**Report 3**

# **A technical summary of experiments conducted in the lab (steps, results, components, code functionality, etc.)**

## **Experiment 1: - A N-bit register with reset and load control**

The main purpose of this experiment is to create and simulate a module representing a n-bit register with load control. N is a generic parameter to have a default value of 8. Use flip flop modules, 2x1 multiplexers.

* **Steps**

A screen shot of a computer program

AI-generated content may be incorrect.After creating new RTL project targeting the Nexys A7 board using Vivado, we will Add a register module in Verilog. We will use it later. However, we will use testbench instead.

A computer screen with text

AI-generated content may be incorrect.A **register** is a small, fast storage location within a computer's central processing unit (CPU) or other hardware components. It used to hold data temporarily during processing, such as instructions, addresses, or operands. Registers are essential for high-speed data access and manipulation, enabling the CPU to perform operations efficiently. The first file under the name DFlipFlop which take inputs clk, rst, D,and output reg Q. After, always block that take posedge clk and rst, if statement to check rest if Q is one bit less than or equal 0, else Q is less than or equal D.

The second file is a Multiplexer under the name MUX\_2x1 to control the N-bit register. That three inputs a, b, and load, while the output is out. Load is a one bit if it is 1 then the output (out) will store a, otherwise it will store b.

A diagram of a circuit

AI-generated content may be incorrect.A diagram of a machine

AI-generated content may be incorrect.

A computer screen shot of a program code

AI-generated content may be incorrect.

The final file is for collecting all the previous files in order to build **figure 2** by using several Multiplexer and DFlipFlop. By passing by parameters the number of Multiplexer and DFlipFlop used, the next step is to generate 8 numbers of Multiplexer and DFlipFlop instead of copy and paste.

* **Results**

The result of adding this number of DFlipFlop and Multiplexer is the register that used to hold values temporarily, in order to use them in CPU. Then, apply testbench is used for making n-bit register in order to make sure that it works.

## **Experiment 2: A n-bit 2x1 multiplexer**

The main purpose of this experiment is to create and simulate a module representing a n-bit 2x1 multiplexer. The module’s structure should be an array of 2x1 multiplexers.

* **Steps**

This experiment will used two files (two modules) the first one is normal under the name mux Multiplexer that works. We mentioned how it works in the first experiment. The second file under the name nbit\_mux is for creating an array of Multiplexer.

A computer screen with text

AI-generated content may be incorrect.By passing by parameters the number of used mux that will be used in the experiment. The next step is to generate 8 numbers of Multiplexer

* **Result**

The result of adding this number of Multiplexer is to simulate a module representing a n-bit 2x1 multiplexer. The module’s structure of 2x1 multiplexers array.

## **Experiment 3: A n-bit shift left 1 module**

The main purpose of this experiment is to create and simulate a module representing an n-bit shift left 1 module. Shifting should be done by rewiring inputs.

* **Steps**

After creating a new RTL project targeting using Vivado, we will use n-bit\_shift\_left module to accomplish this task.

A computer screen with text

AI-generated content may be incorrect.

After then, we will write a continuous assignment statement that assigns value to the output out. Selects all bits of in except the most significant bit (MSB). 1-bit constant with a value of 0. It is concatenation to the least significant (LSB) position of output. Concatenation of two parts, in[n-2:0] (all bits except the MSB) is shifted left by 1 position, and 1’0 which added to the LSB position.

* **Results**

The code assumes a logical left shift, where the LSB is always filled with 0. If we need an arithmetic left shift (where the sign bit is preserved), this code will need modification. The shift amount is fixed at 1 bit. If we need to shift by a variable amount, additional logic would be required.