Q1

```
module Master(ss, MOSI, done, data_in, clk, rst);
    output reg ss, MOSI, done;
    input data_in, clk, rst;

    integer count;

    always @ (posedge clk) begin
        done = 0;
        if (rst != 1) begin
            ss = 1;
            if (ss == 1) begin
                MOSI = data_in;
                if (data_in == 1) begin
                    count = count + 1;
                end
                else if (data_in == 0) begin
                    count = 0;
                end
                if (count == 4) begin
                    ss <= 0;
                    done <= 1;
                end
            end
        end
        else begin
            done <= 0;
            ss <= 0;
            count <= 0;
        end
    end
endmodule
```

Q2

```
module top_module(
    input clk,
    input load,
    input [255:0] data,
    output reg [255:0] q);
```

```
    reg curr [15:0] [15:0];
    reg next [15:0] [15:0];
    integer i, j, k;
    integer count;
    always @ (posedge clk) begin
        if (load == 1) begin
            curr[0] = data[15:0];
            curr[1] = data[31:16];
            curr[2] = data[47:32];
            curr[3] = data[63:48];
            curr[4] = data[79:64];
            curr[5] = data[95:80];
            curr[6] = data[111:96];
            curr[7] = data[127:112];
            curr[8] = data[143:128];
            curr[9] = data[159:144];
            curr[10] = data[175:160];
            curr[11] = data[191:176];
            curr[12] = data[207:192];
            curr[13] = data[223:208];
            curr[14] = data[239:224];
            curr[15] = data[255:240];
        end
        for (i=1 ; i<=254 ; i=i+1) begin
            count = 0;
            for (j=1 ; j<=254 ; j=j+1) begin
                if (curr[i][j+1] == 1) count = count + 1;
                if (curr[i][j-1] == 1) count = count + 1;
                if (curr[i+1][j+1] == 1) count = count + 1;
                if (curr[i-1][j-1] == 1) count = count + 1;
                if (curr[i+1][j] == 1) count = count + 1;
                if (curr[i-1][j] == 1) count = count + 1;
                if (curr[i+1][j-1] == 1) count = count + 1;
                if (curr[i-1][j+1] == 1) count = count + 1;

                if (count == 0 || count == 1) curr[i][j] = 0;
                else if (count == 3) curr[i][j] = 1;
                else if (count >= 4) curr[i][j] = 0;
            end
        end
```

```
    for (j=1 ; j<=254 ; j=j+1) begin //first row
        count = 0;
            if (curr[0][j+1] == 1) count = count + 1;
            if (curr[0][j-1] == 1) count = count + 1;
            if (curr[1][j+1] == 1) count = count + 1;
            if (curr[15][j-1] == 1) count = count + 1;
            if (curr[1][j] == 1) count = count + 1;
            if (curr[15][j] == 1) count = count + 1;
            if (curr[1][j-1] == 1) count = count + 1;
            if (curr[15][j+1] == 1) count = count + 1;

            if (count == 0 || count == 1) curr[0][j] = 0;
            else if (count == 3) curr[0][j] = 1;
            else if (count >= 4) curr[0][j] = 0;
    end

    for (j=1 ; j<=254 ; j=j+1) begin //last row
        count = 0;
            if (curr[15][j+1] == 1) count = count + 1;
            if (curr[15][j-1] == 1) count = count + 1;
            if (curr[0][j+1] == 1) count = count + 1;
            if (curr[14][j-1] == 1) count = count + 1;
            if (curr[0][j] == 1) count = count + 1;
            if (curr[14][j] == 1) count = count + 1;
            if (curr[0][j-1] == 1) count = count + 1;
            if (curr[14][j+1] == 1) count = count + 1;

            if (count == 0 || count == 1) curr[15][j] = 0;
            else if (count == 3) curr[15][j] = 1;
            else if (count >= 4) curr[15][j] = 0;
    end

    for (i=1 ; i<=254 ; i=i+1) begin //first column
        count = 0;
            if (curr[i][0] == 1) count = count + 1;
            if (curr[i][15] == 1) count = count + 1;
            if (curr[i+1][1] == 1) count = count + 1;
            if (curr[i-1][15] == 1) count = count + 1;
            if (curr[i+1][0] == 1) count = count + 1;
```

```
            if (curr[i-1][0] == 1) count = count + 1;
            if (curr[i+1][15] == 1) count = count + 1;
            if (curr[i-1][1] == 1) count = count + 1;

            if (count == 0 || count == 1) curr[i][0] = 0;
            else if (count == 3) curr[i][0] = 1;
            else if (count >= 4) curr[i][0] = 0;
        end

        for (i=1 ; i<=254 ; i=i+1) begin //last column
            count = 0;
            if (curr[i][15] == 1) count = count + 1;
            if (curr[i][14] == 1) count = count + 1;
            if (curr[i+1][0] == 1) count = count + 1;
            if (curr[i-1][14] == 1) count = count + 1;
            if (curr[i+1][15] == 1) count = count + 1;
            if (curr[i-1][15] == 1) count = count + 1;
            if (curr[i+1][14] == 1) count = count + 1;
            if (curr[i-1][0] == 1) count = count + 1;

            if (count == 0 || count == 1) curr[i][15] = 0;
            else if (count == 3) curr[i][15] = 1;
            else if (count >= 4) curr[i][15] = 0;
        end
        count = 0;
        //for 00
        if (curr[0][1] == 1) count = count + 1;
        if (curr[1][0] == 1) count = count + 1;
        if (curr[1][1] == 1) count = count + 1;
        if (curr[15][1] == 1) count = count + 1;
        if (curr[15][15] == 1) count = count + 1;
        if (curr[15][0] == 1) count = count + 1;
        if (curr[0][15] == 1) count = count + 1;
        if (curr[1][15] == 1) count = count + 1;
        if (count == 0 || count == 1) curr[0][0] = 0;
            else if (count == 3) curr[0][0] = 1;
            else if (count >= 4) curr[0][0] = 0;

        count = 0;
        //for 00
```

```
        if (curr[0][15] == 1) count = count + 1;
        if (curr[15][1] == 1) count = count + 1;
        if (curr[14][1] == 1) count = count + 1;
        if (curr[14][0] == 1) count = count + 1;
        if (curr[15][15] == 1) count = count + 1;
        if (curr[0][0] == 1) count = count + 1;
        if (curr[1][0] == 1) count = count + 1;
        if (curr[15][14] == 1) count = count + 1;
        if (count == 0 || count == 1) curr[15][0] = 0;
                else if (count == 3) curr[15][0] = 1;
                else if (count >= 4) curr[15][0] = 0;

        count = 0;
        //for 00
        if (curr[0][0] == 1) count = count + 1;
        if (curr[0][15] == 1) count = count + 1;
        if (curr[15][0] == 1) count = count + 1;
        if (curr[14][15] == 1) count = count + 1;
        if (curr[14][14] == 1) count = count + 1;
        if (curr[15][14] == 1) count = count + 1;
        if (curr[1][15] == 1) count = count + 1;
        if (curr[15][1] == 1) count = count + 1;
        if (count == 0 || count == 1) curr[15][15] = 0;
                else if (count == 3) curr[15][15] = 1;
                else if (count >= 4) curr[15][15] = 0;

        count = 0;
        //for 00
        if (curr[15][0] == 1) count = count + 1;
        if (curr[0][0] == 1) count = count + 1;
        if (curr[15][15] == 1) count = count + 1;
        if (curr[14][15] == 1) count = count + 1;
        if (curr[0][1] == 1) count = count + 1;
        if (curr[1][15] == 1) count = count + 1;
        if (curr[0][14] == 1) count = count + 1;
        if (curr[1][14] == 1) count = count + 1;
        if (count == 0 || count == 1) curr[0][15] = 0;
                else if (count == 3) curr[0][15] = 1;
                else if (count >= 4) curr[0][15] = 0;
```

```
        q[15:0] = curr[0];
        q[31:16]= curr[1];
            q[47:32]= curr[2];
            q[63:48]= curr[3];
            q[79:64]= curr[4];
            q[95:80]= curr[5];
            q[111:96]= curr[6];
            q[127:112]= curr[7];
            q[143:128]= curr[8];
            q[159:144]= curr[9];
            q[175:160]= curr[10];
            q[191:176]= curr[11];
            q[207:192]= curr[12];
            q[223:208]= curr[13];
            q[239:224]= curr[14];
            q[255:240]= curr[15];
    end
endmodule
```

Q3

```
module divider(input [7:0] dividend, divisor,
output reg [7:0] quotient, remainder,
output reg divisor_zero_flag);

initial begin
     quotient = 0;
     remainder = 0;
     divisor_zero_flag = 0;
end
reg dzf;
integer i;
always @ (*) begin
    if (divisor == 0) begin
        divisor_zero_flag = 1;
    end
    if (divisor_zero_flag == 0) begin
    remainder = dividend - divisor;
    quotient = 1;
    for (i=0; i<=dividend; i=i+1) begin
        if (remainder > 0) begin
```

```
            remainder = remainder - divisor;
            quotient = quotient + 1;
        end
        else if (remainder == 0) begin
            $finish;
        end
        else if (remainder < 0) begin
            quotient = quotient - 1;
            remainder = remainder + divisor;
            $finish;
        end
    end
    end
end
endmodule
```

Q4

```
module conv(input [15:0][7:0] signal_a,
input [15:0][7:0] signal_b,
output reg [30:0][7:0] convoluted_signal,
output reg invalid_input_flag);

initial begin
    invalid_input_flag = 0;
end

integer i;
integer j;
integer count_a, count_b;
always @ (*) begin
    count_a = 0;
    for (i=0 ; i<=15 ; i=i+1) begin
        if (signal_a[i] == 0)
            count_a = count_a + 1;
    end
    count_b = 0;
    for (i=0 ; i<=15 ; i=i+1) begin
        if (signal_a[i] == 0)
            count_b = count_b + 1;
    end
```

```
    if (count_a == 16 || count_b == 16) begin
        invalid_input_flag = 1;
    end
end

reg [7:0] rev [15:0];
always @ (*) begin
    if (invalid_input_flag == 0) begin
    for (i=0 ; i<=15 ; i=i+1)begin
        rev[i] = signal_b[15-i];
    end
    for (i=0 ; i<=15; i=i+1) begin
        for (j=0 ; j<i+1 ; j=j+1) begin
            convoluted_signal[i] = convoluted_signal[i] +
signal_a[j]*rev[(15-i)+j];
        end
    end
    for (i=0 ; i<=14; i=i+1) begin
        for (j=0 ; j<i+1 ; j=j+1) begin
            convoluted_signal[30-i] = convoluted_signal[30-i] +
rev[j]*signal_a[15-i+j];
        end
    end
    end
end
endmodule
```

Q5