Laboratory 1

1. Introduction to the Software/Hardware development environment for VHDL based designs.

1.1 Objectives

Familiarize the students with

- Xilinx ISE WebPack CAD tools – ***ISE Quick Start Tutorial***
- Xilinx Vivado WebPack – ***Vivado Design Suite User Guide***
- Xilinx® Synthesis Technology (XST)
- Xilinx Spartan 3E FPGA family
- Xilinx Artix7 FPGA family
- Digilent Development Boards **(DDB)**
  - Digilent Basys Board – Reference Manual
  - Digilent Basys 2 Board – Reference Manual
  - Digilent Basys 3 Board – Reference Manual

1.2 Necessary resources (the kits are available and installed on the workstations from the laboratory)

Digilent Adept Software: download page

One Digilent Development Board:

Basys 1	Basys 2	Basys 3
Reference Manual	Reference Manual	Reference Manual
Schematic	Schematic	Schematic

Development environment:

- Xilinx ISE WebPACK is a part of the Xilinx Design Suite, ISE Design Suite 14.7
- Xilinx VIVADO WebPACK is a part of the Vivado Design Suite – HLx Editions

Xilinx ISE Software manual: XST User Guide

Online Help for VHDL programming http://vhdl.renerta.com/

1.3 Basic Components

1.3.1. Logic Gates

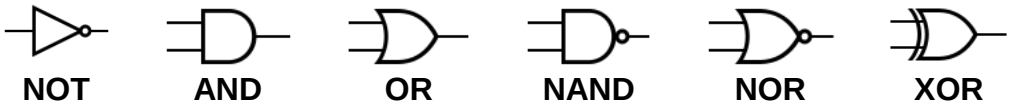

Figure 1-1: Logic Gates Diagrams

A	NOT
0	1
1	0

A	B	AND	OR	NAND	NOR	XOR
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	1	0	0	0

Table 1.1: Logic Gates Truth Tables

1.3.2. Latches

A latch is an electronic circuit which has two stable states and thereby can store one bit of information. XST can recognize latches with asynchronous set/reset control signals. Latches can be described in VHDL by using: processes or concurrent statement assignment. **XST does not support wait statements (VHDL) for latch descriptions.**

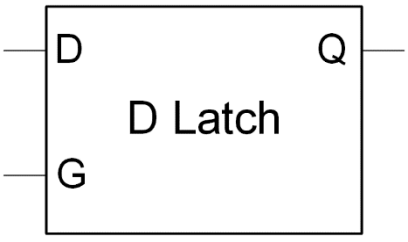

Figure 1-2: Latch with Positive Gate

IO Pins	Description
D	Data Input
G	Positive Gate
Q	Data Output

Table 1.2: Latch with Positive Gate Pin Description

1.3.3. Flip-Flops

A flip-flop is an electronic circuit that has two stable states and is capable of serving as one bit of memory. A flip-flop is usually controlled by one or two control signals and/or a gate or clock signal. XST recognizes flip-flops with the following control signals: asynchronous Set/Reset, synchronous Set/Reset or clock enable.

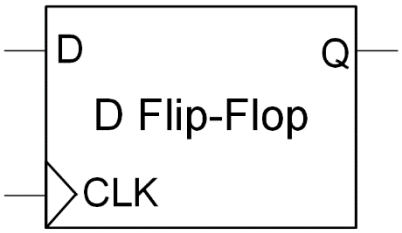

Figure 1-3: Flip-flop with Positive Edge Clock

IO Pins	Description
D	Data Input
CLK	Positive Edge Clock
Q	Data Output

Table 1.3: Flip-Flop with Positive-Edge Clock Pin Descriptions

When using VHDL for a positive-edge clock, instead of using:

```
if (CLK'event and CLK='1') then
```

you can also use:

```
if rising_edge(CLK) then
```

1.3.4. Multiplexers

A multiplexer or mux is a device that performs multiplexing; it selects one of many analog or digital input signals and outputs that into a single line. A multiplexer of 2^n inputs has n select bits, which are used to select which input line to send to the output. XST supports different description styles for multiplexers (MUXs), such as If-Then-Else or Case.

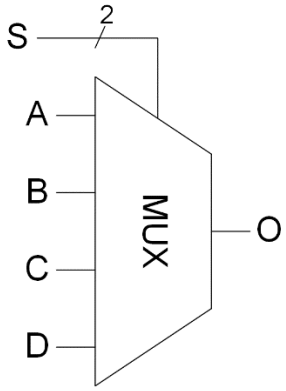

Figure 1-4: 4-to-1 1-Bit MUX

IO Pins	Description
A, B, C, D	Data Inputs
S	Mux Selector
O	Data Output

Table 1.4: 4-to-1 1-Bit MUX Pin Descriptions

1.3.5. Decoders

In digital electronics a decoder is a multiple-input, multiple-output logic circuit that converts coded inputs into coded outputs, where the input and output codes are different. E.g.: n-to-2^n decoders, BCD decoders.

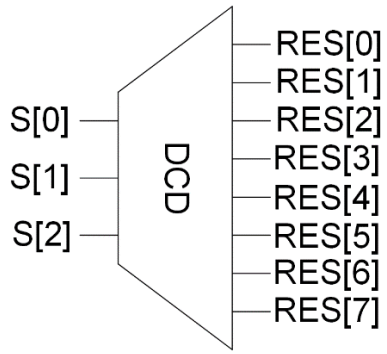

Figure 1-5: 3-to-8 Decoder

IO Pins	Description
S	Selector
RES	Data Output

Table 1.5: 3-of-8 Decoder Pin Descriptions

1.3.6. Counters

In digital logic and computing, a counter is a device which counts the number of times a particular event or process has occurred, often in relationship to a clock signal. XST recognizes counters with the following control signals: asynchronous Set/Reset, synchronous Set/Reset, asynchronous/synchronous Load (signal or constant or both), clock enable, modes (up, down, up/down) or a mixture of all.

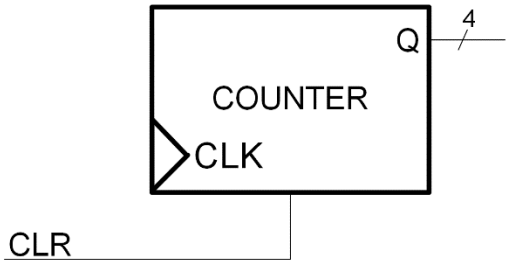

Figure 1-6: 4-Bit Up Counter with Asynchronous Reset

IO Pins	Description
CLK	Positive Edge Clock
CLR	Asynchronous Reset (Active High)
Q	Data Output

Table 1.6: 4-Bit Up Counter with Asynchronous Reset Pin Descriptions

1.4 Laboratory Assignments

Note: If necessary, you can consult the online help for VHDL indicated in the previous section.

1.4.1. Implement a simple VHDL design using Xilinx's ISE 14.7 or the Vivado Design Suite and the development board by carefully and completely covering the tutorial described in Appendix 1 or Appendix 2. You are also encouraged to read / understand / remember section 1.3., which covers the description of basic digital components.

1.4.2. Add an 8-bit counter / 16-bit up-counter to your "*test_env*" design (the width of the counter depends on the chosen board), by describing (in one process) the behavior of the counter in the "*test_env*" architecture. Try to control the counting process from a digital button present on the board.

Start by declaring an 8-bit / 16-bit signal (STD_LOGIC_VECTOR) in the architecture, before the begin statement.

If necessary (until you regain your full capacity in the VHDL programming language), use the *Language Templates* (Appendix 1 or Appendix 2) for extracting the behavioral description of the counter.

Use one of the buttons from the entity's ports in order to control the counting process, as an enable or count-up signal.

Display the counter values on the LEDs available on the board.

Follow the steps in Appendix 1 or Appendix 2 in order to generate the bit file and re-program the development board. Control the counter from the button.

…Is there any problem?

1.4.3. Synchronized (1 clock period) mono pulse generator (MPG)

At this point, you have to work in the same "test_env" project.

In the future, you will need step-by-step control of sequential circuits, to trace and test the required data and control flow of your designs.

The necessary circuit that generates an ENABLE signal once per button push is given in the next figure.

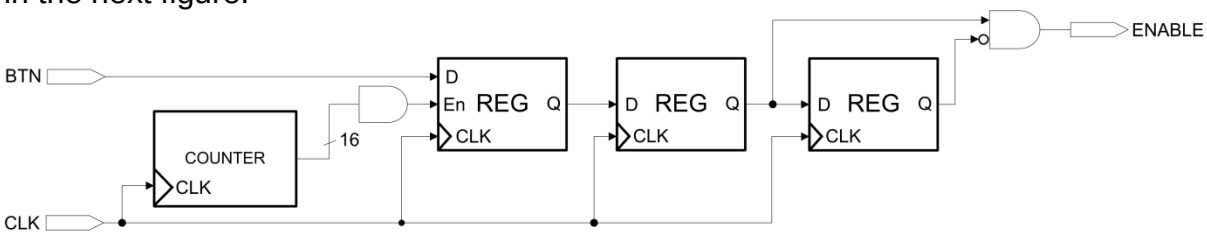

Figure 1-7: Synchronized (1 clock period) mono pulse generator

The role of the first register together with the 16-bit counter is to provide a delay necessary to de-bounce the buttons (physically worn out and / or low quality buttons). According to the "degradation" state of the button, you may have to increase the size of this 16-bit counter in order to increase the sampling interval for the button signal.

The mono pulse generator (MPG) will be implemented in a new entity / new file and will be used in the "*test_env*" by declaring it as a component in the section for declaring signals and by instantiating this component with *port map* in the architecture body after begin.

The hardware components that implement the MPG (counter, registers, and logic gates) will be implemented using the behavioral description: by declaring the necessary signals and by describing the functionality using processes and concurrent assignments in the MPG architecture.

Work to do

a. Draw a timing diagram for the above circuit (paper and pencil / blackboard).
b. The BTN input is a signal from one of the DDB buttons (BTN0), CLK is the clock signal of the DDB (25, 50 or 100 MHz – depending on the used development board)
c. Write and check the VHDL code for this circuit.
d. Include the MPG component in the *test_env.*
e. Use the ENABLE signal as count up for the previously implemented 16-bit counter, from section 3.2. You need to add the condition that ENABLE equals '1' where you test the count up condition for the 16-bit counter.

Synthesize your design and do not forget about the **RTL Schematic / Elaborated Design…**

Load your new design on the development board.

1.4.4. Create a new project, for example *test_new*, using the same ports as for the first project. You have to go again over the steps described in Appendix 1 or Appendix 2, without adding anything to the architecture *test_new*. Now implement the following circuit in the *test_new* architecture.

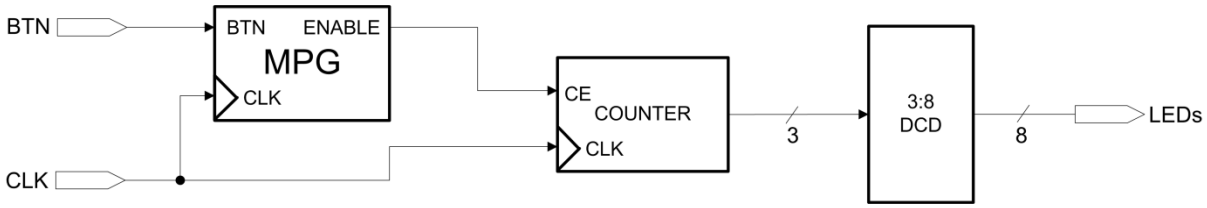

Figure 1-8: Problem 4.4 Schematic

You need to add the MPG source file to the new project (Appendix 1 or Appendix 2). Import the MPG with *component / port map* in the *test_new* architecture and write the code (only processes) for the rest of the digital components, without any additional entities. Add a 3-bit counter and a 3-to-8 decoder, using only signals declared in the *test_new* architecture and concurrent processes / signal assignments.

Do not forget about the **RTL Schematic / Elaborated Design…**

Load the design on the development board.

1.4.5. Redesign the mono pulse generator in order to create a MPG for all the buttons on the DDB.

Show all your designs to the TA

Homework

a. Finish all the laboratory assignments.
b. Re-read the Laboratory regulations, the tutorial from Appendix 1 or Appendix 2 – starting with laboratory 2 these concepts are considered learned. Be attentive about the aspects presented in the tutorial that were not, yet, relevant for this first designs. They will be important in the future
c. (this homework is considered implicit for the next labs) Read the material for the next laboratory (it is available on the web site).

1.5 References

[1] ISE Quick Start Tutorial (www.xilinx.com)
[2] Xilinx® Synthesis Technology (XST) User Guide
[3] Digilent Basys Board – Reference Manual
[4] Digilent Basys 2 Board – Reference Manual
[5] Digilent Basys 3 Board – Reference Manual