**Experiment Report of Digital Logic Circuit with Verilog**

****

Experiment Information:

|  |  |
| --- | --- |
| Index | A |
| Name | Dikshya Kafle |

Student Information:

|  |  |
| --- | --- |
| Student No.： | 2018380039 |
| Student Name： | Dikshya Kafle |
| Grade No.: |  |
| Experiment Time： | 22nd November 2019 |
| Experiment Place： | Computer Science Building Room 118 |

**Northwestern Polytechnical University**

**School of Computer Science**

**Fall 2019**

# 1: Object and Requirement

*Tip: introduce the aim of the experiment task. It may cover the function requirement/adopted algorithm/ protocol analyzing/timing requirement/ area requirement*

1: Model basic sequential circuit module using Verilog and design the ALU design based circuit.

2: Continue to learn how to implement the design on FPGA

3: Continue to learn the design method of testbench

# 2: Environment and Tools

*Tip: simply introduce the EDA tools you need to complete the experiment.*

**ModelSim:** A multi-language HDL simulation environment by Mentor Graphics for simulation of hardware description languages such as VHDL, Verilog and SystemC, and includes a built-in C debugger. ModelSim can be used independently, or in conjunction with Intel Quartos Prime, Xilinx ISE or Xilinx Vivado.Simulation is performed using the graphical user interface (GUI), or automatically using scripts.

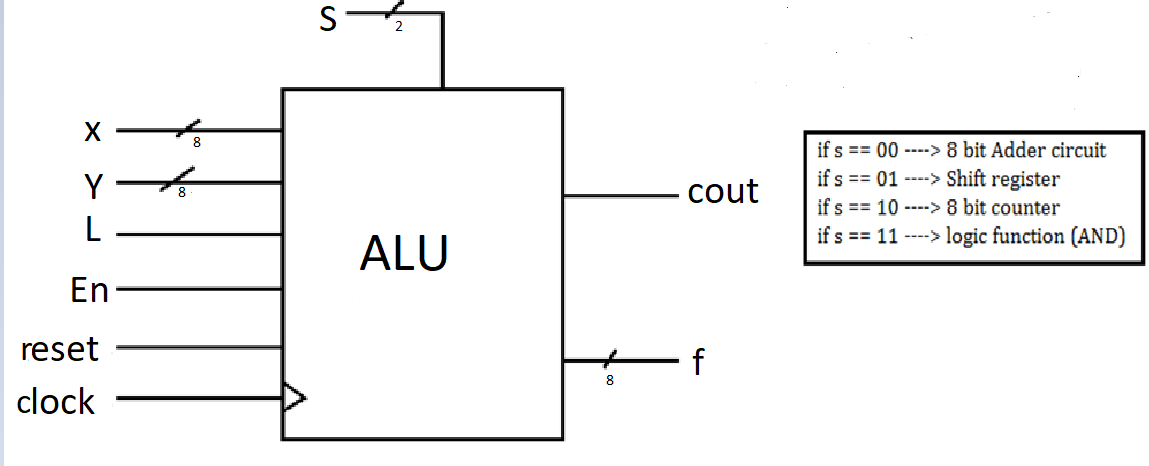
**Altera Quartus II**:The **Altera Quartus II** design software provides a complete, multiplatform design environment that easily adapts to your specific design needs. It is a comprehensive environment for system-on-a-programmable-chip (SOPC) design. The **Quartus II** software includes solutions for all phases of FPGA and CPLD design. It is the programmable logic device design software produced by Intel. It includes an implementation of VHDL and Verilog for hardware description, visual editing of logic circuits, and vector waveform simulation. It enables analysis and synthesis of HDL designs, which enables the developer to compile their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer.

# 3: Procedures

*Tip: show the detailed circuit you designed. It may cover:*

*1: the TOP schematic and explanation*

As shown in the following figure it is the module of an ALU unit, which accompaniments two 8bits input ports (a, b); one control signal (s) of weight 2; four 1bit input ports (L, En, reset, clock); one 1bit output port (cout) ;one 8bits output port (f).



The function of each port of the ALU unit is explained below:

***The input port (x)***: First input of the 8 bit adder circuit, the serial input of the shift register, the number to be loaded to the counter, an input of the logic function.

***The input port (y)***: Second input port of the 8 bit adder circuit, second input port of the logic function.

***The input port L***: The carry in for the adder, the Load port of the counter (If L==1 we load input (a) to the counter circuit).

**The input port En**: The enable port of the 8 bit counter.

***The reset port*** : The reset port of the counter, the reset port of the shift register.

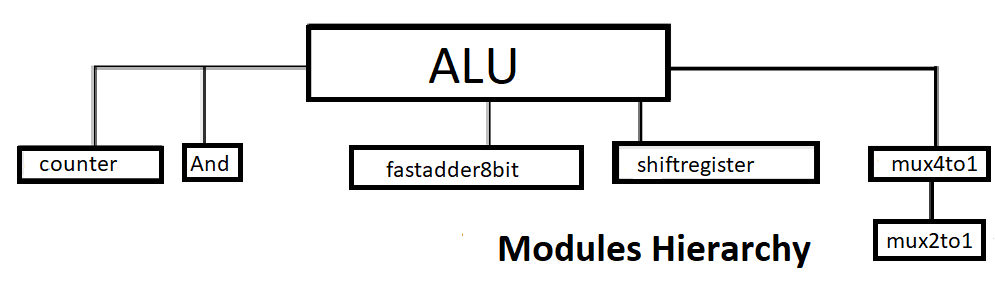
***The clock input port***: The clock signal of both the register and the shift register

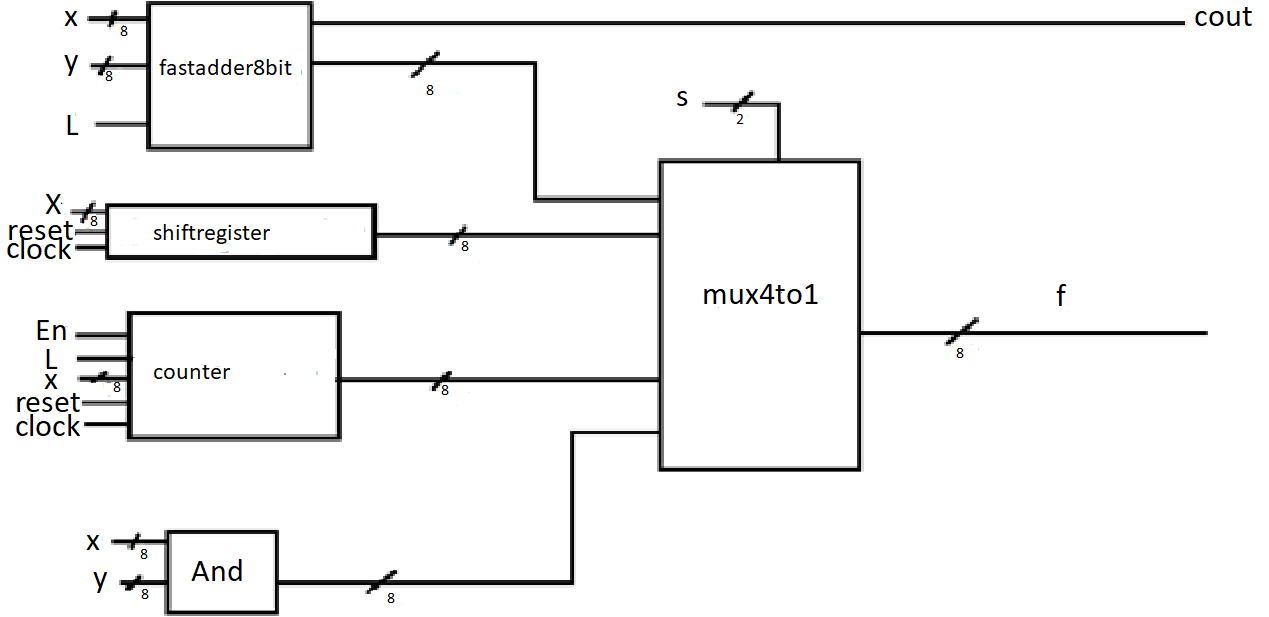
**The control signal (s)**: For selecting which function will the ALU unit perform (check the schematic).

**The output (f)**: The result of the ALU circuit.

***The output port (cout)***: The resulting carry of the binary addition if it is selected.

*2: the interface between sub-modules*

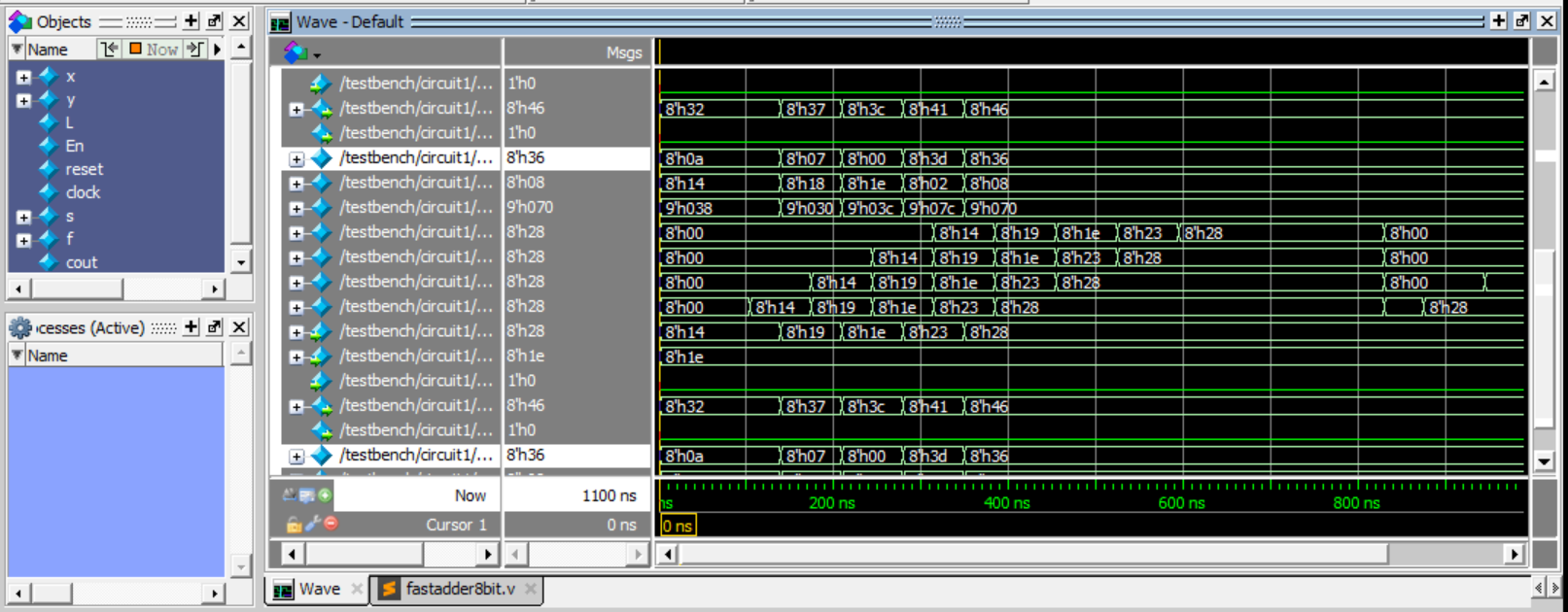




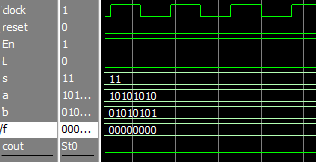
The “ALU” module calls mainly 5 modules:

* “fastadder8bit”: An 8 bits adder.
* “shiftregister”: An 8 bits shift register.
* “Counter”: An 8 bits counter.
* “And”: a circuit performing the AND logic operation.
* “mux4to1”: An 8 bits weight four to one multiplexer.

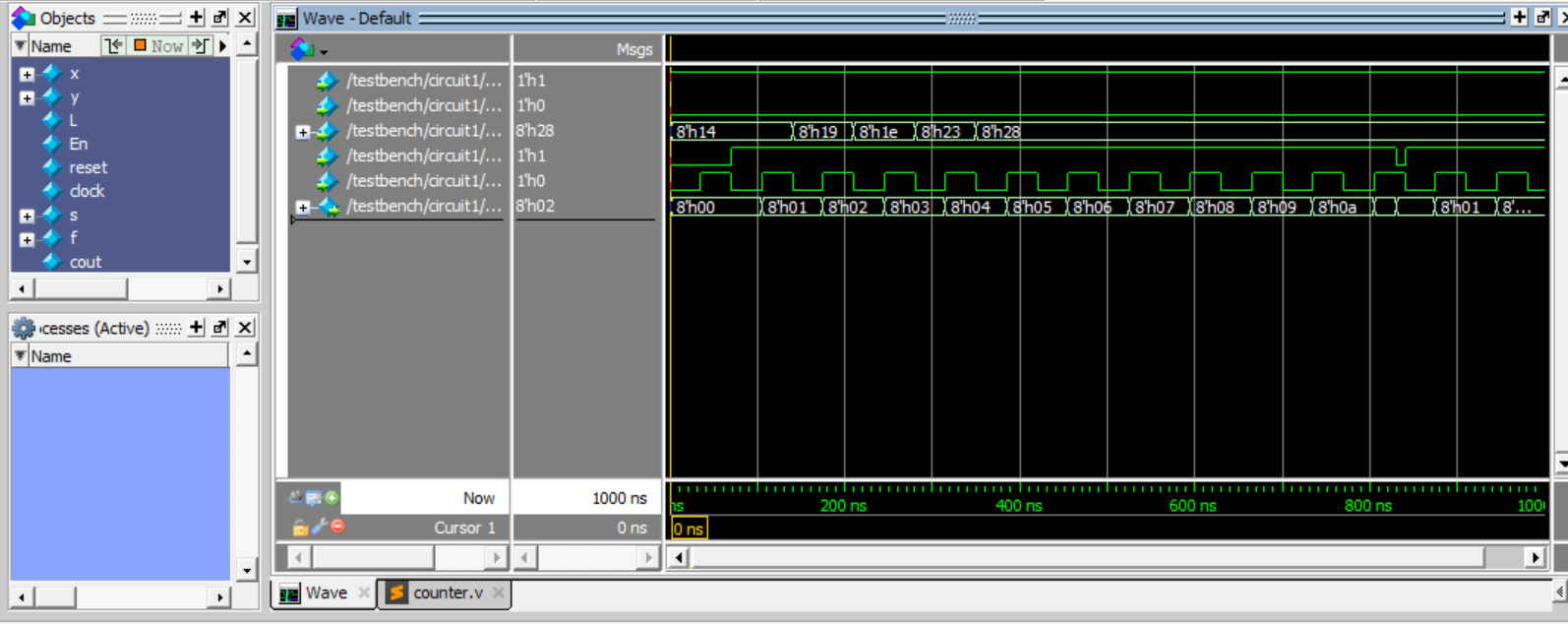
*The expected timing diagram of 8 biit fast adder:*

**

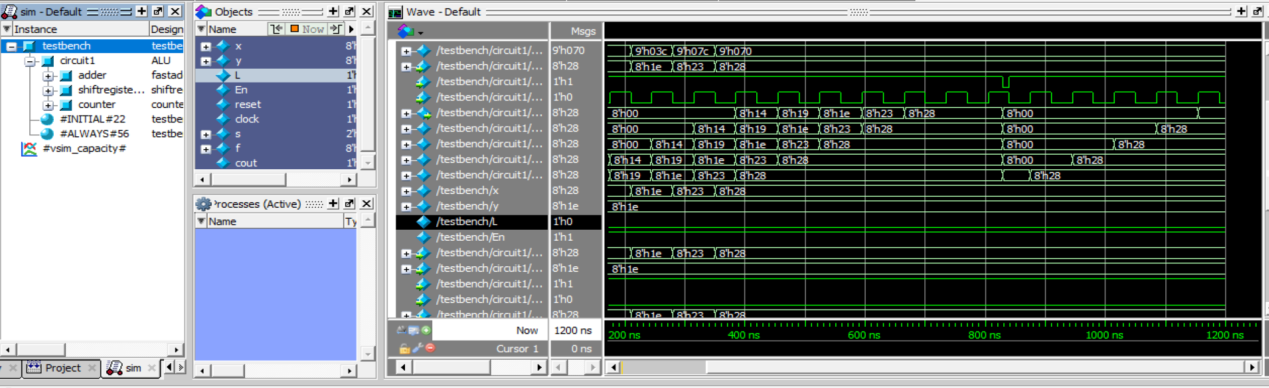
*The expected timing diagram of AND function:*

**

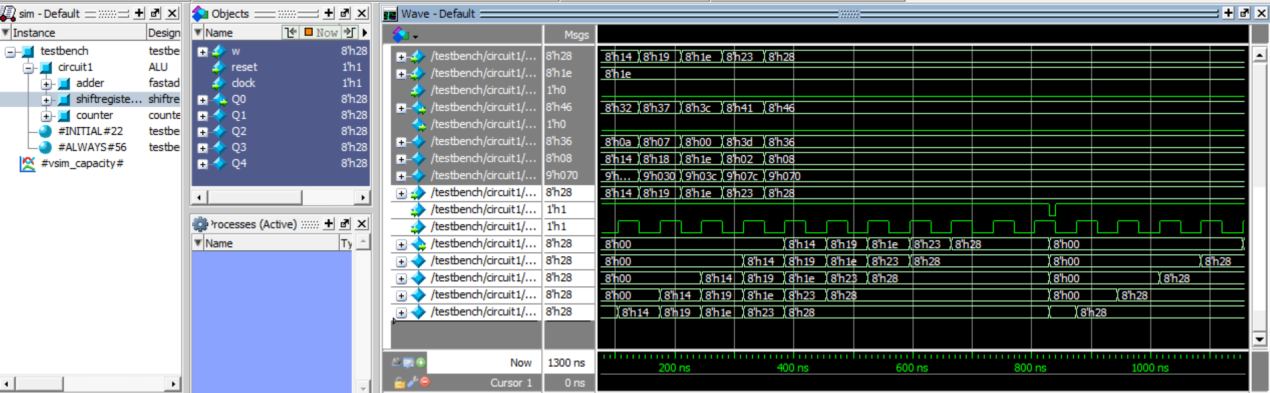
*The expected diagram of the binary counter:*

**

*the binary addition and the shift adder:*

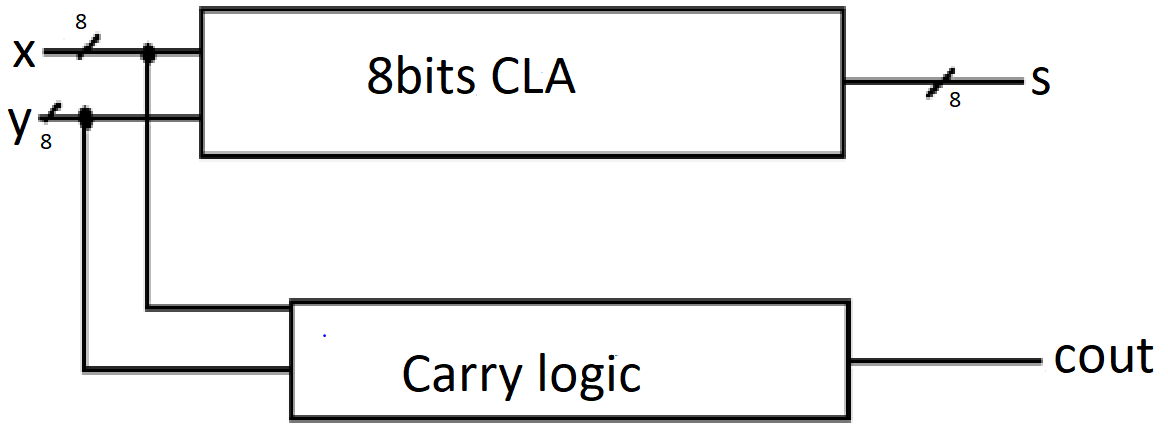
**

*The expected diagram of fastadder and shiftregistration:*

**

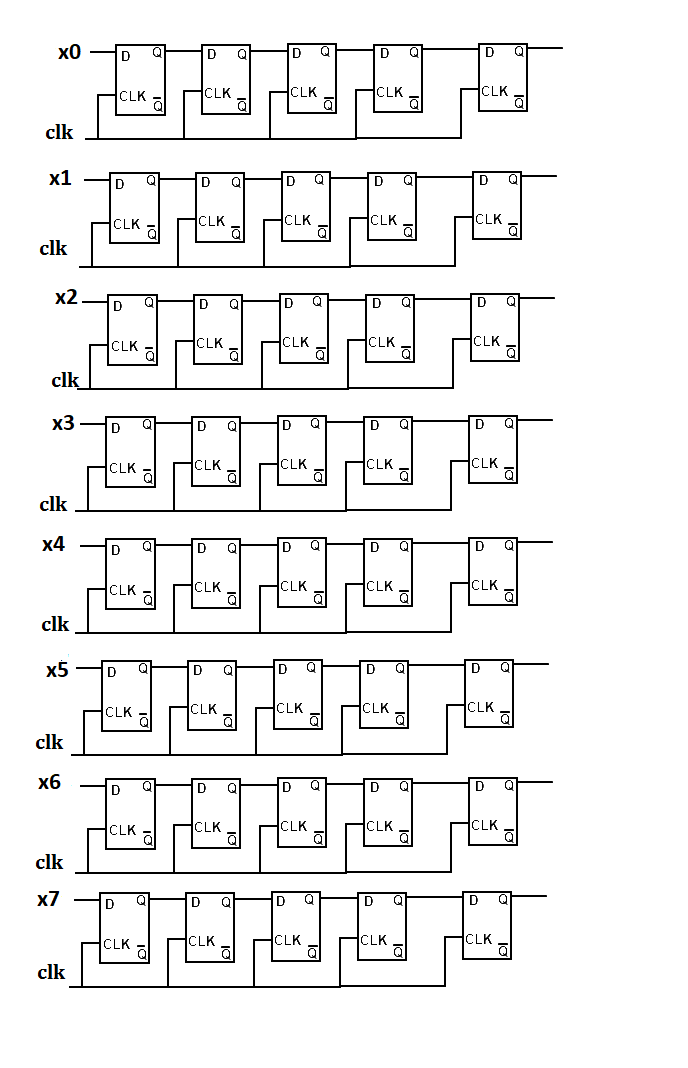
*4: the detailed circuit diagram of each submodule*

***“fastadder8bit”***

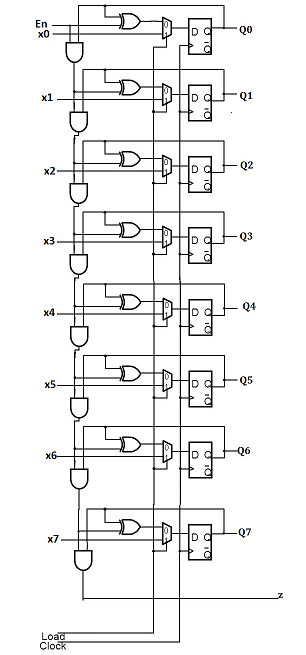


***“shiftregister”:***

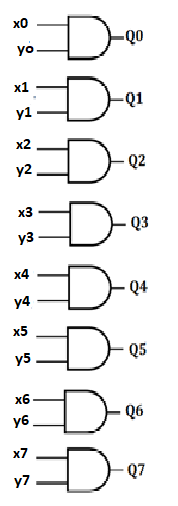
It also supports reset port (not represented): if reset is 0 (0 effect), all outputs are set to 0.

**

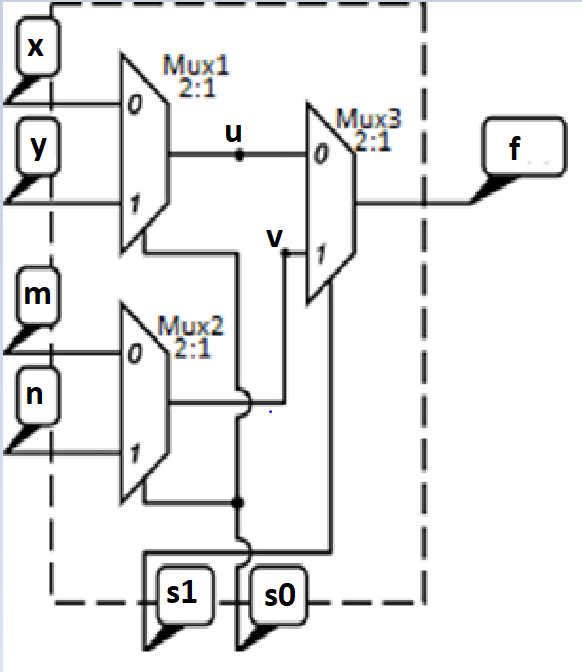
***“counter”***



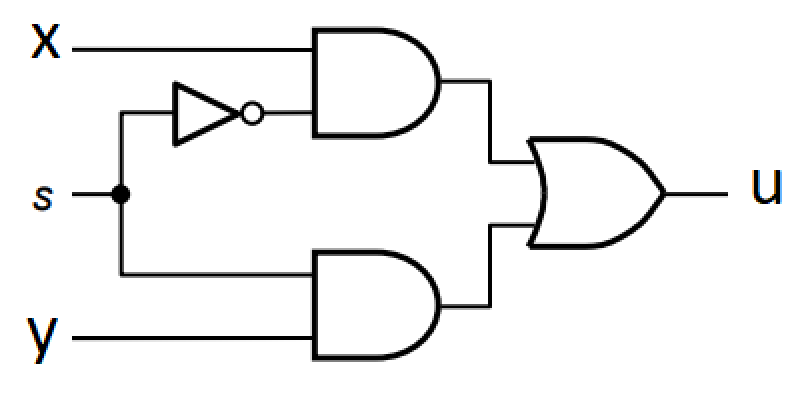
***“And”***



***“Mux4to1”***



***“mux2to1”***



*5: the verification strategy and the testbench design*

# 4: Result and Discussion

*Tip: what about the result, it may cover:*

*1: whether you have simulated all possible cases*

*s*

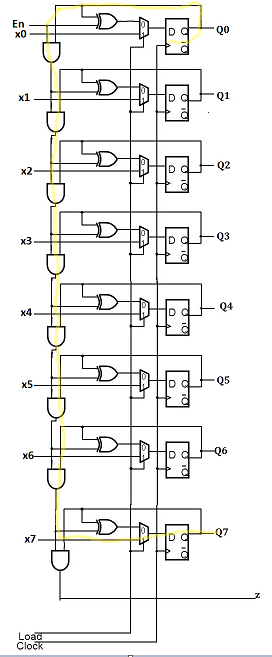
*2: whether the wave generated by the Modelsim is exactly the same as you expected. You could paste some wave pictures to prove your work.*

***3: whether you have detected some bugs in your code and how to fix them.***

Some minor syntax error occurred while compiling the code and I asked for the teacher to help me find out the error. I was able to solve them with the help of teacher. I had forgotten to include 8 bit fast adder file and stated to simulation and the graph was not shown as expected and then I checked the error with the help of teacher and after finding my mistake I added the file and compiled the program and finally the simulation was carried and the bug was solved.

*4: what about the critical path and the area cost.*

The critical path in the circuit is to be the one through the 8 bit adder. The following figure shows that clearly.

**

*5: how to optimize your design*

In order to optimize the circuit, it can be implemented by using Transmission gates which leads to get less circuit area.

*6: whether you have follow some basic coding styles.*

Some followed coding styles:

*-****Naming Varaible:*** Assigning meaningful names to variables.

***-Commenting****:* For the further explanation of some codes comments were used which was denoted by //.

**-*Using Modules:*** The implementation of 4 to 1 Mux could be done with one module but considering the better performance it was implemented using 2 modules.

**-*Separate variables declaration:*** Declaring each variable alone.

-**Creating Project:** After the completion of all the codes a project named “My Design” was created.

**-Compiling**: All the codes were compiled and the errors were corrected.

-**Simulation:** At the final stage simulation was carried which allowed us to view the waves of all the codes of the project.

# 5: Conclusion

*Tip:Give a brief conclusion of your work and evaluate your work by yourself.*

In this experiment, the functionality of an ALU circuit which can perform four different functions i.e, adding, shift, counting and logic function is being stimulated. After the writing the codes project was created and all the files were stimulated together after which the ModelSim displayed the waves and we were able to see the result. In order to be more accurate we can also see the simulation process in the EDA which shows the signals are selected properly. Thus, it proves the experiment succeeded.

Overall, the experiment was very difficult as it was the fisrt time for me to work with ModelSim. However, with the help of teacher, books and the internet I was finally able to understand the code and do the experiment.

# Performance Evaluation

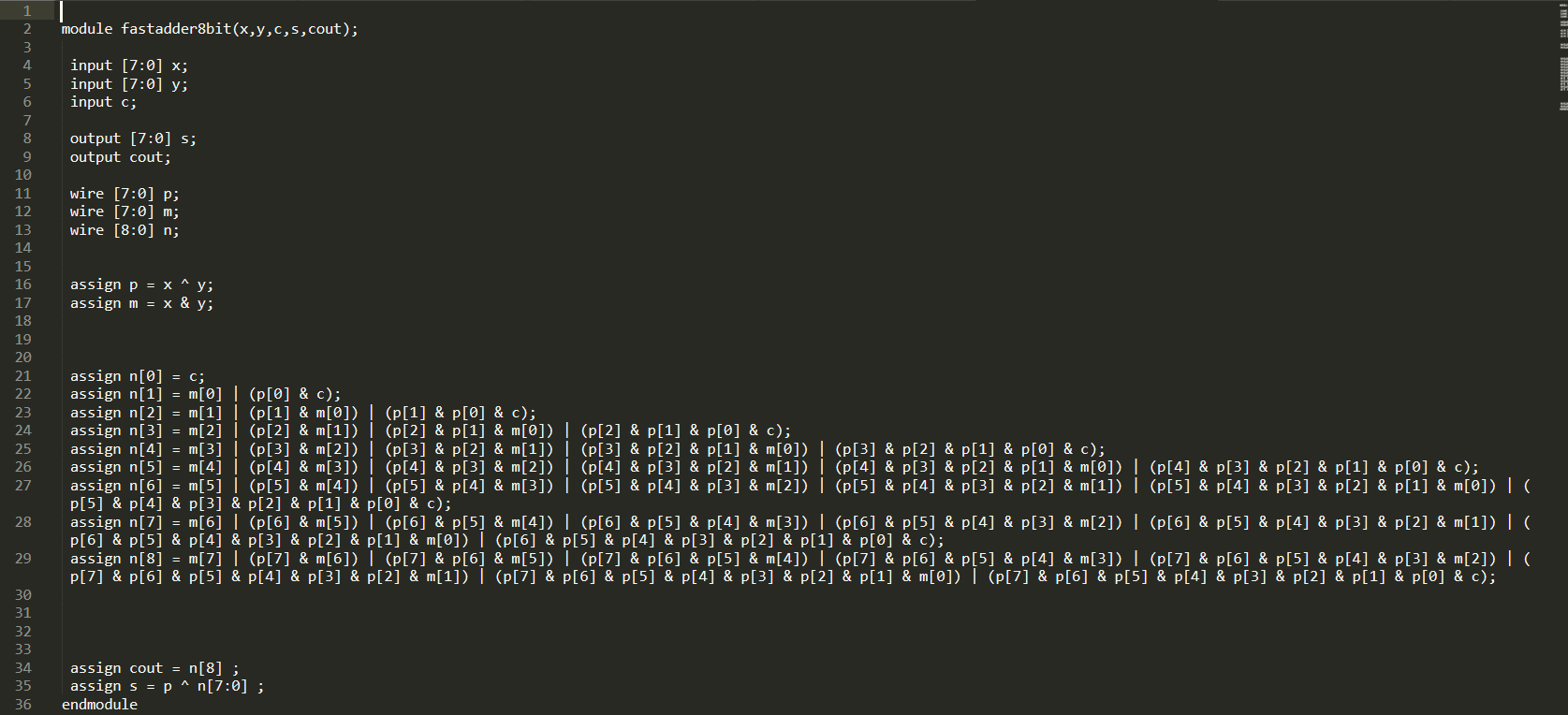
(This part is left only for the instructor)

|  |  |
| --- | --- |
| Comment: | Score:  Signature:  Time: |

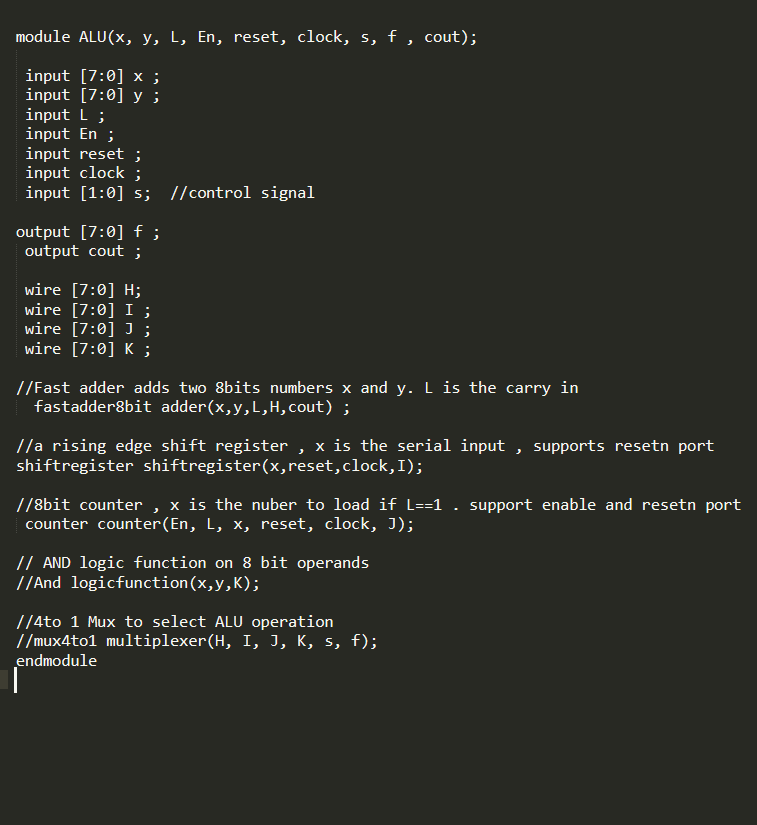
# Appendix:

(Here, please attach key pieces of code of your design and testbench)

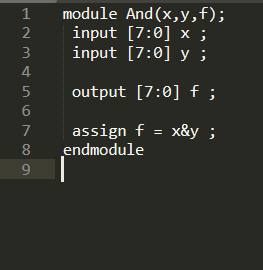
**fastadder8bit**

****

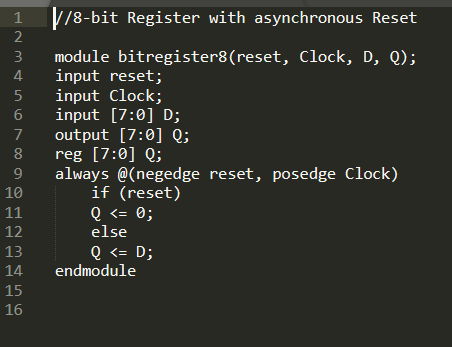
**ALU Design**



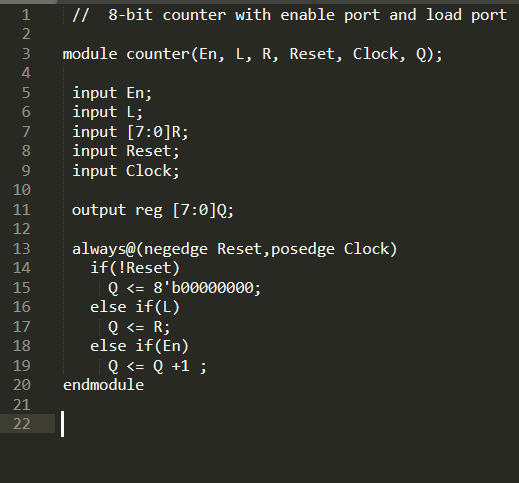
**And**



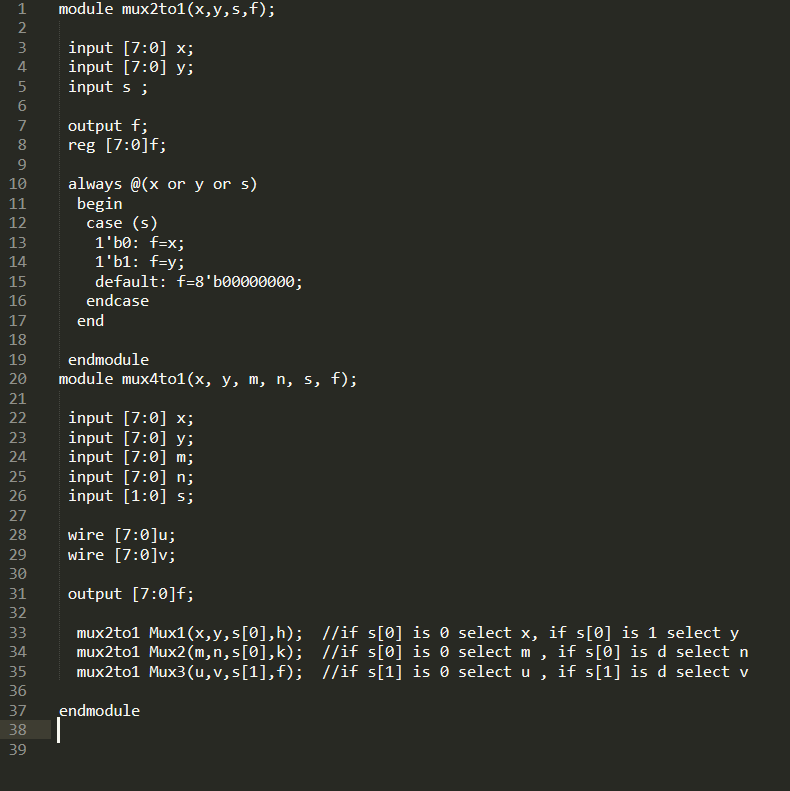
**bitregister8**

****

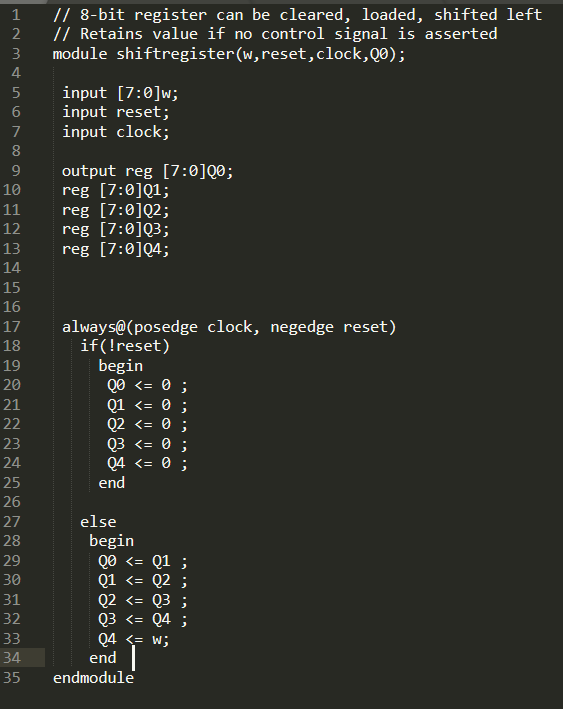
**counter**

****

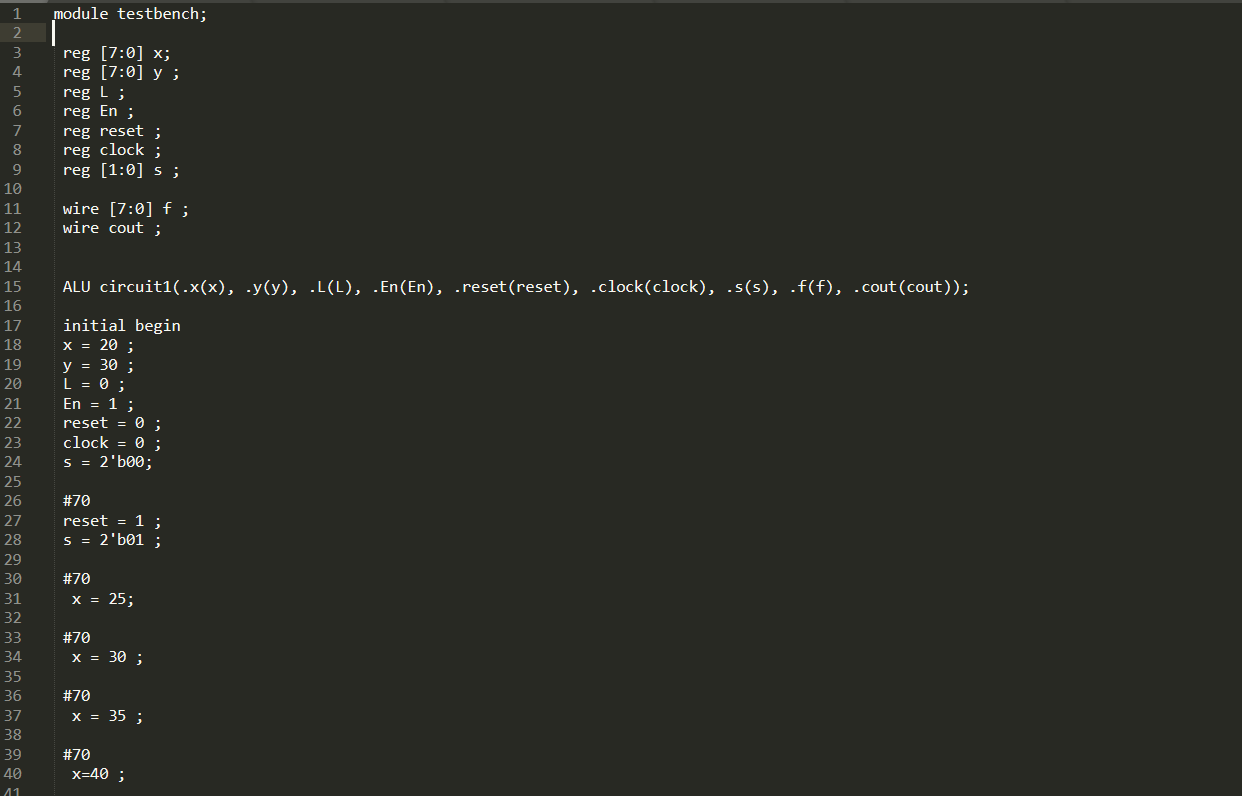
**mux4to1**

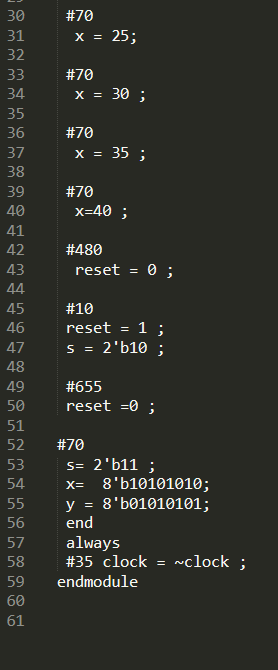
****

**shiftregister**

****

**testbench**

****

****