Verilog

Lab Manual

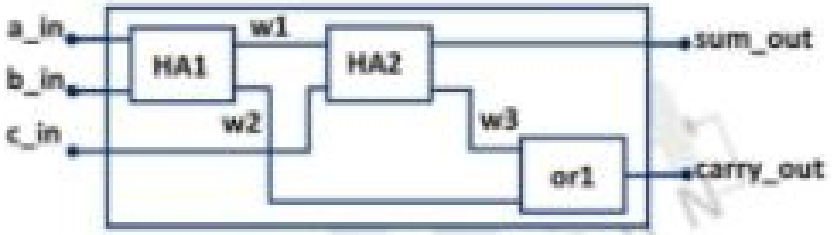
Table of Contents

Lab - 1: Verilog Syntax, Instantiation

Objective : *To understand the different modelling styles in Verilog and learn different port mapping methods.*

Exercise :

Write RTL description and testbench for the Full Adder circuit using half adders and OR gate. The block diagram of full adder, along with complete connections of full adder using half adder and OR gate is shown below.



Working Directory : Verilog_labs/lab1/rtl

Source Code : half_adder.v, full_adder.v

Instructions : The following instructions have been included in the source code as comments. Refer to the comments in the source code and edit the source code.

- ✓ Write down the directions for the ports.
- ✓ Declare the internal wires.
- ✓ Instantiate the Half-Adders.
- ✓ Instantiate the OR gate.

Working Directory : Verilog_labs/lab1/tb

Source Code : full_adder_tb.v

Instructions : The following instructions have been included in the source code as comments. Refer to the comments in the source code and edit the source code.

- ✓ Instantiate the full_adder top with order based port mapping.
- ✓ Understand the global variables required for testbench.
- ✓ Understand how stimulus is generated using a for loop & delays.

Simulation Process :

- ✓ Create a project
- ✓ Add the source files RTL & TB
- ✓ Simulate the TB
- ✓ Observe the waveform & console output

Synthesis Process :

- ✓ Create a project
- ✓ Add the RTL files
- ✓ Synthesize the code
- ✓ Observe the RTL schematic

Learning outcomes :

How to instantiate the sub-blocks in the top RTL module and do port mapping.

Exercises :

1. Write RTL description and test bench for the 4 bits Ripple carry Adder using 1-bit Full adder.
2. Write RTL description and testbench for 4:1 Mux using 2:1 Muxes.

Assignments :

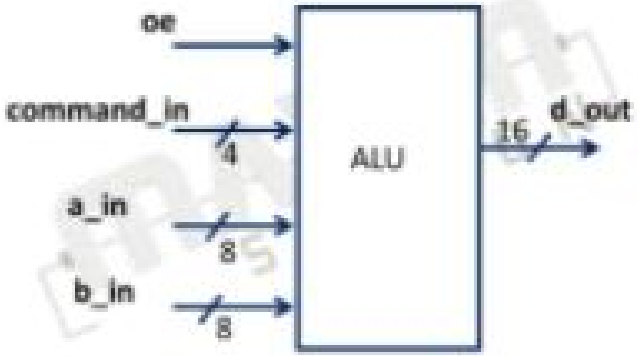
1. Write RTL for 1bit Full adder using Dataflow abstraction.
2. Write RTL for 2x4 decoder using Dataflow abstraction.
3. Write RTL for 8x3 priority encoder using structural model.
4. Write RTL for 4x1 Mux using Decoder & tri-state buffers and verify the same using a Testbench.
5. Write RTL to design a bidirectional buffer and verify the same using a Testbench.

Lab - 2: Operators

Objective : *To understand RTL description for an Arithmetic Logic Unit using arithmetic and logical operators.*

Exercise :

Write RTL description and testbench for an Arithmetic Logic Unit using arithmetic and logical operators. The block diagram and instructions set for ALU are shown below.



Working Directory : Verilog_labs/lab2/rtl

Source Code : alu.v

Instructions : The following instructions have been included in the source code as comments. Refer to the comments in the source code and edit the source code.

- ✓ Write down the functionality of ALU based on commands given.
- ✓ Understand the tri-state output logic.
- ✓ Within always@(command) begin
  - The ALU has to perform 16 different operations using command input on 8bits inputs **a_in** and **b_in.**
  - If the MSB of the **command_in** input is low then the arithmetic operations are performed.
  - If the MSB of **command_in** input is high then the logical operations are performed.
  - Use case construct to perform the operations.
  - The output is of 16bits. Input **oe** enables the ALU, i.e. when oe is low, output is at high impedance.

Working Directory : Verilog_labs/lab2/tb

Source Code : alu_tb.v

Instructions : The following instructions have been included in the source code as comments. Refer to the comments in the source code and edit the source code.

- ✓ Instantiate the design ALU.
- ✓ Understand about the global variables required for testbench.
- ✓ Write a task named "initialize" to initialize the inputs of DUT.
- ✓ Within initial begin
  - Understand how tasks are called and values are passed by "Call by Value" method.
  - Also understand how $monitor is monitoring the changes in the outputs

Simulation Process :

- ✓ Create a project
- ✓ Add the source files RTL & TB
- ✓ Simulate the TB
- ✓ Observe the waveform & console output

Synthesis Process :

- ✓ Create a project
- ✓ Add the RTL files
- ✓ Synthesize the code
- ✓ Observe the RTL schematic

Learning outcomes :

Explore the different operators concept in Verilog.

Assignments :

1. Practise all operators.

Lab - 3: Combinational Logic Design

Objective : *To understand, how to write RTL(behavioral) abstraction for a Multiplexor.*

Exercise :

Write RTL (Behavioral) description and test bench for a 4:1 Multiplexer circuit.The block diagram of 4:1 multiplexer is shown below.

Working Directory : Verilog_labs/lab3/rtl

Source Code : mux4_1.v

Instructions : The following instructions have been included in the source code as comments. Refer to the comments in the source code and edit the source code.

- ✓ Define the port directions with proper datatypes.
- ✓ Write the MUX behaviour as a parallel logic.
- ✓ Within always@(.....) begin
  - Use case construct to represent the behavior of 4:1Mux using its TruthTable as per the select lines.
  - Use Blocking assignments.

Working Directory : Verilog_labs/lab3/tb

Source Code : mux4_1_tb.v

Instructions : The following instructions have been included in the source code as comments. Refer to the comments in the source code and edit the source code.

- ✓ Instantiate the Design.
- ✓ Write down the variables required for testbench.
- ✓ Declare a task to initialize inputs of DUT to 0.
- ✓ Declare tasks with arguments for driving stimulus to DUT.
- ✓ Call the tasks from procedural block.
- ✓ Use $monitor task to display inputs and outputs.
- ✓ Use $finish task to terminate the simulation at 100ns.
- ✓ Within initial begin
  - Call the tasks & pass values by "Call by Value" method.

Simulation Process :

- ✓ Create a project
- ✓ Add the source files RTL & TB
- ✓ Simulate the TB
- ✓ Observe the waveform & console output

Synthesis Process :

- ✓ Create a project
- ✓ Add the RTL files
- ✓ Synthesize the code
- ✓ Observe the RTL schematic

Learning outcomes :

Understand the behavioral abstraction level for designing a combinational circuit.

Exercises :

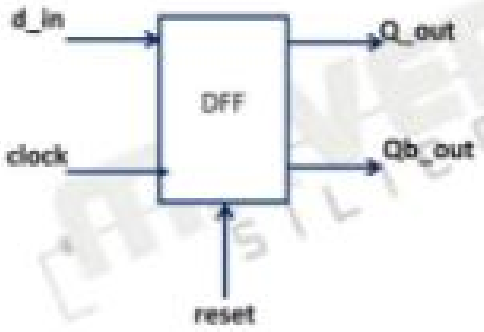
1. Write RTL description and test bench for 3:8 Decoder
2. Write RTL description and testbench for 8:3 Priority encoder

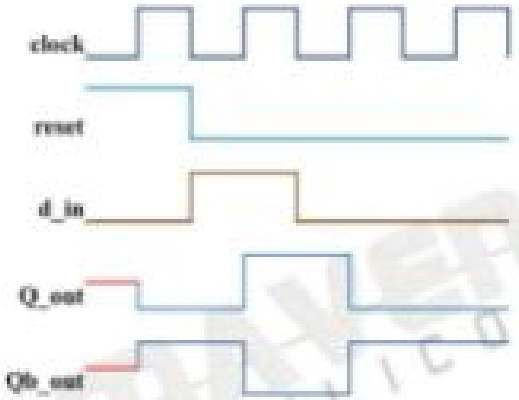
Lab - 4: Sequential Logic Design

Objective : *To understand how to write RTL(behavioral) abstraction for a DFF.*

Exercise :

Write RTL description and testbench for the D flip-flop.
The block diagram, input and output waveforms of D flip-flop is shown below.





Working Directory : Verilog_labs/lab4/rtl
Source Code : dff.v
Instructions : The following instructions have been included in the source code as comments. Refer to the comments in the source code and edit the source code.

- ✓ Define the port directions with proper datatypes.
- ✓ Understand the behavioral logic for D flip-flop functionality.
- ✓ Write the logic for **Qb_out.**
- ✓ Within always@(posedge clk) begin
  - Understand the usage of if construct to reset the DFF and in the else part the normal operation of the Flip-flop.
  - Understand the usage of Non-blocking assignments.

Working Directory : Verilog_labs/lab4/tb

Source Code : dff_tb.v

Instructions : The following instructions have been included in the source code as comments. Refer to the comments in the source code and edit the source code.

- ✓ Instantiate the Design.
- ✓ Understand about the global variables required for testbench.
- ✓ Define a parameter constant with name "CYCLE" which is equal to 10.
- ✓ Understand the clock generation logic.
- ✓ Understand the event based timing control statements within the tasks for generating the stimulus.
- ✓ Use Smonitor to display the various inputs and outputs.
- ✓ Within initial begin
  - Understand how the tasks are called & values are passed by "Call by Value" method.

Simulation Process :

- ✓ Create a project
- ✓ Add the source files RTL & TB
- ✓ Simulate the TB
- ✓ Observe the waveform & console output

Synthesis Process :

- ✓ Create a project
- ✓ Add the RTL files
- ✓ Synthesize the code
- ✓ Observe the RTL schematic

Learning outcomes :

Understand the behavioral abstraction level for designing a sequential circuit.
Also analyze the timing diagram for the circuit using waveform.

Exercises :

1. Write RTL description and test bench for SR latch using Gate level modelling.
2. Write RTL description and testbench for JK Flip Flop, using parameter declaration for the respective scenarios (HOLD, TOGGLE, SET, RESET).
3. Structural model for T Flip Flop using D Flip Flop.
4. Write RTL description and testbench for a 4-bit synchronous and loadable binary up counter.

Assignments :

1. Write RTL to design a 4-bit MOD-12 loadable binary synchronous up counter.
2. Write RTL to design a 4-bit loadable binary synchronous up-down counter.

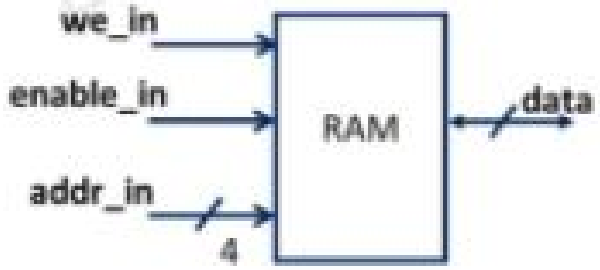
Lab - 5: Memory Logic Design

Objective : *To understand how to write RTL(behavioral) abstraction for an ansynchronous Single Port RAM*

Exercise :

Write RTL description and testbench for the Single Port RAM, which is 8 bit wide and has 16 memory locations. The data can be written on a memory location by providing its address and making "we" high and "oe" low. The data can be read from a memory location by providing the address and making "oe" high and "we" low. This RAM has single port for data writing and reading.

The block diagram of the single port RAM is shown below.



Working Directory : Verilog_labs/lab5/rtl

Source Code : ram.v

Instructions : The following instructions have been included in the source code as comments. Refer to the comments in the source code and edit the source code.

- ✓ Declare a 8 bit wide memory having 16 locations.
- ✓ Understand the logic for writing data into a memory location
- ✓ Understand the logic of reading data from a memory location
- ✓ Within always@(....) begin
  - Understand how the memory write operation is done using a fixed size array such that when write is asserted, read is low.
- ✓ Also understand how read operation is done using continuous concurrent process as data which is an inout port is always of wire type.

Working Directory : Verilog_labs/lab5/rtl

Source Code : ram.v

Instructions : The following instructions have been included in the source code as comments. Refer to the comments in the source code and edit the source code.

- ✓ Declare a 8 bit wide memory having 16 locations.
- ✓ Understand the logic for writing data into a memory location
- ✓ Understand the logic of reading data from a memory location
- ✓ Within always@(....) begin
  - Understand how the memory write operation is done using a fixed size array such that when write is asserted, read is low.
- ✓ Also understand how read operation is done using continuous concurrent process as data which is an inout port is always of wire type.

Simulation Process :

- ✓ Create a project
- ✓ Add the source files RTL & TB
- ✓ Simulate the TB
- ✓ Observe the waveform & console output

Synthesis Process :

- ✓ Create a project
- ✓ Add the RTL files
- ✓ Synthesize the code
- ✓ Observe the RTL schematic

Learning outcomes :

Understand how fixed size array is used to model a Memory design.
Also analyze how inout ports are driven from the Testbench.

Exercise :

Implement a FIFO in Verilog HDL for the below mentioned specification:
A synchronous FIFO 8 bits wide, 16 bytes deep. FIFO has a bank of flip-flops 16X8, to implement the memory. Simultaneous read and write should be possible. Empty, Full should be available and read or write operations will be initiated based on these signals.
Input: clock, reset(asynchronous active low),data_in, read_n and write_n
Output: data_out, full, empty

Assignments :

1. Write RTL to design a 16x8 synchronous dual-port RAM & verify the same.
2. Write RTL to design a 8x16 asynchronous dual-port RAM & verify the same.
3. Write RTL to design a 4-bit right shift Serial-In-Serial-out shift register & verify the same.
4. Write RTL to design a clock buffer & verify the same.

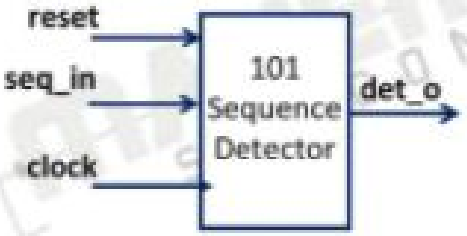
Lab - 6: FSM Design

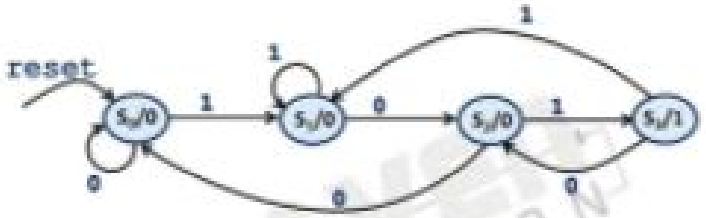
Objective : *To understand how to write RTL(behavioral) abstraction for an overlapping sequence detector 101 using Moore FSM.*

Exercise :

Write RTL description and testbench for the Sequence Detector that detects "101" from input data stream.

The block diagram and state diagram for the "101" sequence detector are shown below.





Working Directory : Verilog_labs/lab6/rtl

Source Code : seq_det.v

Instructions : The following instructions have been included in the source code as comments. Refer to the comments in the source code and edit the source code.

- ✓ Declare the sates as parameter "IDLE","STATE1","STATE2","STATE3".
- ✓ Write down the port direction.
- ✓ Write down the sequential logic for present state.
- ✓ Understand the combinational logic for next state.
- ✓ Write down the logic for output dout.
- ✓ Within always@(posedge clk) begin
  - Write the present state logic using Non-blocking assignments.
- ✓ Within always@(....) begin
  - Understand the next state logic using Blocking assignments.
- ✓ Also understand how output is calculated using continuous assignments.

Working Directory : Verilog_labs/lab6/tb

Source Code : seq_det_tb.v

Instructions : The following instructions have been included in the source code as comments. Refer to the comments in the source code and edit the source code.

- ✓ Generate clock, using parameter "CYCLE".
- ✓ Write a task named "initialize" to initialize the input din of sequence detector.
- ✓ Write a task named "RESET" to reset the design.
- ✓ Write a task named "stimulus" which provides input to design on negedge of clock.
- ✓ Within initial begin
  - Call the tasks & pass values by "Call by Value" method in such a way that the sequence 101 is passed in an overlapping manner.

Simulation Process :

- ✓ Create a project
- ✓ Add the source files RTL & TB
- ✓ Simulate the TB
- ✓ Observe the waveform & console output

Synthesis Process :

- ✓ Create a project
- ✓ Add the RTL files
- ✓ Synthesize the code
- ✓ Observe the RTL schematic

Learning outcomes :

Understand how to use internal registers to write both present state & next state logic. Also analyze how output logic is written for a Moore based sequence detector.

Exercise :

Design a synchronous logic control unit for Vending machine.

The machine can take only two types of coins of denomination 1 and 2 in any order. It delivers only one product that is priced Rs. 3. On receiving Rs. 3, the product is delivered by asserting an output X=1 which otherwise remains 0. If it gets Rs. 4, then the product is delivered by asserting X and also a coin return mechanism is activated by output Y = 1 to return a Re. 1 coin. There are two sensors to sense the denomination of the coins that give binary output as shown in the following table. The clock speed is much higher than human response time, i.e. no two coins can be deposited in same clock cycle.

I	J	Coin
0	X	No coin dropped
1	0	One Rupee
1	1	Two Rupees