# ECSE 281 HW

Tyler Yoak

1. Implement the following function using only 74x138 Binary decoders & NAND Gates

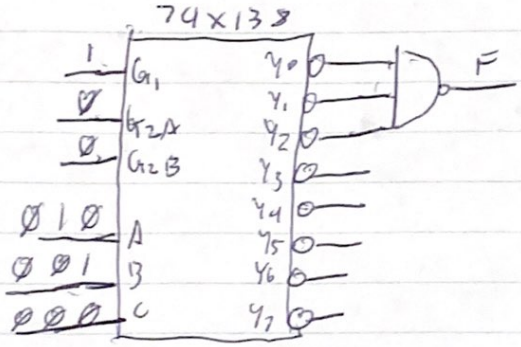
$$F = \prod_{ABC}(3,4,5,6,7)$$

$$F = \sum_{W,X,Y,Z}(2,3,4,5,8,10,12,14)$$

74x138 Binary Decoders → Active Low & NAND gates

a. [redacted]

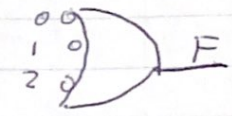
Maxterms: offsets → Product of Sums [redacted]

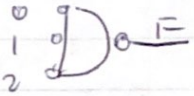



-1: What are the inputs?

$$F = \prod_{ABC}(3,4,5,6,7)$$

$$= F = \sum_{ABC}(1,2,7)$$

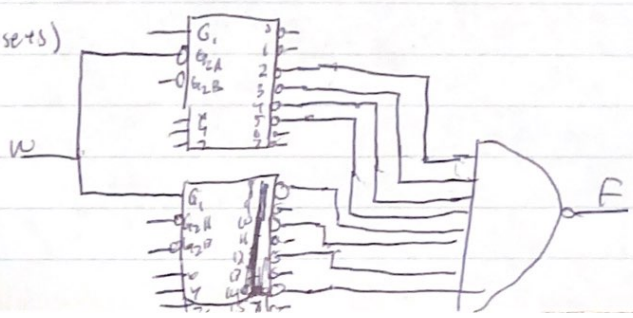



Active Low NAND F=1 AND F=0

b. $F = \sum_{W,X,Y,Z}(2,3,4,5,8,10,12,14)$

Sum of products = Minterms (Onsets)




-1: What are the inputs connected to?

Tyler
Youk

2. Design a 10 to 4 encoder w/ the inputs 1 on at 10 code & outputs coded ~~[illegible]~~ normally for 0-7 [binary 0000 - 0111]
& 8 coded as E [1110]
& 9 coded as F [1111]

Show the internal circuit

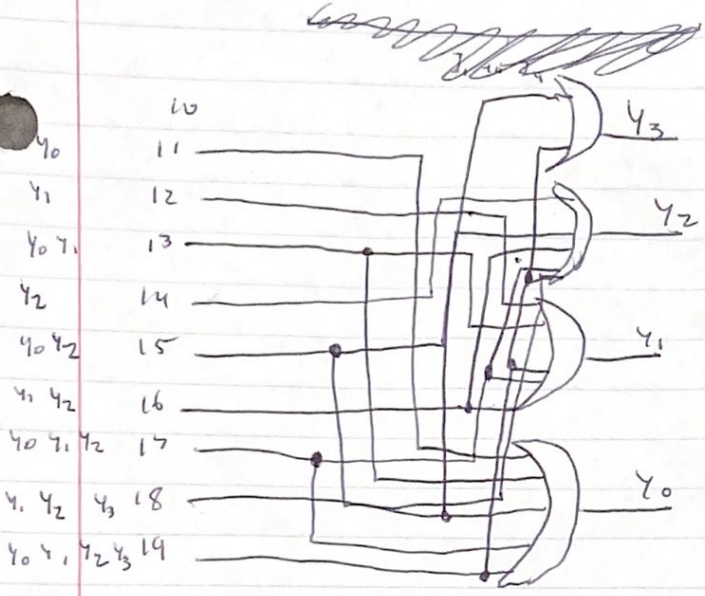
10 to 4 encoder

$Y_3$: $I_8 + I_9$

$Y_2$: $I_4 + I_5 + I_6 + I_7 + I_8 + I_9$

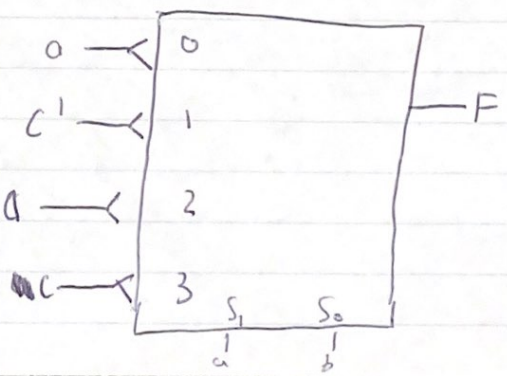
$Y_1$: $I_2 + I_3 + I_6 + I_7 + I_8 + I_9$

$Y_0$: $I_1 + I_3 + I_5 + I_7 + I_9$




3. Implement the following only using a single 4x1 multiplexer & inverters

| a | b | c | F | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | > a |
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 1 | > c' |
| 0 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 1 | > a |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 0 | > c |
| 1 | 1 | 1 | 1 | |

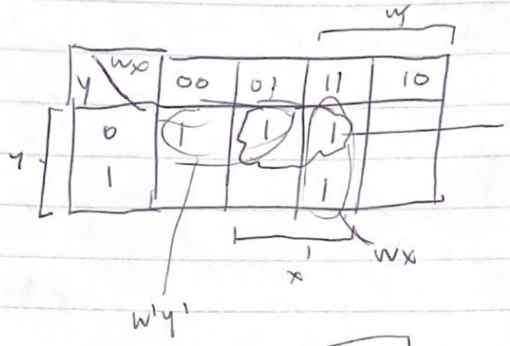


Tyler Rouk

9. For the logic expression below, find all of the static hazards and design a hazard-free circuit that realizes the same logic function.

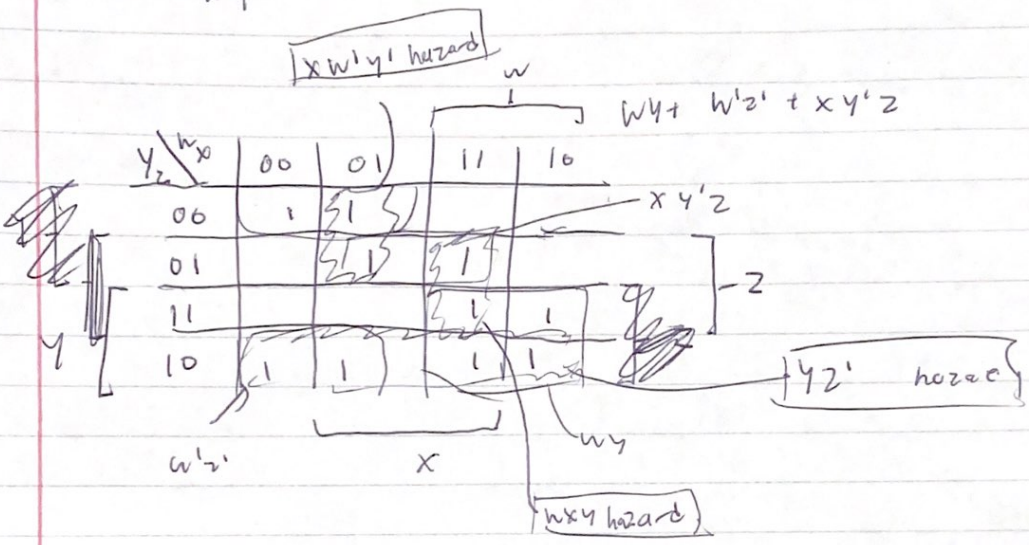
$F = w \cdot x + w' \cdot y'$

$F = w \cdot y + w' \cdot z' + x \cdot y' \cdot z$




Static hazard
$x \cdot y'$

$F = w'y' + wx + xy'$

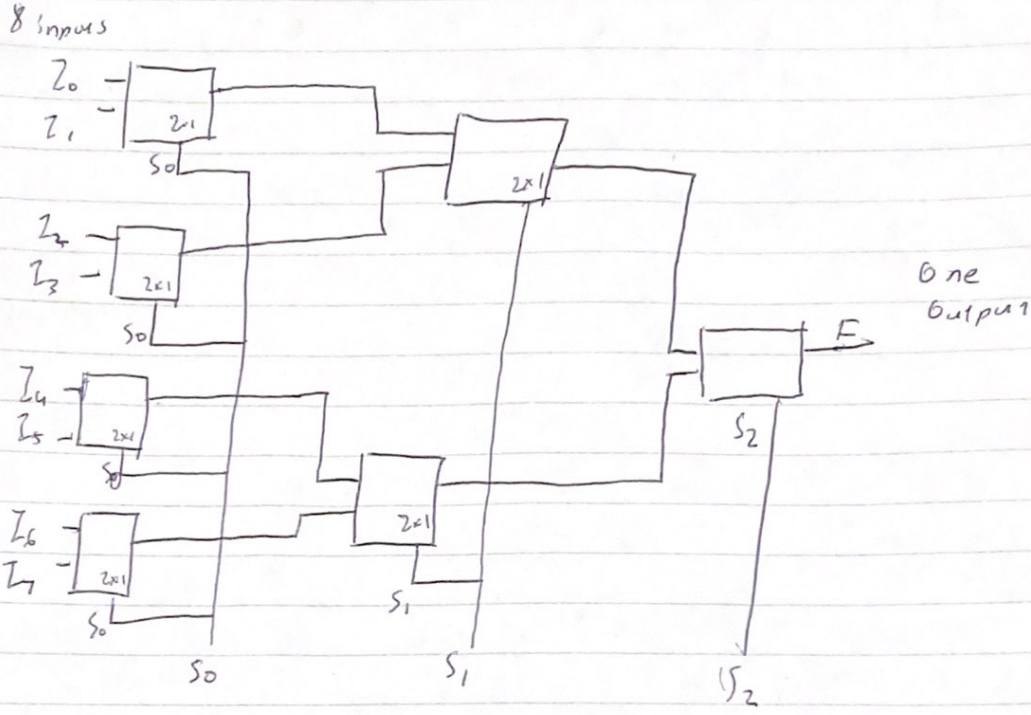



$F = \underbrace{wy + w'z' + xy'z}_{\text{original}} + \underbrace{xw'y' + yz' + wxy}_{\text{removes}}$

Tyler York

5. Design an 8x1 multiplexer (8 data sources / 1 bit of data from each source)

using 2x1 multiplexers only. You can use as many 2x1 multiplexers as needed. Clearly label all inputs and outputs.



```
// Tyler Youk Source Code
// HW Assignment #8

//cla_adder_homework5.sv source code
module cla_adder #(
    parameter N = 8
) (
    input logic [N-1:0] a, b,
    input logic c_in,
    output logic [N-1:0] s,
    output logic c_out

);

    logic [N-1:0] p, g;
    logic [N:0]   c;

    assign p = a ^ b;
    assign g = a & b;

    for (genvar i = 0; i <= N; i++) begin
        if (i == 0)
            assign c[i] = c_in;
        else
            assign c[i] = g[i-1] | p[i-1] & c[i-1];
    end

    assign s = c^p; //COMPLETE
    assign c_out = c[N]; //COMPLETE
endmodule




//Testbench_homework5.sv source code
`timescale 1ns/10ps
module testbench ();

    logic [1:0] a2, b2, s2;
    logic [7:0] a8, b8, s8;
    logic       co2, co8;


//2 bit cla adder
```

```
    cla_adder #(
        .N(2)
    ) UUT2 (
            .a(a2),
            .b(b2),
            .c_in(1'b0),
            .s(s2),
            .c_out(co2)
    );

//8 bit cla adder
    cla_adder #(
        .N(8)
    ) UUT8 (
        .a(a8),
            .b(b8),
            .c_in(1'b0),
            .s(s8),
            .c_out(co8)
    );

    initial begin
        a2 = 0;
        forever
            #10 a2++;
    end

    initial begin
        b2 = 0;
            forever
                    #40 b2++;
    end

// COMPLETE for a8 that will increase by 3 every 10 time units
            initial begin

                    a8 = 0;
                    forever
                            #10 a8+=3;
            end

// COMPLETE for b8 that will increase by 5 every 10 time units
            initial begin
```

```
            b8 = 0;
            forever
                  #10 b8+=5;
      end



   initial begin

      #320 $finish();
end

endmodule
```

//Waveform screenshot with Tyler Youk tag