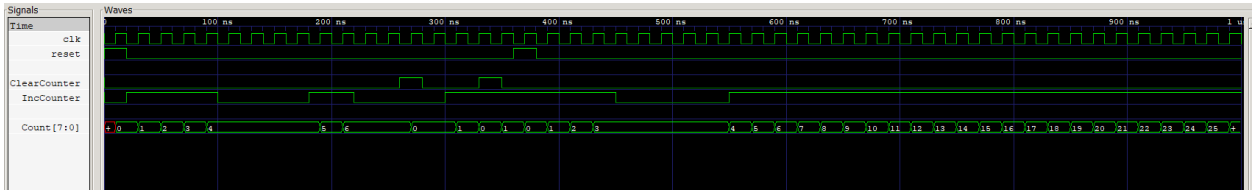
Michael Pate

Problem 1.

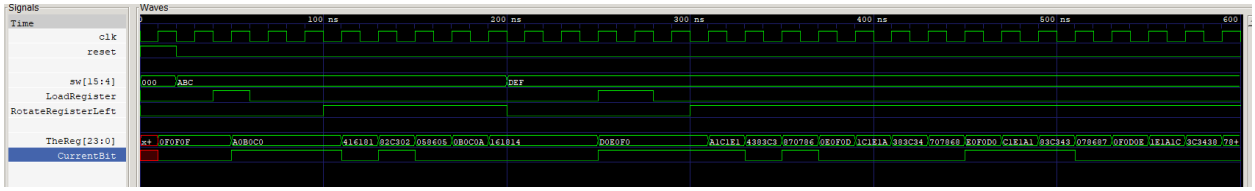



```
// Michael Pate
// Professor Chao Jiang
// EE 4490 HDL Design
// Homework 3 - September 23, 2021
// BitCounter.v

module BitCounter(Count, ClearCounter, IncCounter, clk, reset);
    output [7:0] Count;
    input ClearCounter, IncCounter;
    input clk, reset;

    reg [7:0] Count;

    // Handle reset or clear counter inputs
    always @(posedge clk)
    begin
        if (reset == 1) Count <= 0;
        else if (ClearCounter == 1) Count <= 0;
        else if (IncCounter == 1) Count <= Count + 1;
    end

endmodule
```

```
// Michael Pate
// Professor Chao Jiang
// EE 4490 HDL Design
// Homework 3 - September 23, 2021
// tb_BitCounter.v
```

```
`timescale 10ns/100ps
module tb_BitCounter;
    wire [7:0] t_Count;

    reg t_ClearCounter, t_IncCounter;
    reg t_clk, t_reset;

    BitCounter uut(t_Count, t_ClearCounter, t_IncCounter, t_clk,
t_reset);

    initial begin $dumpfile("tb_BbitCounter.vcd"); $dumpvars(0,
tb_BitCounter);
    end

    initial begin t_clk = 0; forever #1 t_clk = ~t_clk; end

    initial begin t_ClearCounter = 0; t_IncCounter = 0; t_reset = 1; end

    initial
        fork
            #2      begin t_reset = 0; t_IncCounter = 1; end
            #10     t_IncCounter = 0;
            #18     t_IncCounter = 1;
            #22     t_IncCounter = 0;
            #26     t_ClearCounter = 1;
            #28     t_ClearCounter = 0;
            #30     t_IncCounter = 1;
            #33     t_ClearCounter = 1;
            #35     t_ClearCounter = 0;
            #36     t_reset = 1;
            #38     t_reset = 0;
            #45     t_IncCounter = 0;
            #50     t_ClearCounter = 0;
            #55     t_IncCounter = 1;
            #100    $finish;
        join
endmodule
```

Problem 2.



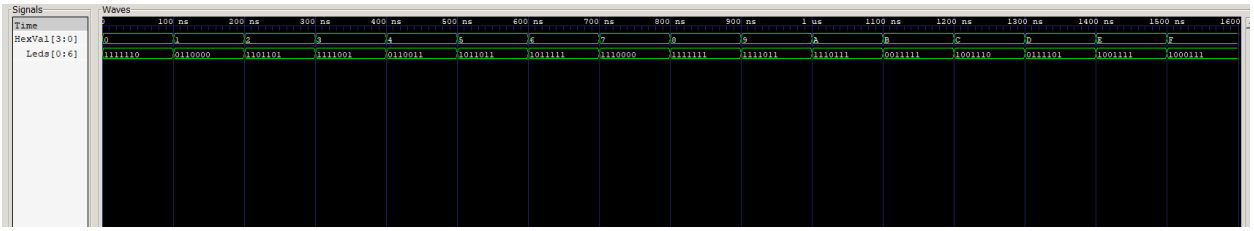

```
// Michael Pate
// Professor Chao Jiang
// EE 4490 HDL Design
// Homework 3 - September 23, 2021
// ShiftRegister.v

module ShiftRegister(CurrentBit, sw, LoadRegister, RotateRegisterLeft,
clk, reset);
    output CurrentBit;
    input [15:4] sw;
    input LoadRegister, RotateRegisterLeft;
    input clk, reset;

    parameter DEFAULTREG = 24'h0F0F0F;

    reg [23:0] TheReg, nTheReg;

    assign CurrentBit = TheReg[23];

    always @(posedge clk)
    begin
        if (reset)
        begin
            TheReg <= DEFAULTREG;
        end
        else if (LoadRegister == 1)
        begin
            TheReg[3:0]     <= 4'b0;
            TheReg[7:4]     <= sw[7:4];
            TheReg[11:8]    <= 4'b0;
            TheReg[15:12]   <= sw[11:8];
            TheReg[19:16]   <= 4'b0;
```

```
            TheReg[23:20]    <= sw[15:12];
        end
        else if (RotateRegisterLeft == 1)
        begin
            TheReg <= {TheReg[22:0], TheReg[23]};
            //TheReg <= (TheReg << 1) | (TheReg >> ~1);
        end
    end


endmodule
```

```
// Michael Pate
// Professor Chao Jiang
// EE 4490 HDL Design
// Homework 3 - September 23, 2021
// tb_ShiftRegister.v

`timescale 10ns/10ps
module tb_ShiftRegister;
    reg t_clk, t_reset;
    wire t_CurrentBit;

    reg t_LoadRegister, t_Rotate;
    reg [15:4] t_sw;

    ShiftRegister uut(t_CurrentBit, t_sw, t_LoadRegister, t_Rotate,
t_clk, t_reset);

    initial begin $dumpfile("tb_ShiftRegister.vcd"); $dumpvars(0,
tb_ShiftRegister);
    end

    initial begin t_clk = 0; forever #1 t_clk = ~t_clk; end

    initial begin t_LoadRegister=0; t_Rotate=0; t_sw=12'b0; t_reset=1;
end
```

```
    initial
        fork
            #2  t_reset = 0;
            #2  t_sw = 12'hABC;
            #4  t_LoadRegister = 1;
            #6  t_LoadRegister = 0;
            #10 t_Rotate = 1;

            #20 t_Rotate = 0;
            #20 t_sw = 12'hDEF;
            #25 t_LoadRegister = 1;
            #28 t_LoadRegister = 0;
            #30 t_Rotate = 1;

            #60 $finish;
        join
endmodule
```

Problem 3.




```
// Michael Pate
// Professor Chao Jiang
// EE 4490 HDL Design
// Homework 3 - September 23, 2021
// Hex27Seg.v

module Hex27Seg(Leds, HexVal);
    output [0:6] Leds;
    input [3:0] HexVal;
    reg [0:6] Leds;

    always @(HexVal)
    begin
        case (HexVal)
            4'b0000: Leds = 7'b1111110;
            4'b0001: Leds = 7'b0110000;
            4'b0010: Leds = 7'b1101101;
            4'b0011: Leds = 7'b1111001;
            4'b0100: Leds = 7'b0110011;
            4'b0101: Leds = 7'b1011011;
            4'b0110: Leds = 7'b1011111;
            4'b0111: Leds = 7'b1110000;
            4'b1000: Leds = 7'b1111111;
            4'b1001: Leds = 7'b1111011;
            4'b1010: Leds = 7'b1110111;
            4'b1011: Leds = 7'b0011111;
            4'b1100: Leds = 7'b1001110;
            4'b1101: Leds = 7'b0111101;
            4'b1110: Leds = 7'b1001111;
            4'b1111: Leds = 7'b1000111;
        endcase
```

```
    end
endmodule
```

```
// Michael Pate
// Professor Chao Jiang
// EE 4490 HDL Design
// Homework 3 - September 23, 2021
// tb_Hex27Seg.v

`timescale 10ns/10ps
module tb_Hex27Seg;

    wire [0:6] t_Leds;
    reg [3:0] t_HexVal;

    initial begin $dumpfile("tb_Hex27Seg.vcd"); $dumpvars(0,
tb_Hex27Seg); end

    Hex27Seg uut(t_Leds, t_HexVal);

    initial t_HexVal = 4'b0;

    initial
    fork
        #00     t_HexVal = 4'b0;
        #10     t_HexVal = 4'h1;
        #20     t_HexVal = 4'h2;
        #30     t_HexVal = 4'h3;
        #40     t_HexVal = 4'h4;
        #50     t_HexVal = 4'h5;
        #60     t_HexVal = 4'h6;
        #70     t_HexVal = 4'h7;
        #80     t_HexVal = 4'h8;
        #90     t_HexVal = 4'h9;
        #100    t_HexVal = 4'hA;
        #110    t_HexVal = 4'hB;
        #120    t_HexVal = 4'hC;
```

```
        #130      t_HexVal = 4'hD;
        #140      t_HexVal = 4'hE;
        #150      t_HexVal = 4'hF;
        #160      $finish;
    join

endmodule
```