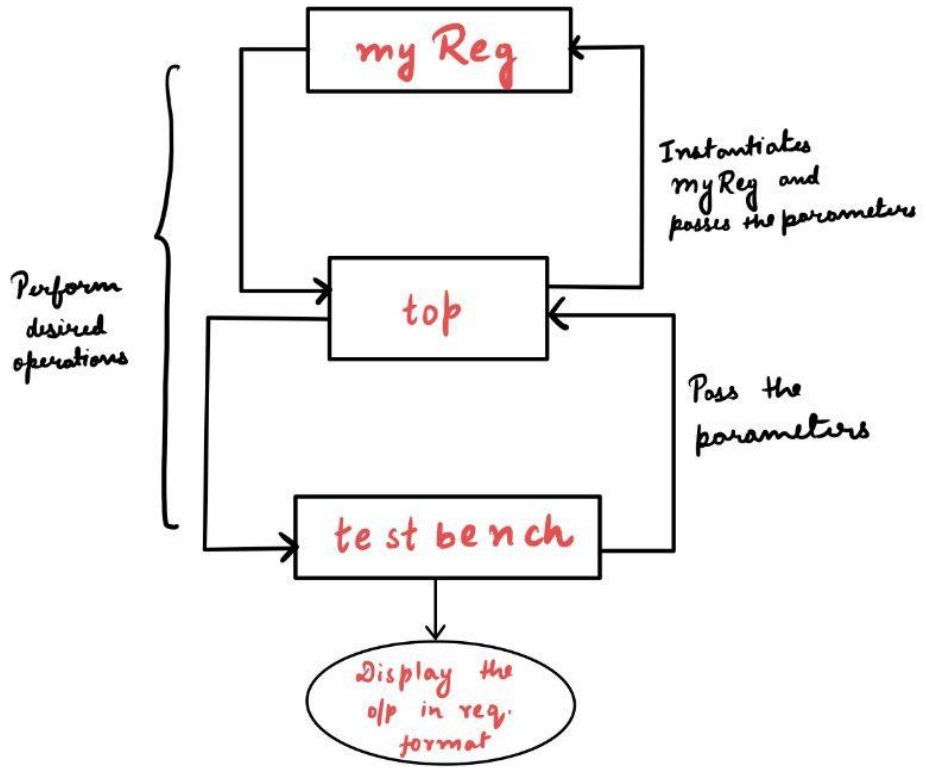
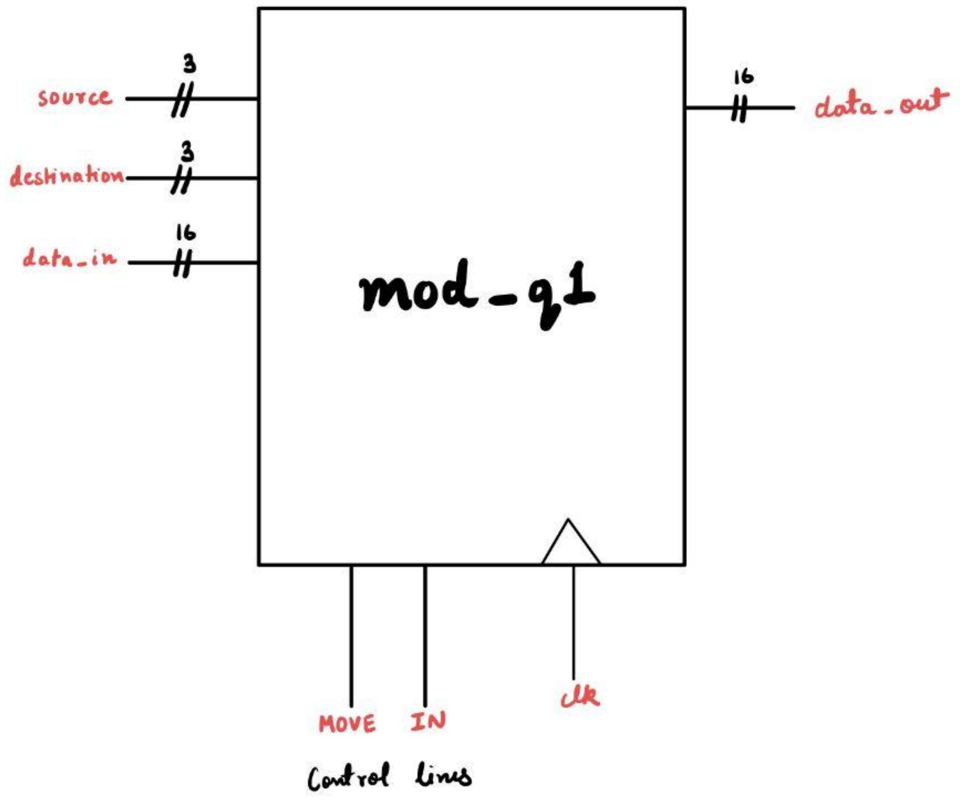
Verilog Assignment 1
Tanishq Prasad (21CS30054)
Krish Khimasia (21CS10037)
Group No. 17

Question 1:

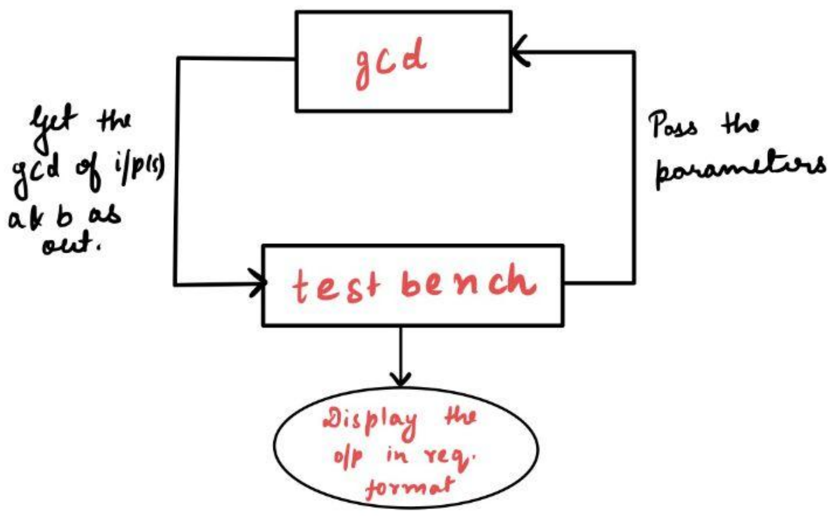

myReg
top
test bench
Instantiates myReg and passes the parameters
Pass the parameters
Perform desired operations
Display the o/p in req. format

- We are using a top-level module called ‘top’, which instantiates the module ‘myReg’.
- The inputs are clock, source register number (3-bits), destination register number (3-bits), move and in control signals, data_in (16-bits) (for writing to register) and data_out (16-bits) (for showing the content of source register).
- The registers R0 to R7 are stored as an array of 16-bit regs.
- When in=1, we write to the destination register and set data_out to high impedance.
- Otherwise, we set the value of data_out as the contents of the source register.
- If in=0, and move=1, we move the contents of the source register to the destination register.
- If both in and move are 0, the module is disabled.


mod_q1
source
3
destination
3
data_in
16
16
data_out
MOVE
IN
clk
Control lines

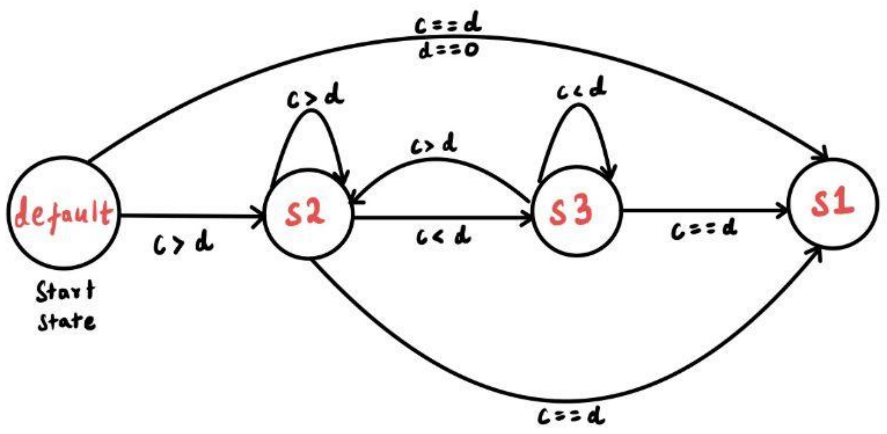
Question 2:

Working of modules:-


gcd
get the gcd of i/p(s) a & b as out.
Pass the parameters
test bench
Display the o/p in req. format

We are using the module gcd for computing the GCD of the two given numbers. The parameters and the inputs are passed through the module testbench where we have initialised our universal clock for all our computations. The parameters are the two inputs, the clock and the output for getting the gcd.

State Diagram:-



Inside the gcd module, we undertake the following steps to calculate:-

- We are taking a and b as inputs and storing the calculated GCD in out. We are using a finite-state machine to calculate the GCD for the given inputs with the states as default, s1, s2 and s3.
- Whenever we get a new a and b pair we set the state as 4.
- For a new pair of a and b, when we first enter the switch case we enter the default state (as the state was set to be 4) where we initialise c and d as the maximum and minimum of a and b respectively. After that, the state is set to 2 and it stays that way unless c==d or d==0 in that case state is set to 1.
- In s2 we update c as (c - d) and then compare c and d. If c > d we transition to s2, c < d we transition to s3 and c == d we transition to s1.
- In s3 we update d as (d - c) and then compare c and d. If c > d we transition to s2, c < d we transition to s3 and c == d we transition to s1.
- s1 is the final state where we save the GCD in out.

Testbench:-

- For displaying the output through the testbench we are using $display whenever a or b gets updated.
- We would need to store a_prev and b_prev as the previous values of a and b as the GCD gets stored in out after some clock cycles instead of the initial clock cycle.
- As we are printing the output previous one when a or b gets updated we need to give a buffer input in the end so that it could trigger the printing of the last result.
- The delays are provided in such a way that each input gets ample time to compute GCD.