Department of Electrical Engineering

ASIC/FPGA Chip Design

Laboratory Manual

ASIC/FPGA Chip Design Laboratory Manual

Dr. Mahdi Shabany

Spring 2025

Contents

Department of Electrical Engineering

ASIC/FPGA Chip Design

Lab 1 - Introduction to FPGA

Lab 1 - Introduction to FPGA

1.1 Objective

In this lab, you will describe digital circuits using Hardware Description Language (HDL) and implement them on the FPGA to test their functionality.

1.2 Introduction

FPGAs are used to create any specific hardware you want, there is almost no built-in fixed hardware on the chip, you need to create the hardware dedicated to performing your particular application. Because the designed hardware is customized for the specific task, you can benefit from its high performance and parallelism. The drawback might be high power consumption.
The board used in this lab includes the Zynq 7010 chip. To observe the results and test the implemented simple circuits, as well as to establish communication with the external world, we need to manage and monitor the circuit's input and output signals through the board's I/O interfaces. In practice, we connect the circuit inputs to a series of push buttons and the outputs to LEDs to observe its functionality after implementing it on the chip. The FPGA board used in this lab (which is a Zynq 7010 model manufactured by Xilinx) has four simple input push buttons and five single LEDs, all of which are connected to the PL (FPGA) side of the chip.

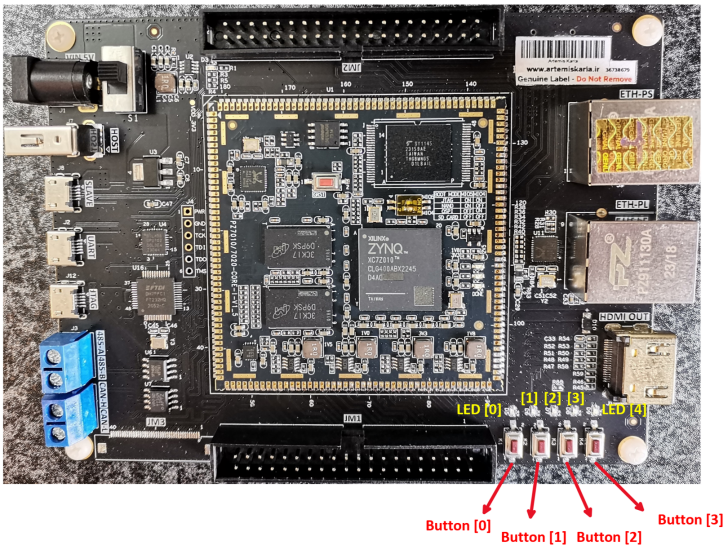

Figure 1.1: ZYNQ board (AXPZ7010) containing Zynq 7010 chip

To implement your described digital circuit on Zynq FPGAs, you need to use Xilinx's own software, Vivado. Before you begin the lab, please watch videos 1 to 6 from the Zynq tutorial series to learn more about Zynq, installing Vivado, becoming familiar with Vivado, performing simulations, synthesis, implementation, and programming the Zynq chip using Vivado. These tutorials build on your basic knowledge and preparation. Additionally, please read the following document thoroughly, as it should answer many of your questions. If you have any further questions or issues, feel free to ask!

Extension Board: To implement circuits that require a large number of input and output components, we need a large number of switches and LEDs. In this lab we are using an industrial board with fewer LEDs and switches compared to educational boards, also known as an Evaluation Board.

To address this issue and facilitate the implementation of more advanced circuits, an extension board is provided, adding the following peripherals to the board:

- 10 LEDs
- 8 Dip switches
- 2 Push buttons
- 4-digit 7-segment display
- LCD display

The main board is hardwired to some Zynq chip pins (for example, LED 5 on the main board), and several other pins have been left open by the manufacturer for user customization. If additional components are needed, two sets of expansion port are available, allowing an extension board to be connected, thereby increasing the number of inputs and outputs that can be connected to the board.

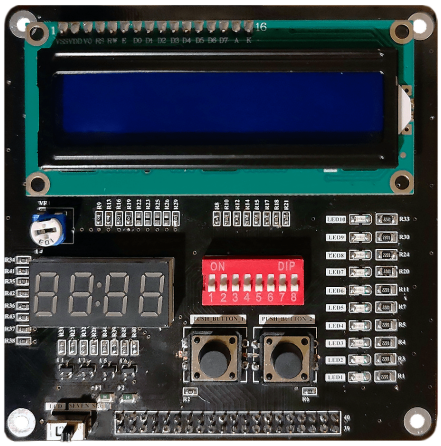

Figure 1.2: Extension Board

In the following table , you can see the connection of the extension board to the Zynq chip pins.

Pinout					
Id.	Ext. board	Zynq	Id.	Ext. board	Zynq
1	Vdd 5V	Vdd 5V	21	Vdd 3.3V	Vdd 3.3V
2	GND	GND	22	GND	GND
3	7-seg “dig3” LCD “RS”	B19	23	7-seg “dig2” LCD “RW”	G20
4	7-seg “dig1” LCD “EN”	D19	24	7-seg “dig0”	D20
5	7-seg “a” LCD “D0”	F16	25	7-seg “b” LCD “D1”	F17
6	7-seg “c” LCD “D2”	H15	26	7-seg “d” LCD “D3”	G15
7	7-seg “e” LCD “D4”	J18	27	7-seg “f” LCD “D5”	H18
8	7-seg “g” LCD “D6”	H16	28	7-seg “dp” LCD “D7”	H17
9	Push button1	J20	29	Push button2	H20
10	DIP 1	K14	30	DIP 2	J14
11	DIP 3	M14	31	DIP 4	M15
12	DIP 5	L19	32	DIP 6	L20
13	DIP 7	M19	33	DIP 8	M20
14	LED 10	N15	34	LED 9	N16
15	LED 8	C20	35	LED 7	B20
16	LED 6	E17	36	LED 5	D18
17	GND	GND	37	GND	GND
18	GND	GND	38	GND	GND
19	LED 4	E18	39	LED 3	E19
20	LED 2	G17	40	LED 1	G18

Figure 1.3: The interconnections between the extension board and the Zynq board

⚠ Important Notes

- LEDs, push buttons, and dip switches are Active HIGH.
- Dig3-Dig0 are Active HIGH but “a-g” and “dp” are Active LOW.
- Set the bottom-left switch to 7-seg mode.
- Connect the extension board via JM2 pins on the main board.

1.3 Instructions

1.3.1 Getting started with FPGA

1.3.1.1 Implementing a simple circuit

Describe the following circuit in Verilog using only **Continuous Assignment** statements.

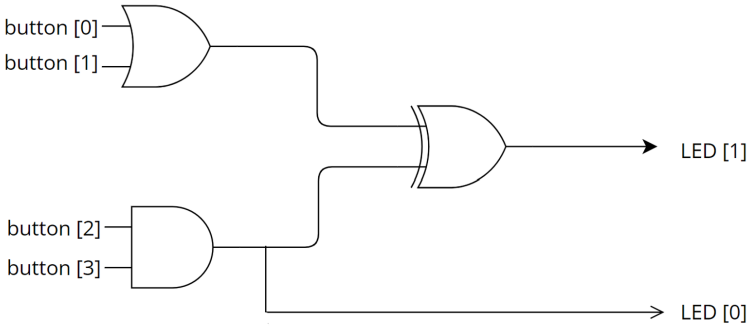

Figure 1.4: A simple circuit

You need to create the module, the constraint file and implement the circuit on the FPGA. You must use the board's user manual documentation to check which pin of the FPGA is connected to what peripheral of the board. Note that LEDs are **Active HIGH** but the keys are **Active LOW**.

1.3.1.2 Using Block Design in Vivado

At this stage, you will use the graphical environment (Block Design or Diagram) in Vivado. Vivado provides this graphical environment to make your work easier, allowing you to effortlessly add your modules, IP cores, and Xilinx IPs, connect them together, and visually comprehend the final hardware that will be implemented on the FPGA.
Follow this fellow:

- create a Block Design.

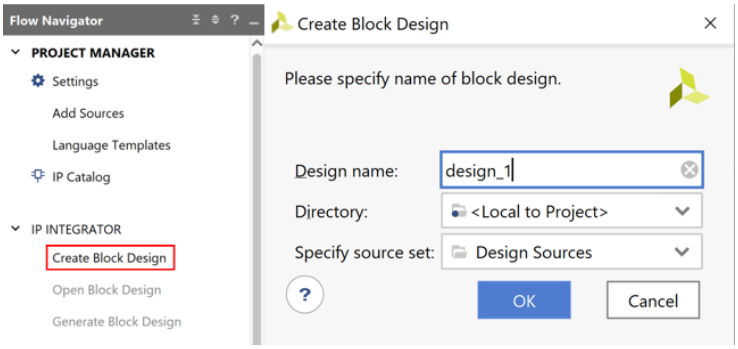

Figure 1.5: Add module to block design

- Add your module to the Vivado block design.

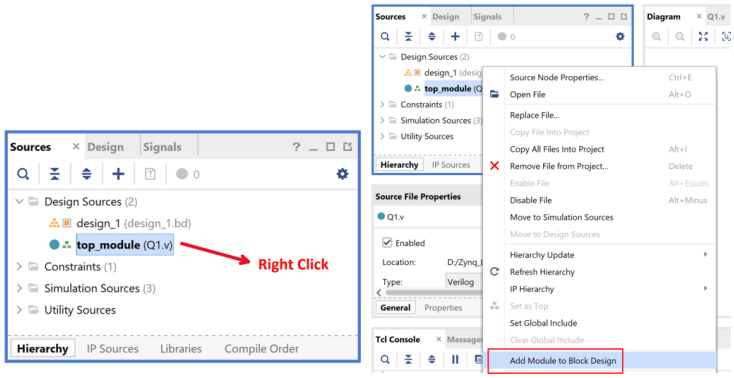

Figure 1.6: Add module to block design

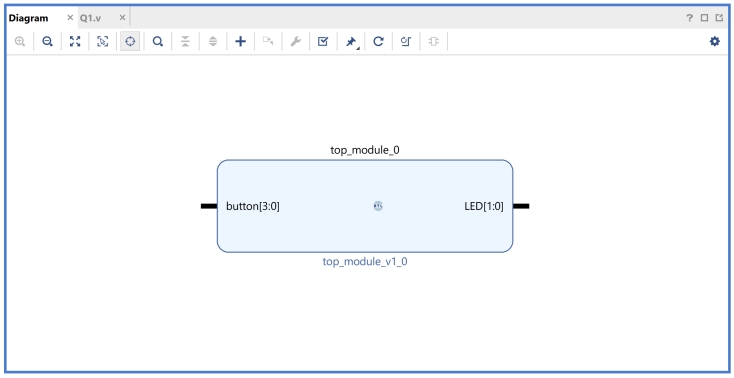

Figure 1.7: Module Graphical representation in the design

- Make External the inputs and outputs of your module so that it can connect to the chip pins.

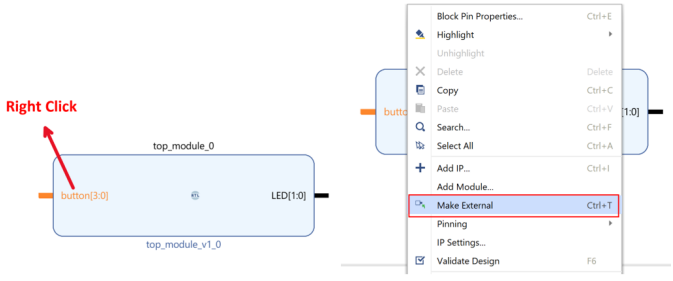

Figure 1.8: Make External

- Update your constraint file based on the new names of the external ports. (You can also change the name of any external port by clicking on it and using the External Port Properties window.)
- By selecting the **Validate Design** option, you can ensure the initial correctness of your graphical diagram in terms of Connection types , Matching bit-widths of connected ports , Consistency of Active High and Active Low signals , Proper clock and reset connections and other basic validations.
  Note: This option only checks for basic diagram correctness and does not replace synthesis in any way.
- An important point when working with graphical diagrams is that you should understand that this environment is solely designed to make your work easier. The Synthesis tool in Vivado software is responsible for synthesizing the circuit, and in the end, what you actually need is a Verilog code. Therefore, there must be a way to convert the designed hardware in the diagram into Verilog code and describe it accordingly. Right click on the design and select **Create HDL Wrapper**, so that Vivado creates an HDL Wrapper from your design.
- Now you need to repeat the implementation steps on the bord and test the results.

1.3.2 Priority Encoder

In this section, design a Priority Encoder. Your circuit has four inputs with arbitrary numbers from 1 to 4, which correspond to the four buttons on the board. The output displays the highest-numbered pressed button on the LEDs.
(You need three LEDs since the highest number is 4, which equals 3'b100).
For example, if buttons 1 and 3 are pressed simultaneously, the LEDs should display the number 3, which is represented as 3'b011 = 3.
Simulate your designed module to check its functionality. Here's a basic simulation method without designing a testbench code:

- From "Flow Navigator" bar in Vivado, select "Simulation > Run Simulation > Run Behavioral Simulation" to open the simulation window.
- In the wave window, right click on your module's inputs and select "Force Constant" to set a value for the inputs.
- Select "Run for ..."  in the top bar to run the simulation.

1.3.3 LED Blinking Implementation

1.3.3.1 Basic LED Blinking

Implement a blinking LED. The blinking frequency should be 1 Hz. In this part's top module, the input is clk, which is the clock signal generated by PL's clock crystal (pin U18 on the board). And the module's output is LEDblink, which will be connected to one of the ZYNQ board's LEDs and

switches on/off at a 1 Hz frequency.
Hint: Use a counter variable. When the counter reaches its maximum (determined based on both clock frequency and our desired blinking frequency), it will be reset and the LED switches on/off.

1.3.3.2 Variable Frequency LED Blinking

Now implement a blinking LED with different blinking frequencies. Use button[0] – button[3] to select the blinking frequency. In Table 1-2, there are the blinking frequencies corresponding to each button.

Button	Blinking Frequency
button[0]	0.5 Hz
button[1]	1 Hz
button[2]	2 Hz
button[3]	4 Hz

Table 1.1: Blinking frequency corresponding to each button

Use the priority encoder module that you implemented in previous parts to avoid conflicts when more than one button is pushed simultaneously. The output of this module determines the blinking frequency.
In this part's top module, the inputs are a 4-bit signal button (connected to the ZYNQ board's buttons) and 1-bit clk. And the output is LEDblink, the same as in the first part.

1.3.4 Counter,7-Segment

1.3.4.1 BCD to 7-Segment Display Converter

In this part, you're going to use the extension board. Implement a BCD-to-7-segment converter module that gets a 4-bit BCD input and converts it to an 8-bit value that determines which LEDs of the 7-segment should be on to represent a certain decimal number.

Figure 1.9: 7-segment display character representations

The input of the top module is a 4-bit signal Dip, which is the BCD number we want to display on 7-segment. It will be connected to the extension board's DIP switches. The top module's outputs are *a*, *b*, *c*, *d*, *e*, *f*, *g*, *dp*, *dig*.

a, *b*, *c*, *d*, *e*, *f*, *g* are the 7-segment segments used to display the number. *dp* is the decimal point character and is assigned to "1" so that it remains off all the time (we don't need it in this part). And *dig* is a 4-bit signal in which each bit determines whether the corresponding digit on the extension board's 7-segment is on or off. Figure 1.10 shows a schematic of the module and its inputs/outputs.

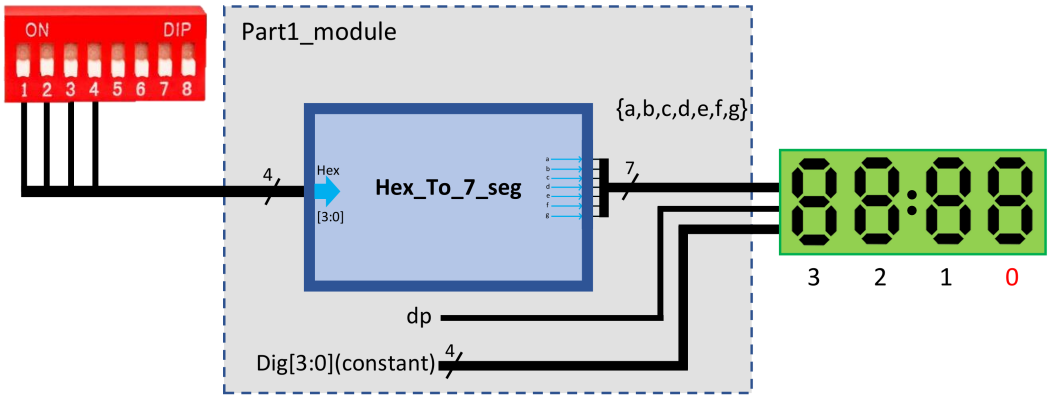

Figure 1.10: Schematic of the module and its input/output ports

1.3.4.2 4-Bit Counter with 7-Segment Display

Implement a 4-bit counter and display its output on the 7-segment. Design a 4-bit counter module with synchronous reset and connect its output to the input of the BCD-to-7-segment module that you designed in previous part. The inputs of the top module are clk (the counter's clock signal) and rst (the counter's reset signal). And the outputs are *a*, *b*, *c*, *d*, *e*, *f*, *g*, *dp*, *dig* – the same as in the previous part.

Department of Electrical Engineering