Nem Negash

CMPE 415

Prof. Mohsenin

February 21th 2021

Homework 1 Report

Statement of Project Completion

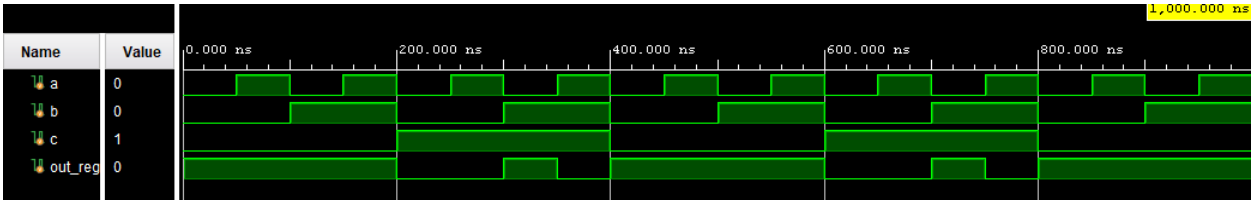
What I got to work:

Problem 1:

a) Implemented truth table as shown in the wave form.

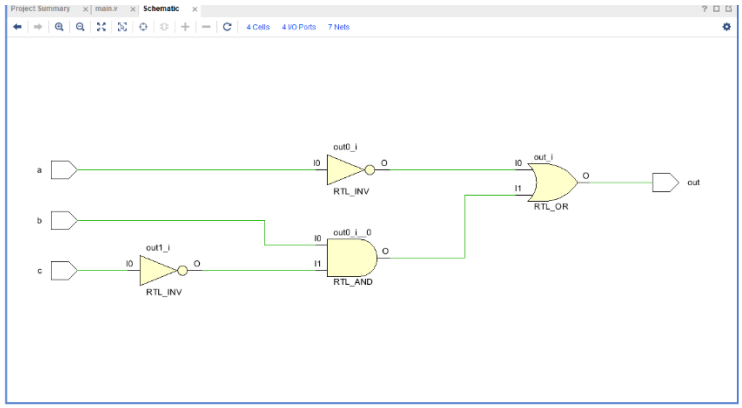
```
`timescale 1ns / 1ps
module part1();
   //inputs
   reg a = 1'b0;
   reg b = 1'b0;
   reg c = 1'b0;
  //outputs
   reg out_reg = 1'b0;

   always @(*)
   begin
```

```
        case ({c,b,a})//case statment using the inputs
            3'b000  : out_reg = 1;
            3'b001  : out_reg = 1;
            3'b100  : out_reg = 0;
            3'b101  : out_reg = 0;
            3'b110  : out_reg = 1;
            3'b111  : out_reg = 0;
    endcase
  end
//testbench
initial begin
//forloop that checks using all forms of inputs
for(reg[2:0] temp = 3'b000; temp < 8; temp = temp + 1)
begin
    a = temp[0];
    b = temp[1];
    c = temp[2];
    #50;
end


end


endmodule
```



The waveform shows the exact truth tables with the ‘don’t care’ values as ones.

b) Completed the RTL Schematic.



Problem 2:

a) Designed and implemented each ALU operations

```
`timescale 1ns / 1ps


module part2(
 input signed [8:0] A,B,  // inputs
        input signed [3:0] S,// select
        output signed [8:0] Q // output
   );
   reg signed [8:0] Result;
   assign Q = Result; // out
```

```
reg [8:0] count = 9'b000000000;//counter for rotate
always @(*)
begin
    case(S)
    4'b0000: begin // Addition0
       Result = A + B ;
       end
    4'b0001: begin // Subtraction1
       Result = A - B ;
       end
    4'b0010: begin// Multiplication2
       Result = A * B;
       end
     4'b0011: begin// Logical shift right3
       Result = B>>A;
       end
     4'b0100: begin// Rotate left4
       Result = {B[9:0]};
       for(count = 0; count < A; count  = count + 1'b1)
       begin
       Result = {Result[7:0],Result[8]};
       end
       end
```

```
4'b0101: begin//  Logical or 5
 Result = A | B;
 end
4'b0110: begin//  Logical and 6
 Result = A & B;
 end
4'b0111: begin// Logical nand 7
 Result = !(A & B);
 end
4'b1000: begin// Logical xnor 8
 Result = 9'b111111111 ^ (A ^ B);
 end
4'b1001: begin// complement of B 9
 Result = (B ^ 9'b111111111) + 1'b1;
 end
default: Result = 9'b000000000 ;
endcase
end

endmodule
```

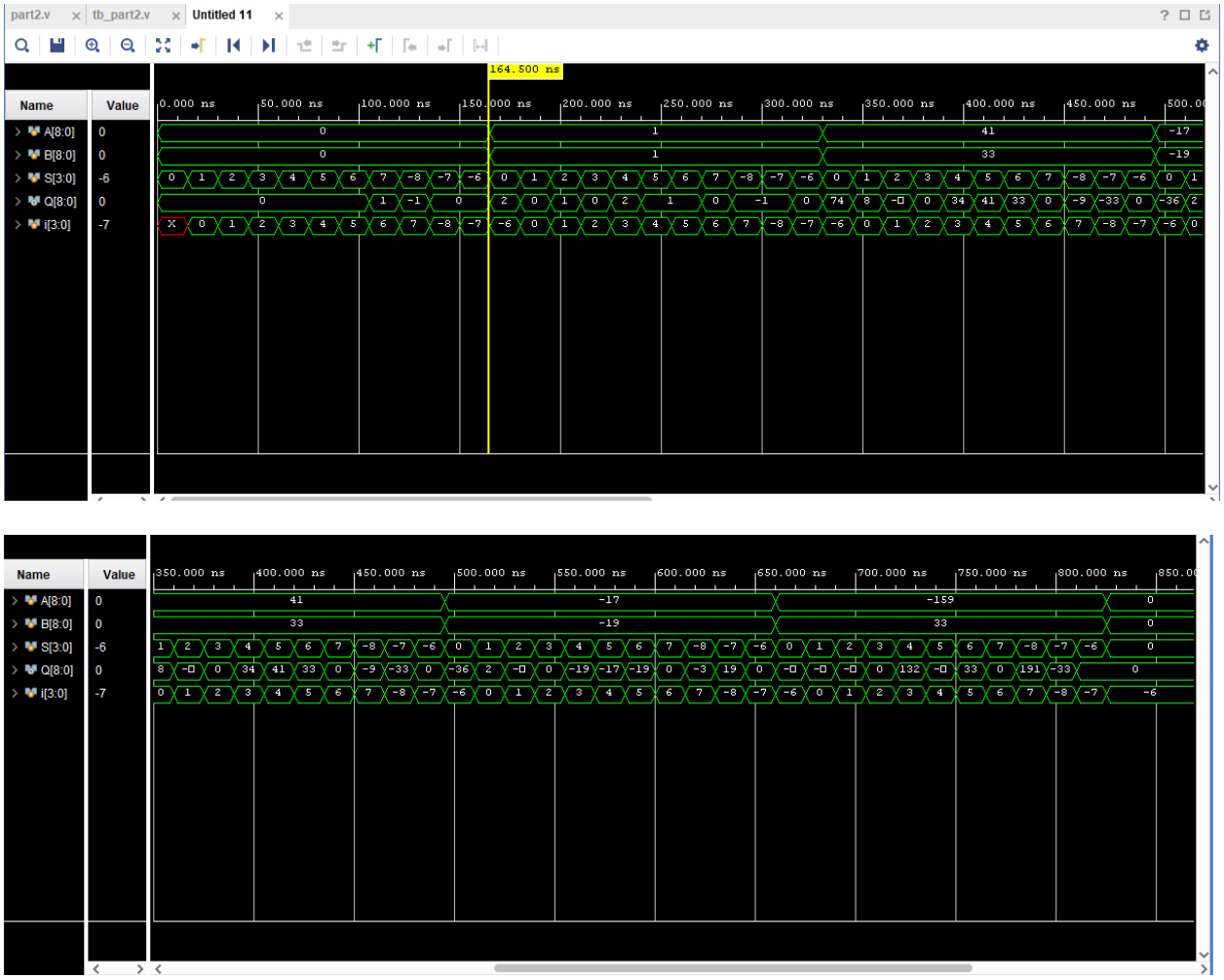
b) Conducted a test bench for each operation with all possible edge cases.

```
`timescale 1ns / 1ps
```

```
module tb_part2;
 reg[8:0] A,B;
 reg[3:0] S;

//Output
 wire[8:0] Q;
 //counter
 reg[3:0] i;
 part2 test_unit(
          .A(A),.B(B),  //Inputs
          .S(S),//Select
          .Q(Q)//output
    );
   initial begin
   // testing all 0s
     A = 9'b000000000;
     B = 9'b000000000;
     S = 4'b0000
     #15
     for (i=0;i <= 9;i = i+ 1'b1)
     begin
      S = S + 4'b0001;
      #15;
```

```
end
//testing all 1s
A = 9'b000000001;
B = 9'b000000001;
S = 4'b0000;
#15
for (i=0;i <= 9;i = i+ 1'b1)
begin
 S = S + 4'b0001;
 #15;
end;
//testing all positive inputs
A = 9'b000101001;
B = 9'b000100001;
S = 4'b0000;
#15
for (i=0;i <= 9;i = i+ 1'b1)
begin
 S = S + 4'b0001;
 #15;
end;
//testing all negative inputs
A = 9'b111101111;
```

```
    B = 9'b111101101;
    S = 4'b0000;
    #15
    for (i=0;i <= 9;i = i+ 1'b1)
    begin
     S = S + 4'b0001;
     #15;
    end;
    //testing one positive and one negative input
    A = 9'b101100001;
    B = 9'b000100001;
    S = 4'b0000;
    #15
    for (i=0;i <= 9;i = i+ 1'b1)
    begin
     S = S + 4'b0001;
     #15;
    end;
    A = 9'b000000000;
    B = 9'b000000000;
    S = 4'b0000;
  end
endmodule
```



The waveform diagram shows the input values with all edge cases and their output for each operation.

Problem 3:

This part was not completed (Look at the “What I did not get to work” section).

Problem 4:

a) Implement Verilog code to represent a 4-bit counter.

timescale 1ns / 1ps

module part4(input CLK, CLR,ENABLE, UP,LOAD, //single bit inputs

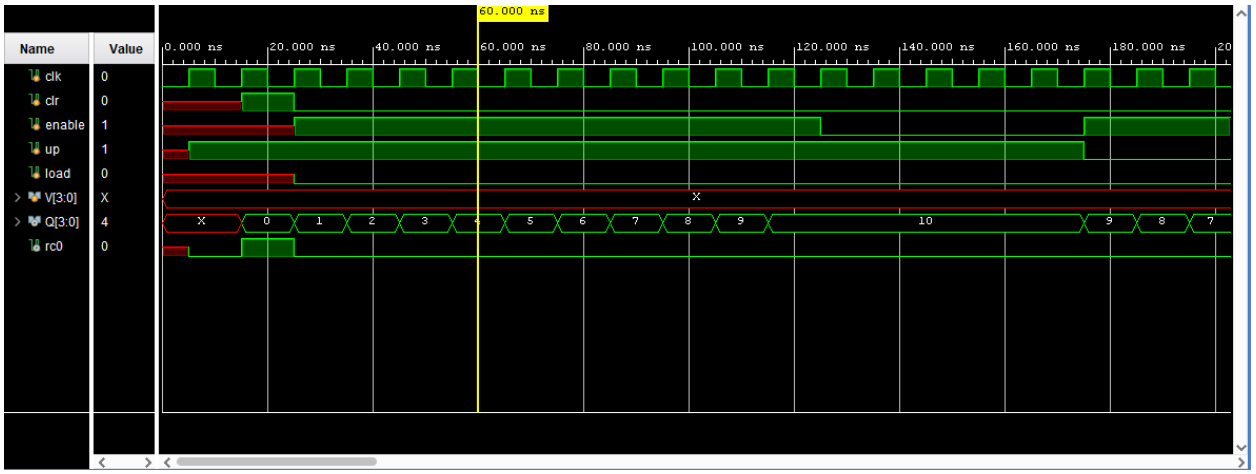
```
    output reg [3:0] Q, //output
    input [3:0] V, //load value
    output reg RC0); //single bit output
  //at clock edge
   always @(posedge CLK )
    begin
      RC0 = 1'b0;//start with RC0 as 0
     if (CLR) //if clear is 1
      Q = 4'b0000;
     else //if clear is 0
      if(ENABLE)
        if(LOAD)//is load is 1
          Q = V;
        else//if load is 0
          if (UP) //if up is 1
            Q = Q + 1'b1;
          if(!UP) //if up is 0
            Q = Q - 1'b1;
          if(Q == 4'b1111 | Q == 4'b0000)//if the output is 15 or 0
              RC0 = 1'b1;
       end
    endmodule
```

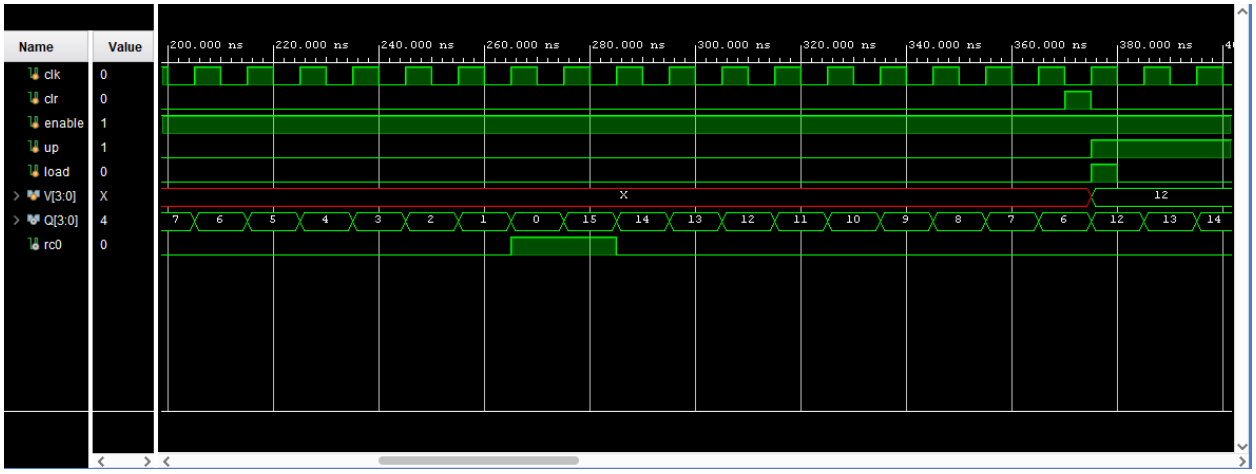
b) Create a test bench for the test bench as specified in the project description

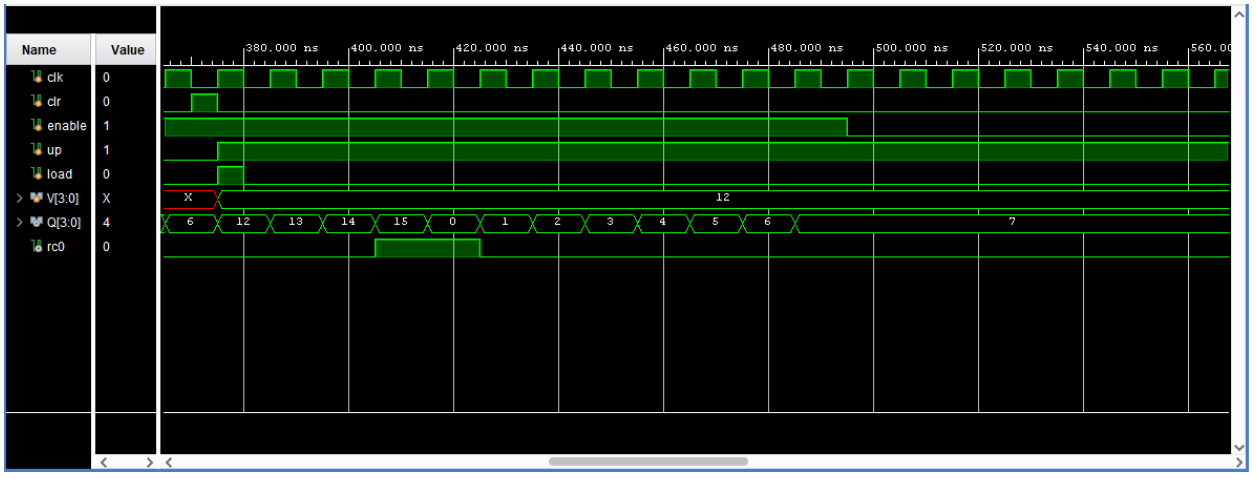
```
`timescale 1ns / 1ps

module tb_part4;

//variables

reg clk,clr,enable,up,load;

reg[3:0] V;

wire [3:0] Q;

wire rc0;


part4 test_unit(.CLK(clk),.CLR(clr), .UP(up), .Q(Q), .ENABLE(enable),

.LOAD(load), .V(V), .RC0(rc0));

//start clock

always #5 clk = (clk ^ 1'b1);


initial begin

//setup

clk = 0;

#5

up = 1;

#10

clr = 1;


//first 10 cycles

#10
```

```
clr = 0;

enable = 1;

up = 1;

load = 0;

//hold value for 5 cycles

#100 enable = 0;

//count down  for 20 cycles

#50 enable = 1;up = 0;

//clear counter

#195 clr = 1;

//load V value into counter

#5 clr = 0; V = 4'b1100;load = 1; up = 1;

#5 clr = 0; load = 0;

//count up for 11 cycles

#115 enable = 0;

end

endmodule
```

c) Verify code with wave form diagram







What I did not get to work:

Problem 1:

a) I was not able to have a separate test bench file as a simulation source. My test bench is written within the design file

Problem 2:

a) I wasn't able to have each operation within it's own module but rather all in one file.

Problem 3:

a) I wasn't able to figure out how to have the output be a registered output without just assigning the output to a reg variable.

Problem 4:

a) Completed with no exceptions.