VLSI System Design (CS60067) : Autumn Semester 2018

Assignment 2 (Using Verilog)

1. Write a Verilog module X to compute the GCD of two 16-bit numbers. Instantiate the module X as many times as required to find the GCD of 4 16-bit numbers. Modify the design to compute the GCD of N numbers (for any arbitrary N), where the numbers are fed one at a time.

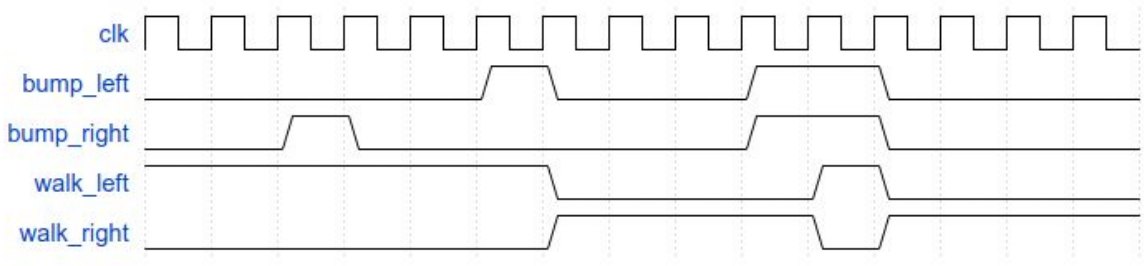
2. The state transition table for a Moore state machine is shown below. It has one input, one output, and four states. Implement this state machine in a Verilog behavioural model. Include a asynchronous reset that resets the FSM to state A.

State	Next State		Output
	in = 0	in = 1	
A	A	B	0
B	C	B	0
C	A	D	0
D	C	B	1

3. Implement a Verilog module *verify_data* which checks if the data going through a transmission channel is golden or not. It has a 10bit register, *golden_data*, preloaded with an expected value. It receives serial data on its *data_in* port. If the first 10bits of the data received matches with the expected value, it notifies by setting the output *data_ok* high, else *data_ok* remains low. The testbench should be designed such that atleast 30 bits of data is fed to *data_in*.

4. Write a Verilog module to implement a 4-to-1 multiplexer using logic gates (through instantiation). Use this module and other gates as necessary to implement a 16-to-1 multiplexer.

5. In the Mario's 2D world, Mario can be in one of two states: walking left or walking right. It will switch directions if it hits an obstacle. In particular, if a Mario is bumped on the left, it will walk right. If it's bumped on the right, it will walk left. If it's bumped on both sides at the same time, it will still switch directions. Implement a Moore state machine with two states, two inputs, and one output that models this behaviour. Design the testbench to show all possible events.



6. Write a Verilog module to implement an 8-bit shift register, with facilities of parallel load, synchronous clear, and shift right. Hence write another module to convert the shift register into a linear feedback shift register such that :

$$x[7] = x[5] \oplus x[4] \oplus x[2] \oplus x[0]$$

, where x[7] is the left most bit, and x[0] is the rightmost.

7. Write a Verilog module to check even parity for a serial data stream. The module receives serial data on its port *data_in* and checks every 6bits of data. If no. of 1 in the 6bits is even then it sets output port *data_parity* for 1clk cycle. Design the testbench which provides *data_in* with at least 24bit serial data stream.

8. Write a Verilog module to implement a full adder. Using the full adder module, write another module to implement a 4-bit ripple carry adder. Write another module to implement a 4-bit carry look-ahead adder.

Submission Guidelines

1. Write clean code with sufficient comments, otherwise it becomes an assignment for us to decipher the code. In each module write what each port is for and what the module does.
2. Give meaningful module and port names, like *testbench_gcd, parity_checker,* etc.
3. Submit a single zip file named ***Rollno1_Rollno2*.zip** .
   If the roll numbers of the group members are 18CS61R00 and 18CS61R01, the file name should be 18CS61R00_18CS61R01.zip
4. The zip will contain the following directory structure.
   ./src
   testbench.v
   gcd.v
   ./output
   <screenshot of output waveforms clearly showing all ports>
5. Mail the assignments to
   <isg@iitkgp.ac.in>
   <brojogopal.sapui@gmail.com>
   <sayandeep.sanyal@gmail.com>
   with subject "**Assignment 2 : Verilog**"