1.(3-70)

module decoder_2_to_4_st(B, E, D);
    input [1:0] B;
    input E;
    output [3:0] D;
    wire [1:0] not_A;
    not(
        not_A[0],
        B[0],
    );
    not(
        not_A[1],
        B[1],
    );
    nand
        g2(D[0], not_A[0], not_A[1], E),
        g3(D[1], B[0], not_A[1], E),
        g4(D[2], not_A[0], B[1], E),
        g5(D[3], B[0], B[1], E);
endmodule
g5: NAND3 port map (A(0), A(1), E, D(3));
end structural_1;

2.(3-72*)

3.(3-75)
4.(3-81)

/ 4-bit Adder: Behavioral Verilog Description
module adder_4_b_v(A, B, Sel, S, C4);
  input[3:0] A, B;
  input Sel;
  output[3:0] S;
  output C4;
  assign {C4, S} = Sel ? (A - B):(A + B);
endmodule

5.(4-44)
module problem_4_44 (S, D, Y);
  input [1:0] S;
  input [3:0] D;
  output Y;
  reg Y;
  // (continued in the next column)
  always @(S or D)
  begin
    case (S)
      2'b00 : Y <= D[0];
      2'b01 : Y <= D[1];
      2'b10 : Y <= D[2];
      2'b11 : Y <= D[3];
    endcase;
  end
endmodule

6.(4-47*)
module JK_FF (J, K, CLK, Q);
  input J, K, CLK;
  output Q;
  reg Q;
  // (continued in the next column)
  always @(negedge CLK)
  begin
    case (J)
      0'b0 : Q <= K ? 0: Q;
      1'b1 : Q <= K ? ~Q: 1;
    endcase
  end
endmodule
7.(4-48)  (See Errata: Delete the sentence “Use the D flip-flop ...”)

module seq_circuit (DIN, CLK, RESET, DOUT);
input DIN, CLK, RESET;
output DOUT;
reg [2:0] state, next_state, DOUT;

parameter A = 3'b000, B = 3'b001, C = 3'b100, D = 3'b110, E = 3'b101, F = 3'b111, G = 3'b110, H = 3'b111;

//State register process
always @ (posedge CLK or posedge RESET)
begin
  if (RESET) //asynchronous RESET active High
    state <= A;
  else //use CLK rising edge
    state <= next_state;
end

//Next state function
always @(DIN or state)
begin
  case (state)
    A: next_state <= DIN ? C : B;
    B: next_state <= DIN ? E : D;
    C: next_state <= F;
    D: next_state <= G;
    E: next_state <= DIN ? H : G;
    F: next_state <= DIN ? H : G;
    G: next_state <= A;
    H: next_state <= A;
  endcase
end

//Output function
always @(DIN or state)
begin
  case (state)
    A: DOUT <= DIN ? 1'b0 : 1'b1;
    B: DOUT <= DIN ? 1'b0 : 1'b1;
    C: DOUT <= DIN ? 1'b1 : 1'b0;
    D: DOUT <= DIN ? 1'b1 : 1'b0;
    E: DOUT <= DIN ? 1'b0 : 1'b1;
    F: DOUT <= DIN ? 1'b0 : 1'b1;
    G: DOUT <= DIN ? 1'b1 : 1'b0;
    H: DOUT <= 1'b1;
  endcase
end
endmodule

Notice that DOUT is equal to the inverse of the input at states 0, 1, 4, and 5, while it is equal to the input at states 2, 3, and 6. For state 7, DOUT equals 1.

8.(5-3)*

1000, 0100, 1010, 1101, 0110, 1011, 1101, 1110

9.(5-6)*

<table>
<thead>
<tr>
<th>Shifts:</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0110</td>
<td>1011</td>
<td>0101</td>
<td>0010</td>
<td>1001</td>
</tr>
<tr>
<td>B</td>
<td>0011</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>C</td>
<td>00</td>
<td>00</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Problem Solutions – Homework 4

10. (5-7)

The SO output to register B must be inverted and the initial carry must be set to 1, this forms the 2’s complement of the contents of register B.
If A < B, the final value left in the carry flip-flop will be 0.
If A < B, then the result will be in 2’s complement form.

11. (5-8)*

Replace each AND gate in Figure 5-6 with an AND gate with one additional input and connect this input to the following:

\[ S_1 + \overline{S_0} \]

This will force the outputs of all the AND gates to zero, and, on the next clock edge, the register will be cleared if S1 is 0 and S0 is logic one.

Also, replace each direct shift input with this equation: \( S_1S_0 \) This will stop the shift operation from interfering with the load parallel data operation.

12. (5-10)*

a) 1000, 0100, 0010, 0001, 1000
b) # States = n

13. (5-13)

Max Delay = 16(2 ns) = 32 ns
\[ f = \frac{1}{T} = \frac{1}{32 \text{ ns}} = 31.14 \text{ MHz} \]

14. (5-14)

a) 4
b) 6

15. (5-19)
16. (5-21)*

17. (5-22)

18. (5-25)

19. (6-1)*

20. (6-2)

21. (6-3)*
22.(6-4)*

Number of bits in array = \(2^{15} \times 2^3 = 2^{18} = 2^9 \times 2^9\)
Row Decoder size = \(2^9\)
Column Decoder size = \(2^{15} - 2^9 = 2^6\)
a) Row Decoder = 9 to 512, AND gates = \(2^9 = 512\)
Column Decoder = 6 to 64, AND gates = \(2^6 = 64\)
Total AND gates required = 512 + 64 = 576

b) \((21000)_{10} = (0101 0010 0000 1000)_2\), Row = 328, Column = 8

23.(6-7)

With 4-bit data, the RAM cell array is a square of \(2^{13} \times 2^{13}\).
Hence number of address pins is \(2 \times 13 = 26\).

24.(6-9)*

a) 32  b) 20, 15  c) 5, 5–to–32

25.(6-10)

Replace the 64k x 8 RAMs in figure 6-13 with the following basic cell to realize a 256k x 32 RAM.

26.+ (a) (5) Draw the circuit diagram corresponding to the given Verilog description.

```verilog
module prob3_5a (X, Y);
  input [3:0] X;
  output [1:0] Y;
  wire [3:0] N;

  not N1 (N[0], X[3]),
    N2 (Y[0], N[1]);
  and A1 (N[1], N[0], X[2]),
    A2 (N[2], X[1], X[0]);
  or O1 (Y[1], N[0], N[2]);
endmodule
```

(b) (6) Draw the state diagram corre-
sponding to the given Verilog description.

```verilog
module prob3_5b (X, RESET, CLK, Z);
input CLK, RESET, X;
output Z;
reg[1:0] state, next_state;
reg Z;
always@(posedge CLK or posedge RESET)
begin
  if (RESET)
    state <= 2'b00;
  else
    state <= next_state;
end
always@(X or state)
begin
  //Ignore the next statement.
  next_state <= 2'bxx; Z = 1'bx;
  case (state)
    2'b00: begin
      Z = 1'b0;
      if (X == 1)
        next_state <= 2'b01;
      else
        next_state <= 2'b00;
      end
    2'b01: begin
      Z = 1'b0;
      if (X == 1)
        next_state <= 2'b01;
      else
        next_state <= 2'b10;
      end
    2'b10: begin
      if (X == 1)
        begin
          next_state <= 2'b00;
          Z = 1'b1;
        end
      else
        begin
          next_state <= 2'b10;
          Z = 1'b0;
        end
    endcase
  end
endmodule
```

State Diagram:

*Also, for states 00 and 01, the output can be shown on the state and omitted on the arcs.*
(c) (7) Complete the following Verilog dataflow description which is to compute:
   1) the majority function $M$ of $A$, $B$, and $C$, and
   2) a selection function $Y$ between $A$ and $B$ with $C$ as the selection input.

   The majority function is 1 if two or more of $A$, $B$, and $C$ have value 1; otherwise, it is 0.
   The select function is to form $Y = A$ if $C = 0$ and $Y = B$ if $C = 1$.

   module prob3-5c (A, B, C, M, Y);
      input A, B, C;
      output M, Y;
      assign M = A & B | A & C | B & C
      assign Y = C ? B : A // also, ~C & A | C & B
   endmodule

27.+

(a) The block diagram of a bidirectional shift register is given below. This register is controlled
   by a 2-bit code (SL, SR) with the operations performed listed in the table below.

   Manually simulate the register for 8 clock cycles with the initial contents and inputs as given in
   the simulation table below. The results due to any given line of the table are to appear on the
   following line due to the positive clock edge triggering of the register.

<table>
<thead>
<tr>
<th>Clock Cycle</th>
<th>SL</th>
<th>SR</th>
<th>SL_In</th>
<th>SR_In</th>
<th>Shift Register Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1 0 0 1 0 1 1 0</td>
</tr>
</tbody>
</table>
Equation: \( D = \overline{SL} \overline{SHR(i - 1)} + SR \overline{SHR(i + 1)} + SL \overline{SR} \overline{SHR(i)} \)
29.+

a) Three positive edge-triggered flip-flops with asynchronous clear are shown. Connect the flip-flops shown to form a **ripple down counter** (counts 0, 7, 6, 5, 3, 2, 1, 0, ...). No added logic may be used.

b) (6) Three positive edge-triggered flip-flops with asynchronous clear are shown. Connect the flip-flops and additional logic to form a **synchronous binary down counter with parallel**
gating (counts 0, 7, 6, 5, 3, 2, 1, 0, ...). Use only the logic gate given.

30.

A Verilog description of a circuit that is to shift right and count up in binary is given. Add the statements in the boxes that implement the shift right (use a concatenate operation) and binary count up.

```verilog
module shift_count (CK, RESET, CU, SR, SR_IN, SC_OUT)
input CK; // The clock
input CU; // CU = 1 causes count up to occur
input SR; // SR = 1 causes a right shift to occur
input SR_IN; // Input to the most significant bit for right shift
output[3:0] SC_OUT // The four bit output from the circuits flip-flops
reg [3:0]SC_OUT;
always@ (posedge CK or posedge RESET)
begin
    if (RESET == 1)
        SC_OUT <= 4'b0;
    else if (SR == 1)
        SC_OUT <= {SR_IN, SC_OUT[3:1]}
    else if (CU == 1)
        SC_OUT <= SC_OUT + 1
    else
        SC_OUT <= SCOUT;
end
endmodule
```

31.

a) A memory is made up of 32 64M X 8 RAM chips. The memory has 32-bit words.

1) How many words are there in the memory? 512M or $2^{29}$
2) How many address bits are there for the memory assuming addressing is to words? 29

3) How many address bits are there for a RAM chip? 26

4) What is the size of the decoder required to decode address bits into chip select signals? 3-to-8

b) (4)

1) Explain the difference between the storage method used in a static RAM and a dynamic RAM:

**SRAM** uses a latch for storage.

**DRAM** uses a capacitor for storage.

2) Circle the RAM type(s) that requires a refresh operation: Static Dynamic

3) Typically, dynamic RAM is (circle one): Volatile Non-Volatile

c) (6) Connect together the inputs and 3-state buffers given below to implement the following function, assuming that A, C, and E are such that A C = C E = A E = 0 and A + C + E = 1.

\[ G = A B + C D + E F \]