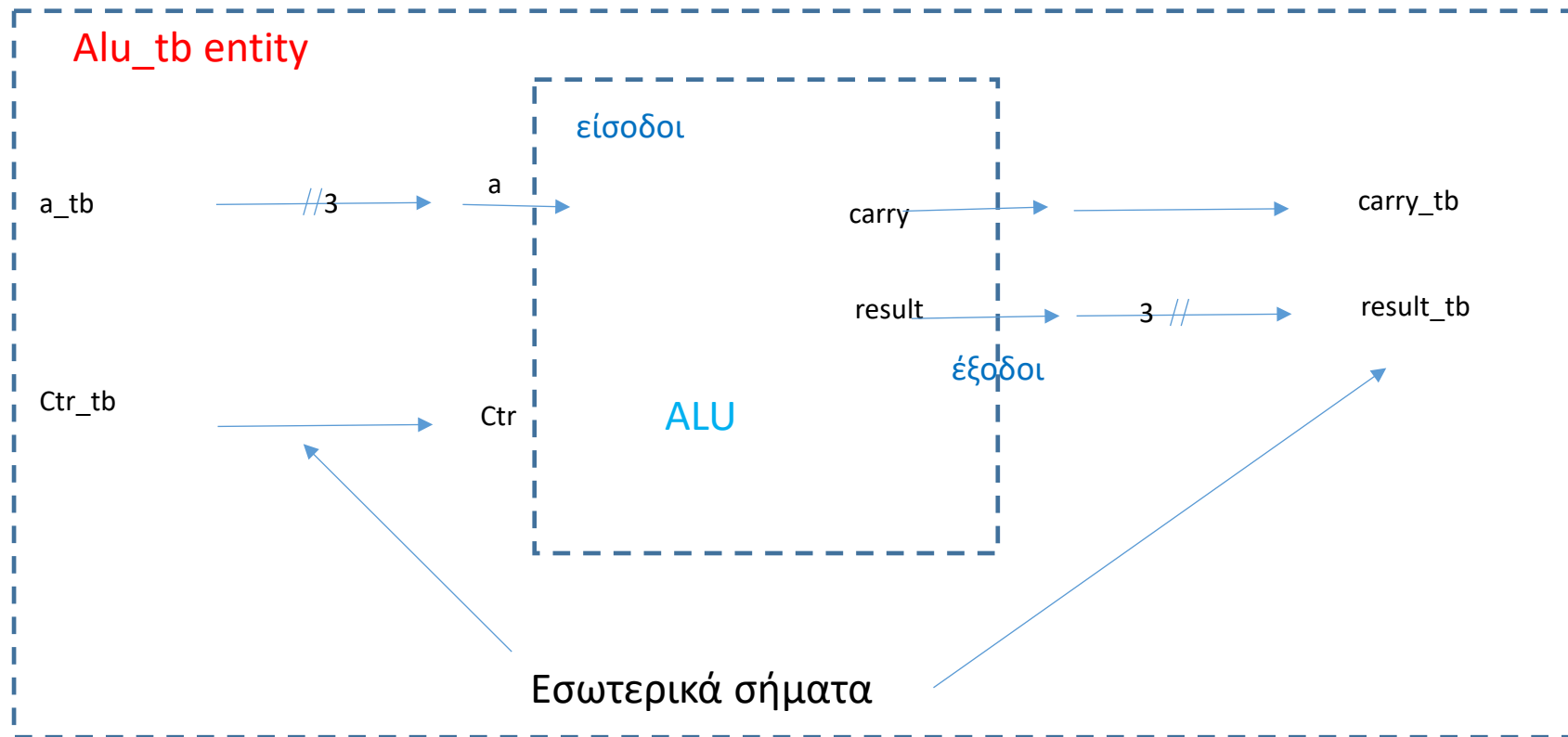
apalorohapa
8,329 ● 2 ● 28 ● 39

Add a comment

Υλοποίηση LUT

1^ο Εργαστηριακό μάθημα

Simulation



1^ο Εργαστηριακό μάθημα

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);
          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 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,Ctr_tb,Result_tb, Carry_tb);
```

```
test: process is
begin
    Ctr_tb<='0';
    a_tb<="000";wait for 10ns;
    a_tb<="001";wait for 10ns;
    a_tb<="010";wait for 10ns;
    a_tb<="011";wait for 10ns;
    a_tb<="100";wait for 10ns;
    a_tb<="101";wait for 10ns;
    a_tb<="110";wait for 10ns;
    a_tb<="111";wait for 10ns;
    Ctr_tb<='1';
    a_tb<="000";wait for 10ns;
    a_tb<="001";wait for 10ns;
    a_tb<="010";wait for 10ns;
    a_tb<="011";wait for 10ns;
    a_tb<="100";wait for 10ns;
    a_tb<="101";wait for 10ns;
    a_tb<="110";wait for 10ns;
    a_tb<="111";wait for 10ns;
end process test;
```

```
end architecture Behavioral;
```

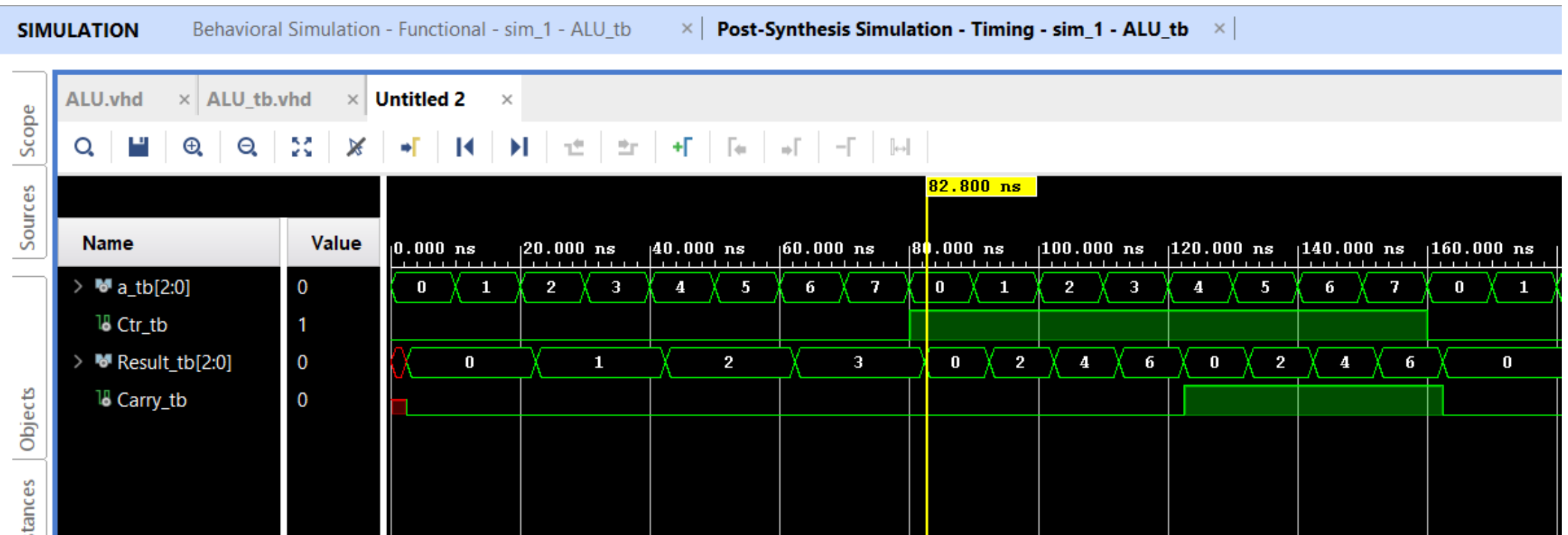
1^ο Εργαστηριακό μάθημα

Simulation – Χρονοσειρά RTL



1^ο Εργαστηριακό μάθημα

Simulation – Χρονοσειρά post Synthesis



1^ο Εργαστηριακό μάθημα

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