use the basic element (at most two input for each) to make a 4:16 encoder. What is the least number of elements you need to use in order to make this encoder?
In Verilog, implement an adder that adds two sign-magnitude 3-bit binary numbers. If there is overflow, the adder outputs the last non-overflowing sum it gave (i.e. remembers old values and can output the most recent one in case of overflow). A warning variable also gets outputted, and is true in the case of overflow.
HW #8

The following Verilog code has 2 problems. Please find them and write down how to fix them.

module choose(in, out);
  input logic [1:0] in;
  output logic [1:0] out;

  always_comb begin
    case (in):
      2'b00: out = in;
      2'b01: out = ~in;
      2'b10: out = 2'b00;
      2'b11: out = 2'b11;
    endcase
  end
endmodule

Problem 1:

How to fix:

Problem 2:

How to fix:
Assume $T_{\text{setup}}$ AND $T_{\text{hold}}$ ARE 0.1 NS

All gate delays ARE 1.0 NS

Clock period: 100 NS

"In" is external input

For each circuit, determine whether OUT IS STABLE. If it is unstable, modify the circuit to make it stable.

For all possible inputs, there is the same output.
DESIGN A COUNTER WITH FOLLOWING SEQUENCE
\[1 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 0 \rightarrow 4 \rightarrow 2 \rightarrow 6] \rightarrow 1 \rightarrow 3 \ldots.
DRAW THE CIRCUIT DIAGRAM. DDFS AND GATES ARE AVAILABLE.
HW 8A

Find the smallest clock period:
$T_{\text{delay}} = T_{\text{prop}} + C_{\text{inv}} + Q = 0.5\text{ns}$
Please build a 4:16 decoder using standard gates (no just draw a decoder)
Design a circuit that can tell if a 3-bit, unsigned number is prime.
perform the following operations in sign-magnitude & then in 2's complement.

a. 1010001
   + 1000010

b. 0111001
   - 0010111
   = 0010111
module mystery (a, b, c, d);
    input logic[3:0] a, b;
    output logic c;
    output logic[3:0] d;
    logic[3:0] e;
    integer i;
    always_comb begin
        c = 0;
        d = 4'b0000;
        e = 4'b0000;
        for (i=0; i<4; i++) begin
            if (a[i] | b[i])
                e[i] = 1;
            if (a[i] ^ b[i])
                d[i] = 1;
        end
        c = (e == 4'b1111);
    end
endmodule
In review problem 53 we found out how to increase the number of bits a RAM system had by combining 8x2 RAM units to make an 8x6 RAM system. How would you now increase the number of addresses in a RAM system? Describe how by creating an 8x2 RAM system using 4x2 RAM units. You may use simple gates (ANDs, ORs, NANDs, NORs, XORs, XNORs) and basic circuit elements (encoders, decoders, muxes).

<table>
<thead>
<tr>
<th>4x2 RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
</tr>
<tr>
<td>A0</td>
</tr>
<tr>
<td>Write</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Build a 1 bit counter to do the following:
Can use only basic gates, DFFs, and muxes.

<table>
<thead>
<tr>
<th>$C_i$</th>
<th>$C_o$</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Reset (output = 0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Parallel load</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Multiply the current value by 2. (if value too large to hold, output: value - 1)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Divide the current value by 2, rounding down if the current value is odd</td>
</tr>
</tbody>
</table>
Derive a state table for the circuit below and write the corresponding Verilog code.
SIMPLIFY: $ABC(A+D) + BD(AC) = F$
DRAW THE DIAGRAM FOR THE FOLLOWING CIRCUIT:

BUILD A FSM FOR HIGHWAY INTERSECTION. THE FSM HAS A INPUT FROM THE EMERGENCY SERVICES. IF THE SIGNAL FROM THE EMERGENCY DEPARTMENT IS TRUE, THE SIGNAL SHOULD SHOW GREEN(GO) FOR THE SIDE OF THE ROAD ON WHICH EMERGENCY VEHICLE IS PASSING BY, AND IT SHOULD TURN ALL OTHER SIGNALS TO RED. IF THE SIGNAL FROM EMERGENCY DEPARTMENT IS FALSE THE FSM SHOULD OPERATE NORMALLY.
Using the following equation
\[ F = \overline{ABC} + (\overline{A+C}) + \overline{BC} \]
to implement these technologies.

a) 8:1 MUX
b) 3:8 Decoder
c) 4:1 MUX