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

Programming Combinational Logic on Basys FPGA Board

PART 1[*] **Design, FPGA Synthesis and Testing of an AND Gate**

PART 2 **Implementation of Sprinkler Controller**

PART 3 **BCD to 7 Segment LED Display**

[*] Design Example guided through by TA

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.

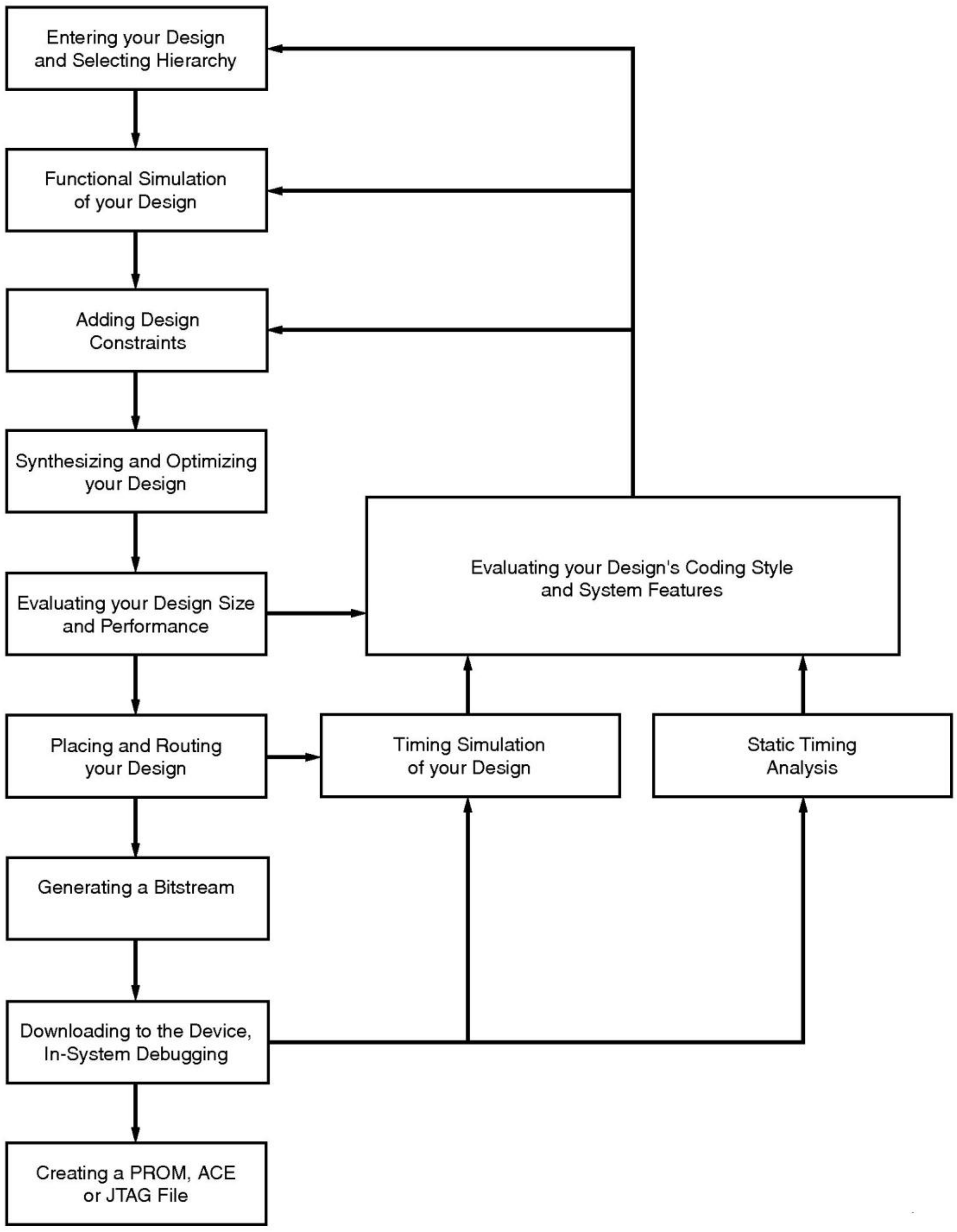
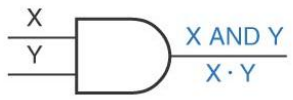

Entering your Design and Selecting Hierarchy
Functional Simulation of your Design
Adding Design Constraints
Synthesizing and Optimizing your Design
Evaluating your Design Size and Performance
Placing and Routing your Design
Generating a Bitstream
Downloading to the Device, In-System Debugging
Creating 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

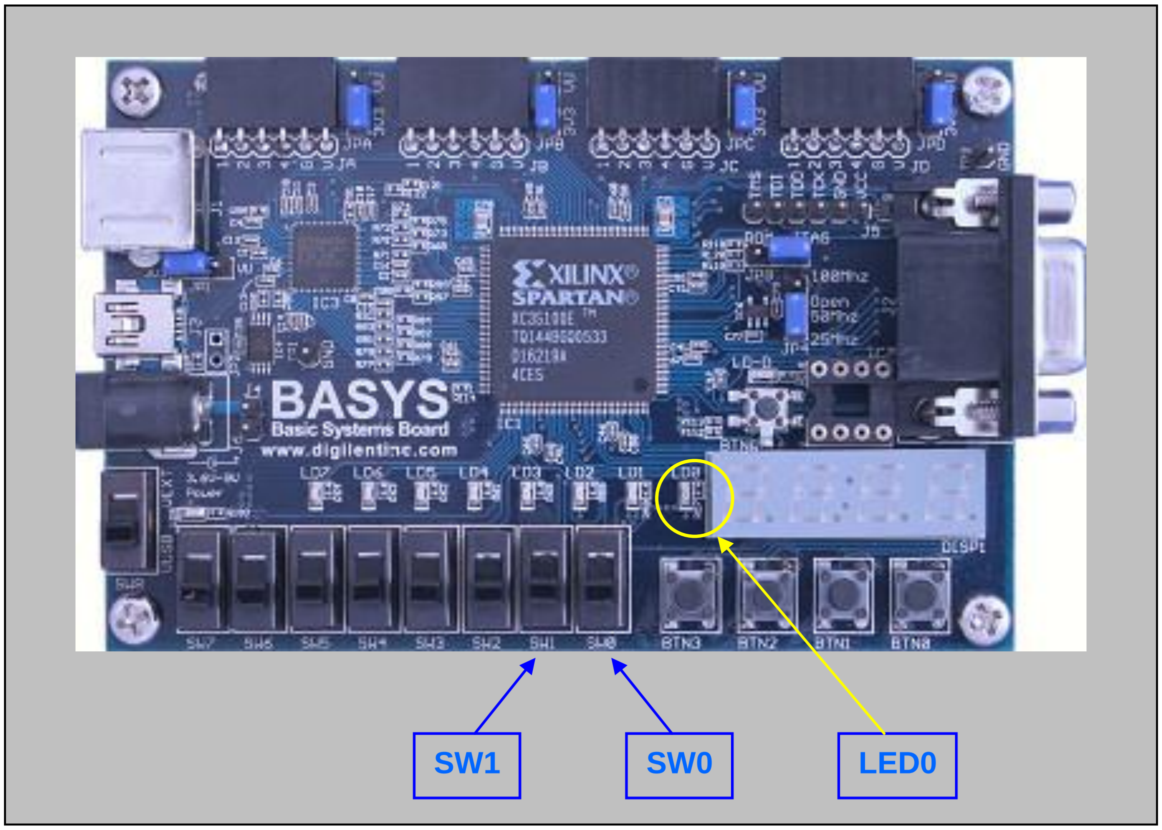


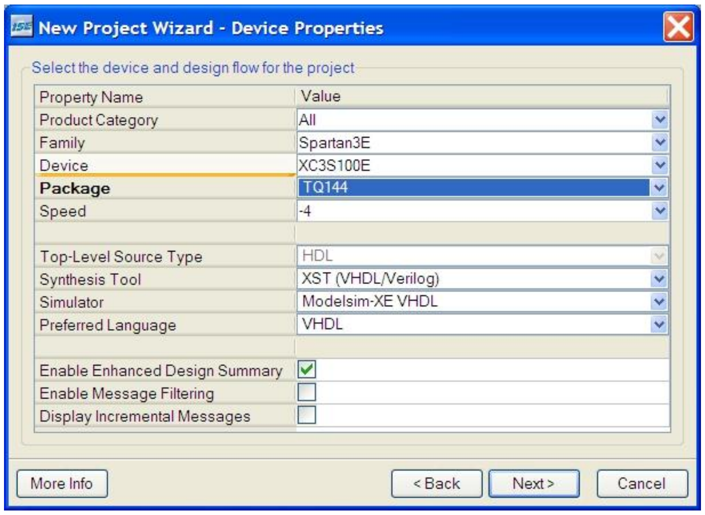
Figure 3. Digilent Basys AND Gate Application

[†] Guided by TA

AND gate and its associated truth table are shown in **Figure 1**. 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.



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 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

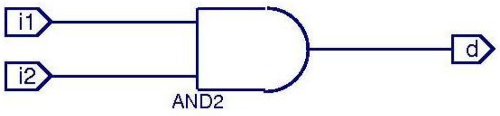


Figure 5. AND gate schematic entry

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.tbw**
Association:	**and_gate**

Table 2. Testbench entry

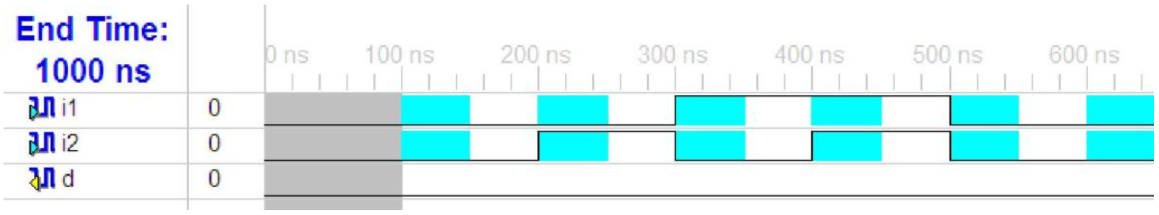


Figure 6. Testbench example timings

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:

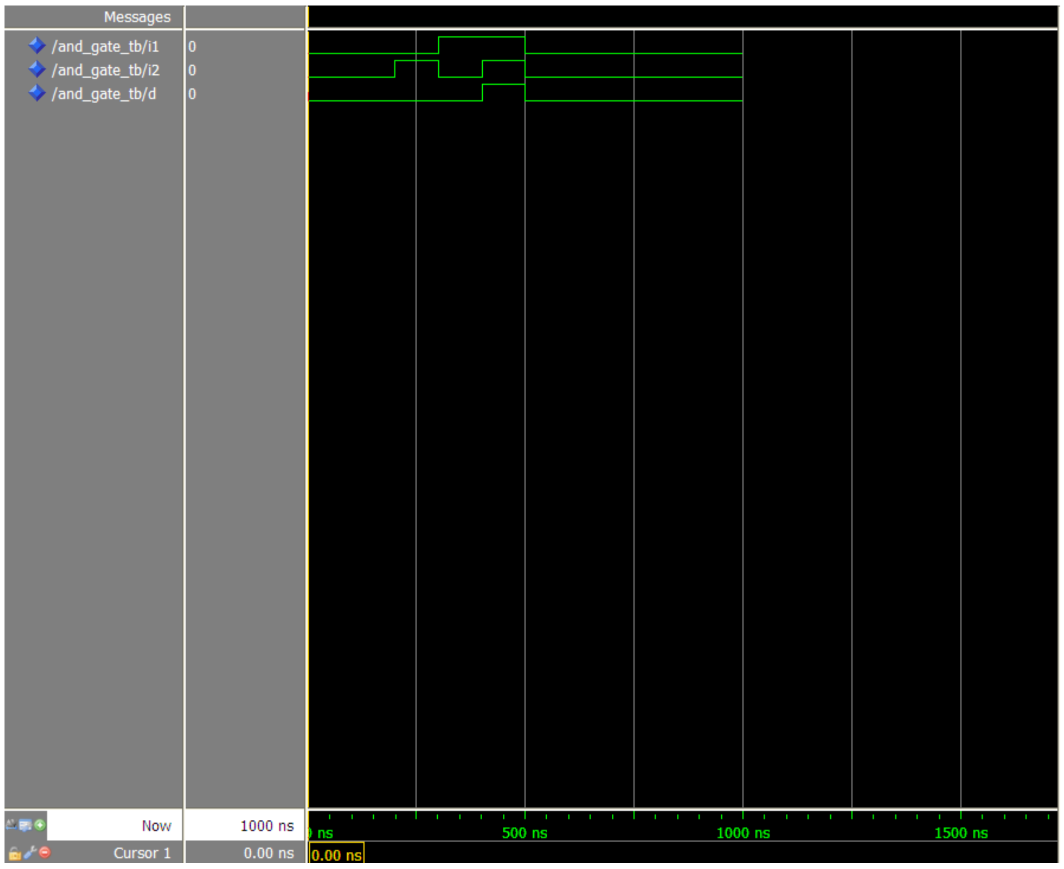


Figure 7. ModelSim Logic Verification of **and_gate** Performance

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

NOTE in passing, internally ISE creates a VHDL source code for the schematic entered graphically (**`and_gate.vhf`**[‡]) *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 VHDL Functional Model":

```
--   ____  ____
--  /   /\/   /
-- /___/  \  /    Vendor: Xilinx
-- \   \   \/     Version : 10.1.03
--  \   \         Application : sch2vhdl
--  /   /         Filename : and_gate.vhf
-- /___/   /\     Timestamp : 01/24/2009 16:39:43
-- \   \  /  \
--  \___\/\___\
--Command:                                            C:\Programs\Xilinx\ISE-
WebPack\10.1\ISE\bin\nt\unwrapped\sch2vhdl.exe    -intstyle    ise    -family
spartan3e                 -flat                 -suppress                  -w
"C:/…/Laboratory/Lab_3/Part_1/ISE/ee120a_L3P1_basys_AND_gate/and_gate.sch"
and_gate.vhf
--Design Name: and_gate
--Device: spartan3e
--Purpose:
--    This vhdl netlist is translated from an ECS schematic. It can be
--    synthesis and simulted, but it should not be modified.
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity and_gate is
   port ( i1 : in    std_logic;
          i2 : in    std_logic;
          d  : out   std_logic);
end and_gate;

architecture BEHAVIORAL of and_gate is
   attribute BOX_TYPE   : string ;
   component AND2
      port ( I0 : in    std_logic;
             I1 : in    std_logic;
             O  : out   std_logic);
   end component;
   attribute BOX_TYPE of AND2 : component is "BLACK_BOX";

begin
   XLXI_1 : AND2
      port map (I0=>i2,
                I1=>i1,
                O=>d);
end BEHAVIORAL;
```

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

[‡] VHDL file extension is .vhd which can be modified. Xilinx uses .vhf files for non-modifiable files

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.

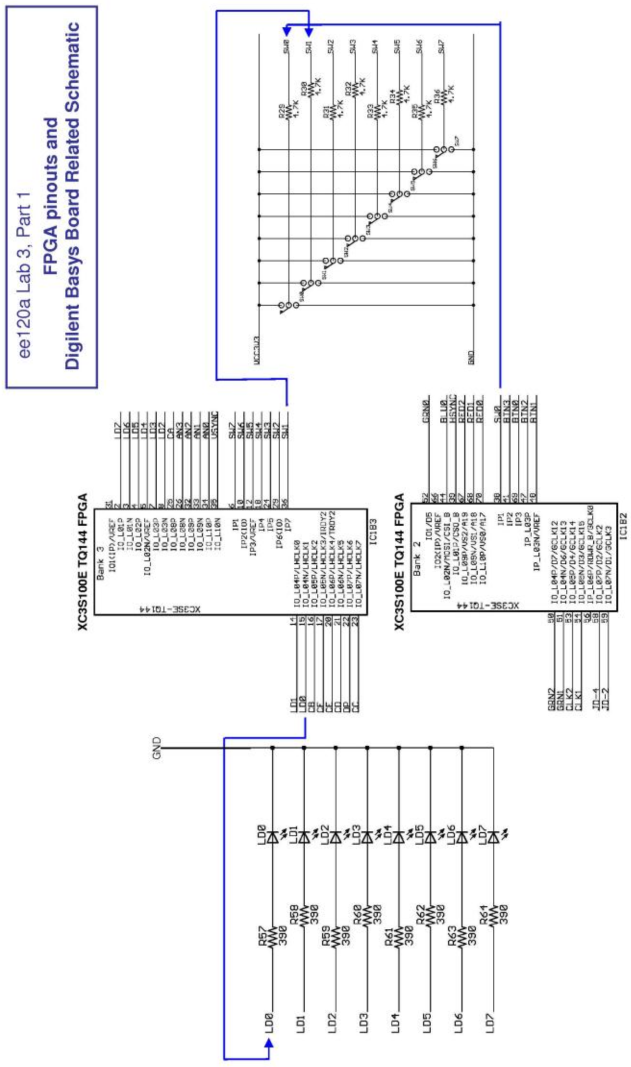


Figure 8. Digilent Basys Board Schematic[§] related to FPGA signal routing

[§] Compare to ***Lab 1*** hardware components and schematic. Check out also ***Lecture 2***

```
# Project: ee120a_L3P1_AND_gate
# File:    and_gate.ucf
# Dreated: R.Chomko
# Date:    2009/01/24

# Pin assignment for LEDs

NET "d" LOC = "p15" ; # Bank = 3, Signal name = LD0

#NET "Led<7>" LOC = "p2" ; # Bank = 3, Signal name = LD7
#NET "Led<6>" LOC = "p3" ; # Bank = 3, Signal name = LD6
#NET "Led<5>" LOC = "p4" ; # Bank = 3, Signal name = LD5
#NET "Led<4>" LOC = "p5" ; # Bank = 3, Signal name = LD4
#NET "Led<3>" LOC = "p7" ; # Bank = 3, Signal name = LD3
#NET "Led<2>" LOC = "p8" ; # Bank = 3, Signal name = LD2
#NET "Led<1>" LOC = "p14" ; # Bank = 3, Signal name = LD1
#NET "Led<0>" 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

#NET "sw<7>" LOC = "p6"; # Bank = 3, Signal name = SW7
#NET "sw<6>" LOC = "p10"; # Bank = 3, Signal name = SW6
#NET "sw<5>" LOC = "p12"; # Bank = 3, Signal name = SW5
#NET "sw<4>" LOC = "p18"; # Bank = 3, Signal name = SW4
#NET "sw<3>" LOC = "p24"; # Bank = 3, Signal name = SW3
#NET "sw<2>" LOC = "p29"; # Bank = 3, Signal name = SW2
#NET "sw<1>" LOC = "p36"; # Bank = 3, Signal name = SW1
#NET "sw<0>" LOC = "p38"; # Bank = 2, Signal name = SW0
```

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

We need to do it since ISE while synthesizing internal configuration doesn't know which of its pins to route the inputs and outputs of the design to. Of course, we want to direct ISE to use FPGA pins that are 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.

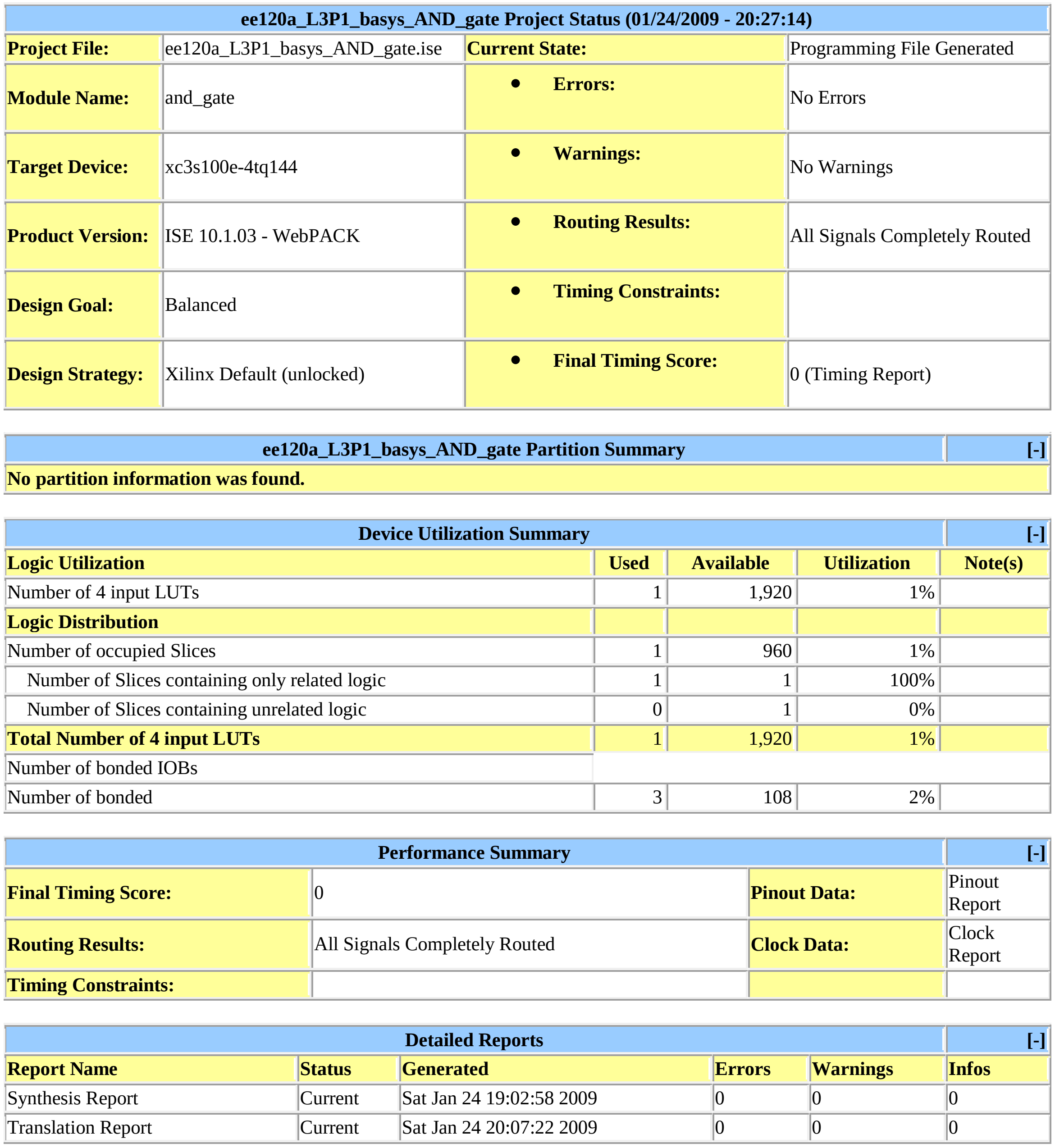
Go to Sources panel, select **and_gate** and in the Processes panel click "Implement Design". Now ISE did all the mapping and routing to FPGA pins.

[**] Observe that FPGA pins must be prefixed with "p" or "P" in .ucf files. Also, **IMPORTANT !!!** a template configuration file that contains all the information related to components on the Digilent Basys Board is uploaded into iLearn: ***Basys_Configuration_Template.ucf***

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.



ee120a_L3P1_basys_AND_gate Project Status (01/24/2009 - 20:27:14)			
Project File:	ee120a_L3P1_basys_AND_gate.ise	**Current State:**	Programming File Generated
Module Name:	and_gate	• **Errors:**	No Errors
Target Device:	xc3s100e-4tq144	• **Warnings:**	No Warnings
Product Version:	ISE 10.1.03 - WebPACK	• **Routing Results:**	All Signals Completely Routed
Design Goal:	Balanced	• **Timing Constraints:**	
Design Strategy:	Xilinx Default (unlocked)	• **Final Timing Score:**	0 (Timing Report)

ee120a_L3P1_basys_AND_gate Partition Summary	[-]
No partition information was found.	

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	1	1,920	1%	
Logic Distribution				
Number of occupied Slices	1	960	1%	
Number of Slices containing only related logic	1	1	100%	
Number of Slices containing unrelated logic	0	1	0%	
Total Number of 4 input LUTs	1	1,920	1%	
Number of bonded IOBs				
Number of bonded	3	108	2%	

Performance Summary			[-]
Final Timing Score:	0	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:			

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Sat Jan 24 19:02:58 2009	0	0	0
Translation Report	Current	Sat Jan 24 20:07:22 2009	0	0	0

Map Report	Current	Sat Jan 24 20:07:34 2009	0	0	2 Infos
Place and Route Report	Current	Sat Jan 24 20:07:43 2009	0	0	1 Info
Static Timing Report	Current	Sat Jan 24 20:07:46 2009	0	0	3 Infos
Bitgen Report	Current	Sat Jan 24 20:27:13 2009	0	0	0

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.

<u>Digilent Basys Board Setup and Programming</u>

Make sure that

1. The board is set to be powered by USB port;
2. **ROM/JTAG** jumper is located at **JTAG**[††]

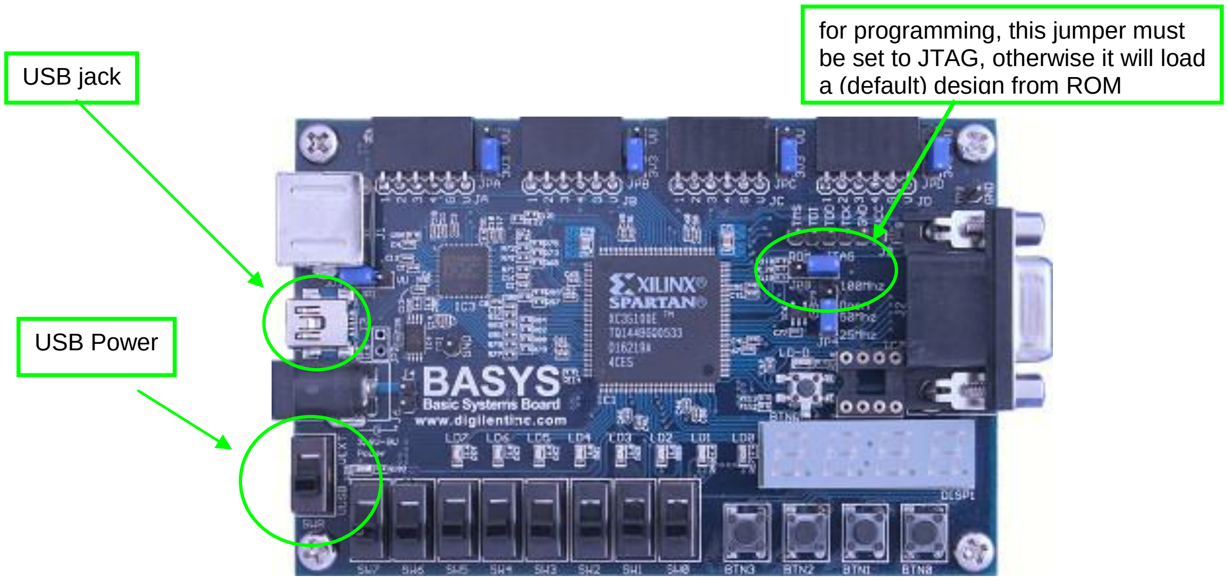
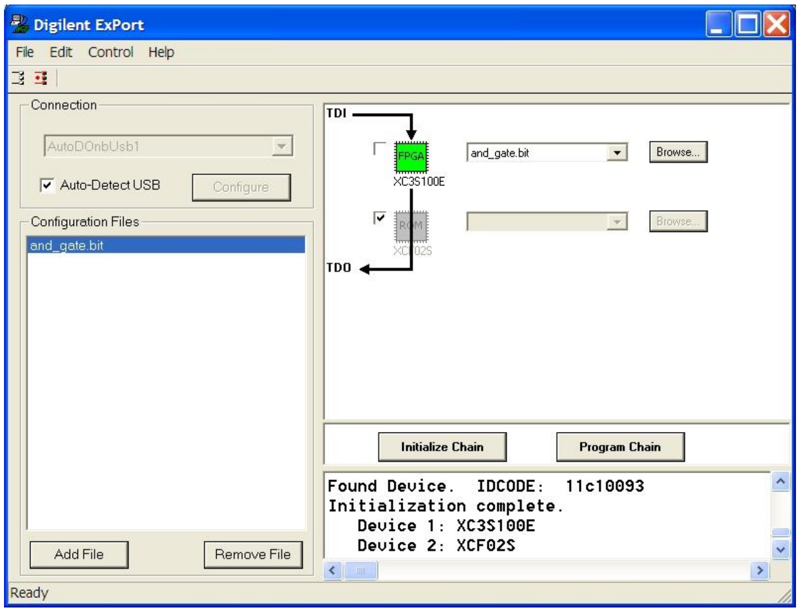

USB jack
USB Power
for programming, this jumper must be set to JTAG, otherwise it will load a (default) design from ROM

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

[††] Otherwise it will load a design from the on-board ROM. We will be programming FPGA directly with JTAG in EE120A labs. Programming ROM, so designs are permanently built into the board, is not difficult.
[‡‡] Xilinx devices are typically programmed with JTAG cables. In order to program the board with a JTAG cable Xilinx iMPACT software is used. Since we are programming FPGA with a USB cable we will have to use Digilent's proprietary Adept software. Note if you have Digilent's JTAG-USB cable you can use iMPACT following the identical procedure with one exception: iMPACT can be called directly from within ISE.

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**.

Demonstration

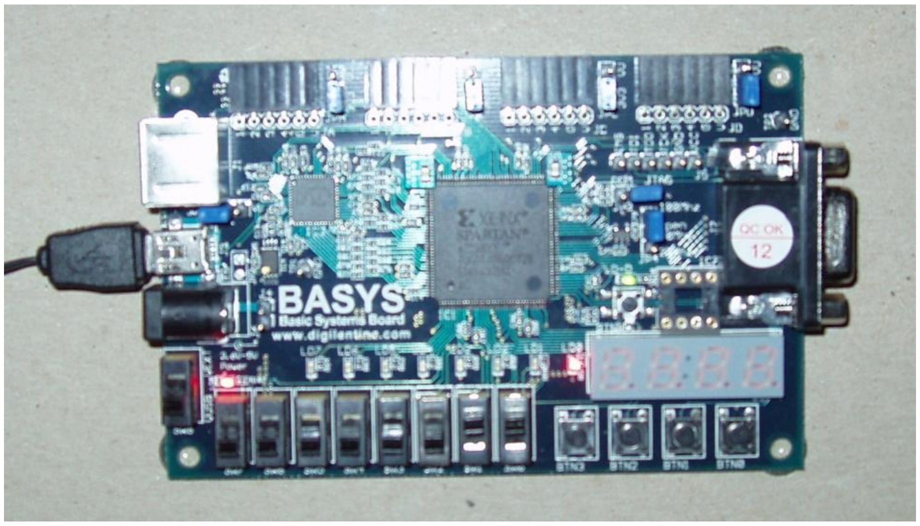


Figure 11. and_gate: **SW1=1** (i1), **SW0=1** (i2): **LED0** is **ON** (d) as needed

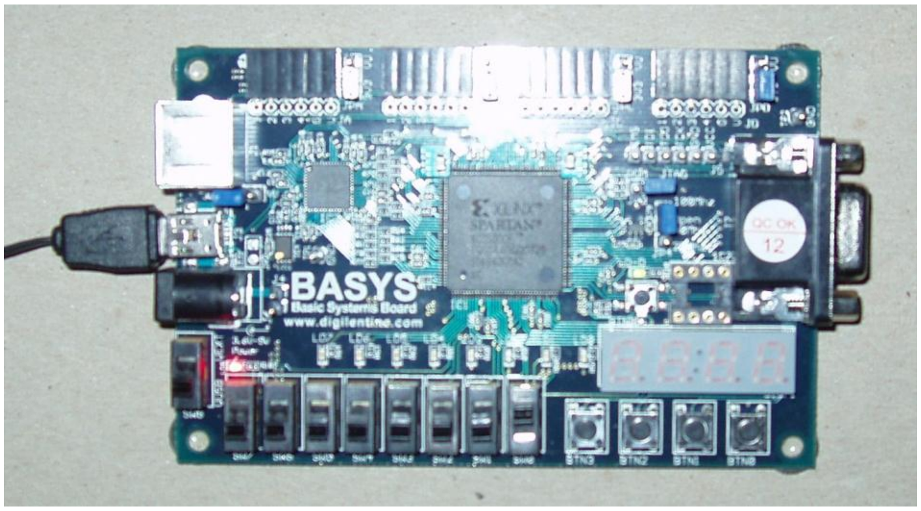


Figure 12. and_gate: **SW1=0** (i1), **SW0=1** (i2): **LED0** is **OFF** (d) as needed

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

In this assithis assignment it is required to implement 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

Demonstration

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

[§§] Review the material of Lab 2
[***] Review the material of Lab 1 and corresponding Lectures

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