LAB # 1
Verilog (HDL) Programing Fundamentals

Objective:

- To understand the Fundamental concepts of Verilog HDL
- Simulate & verify the different HDL designs on Visual Studio Code

System Module (Hardware/Software)

- Visual Studio Code

Theoretical Background:

Hardware descriptive language:
In the introduction lab we went through the basic introduction of Hardware descriptive languages. This lab is aimed at designing the Combinational and sequential circuits via use of HDL. As discussed in earlier labs there are three modeling techniques used for modeling the digital circuitry however the modeling techniques which are practically used are only dataflow modeling and behavioral modeling. DataFlow modeling is used mainly to design combinational logic circuits for eg, Adders,Multiplexers, Decoders etc. On the other end Behavioral modeling is used for designing the sequential logic circuits such as FSM, Memory circuits Pipelined Datapaths etc.

In the labs we'll first shift our focus on combinational logic circuits and will see how we can code them using data flow modeling techniques.

Combinational Circuits:
Combinational circuit is a circuit in which we combine the different gates in the circuit, for example encoder, decoder, multiplexer and demultiplexer. Some of the characteristics of combinational circuits are following:

1) The output of a combinational circuit at any instant of time, depends only on the levels present at input terminals.
2) The combinational circuit does not use any memory. The previous state of input does not have any effect on the present state of the circuit.
3) A combinational circuit can have an **N** number of inputs and **M** number of outputs.

Diagram

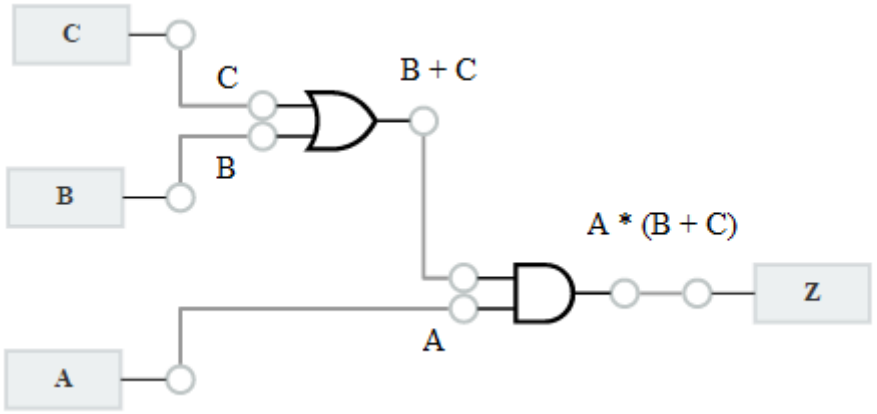
Examples:

- Adders
- Subtractors
- Multiplexers
- Demultiplexers

- Encoder
- Decoder
- BCD
- 7 segment

Data Flow Modeling:

In this form of digital circuit designing the implementation is carried out by means of defining how the data flows in a given path this is easily understandable by means of the following example of a digital design,



The Digital design in the figure above is modeled in verilog using dataflow model in the following way.

LAB Exercise

Verilog Code	Testbench

//Dataflow model
 	
module Dataflow_Model(A,B,C,Z);
 	
//definition of input ports of the module
input A,B,C;
 	
//definition of output ports of the module
output Z;
 	
//definition of internal wires of the design
wire B_or_C ;
wire A_and_B_or_C;
 	
// assignment in the relevant wires via 'assign keyword'
assign B_or_C = B | C;
	module tb;
	reg A; reg B; reg C;
	wire Z;
	
	Dataflow_Model dut (
	.A(A),
	.B(B),
	.C(C),
	.Z(Z)
	);
	
	initial begin
	A <= 1'b1;
	B <= 1'b1;
	C <= 1'b1;
	#100;

```
assign A_and_B_or_C = A & (B_or_C) ;

//assignment in the output port Z of the module
assign Z = A_and_B_or_C;


endmodule
```

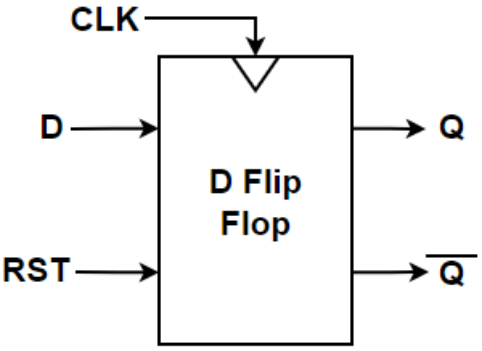
```
A <= 1'b1;
B <= 1'b0;
C <= 1'b1;
#100;
A <= 1'b0;
B <= 1'b1;
C <= 1'b0;
#100;
end

initial begin
   $dumpfile("dump.vcd");
   $dumpvars(0);
end
endmodule
```

The key to dataflow modeling is the Keyword 'assign', which as the name suggests solves the RHS and assigns the value to the LHS; however, there are few rules which are to be obeyed while using this keyword.

1. The LHS must be a wire data-type variable.
2. RHS could have a wire register or a mix of both.
3. The length of signals on either side must be equal.
4. Feedback of the signal is not allowed

Sequential Elements and Circuits:

Sequential elements are designed to be used as memory elements in digital circuitry; they possess the ability to store the output. The famously known sequential elements are D-Flip Flops, J-K Flip Flop, S-R Latch , etc. Most famously used sequential elements in practice by digital logic designers is D-Flip Flop,

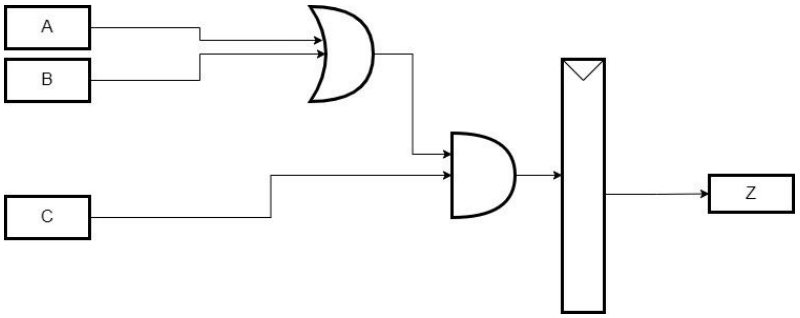


D Flip Flop has three inputs D, CLK, and RST, and 2 outputs Q, and Qbar. At the input side, port D of the Flip Flop is where the data to be stored is applied, while Port CLK is connected to the digital clock, and Port RST is used to reset the Flip Flop. While at the output side Port Q is the main output, which is the stored data and Qbar is simply the negative of the Q. The data applied at the port D of the flip flop is flopped(stored) in the flip flop only at the positive edge (or negative edge, depending on the type of flip flop used) of the clock applied at the CLK port. Once the data is stored at the edge of the clock, the change of data at port D doesn't affect the output at port Q, for the rest of the clock. If at any instant it is required to

reset the flip flop to its initial condition of Q being 0, a signal either high or low depending on the type of flip flop used is applied at the RST port.

Sequential circuits can be described as a combinational circuit which through the use of sequential elements gains the capability to store data. Some examples of such circuits include, counter, memory, FSM, etc. Some of the characteristics of sequential circuits are following:

Behavioral modeling:
In this form of modeling, rather than defining the dataflow strictly, we define the behavior of the circuit. Mainly behavioral modeling is carried out in order to model sequential circuits in verilog. The most commonly used sequential element is a D-flip flop which is modeled in verilog via declaring a variable of "reg" type and assigning the value in it via "always" block on "positive edge" of the clock. In the following verilog code we connect a D-Flip Flop at the end of our combinational circuit designed above, you'll see these kinds of circuits more and more as we move on in this course.



Behavioral model	Test bench

//Dataflow model
 	
module Behavioral_Model(A,B,C,clk,rst,Z);
 	
//definition of input ports of the module
input A,B,C;
input clk,rst;
 	
//definition of output ports of the module
output Z;
 	
//definition of internal wires of the design
wire B_or_C ;
wire A_and_B_or_C;
 	
//definition of flip flop to store the data of the combinational circuit
reg Result_Comibnational_Circuit;
	module tb;
	reg A; reg B, reg C;
	reg clk,rst;
	wire Z;
	
	Behavioral_Model dut (
	.A(A),
	.B(B),
	.clk(clk),
	.rst(rst),
	.C(C),
	.Z(Z)
	);
	initial begin
	$dumpfile("dump.vcd");
	$dumpvars(0);
	end
	
	//Clock generation

```
//assignment in the relevant wires via 'assign keyword'
assign B_or_C = B | C;
assign A_and_B_or_C = A & (B_or_C) ;


always @(posedge clk)
  begin
    if (rst ==1'b1)
       begin
        Result_Comibnational_Circuit <= 1'b0;
       end
    else
       begin
        Result_Comibnational_Circuit <= A_and_B_or_C;
       end

  end

//assignment in the output port Z of the module
assign Z = Result_Comibnational_Circuit;


endmodule
```

```
always
 begin
 clk = 1'b1;
#50;
 clk = 1'b0;
 #50;
 end

initial begin
rst = 1'b1;
#100;
rst = 1'b0;
A= 1'b0;
B= 1'b1;
C = 1'b1;
#100;
$finish;
end
endmodule
```

While using "always" blocks following rules must be thought of beforehand,

1. LHS of the assignment must be a reg type variable
2. RHS could be a wire, a reg or a mix of both.
3. Signal size on either side of the assignment must be consistent.
4. Feedback of the signal is allowed.

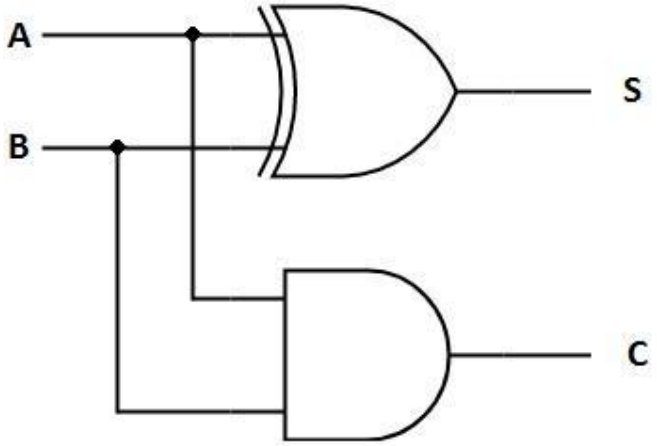
Please find more theoretical details of the always block and behavioral modeling in the Supplementary document provided.

Procedure:

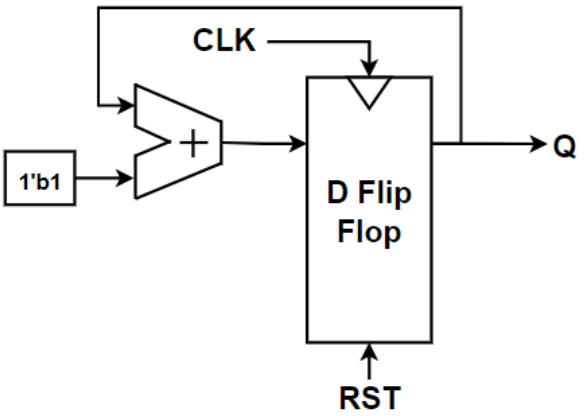
Open Visual Studio Code and perform the procedure mentioned in lab #00 in order to make a Verilog module and test bench.

In-Lab Tasks:

1. **Create a Verilog module for half adder. Make a test bench for all the possible input combinations and attach the waveforms.**

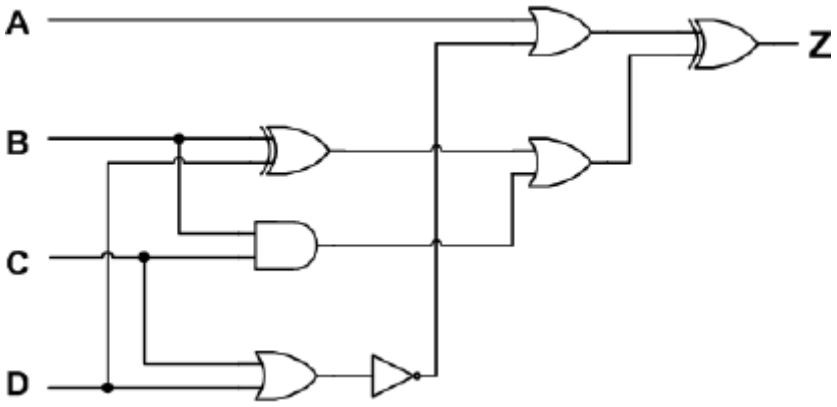


2. **Create a two bit counter. Make a test bench for all the possible input combinations and attach the waveforms.**



Post-Lab Tasks:

1. **Below is the gate level circuit for the combinational circuit. Write down a data flow model for it. Hint: First write the Boolean expression for the below circuit.**



Conclusion:

What have you learnt from this lab?

_____

_____

_____

_____

Learning Outcomes:

Upon successful completion of the lab, students will be able to:

LO1: Understand the fundamentals of Verilog HDL.
LO2: Design & Simulate different designs using Verilog.