LABORATORY # 3
L A B   M A N U A L

Programming Combinatorial Logic on the Basys FPGA Board

Objectives

Lab 3 contains 3 parts: **Part 1** – guided design and **Parts 2, 3** – individual or in group design. Its purposes are to get familiar with:

1. Xilinx ISE Design, Synthesis and Basys Board FPGA Programming.

2. Learning Basys Board components and FPGA pin routing.

3. Understanding of Configuration files.

4. Synthesis and Implementation of combinational logic applications on FPGA.

5. Basys Board Programming

Equipment

- PC or compatible
- Digilent's Basys Spartan-3E FPGA Evaluation Board

Software

- Xilinx ISE Design Software Suite 10.1
- ModelSim XE III modeling software
- Digilent's Adept ExPort Software

Parts

N/A

Introduction

In all the labs we will adhere to the following industry standard design flow in applications development which utilizes FPGA devices.

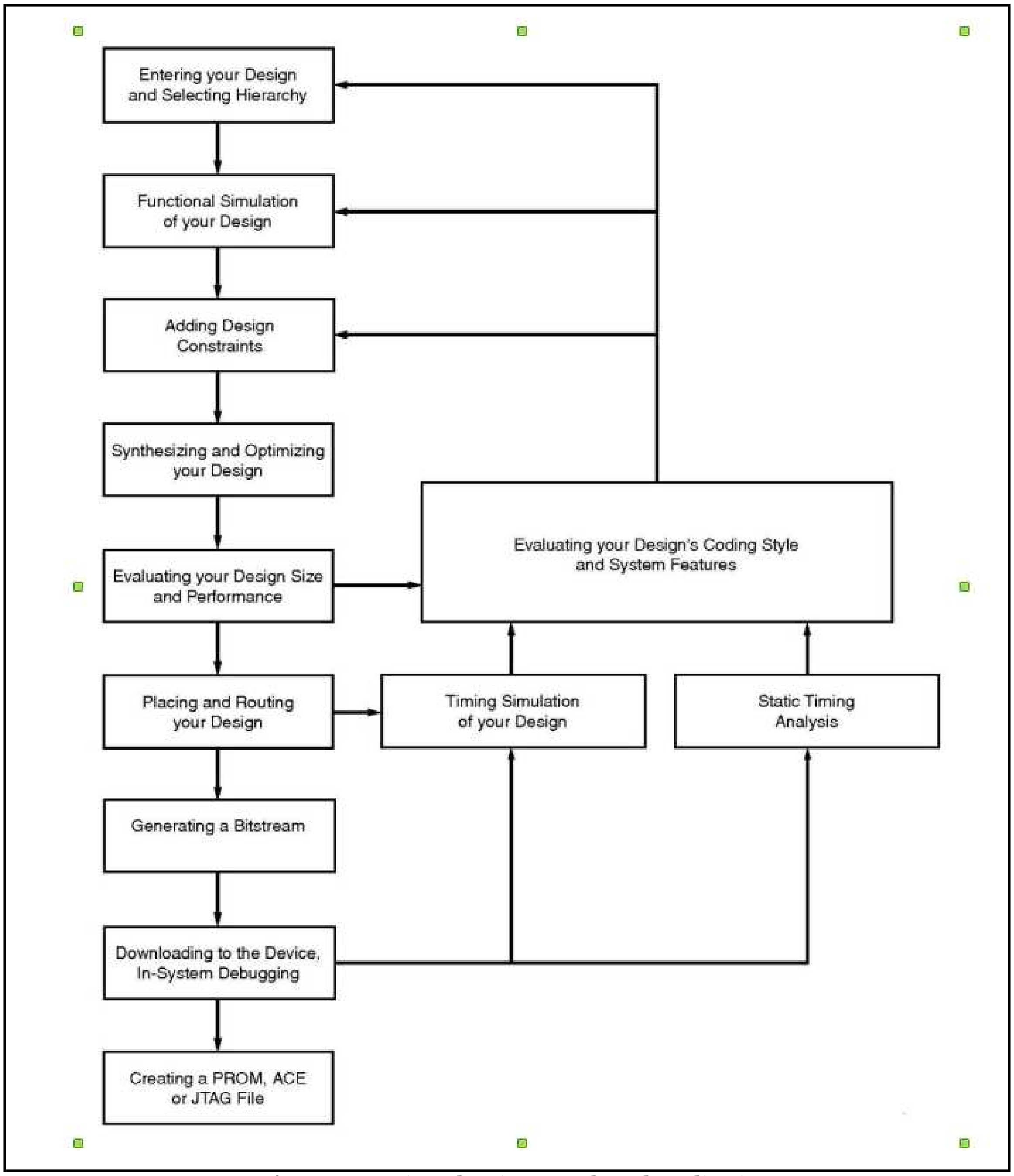
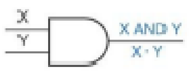

Entering your Design and Selecting Hierarchy
Functional Simulation of your Design
Adding Design Constraints
Synthesizing and Optimizing your Design
Evaluating your Design Size and Performance
Placing and Routing your Design
Generating a Bitstream
Downloading to the Device, In-System Debugging
Creating a PROM, ACE or JTAG File
Evaluating your Design's Coding Style and System Features
Timing Simulation of your Design
Static Timing Analysis

Figure 1. Design Flow in FPGA based applications

PART 1. Design, FPGA Synthesis and Testing of an AND Gate

In this guided FPGA application development experiment, we will design and test combinational AND gate and test it on the Digilent's Basys Board:

Specification



X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

Figure 2. AND gate

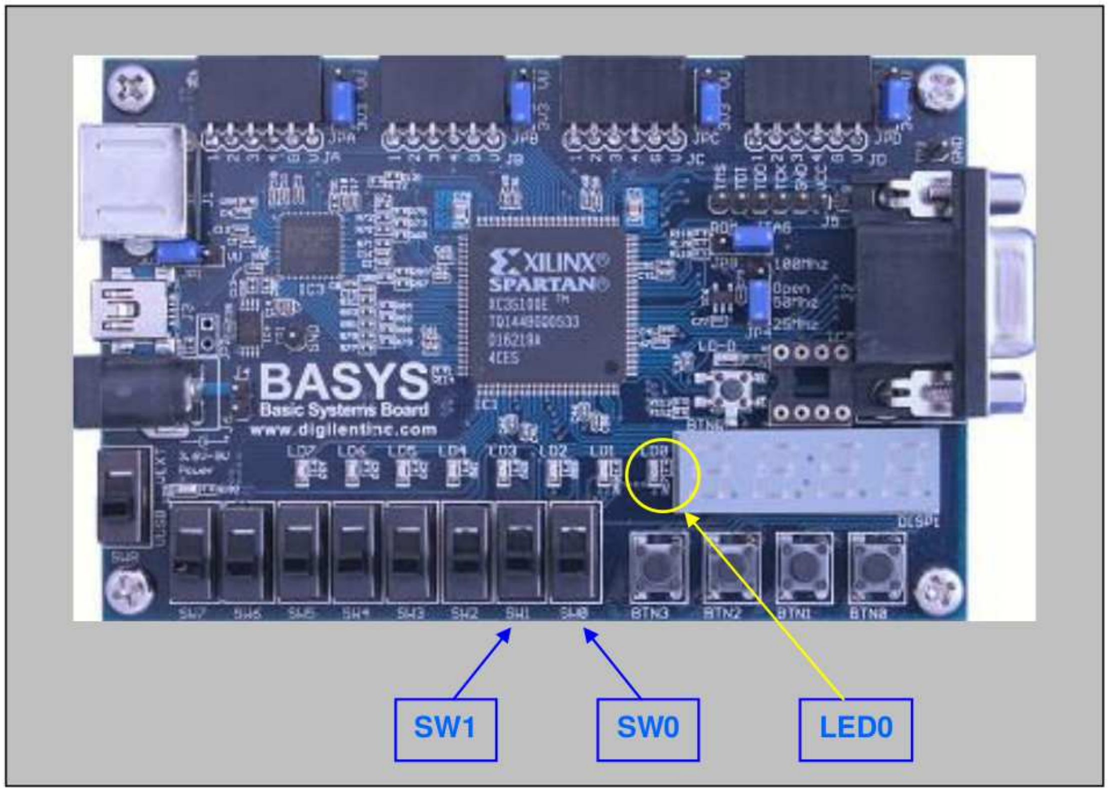
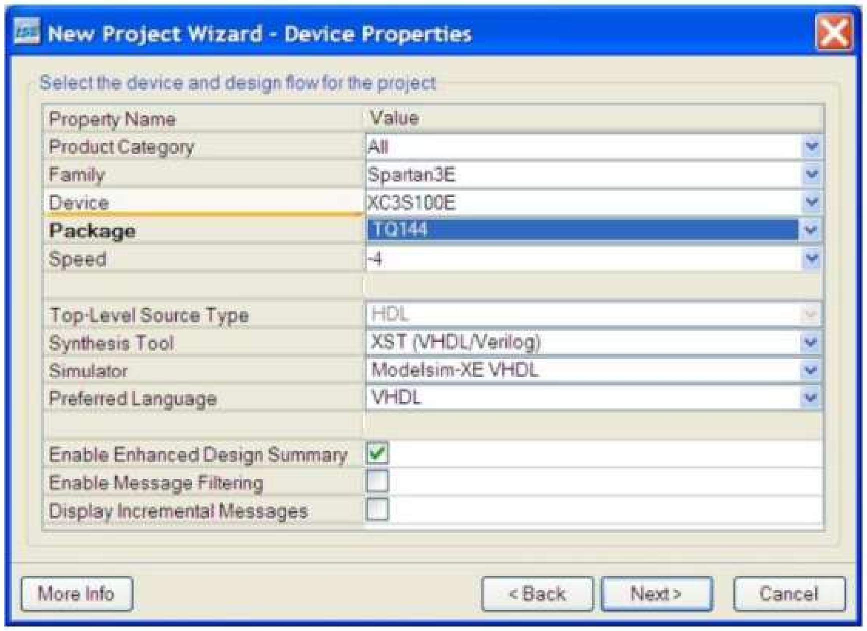


Figure 3. Digilent Basys AND Gate Application

AND gate and its associated truth table are shown in Figure 2. It is required to realize this gate on the Digilent Basys board so that the action of switches SW0 and SW1 (ON/OFF) correspond to X, Y inputs in the truth table and Z output corresponds to the LED0 (lit UP/turned OFF).

Procedure

Proceed with ISE as you would for a software behavioral simulation project except that now you have to configure the project for a specific FPGA which is XC3S100E-TQ144 on the Digilent's Basys board.


New Project Wizard - Device Properties

Select the device and design flow for the project

Property Name	Value
Product Category	All
Family	Spartan3E
Device	XC3S100E
Package	TQ144
Speed	-4
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	Modelsim-XE VHDL
Preferred Language	VHDL
Enable Enhanced Design Summary	✓
Enable Message Filtering	☐
Display Incremental Messages	☐

Figure 4. Device Properties Configuration

Add a new source file as shown:

Add to Project: Yes
Source Directory: C:\...\Laboratory\Lab_3\Part_1\ISE\ee120a_L3P1_basys_AND_gate
Source Type: Schematic
Source Name: and_gate.sch

Table 1. and_gate top level source (code) schematic

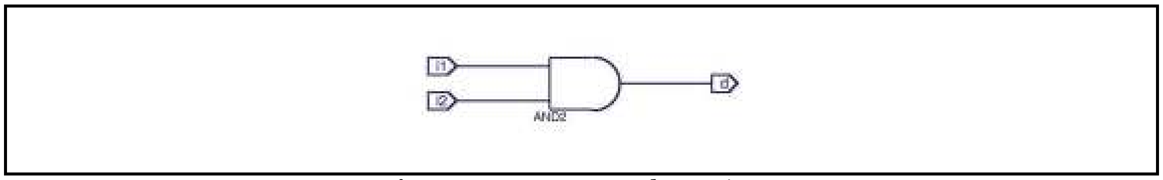


Figure 5. AND gate schematic entry

We need to verify our "design" by creating a testbench file and_gate_tb (click New Source in the Processes panel) and using for example the following info:

Add to Project: Yes_________________
Source Directory: C:\...\Laboratory\Lab_3\Part_1\ISE\ee120a_L3P1_basys_AND_gate
Source Type: Test Bench Waveform
Source Name: and_gate_tb.tb___________
Association: and_gate

Table 2. Testbench entry

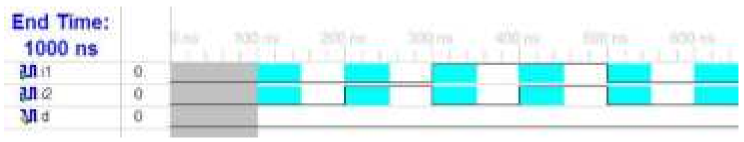


Figure 6. Testbench example timings

Select the testbench and_gate_tb in the Sources for Behavioral Simulation then go to Processes panel and click "ModelSim Simulator -> Simulate Behavioral Model" to perform the software simulation/verification of the logic design:

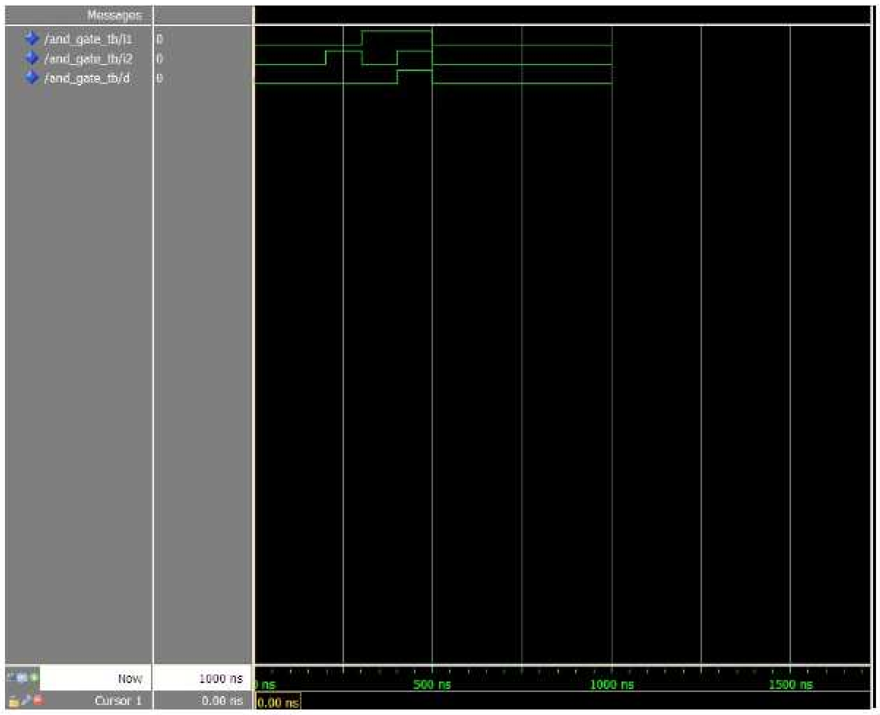


Figure 7. ModelSim Logic Verification of **and_gate** Performance

Observe, only when i1,i2 = 1,1 then d=1, otherwise d=0 as needed.

NOTE in passing, internally ISE creates a HDL source code for the schematic entered graphically (**`and_gate.vf`**) *which should not be modified*. It is the code (or a set of codes) that is submitted to ModelSim (or itself if so chosen) to perform the behavioral simulation. To view it, select **`and_gate`** in sources panel, go to the processes panel window "Design Utilities -> View HDL Functional Model":

```
//    This verilog netlist is translated from an ECS
schematic.It can be
//    synthesized and simulated, but it should not be
modified.

`timescale 1ns / 1ps
module and_gate(i1,  i2,  d);

  input i1;
  input i2;
  output d;

   AND2  gate_and (.I0(i2),
                  .I1(i1),
                  .O(d));
endmodule
```

Listing 1. Xilinx ISE Internal VHDL code for **`and_gate.sch`**

Synthesis and Routing

Right-click on **and_gate** in the Sources panel "New Source -> Implementation Constraints -> and_gate". This creates a constraint file **`add_gate.ucf`**. Select it and go to the Processes panel than "User Constraints -> Edit Constraints" and enter information as shown in **Listing 2**. We need to assign Spartan-3E FPGA pins to the corresponding components on the Basys board.

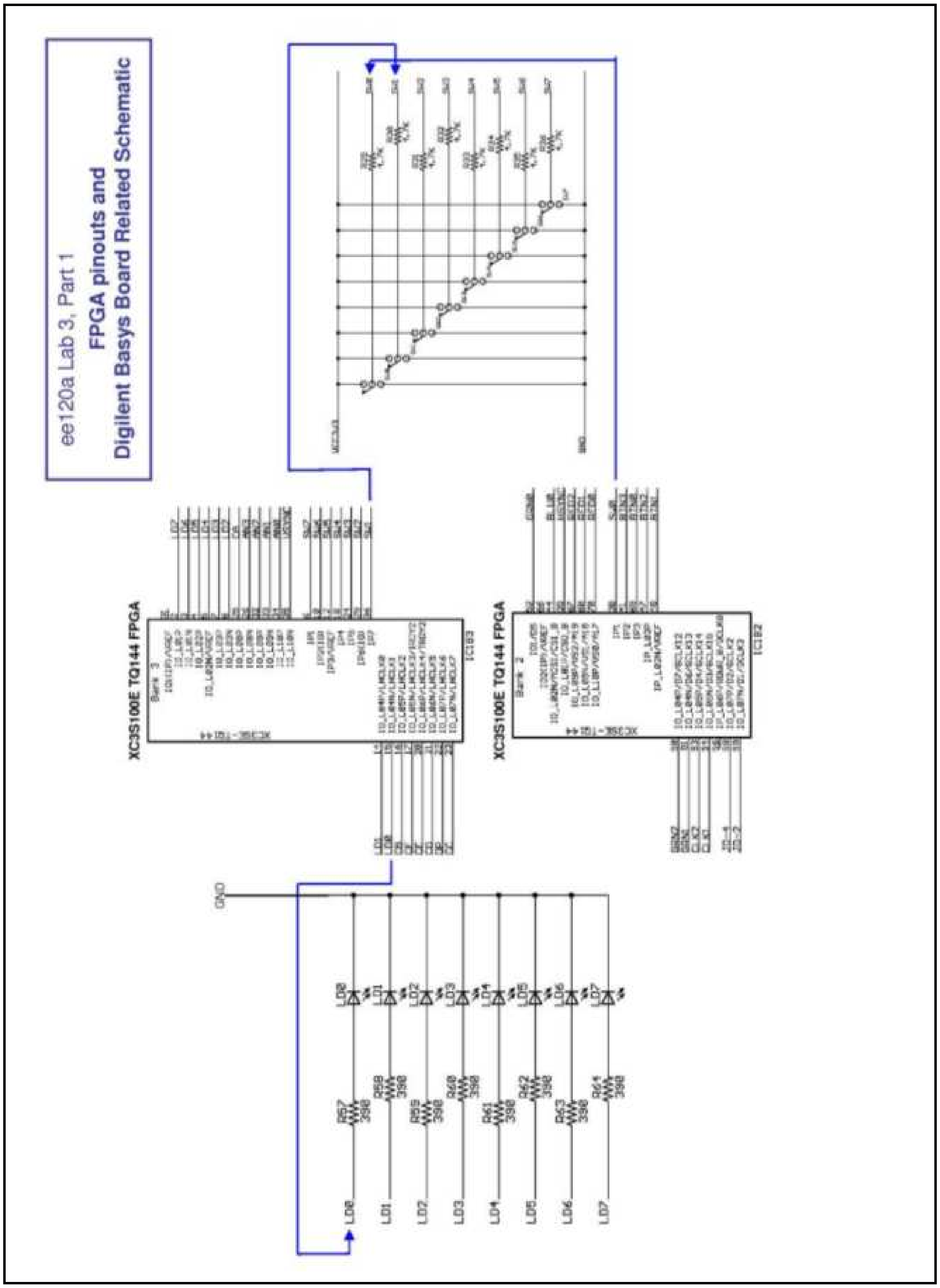

ee120a Lab 3, Part 1
FPGA pinouts and
Digilent Basys Board Related Schematic
XC3S100E TQ144 FPGA
XC3S100E TQ144 FPGA
IC1B3
IC1B2
VCC3V3
GND
GND
LD0
LD1
LD2
LD3
LD4
LD5
LD6
LD7
390

Figure 8. Digilent Basys Board Schematic related to FPGA signal routing

```
# Pin assignment for LEDs
NET "d" LOC = "p15" ; # Bank = 3, Signal name = LD0

# Pin assignment for SWs
NET "i1" LOC = "p36"; # Bank = 3, Signal name = SW1
NET "i2" LOC = "p38"; # Bank = 2, Signal name = SW0
```

Listing 2. Configuration file **`and_gate.ucf`**

We need t

connected directly to components provided by the Basys board as shown in **Figure 8**.

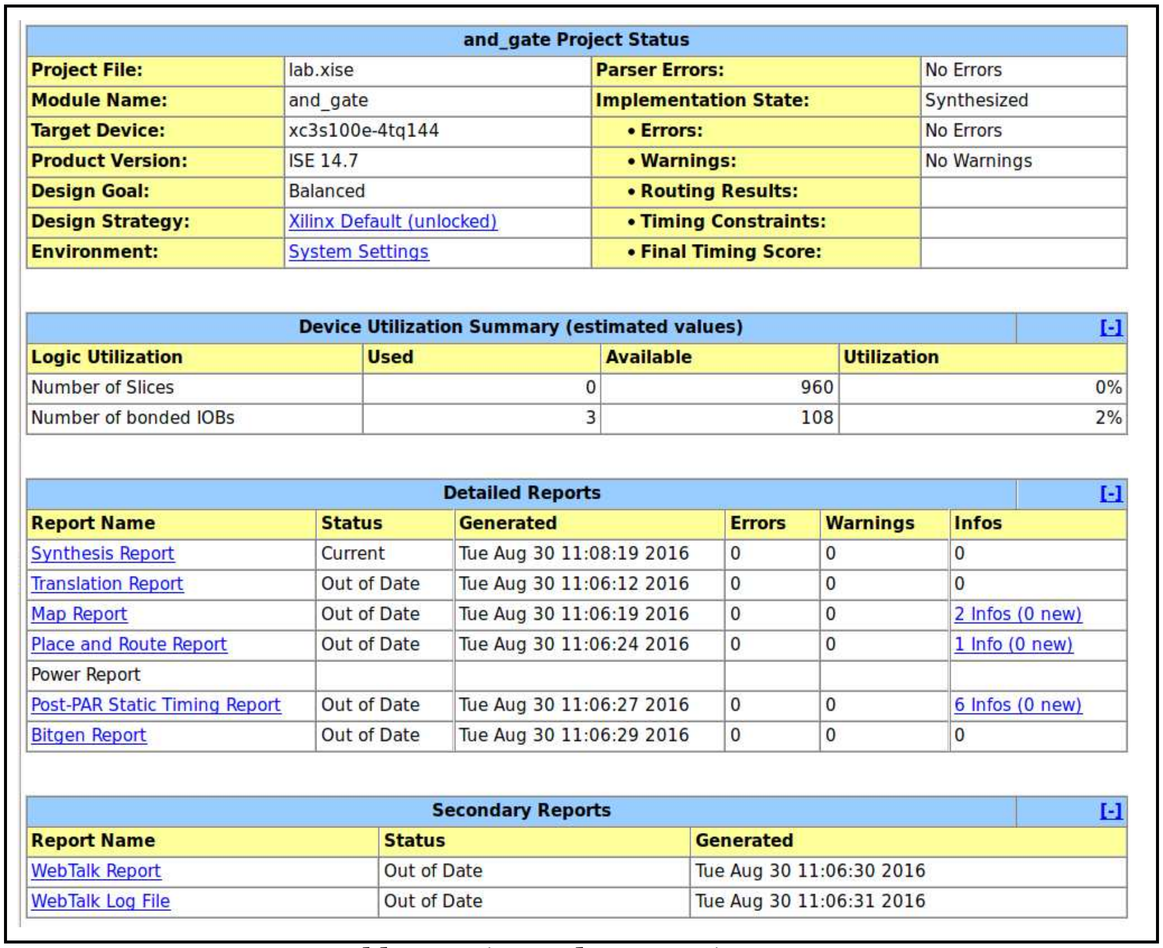
Save the constraint file. Go to Sources panel, select **and_gate** and in the Processes panel click "Synthesize - XST". At this point ISE synthesized its internal code with its own internal logic that MAY BE DIFFERENT from what you think it is. This depends very much on the FPGA architecture and how it creates its logic functions.

Go to Sources panel, select and_gate and in the Processes panel click "Implement Desig". Now ISE did all the mapping and routing to FPGA pins

Generating FPGA Programming File

CRITICAL!!! In the Processes panel right-click on "Generate Programming File", choose Properties and in the pop-up window select "Startup Options". Make sure that **FPGA startup clock** is **JTAG Clock** (not the default CCLK).

Now, by clicking "Generate Programming File" in the Processes panel we create a file **and_gate.bit** which we will download to the FPGA in the Digilent Basys board.



and_gate Project Status			
Project File:	lab.xise	**Parser Errors:**	No Errors
Module Name:	and_gate	**Implementation State:**	Synthesized
Target Device:	xc3s100e-4tq144	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	No Warnings
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values) [-]			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	0	960	0%
Number of bonded IOBs	3	108	2%

Detailed Reports [-]					
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Tue Aug 30 11:08:19 2016	0	0	0
Translation Report	Out of Date	Tue Aug 30 11:06:12 2016	0	0	0
Map Report	Out of Date	Tue Aug 30 11:06:19 2016	0	0	2 Infos (0 new)
Place and Route Report	Out of Date	Tue Aug 30 11:06:24 2016	0	0	1 Info (0 new)
Power Report					
Post-PAR Static Timing Report	Out of Date	Tue Aug 30 11:06:27 2016	0	0	6 Infos (0 new)
Bitgen Report	Out of Date	Tue Aug 30 11:06:29 2016	0	0	0

Secondary Reports [-]		
Report Name	**Status**	**Generated**
WebTalk Report	Out of Date	Tue Aug 30 11:06:30 2016
WebTalk Log File	Out of Date	Tue Aug 30 11:06:31 2016

Table 3. Project **and_gate** Design Summary

If all goes well we end up with a bit file (**and_gate.bit**) in the project directory with a final FPGA configuration that can be downloaded to the FPGA on the Basys board and which hopefully does what we need it to do.

Digilent Basys Board Setup and Programming

Make sure that

1. The board is set to be powered by USB port;

2. **ROM/JTAG** jumper is located at **JTAG**

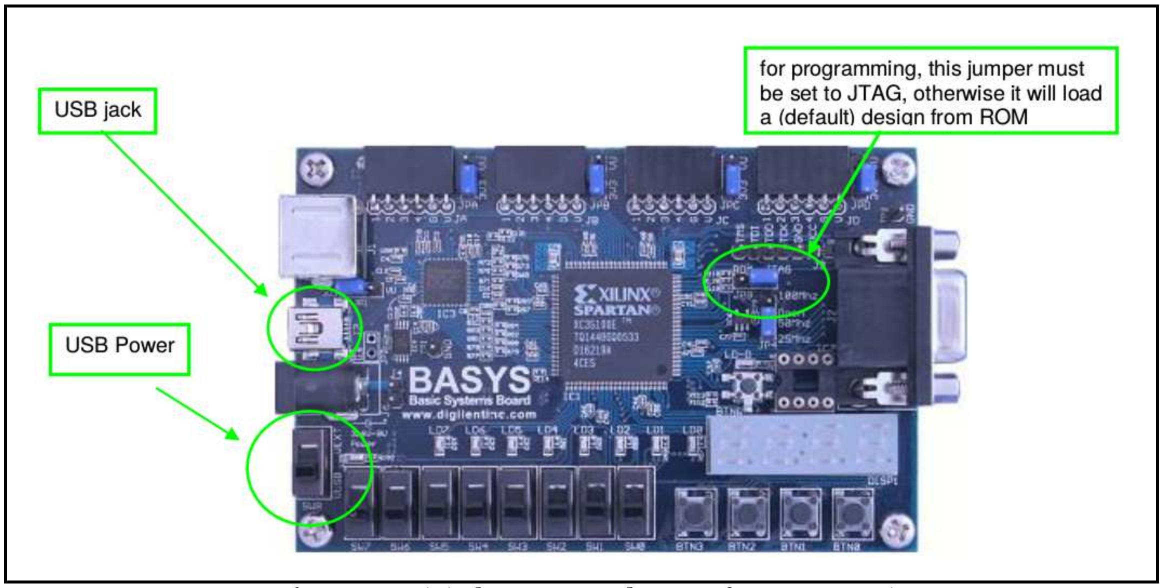
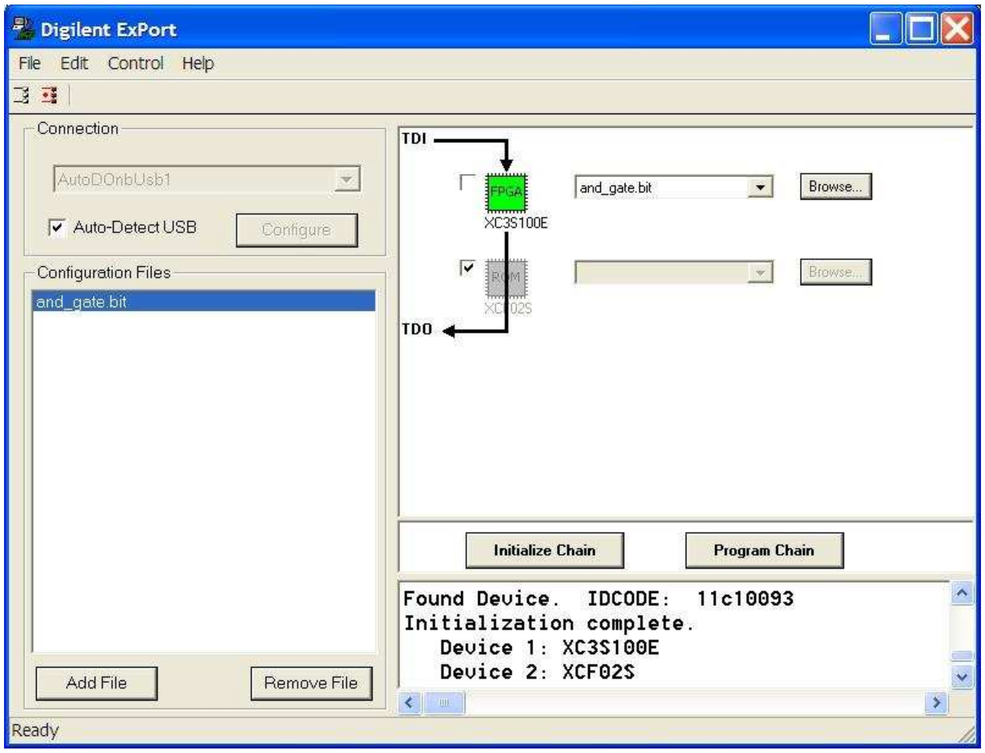

USB jack
for programming, this jumper must be set to JTAG, otherwise it will load a (default) design from ROM
USB Power

Figure 9. Digital Basys Board set-up for programming

Start the Digilent Adept ExPort program , "Add " the bit file and_gate.bit created by ISE in the project's directory and follow the procedure reproduced in the **Figure 10** below and click "ProgramChain" to do the actual download of the bit code into FPGA.



```
Found Device.  IDCODE:  11c10093
Initialization complete.
   Device 1: XC3S100E
   Device 2: XCF02S
```

Figure 10. Digital Basys Board Programming with Adept tool ExPort

At this point the on-board FPGA is programmed with the application. By playing with switches **SW0** and **SW1** observe the reaction of **LED0**.

<u>Demostration</u>

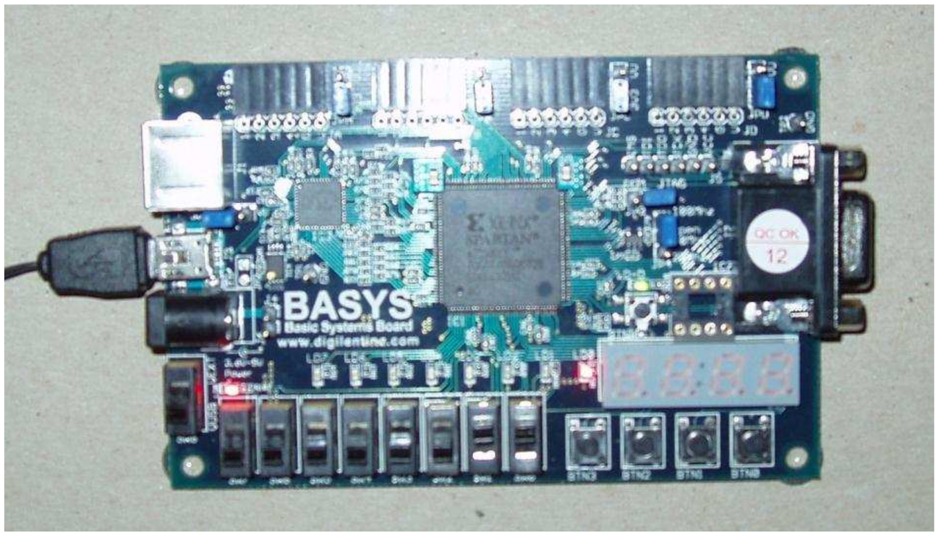


_________Figure 11. and_gate: SW1=1 (i1), SW0=1 (i2): LED0 is ON (d) as needed

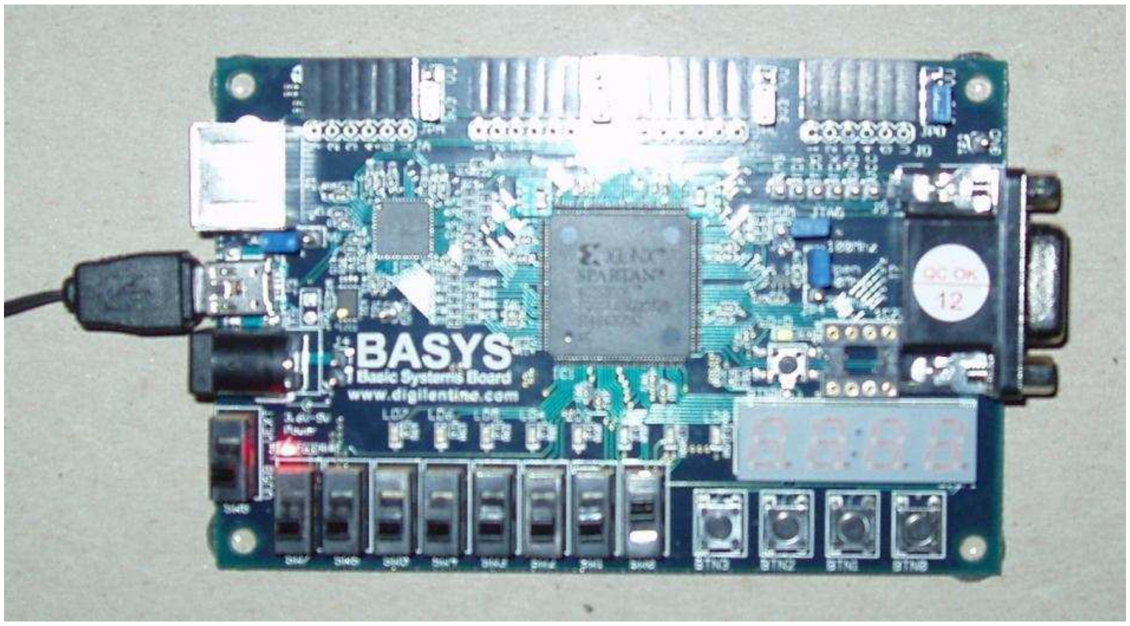


Figure 12. and_gate: **SW1=0** (i1), **SW0=1** (i2): **LED0** is **OFF** (d) as needed

Questions

1. Can there be a difference in logical behavior between the intended logic entered and simulated and, the logic actually synthesized for FPGA? Why?

2. Why do we need a configuration file?

3. Is there a functional difference in circuitry between **Lab 1, Part 3** and **Basys board** for this particular application?

4. What must be done in order to use switches **SW3** and **SW7** instead of **SW0** and **SW1**? How about using **LED5** instead of **LED0**?

Conclusion for Part 1

We have gone through the whole cycle of system design, analysis, synthesis and FPGA based hardware implementation of a combinational logic application.

PART 2. Implementation of Sprinkler Controller

Specification

In this assignment it is required to actually implement the Sprinkler Controller system on Digilent's Basys Board so that

1. Use the Specs from Lab 2 regarding the system function;
2. Assume that on-board LEDs act as sprinkler valves;
3. Control the sprinkler valves using Basys switches SW2 = A, SW1 = B, SW0 = C and SW7 = E

Demonstration

For completeness provide in the report the logic circuits used.

PART 3. BCD to 7 Segment LED Display

Specification

Part A: In this this assignment it is required to implement the **schematics** of a BCD-to-7seg decoder so that switches **SW[3:0]** control the decimal number displayed on the rigt-most LED Diplay **AN3** on the Basys Board.

Part B: In this part of the lab you are required to implement the structural (slow to code) or behavioral (faster to code) description of the BCD-to-7seg decoder described above. The module you are about to implement should have the set of input and output ports given in listing 3. In here, we have provided a basic template that can be used in the task at hand.

```
module bcd_to_7led_bh (
 input wire sw0 ,   // Switches
 input wire sw1 ,
 input wire sw2 ,
 input wire sw3 ,
  output reg a ,  // LED segments
 output reg b ,
 output reg c ,
 output reg d ,
 output reg e ,
 output reg f ,
 output reg g ,
 output reg an0,  // LED display control
 output reg an1,
 output reg an2,
 output reg an3
  );

// Internal wire
wire [3:0] bundle ;
assign bundle = {sw3,sw2,sw1,sw0 } ;

always @(*) begin

     // Setting the ANs signals
     an0 = 1'b1;
     an1 = 1'b1;
     an2 = 1'b1;
     an3 = 1'b0; // Display in the module AN3
     // Setting the segments signals
     a = 1'b1 ;
     b = 1'b1 ;
     c = 1'b1 ;
     d = 1'b1 ;
     e = 1'b1 ;
     f = 1'b1 ;
     g = 1'b1 ;

 case ( bundle )

     4'b0000 : begin // 0
          a = 1'b0 ;
          b = 1'b0 ;
          c = 1'b0 ;
          d = 1'b0 ;
          e = 1'b0 ;
          f = 1'b0 ;
     end

// Your code goes here

     endcase
end

endmodule
```

Listing 3. BCD-to-7seg decoder behavioral model

Recall from previous lab notes that the AN signals are required to be low for the 7-segments units to work. In addition, to facilitate the implementation of the BCD to 7-Seg module, the following set of constraints are given.

```
# Inputs
NET "sw0" LOC = "p38";
NET "sw1" LOC = "p36";
NET "sw2" LOC = "p29";
NET "sw3" LOC = "p24";

# Outputs
NET "a" LOC = "p25";
NET "b" LOC = "p16";
NET "c" LOC = "p23";
NET "d" LOC = "p21";
NET "e" LOC = "p20";
NET "f" LOC = "p17";
NET "g" LOC = "p83";

// ANx
NET "an0" LOC = "p26";
NET "an1" LOC = "p32";
NET "an2" LOC = "p33";
NET "an3" LOC = "p34";
```

Demonstration

Provide the truth tables, circuit schematic and functionality of the design.

Procedures

1. Xilinx ISE Design and Synthesis environment;
2. Creation of Configuration files;
3. Usage of Adept ExPort download software;

Presentation and Report

Must be presented according to the general EE120A lab guidelines posted in iLearn.

Prelab

1. Familiarize yourself with ISE and ModelSim tutorials posted in iLearn.
2. Review Lectures 1-8.
3. Try to answer all the questions, prepare logic truth tables, do all necessary computations.