dscal
DIGITAL SYSTEMS & COMPUTER ARCHITECTURE LABORATORY

Εργαστήριο Λογικής Σχεδίασης

1ο Εργαστηριακό Μάθημα

Βασιλόπουλος Διονύσης

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

1ο 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

1ο Εργαστηριακό μάθημα

Άσκηση

Να σχεδιάσετε και να προσομοιώσετε στο Vivado ένα απλό κύκλωμα ηλεκτρονικής κλειδαριάς που θα δέχεται ως είσοδο/κωδικό κλειδαριάς έναν ακέραιο αριθμό 4-bits (στο δυαδικό) και θα ενεργοποιεί (λογικό 1) την έξοδο της κλειδαριάς (lock_out) μόνο όταν ο αριθμός αυτός ταυτίζεται με το τελευταίο ψηφίο του ΑΜ σας. Για παράδειγμα, για τον ΑΜ 1115201900205, ο κωδικός έχει την τιμή 5 (στο δυαδικό 0101).
Το όνομα του project θα είναι Lab1, το όνομα του αρχείου (design source) αλλά και η οντότητα σας θα λέγεται locker, ενώ η αρχιτεκτονική Dataflow. Τα αντίστοιχα ονόματα για την προσομοίωση θα είναι locker_tb, και Dataflow_tb.
Σας δίνεται ο ορισμός της οντότητας

```
entity locker is
port(
  digit3, digit2, digit1, digit0   : in std_logic;
  lock_out : out std_logic);
end locker;
```

Το digit0 αντιστοιχεί στο λιγότερο σημαντικό bit του κωδικού ενώ το digit3 στο πιο σημαντικό bit (στο παράδειγμά μας digit0=’1’ και digit3=’0’). Γράψτε την αρχιτεκτονική που αντιστοιχεί στον ΑΜ σας. Εμφανίστε το RTL διάγραμμα, κάντε τη σύνθεση, εμφανίστε το διάγραμμά της (Schematic), κάντε το ίδιο για την υλοποίηση, προγραμματίστε την κάρτα fpga.

1ο Εργαστηριακό μάθημα

Πραγματικό πρόβλημα – Βήματα επίλυσης

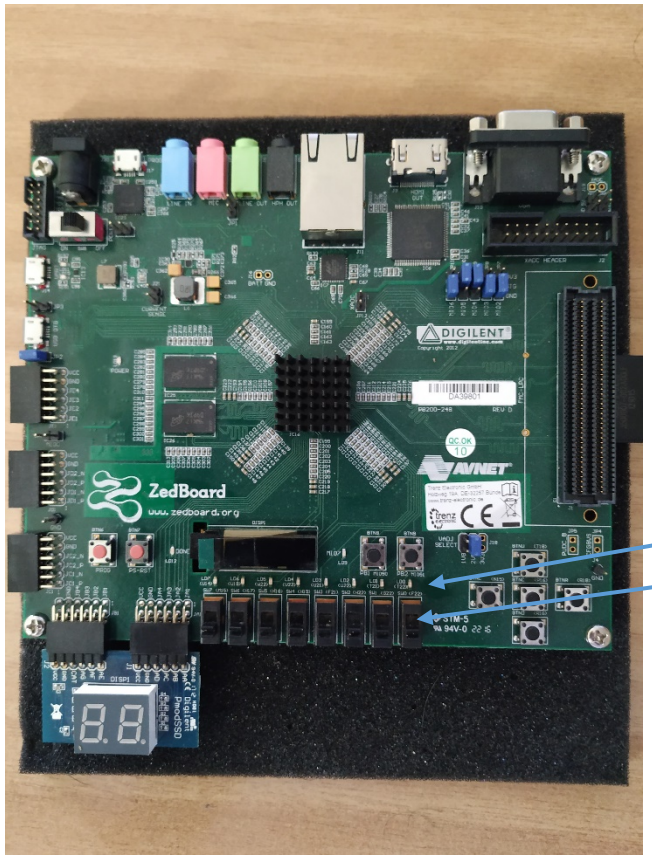
1. Εντοπισμός Input/Output του συστήματος
2. Εύρεση πίνακα αληθείας για κάθε έξοδο του συστήματος (**αν χρειάζεται**)
3. Δημιουργία νέου project
4. Δημιουργία Entity/Αρχείου constraints
5. Δημιουργία Architecture – Θα έχετε τουλάχιστον τόσες εντολές όσες είναι και οι έξοδοι του συστήματος. Κάθε μία εντολή αντιστοιχεί σε μία έξοδο.
6. Δημιουργία RTL αναπαράστασης
7. Σύνθεση
8. Υλοποίηση

Προγραμματισμός κάρτας (Γίνεται μόνο στο Εργαστήριο)

9. **Προσομοίωση** (Θα παρουσιαστεί σε διάλεξη)

1ο Εργαστηριακό μάθημα

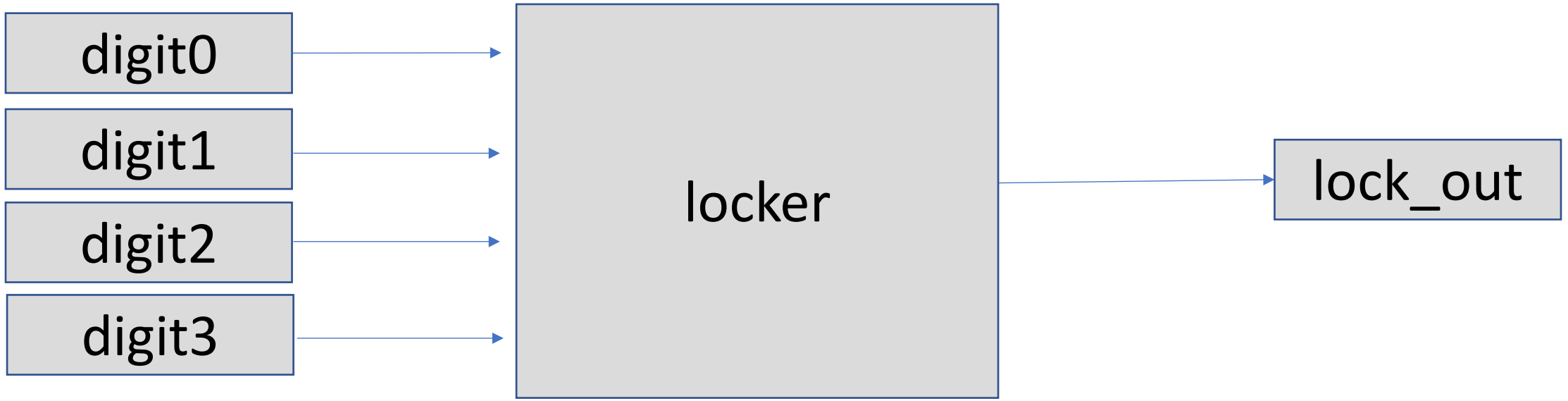
Γνωριμία με την κάρτα



Led: ld0
SW: sw0-sw3
Το πραγματικό όνομα (pin) είναι σε παρένθεση

1ο Εργαστηριακό μάθημα

Απλοποιημένη μορφή κυκλώματος – Είσοδοι/Έξοδοι



1ο Εργαστηριακό μάθημα

Βήμα 2: Περιγραφή Οντότητας

```
entity locker is

port(
   digit3, digit2, digit1, digit0   : in std_logic;
   lock_out : out std_logic);

end locker;
```

1ο Εργαστηριακό μάθημα

Βήμα 3: Πίνακας αληθείας κυκλώματος

Πίνακας Αληθείας
Truth Table
για ΑΜ που λήγει σε
5 =>0101

Είσοδοι				Έξοδοι
Digit3	Digit2	Digit1	Digit0	lock_out
0	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**	**1**
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Υπάρχει μία μόνο γραμμή με ‘1’ και άρα η συνάρτηση του lock_out αντιστοιχεί σε ένα μόνο ελαχιστόρο (γινόμενο)
lock_out= !Digit3*Digit2*!Digit1*Digit0

Η ανωτέρω παράσταση σε VHDL είναι:
lock_out<= not Digit3 and Digit2 and not Digit1 and Digit0;

1ο Εργαστηριακό μάθημα

Βήμα 4: Περιγραφή Αρχιτεκτονικής

Αρχιτεκτονική για ΑΜ που λήγει σε 5

```
lock_out<=not digit3 and digit2 and not digit1 and digit0;
ή
lock_out<=digit0 and not digit1 and digit2 and not digit3;
ή
lock_out<=(not digit3) and digit2 and (not digit1) and digit0;
ή
……
```

1ο Εργαστηριακό μάθημα

Άσκηση – Συσχέτιση port με FPGA-1

Είσοδοι	DIP Switch
digit3	SW3
digit2	SW2
digit1	SW1
digit0	SW0

Έξοδοι	LED
lock_out	LD0

1ο Εργαστηριακό μάθημα

Άσκηση – Συσχέτιση port με FPGA-2 (Αρχείο constraints: locker.xdc)

```
# ZedBoard Pin Assignments
##############################
# On-board Slide Switches  #
##############################

set_property -dict { PACKAGE_PIN F21   IOSTANDARD LVCMOS33 } [get_ports { digit3 }];
set_property -dict { PACKAGE_PIN H22   IOSTANDARD LVCMOS33 } [get_ports { digit2 }];
set_property -dict { PACKAGE_PIN G22   IOSTANDARD LVCMOS33 } [get_ports { digit1 }];
set_property -dict { PACKAGE_PIN F22   IOSTANDARD LVCMOS33 } [get_ports { digit0 }];

##############################
# On-board led           #
##############################
set_property -dict { PACKAGE_PIN T22   IOSTANDARD LVCMOS33 } [get_ports { lock_out }];
```

ΠΡΟΣΟΧΗ στις διαφορές με τον Κώδικα VHDL

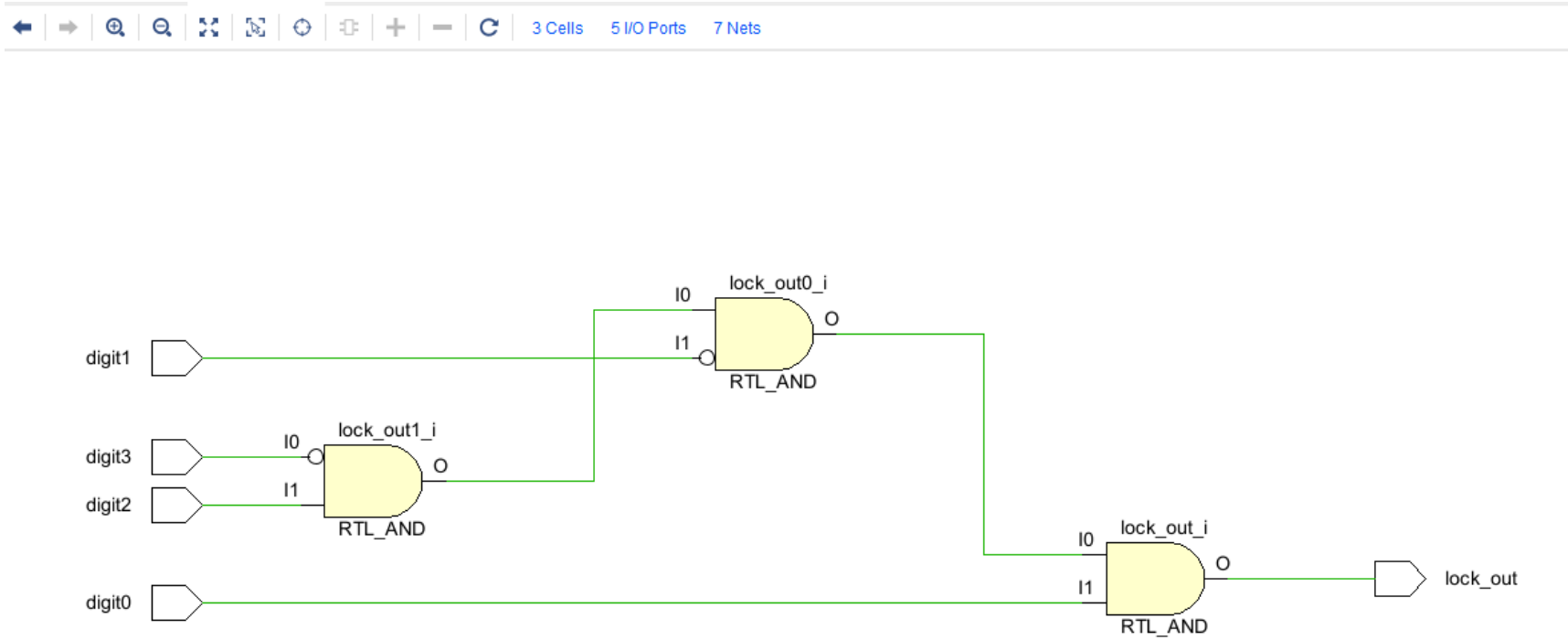
1.Τα σχόλια εδώ είναι με #

2. Τα ονόματα των σημάτων, πρέπει να είναι **ΑΚΡΙΒΩΣ** ίδια με τη δήλωση στην οντότητα (**case sensitive**)

3.Κάρτα Zynq 7(000) ZC702 – Evaluation Board

1ο Εργαστηριακό μάθημα

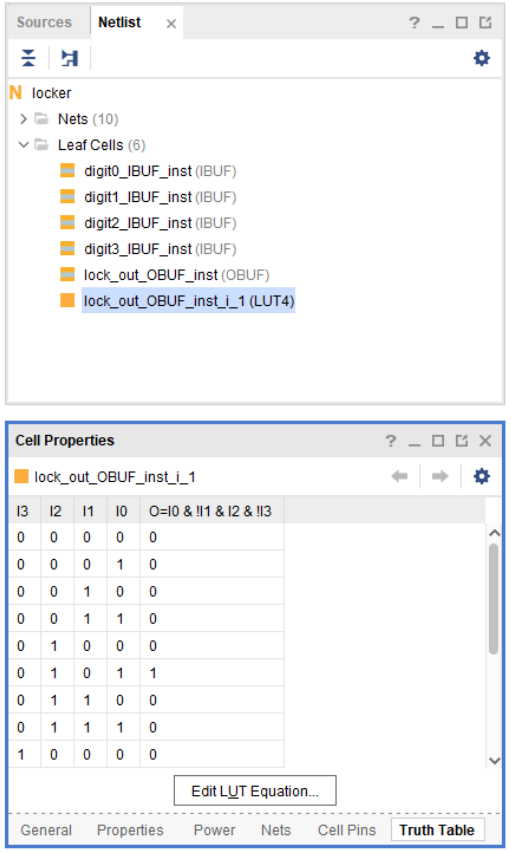
Βήμα 5: RTL Analysis

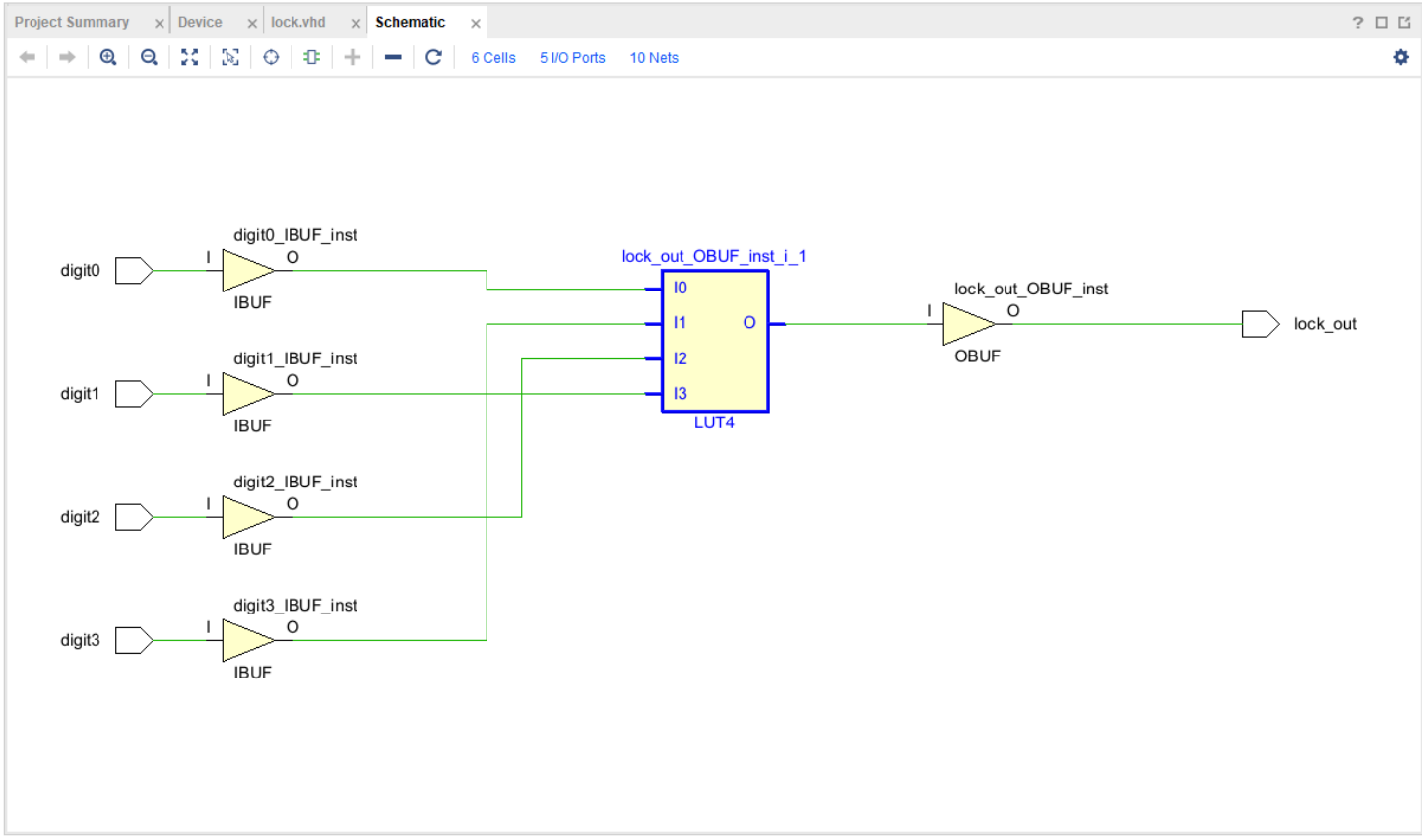


1ο Εργαστηριακό μάθημα

Βήμα 6: Synthesis

Πίνακας Αληθείας LUT όπως Βήμα 3.

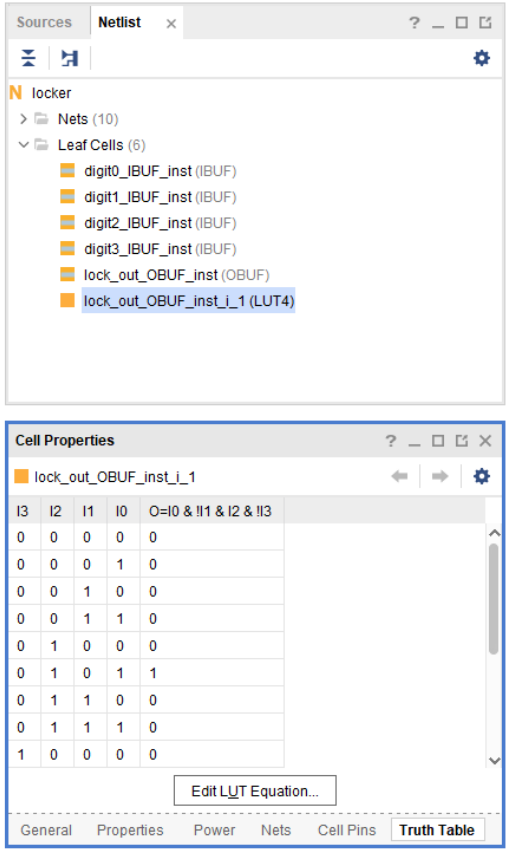


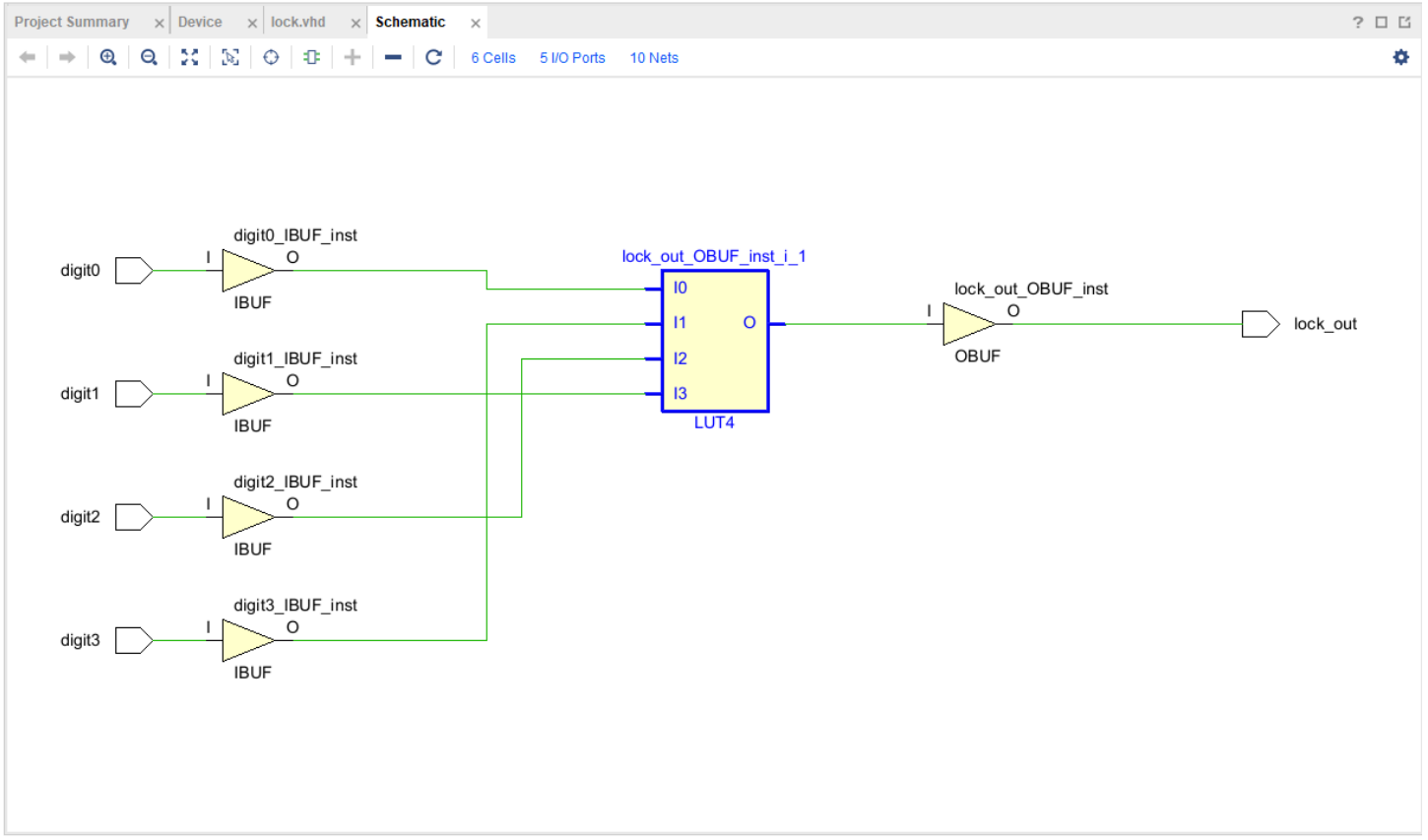


1ο Εργαστηριακό μάθημα

Βήμα 7: Implementation

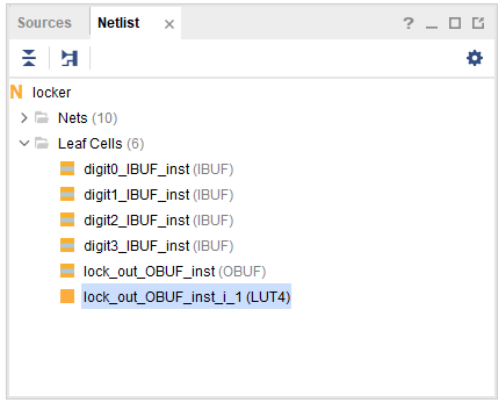
Πίνακας Αληθείας LUT όπως Βήμα 3.

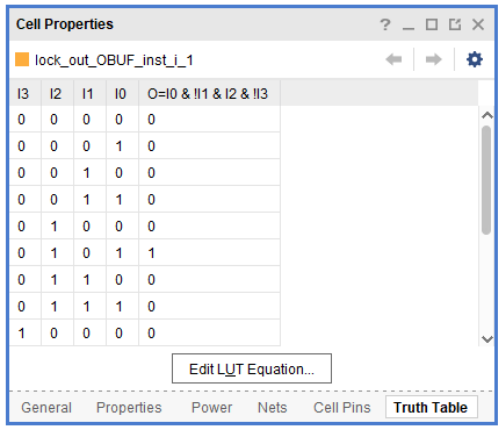




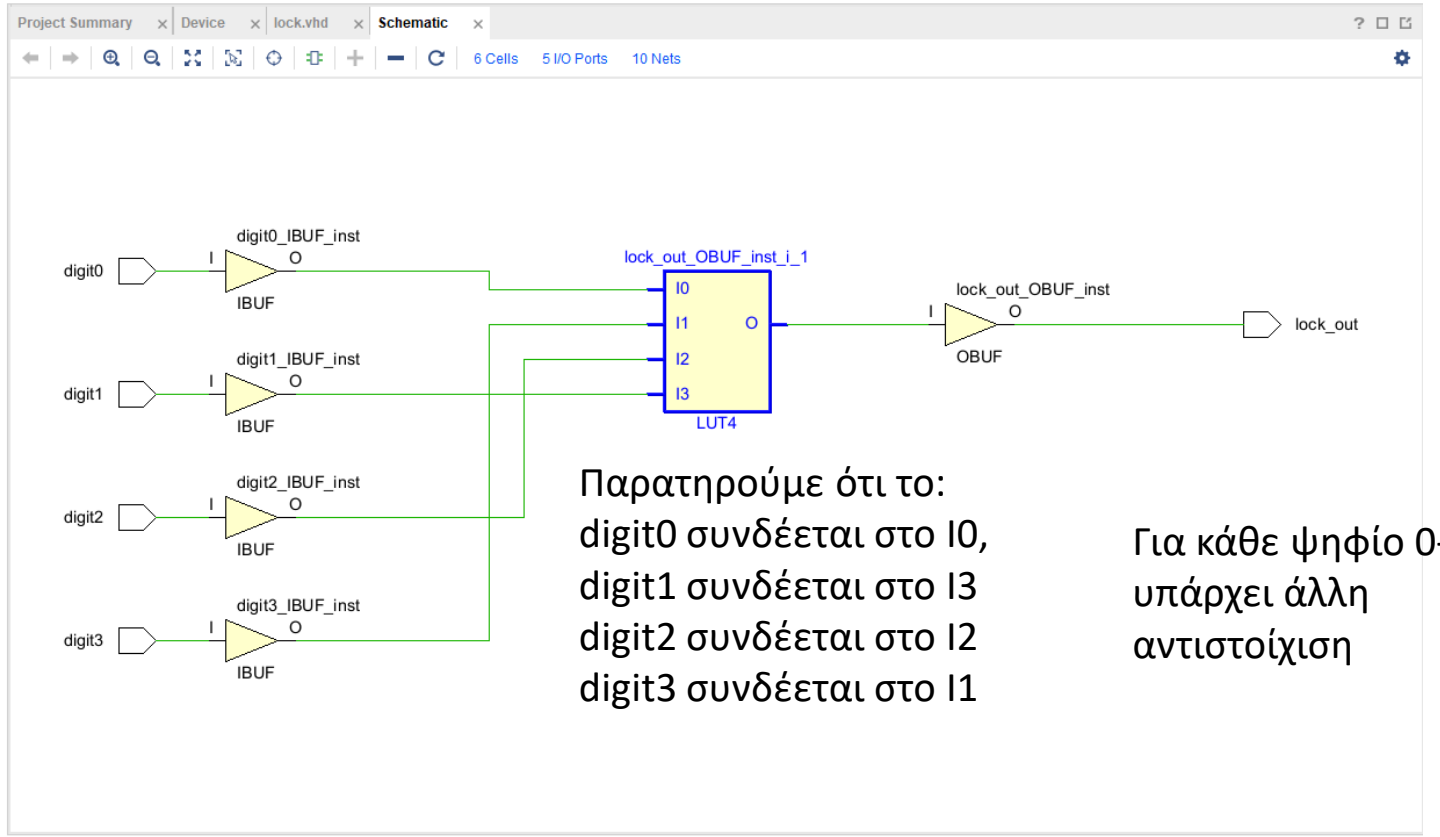
1ο Εργαστηριακό μάθημα

Βήμα 7a: Implementation





Πίνακας Αληθείας LUT όπως Βήμα 3.



Παρατηρούμε ότι το:
digit0 συνδέεται στο I0,
digit1 συνδέεται στο I3
digit2 συνδέεται στο I2
digit3 συνδέεται στο I1

Για κάθε ψηφίο 0-9 θα υπάρχει άλλη αντιστοίχιση

1ο Εργαστηριακό μάθημα

Βήμα 7b: Implementation: Modified Truth Table (αντιστοιχίες $Digit_X$ <=> I_X στο LUT)

Για κάθε Digit υπάρχει το αντίστοιχο I(nput) του LUT

Πίνακας Αληθείας Truth Table

για ΑΜ που λήγει σε 5 =>0101

Είσοδοι				Έξοδοι
Digit3/I1	Digit2/I2	Digit1/I3	Digit0/I0	lock_out
0	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**	**1**
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Ο αρχικός Πίνακας Αληθείας και ο αντίστοιχος του LUT είναι ίδιοι

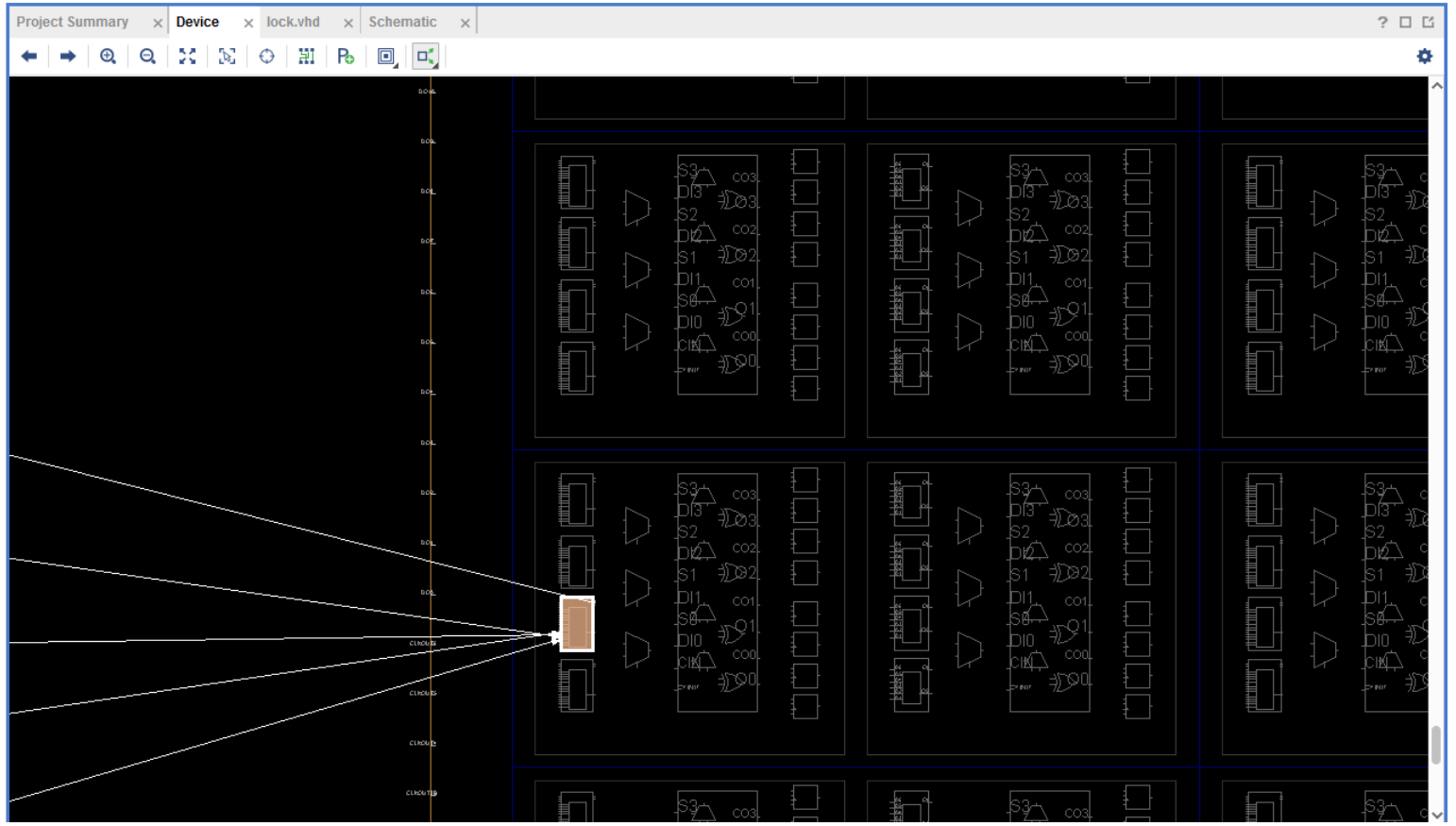
Υπάρχει μία μόνο γραμμή με ‘1’ και άρα η συνάρτηση του lock_out αντιστοιχεί σε ένα μόνο ελαχιστόρο (γινόμενο)

lock_out= !Digit3*Digit2*!Digit1*Digit0
ή στο LUT
O=!I1*I2*!I3*I0

Η ανωτέρω παράσταση σε VHDL είναι:
lock_out<= not Digit3 and Digit2 and not Digit1 and Digit0;

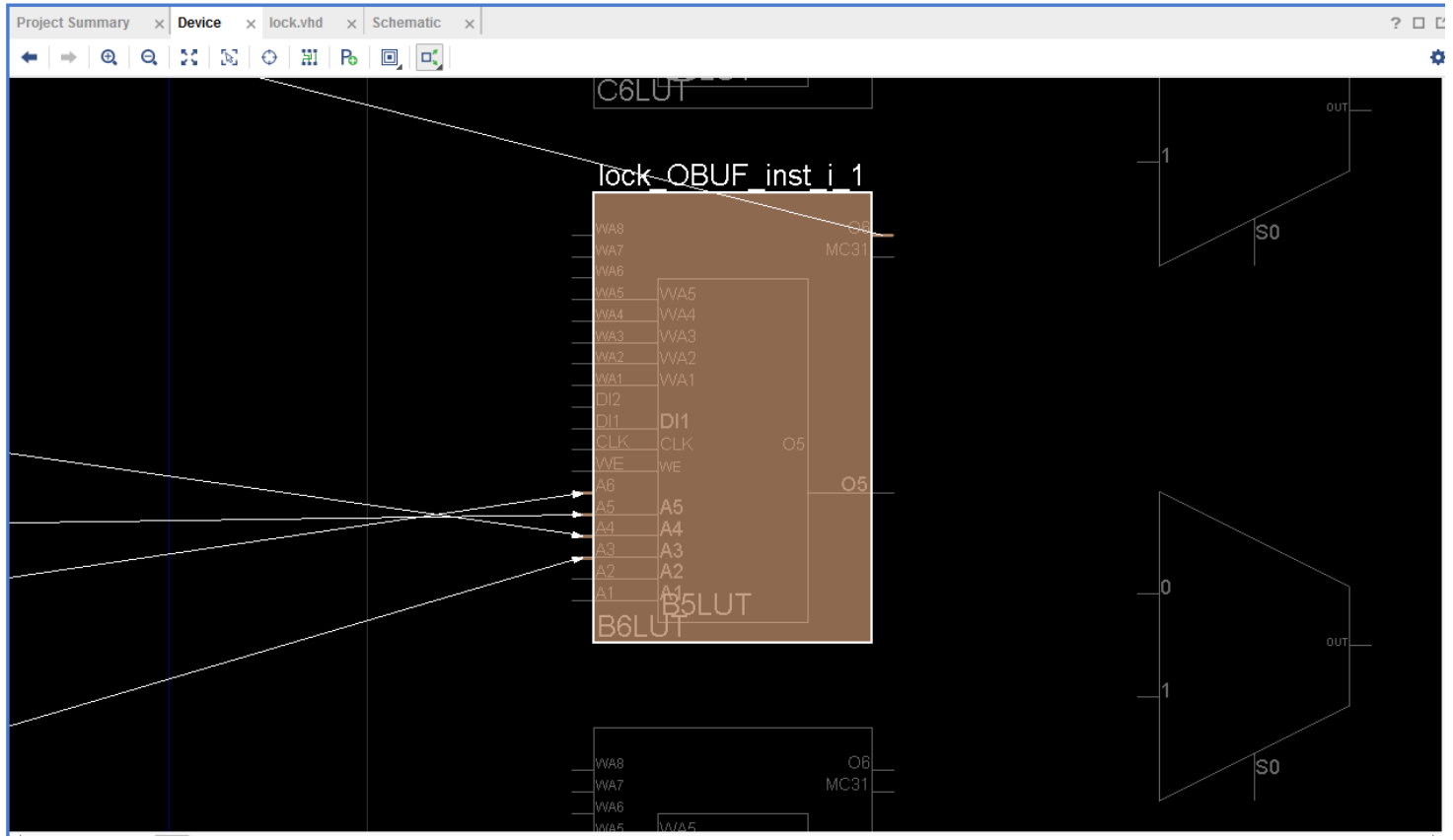
1ο Εργαστηριακό μάθημα

Βήμα 7c: Implementation – Design: LUT on Board



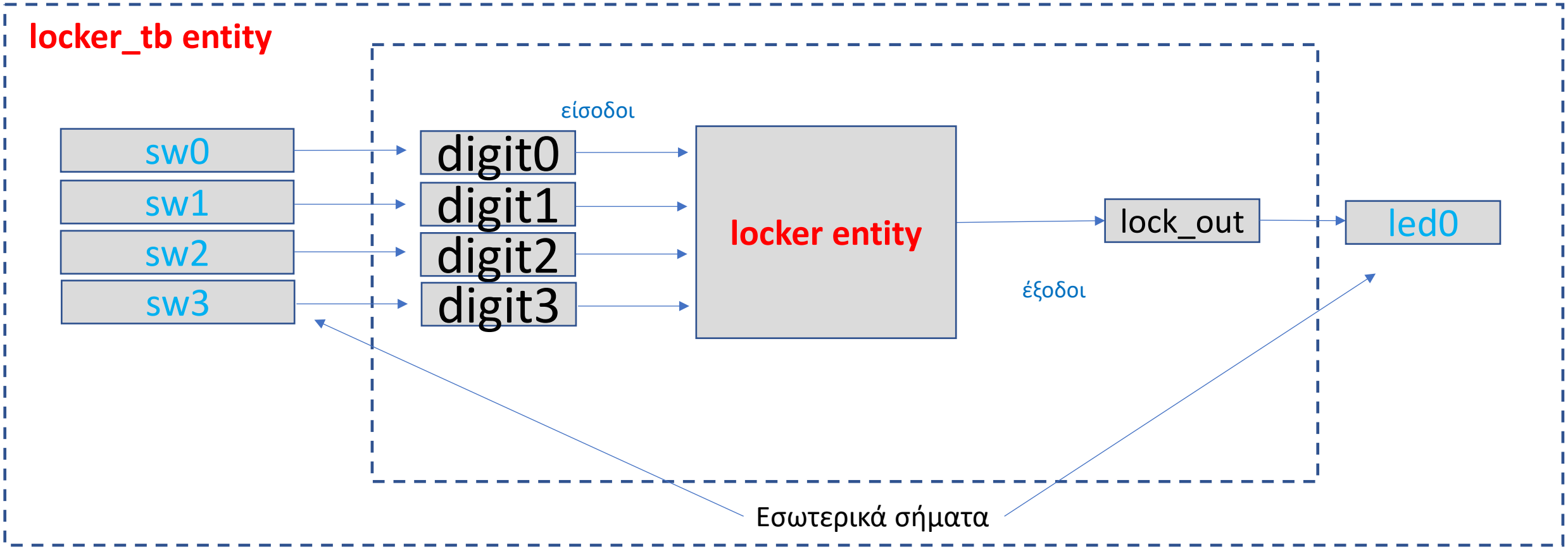
1ο Εργαστηριακό μάθημα

Βήμα 7d: Implementation – Design: LUT on Board



1ο Εργαστηριακό μάθημα

Βήμα 8a: Simulation



1ο Εργαστηριακό μάθημα

Βήμα 8b: Simulation

```
LIBRARY ieee; USE ieee.std_logic_1164.ALL;

entity locker_tb IS
end locker_tb;

architecture behavior OF locker_tb IS
    -- Component Declaration for the Unit Under Test (UUT)
    component locker
   port(
       digit3, digit2, digit1, digit0   : in std_logic;
                                lock_out : out std_logic);
 end component;

signal sw3, sw2, sw1, sw0 : std_logic; -- Input
signal led0 : std_logic;                        --Output

begin
-- Instantiate the Unit Under Test (UUT)
  uut: locker PORT MAP (digit0 => sw0,digit1 => sw1,digit2 => sw2,digit3 => sw3,
                                       lock_out => led0);
```

```
 -- Test process
test_proc: process
begin
--
sw3<='0';sw2<='0';sw1<='0';sw0<='0';wait for 20 ns;
sw3<='0';sw2<='0';sw1<='0';sw0<='1';wait for 20 ns;
sw3<='0';sw2<='0';sw1<='1';sw0<='0';wait for 20 ns;
sw3<='0';sw2<='0';sw1<='1';sw0<='1';wait for 20 ns;
sw3<='0';sw2<='1';sw1<='0';sw0<='0';wait for 20 ns;
sw3<='0';sw2<='1';sw1<='0';sw0<='1';wait for 20 ns;
sw3<='0';sw2<='1';sw1<='1';sw0<='0';wait for 20 ns;
sw3<='0';sw2<='1';sw1<='1';sw0<='1';wait for 20 ns;
sw3<='1';sw2<='0';sw1<='0';sw0<='0';wait for 20 ns;
sw3<='1';sw2<='0';sw1<='0';sw0<='1';wait for 20 ns;
sw3<='1';sw2<='0';sw1<='1';sw0<='0';wait for 20 ns;
sw3<='1';sw2<='0';sw1<='1';sw0<='1';wait for 20 ns;
sw3<='1';sw2<='1';sw1<='0';sw0<='0';wait for 20 ns;
sw3<='1';sw2<='1';sw1<='0';sw0<='1';wait for 20 ns;
sw3<='1';sw2<='1';sw1<='1';sw0<='0';wait for 20 ns;
sw3<='1';sw2<='1';sw1<='1';sw0<='1';wait for 20 ns;
end process;
end behavior;
```

1ο Εργαστηριακό μάθημα

Βήμα 8c: Simulation