I have provided my verilog answers as files.

CPE 322 **EXAM II** **Name** Nolan Anderson

Spring Semester 2021

Work should be performed systematically and neatly with the final answer being underlined. This exam is open book, open notes, closed neighbor/device/browser. Allowable items on desk include: exam, pencils, and pens. All other items must be removed from student's desk. Students have Approximately 90 minutes (1 1/2 hours) to complete this exam. Best wishes!

1. [20 points] Use Shannon's expansion theorem around a and b for the following function

   Y(a,b,c,d,e,f) = ((a ^ b) & c & ~d & f) + (~a & ~b & ~c & d & e & f) +
   (a & b & (c | e | ~f))

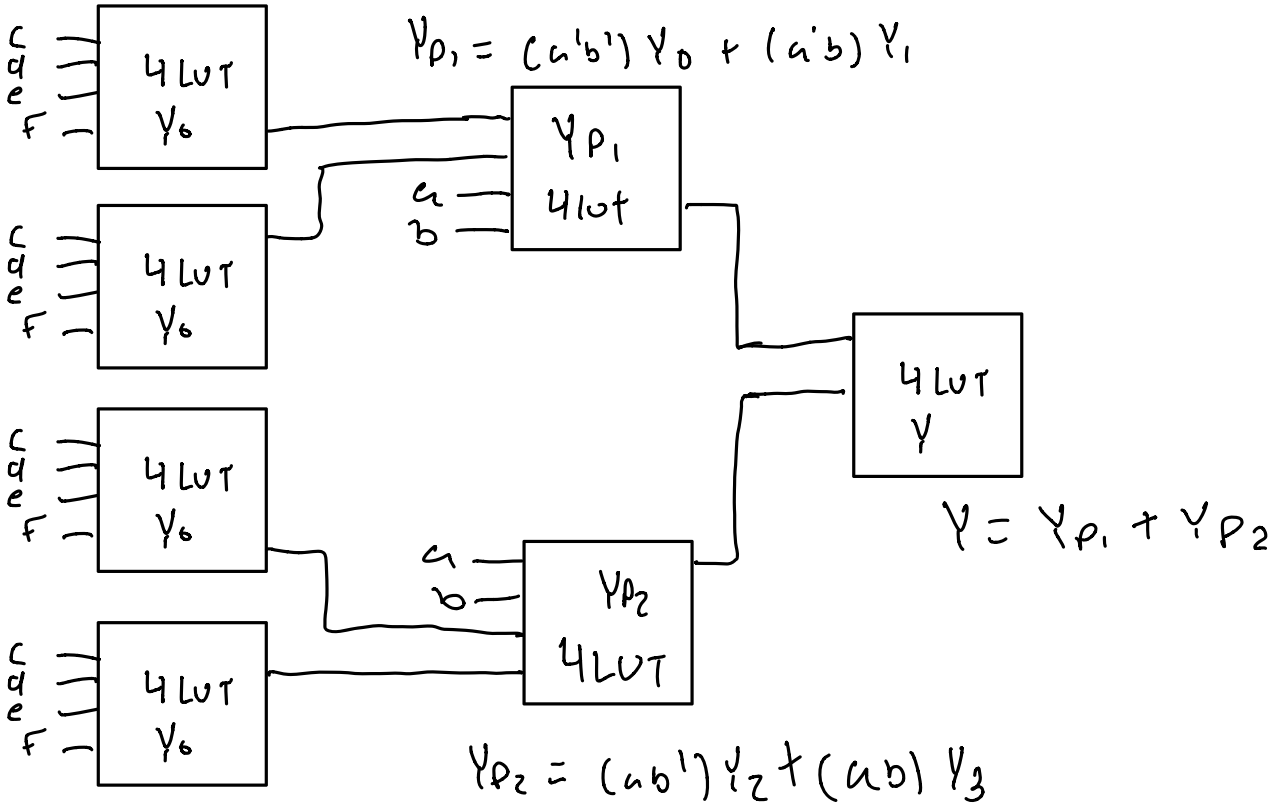
   so that it can be implemented using only four-variable function generators (4-input LUTs). Draw a block diagram to indicate how Y can be implemented using only four-variable function generators. Indicate the function realized by each four-variable function generator.

   Note the operators in the above are written in Verilog syntax; ~ inversion, & bitwise-AND, | bitwise-OR, ^ bitwise-XOR

$Y(a,b,c,d,e,f)$

$Y_0 = Y(a'b') = c'def$ $Y_1 = Y(a'b) = cd'f$

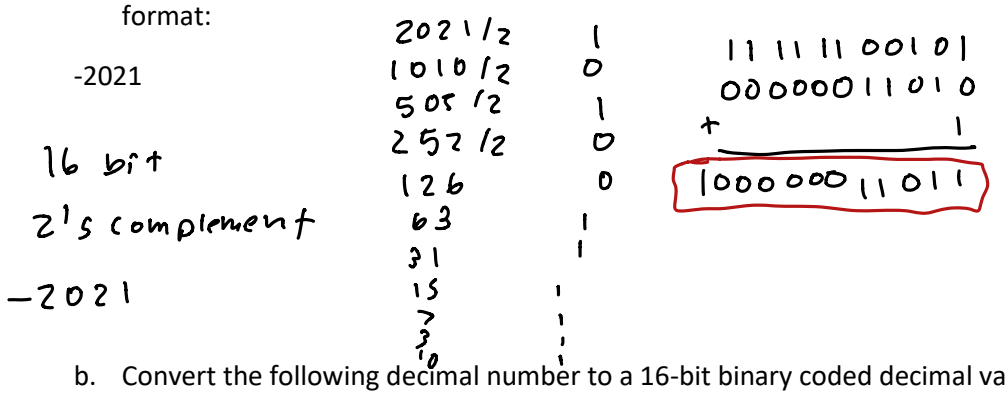
$Y_2 = Y(ab') = cd'f$ $Y_3 = Y(ab) = (c+e+f')$



2. [20 points] Number Conversions

   a. Convert the following signed decimal number to a 16-bit 2's-complement signed binary format:

   -2021



   b. Convert the following decimal number to a 16-bit binary coded decimal value:

   322

   c. Convert the following decimal number into a 32-bit unsigned hexadecimal value:

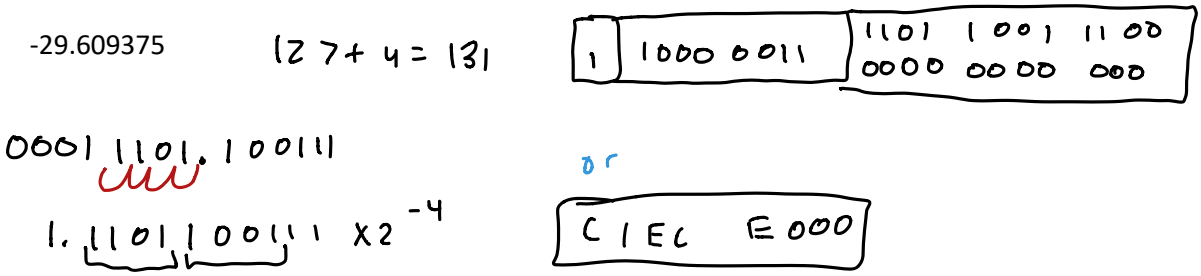
   12,648,430

   d. Convert the following decimal number into 32-bit short precision floating point, using 1 bit for sign, 8 bits for the exponent with a bias of 127, and with 23 bits for the mantissa:

$$\text{value} = (-1)^{\text{sign}} \times 2^{(E-127)} \times \left(1 + \sum_{i=1}^{23} b_{23-i} 2^{-i}\right)$$

   expressed as the contatenated value {sign, E, b}

   -29.609375

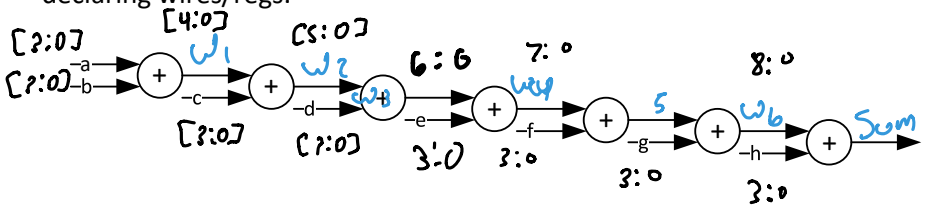


3. [20 points] As discussed in class, the combination of linear logical functions can be performed either sequentially or half-at-a-time. Either method can get to the same solution, but often times the method that performs half of the operations at a time is the best choice because it reduces the number of levels of logic and the number of routing paths.

   Take the sum operation:

   sum[6:0] = a[3:0] + b[3:0] + c[3:0] + d[3:0] + e[3:0] + f[3:0] + g[3:0] + h[3:0]

   Write Verilog code that will generate a sequential set of adders, noting that when 2 numbers are added together, the range expands by a bit. 3-4 numbers grow by 2 bits, and 5-8 numbers make it grow by 3 bits, etc. Be sure to indicate the width of intermediate signals in your code when declaring wires/regs.

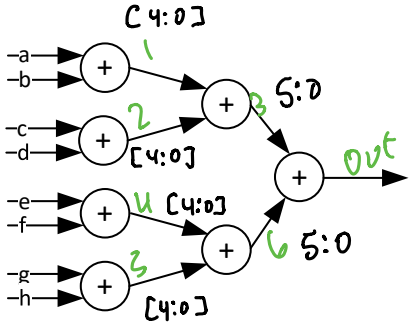



```
module sum_seq (input [3:0] a,b,c,d,e,f,g,h, output [6:0] sum);
```

Sum - seq . v

```
endmodule
```

   Write Verilog code that twill generate a half-at-a-time set of adders, noting that every stage the intermediate sums will increase by one bit. Be sure to indicate the width of intermediate signals in your code when declaring wires/regs.



(Problem 3, Continued)

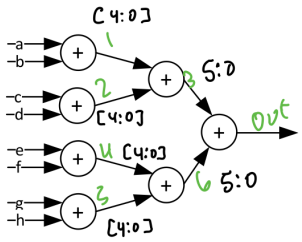
```
module sum_log (input [3:0] a,b,c,d,e,f,g,h, output [6:0] sum);
```

Sum - log. v

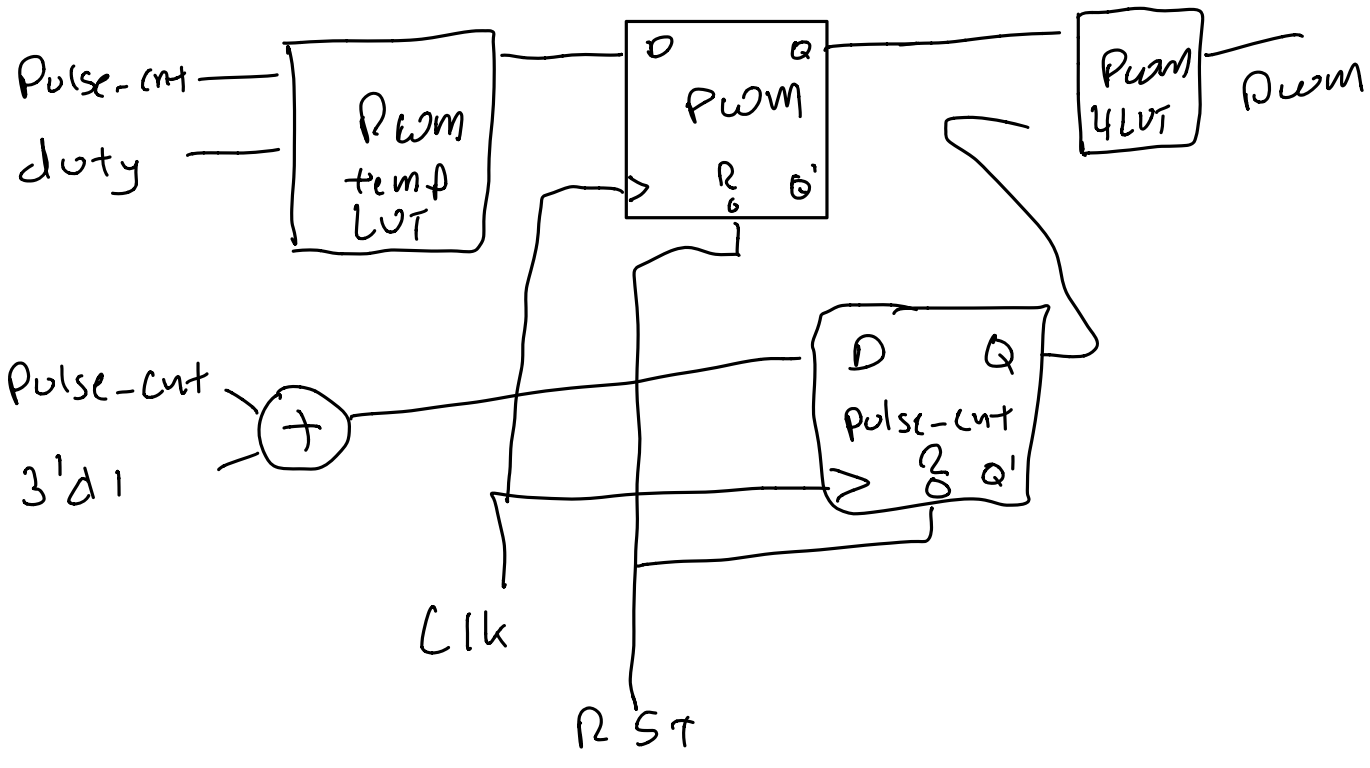
```
endmodule
```

4. [20 points] Read the following Verilog code module, and answer the following questions from the perspective of an FPGA synthesis tool:

```
module pulse_width_mod (input clk, rst, [2:0] duty, output reg pwm);
  reg [2:0] pulse_cnt;
  always @(clk) begin
    if (rst) begin
      pwm <= 1'b0;
      pulse_cnt <= 3'b000;
    end else begin
      pulse_cnt <= pulse_cnt + 3'd1;
      if (pulse_cnt[2:0] > duty[2:0])
        pwm <= 1'b0;
      else
        pwm <= 1'b1;
    end
  end
endmodule
```



a. Assume that all flip-flops are DQ flip-flops with active-high synchronous reset inputs R. Also assume that the (+) adder symbol shown in Problem 3 can be used in your block diagram. Draw a small block diagram of the circuit, including registers, input and output signals.



(Problem 4, continued)

b. Based on your circuit diagram from a, how many DQ Flip-Flops are required to create this circuit?

2 DQ flip flops, one for temp pulse_cnt and output pwm

c. Based on your circuit diagram from a, and based on your own intuition and reasoning, determine how many 4-input look-up-tables (LUTs) are required to realize this circuit. DO NOT SIMPLY PUT A NUMBER, but rather please explain how many LUTs are required to realize all of the combinatorial paths in the block diagram from part a.

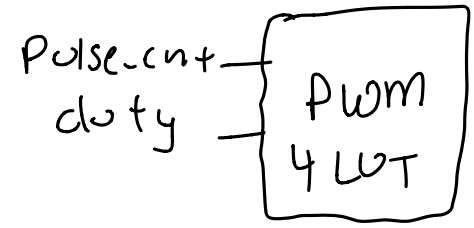
1 lut for comparison of pulse_cnt and duty where output is pwm

1 lut for addition, pulse_cnt + 3'd1.

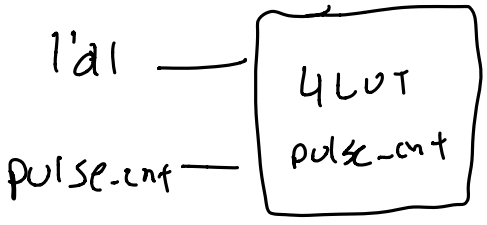
1 lut for final output, pwm

3 total probably don't need last lut

d. Based on your answer for part c, briefly describe the function handled by each 4-input LUT. This can be described either by a logical equation/formula, or by a brief description of the operation(s) each LUT handles. NOTE: THIS MUST BE COMPLETED ON A LUT-BY-LUT BASIS FOR EACH LUT IDENTIFIED IN PART C.



pwm = pulse_cnt > duty



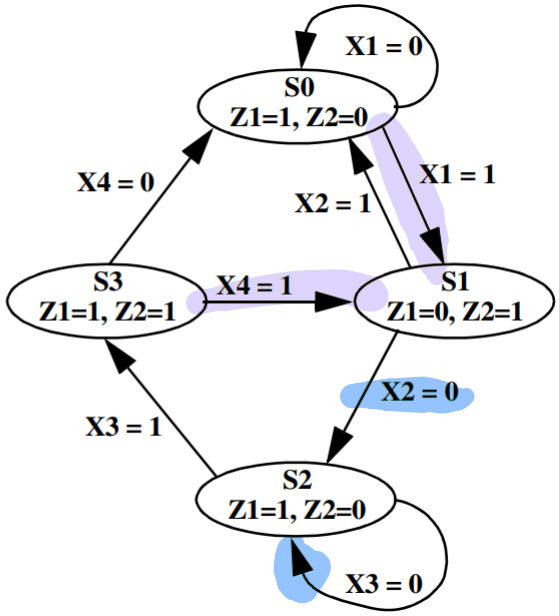
pulse_cnt = pulse_cnt + 3'd1

Not sure if I need the 3rd, honestly. uses a bit lost honestly.

5. [20 points] For the given state graph:

Inputs:
X1, X2, X3, X4

Outputs:
Z1, Z2



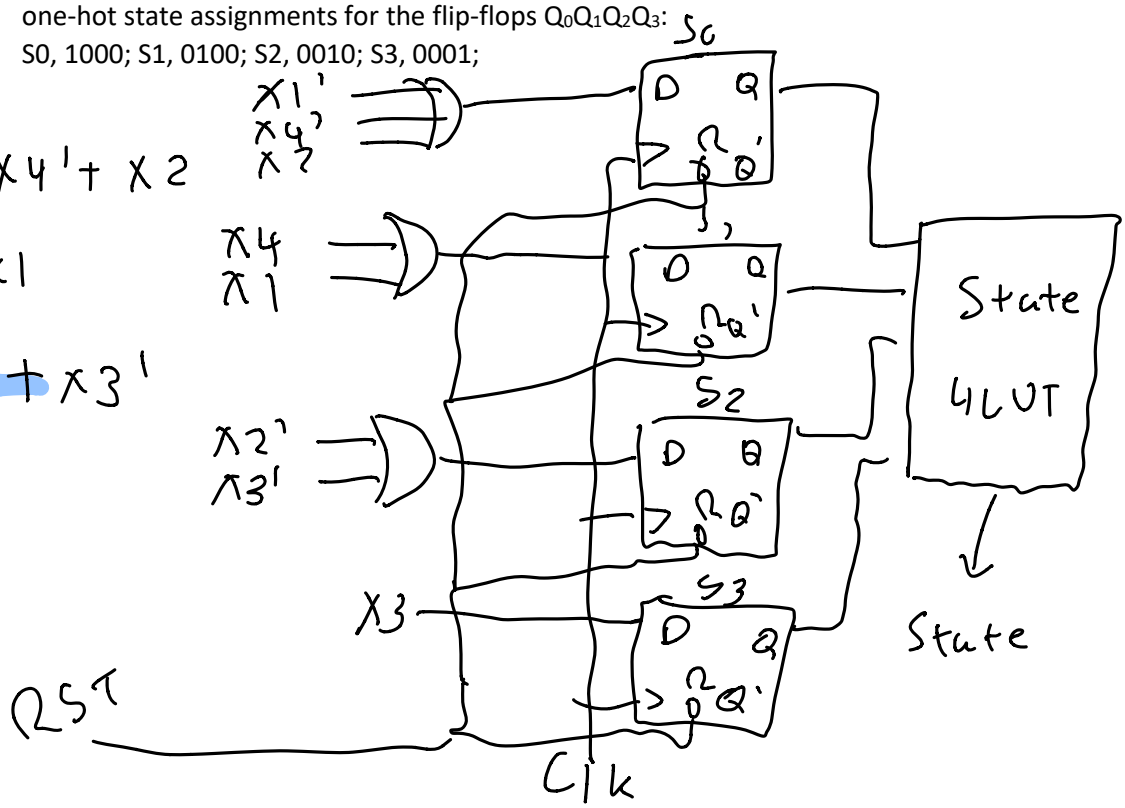
a. Derive the simplified next-state and output equations by inspection. Use the following one-hot state assignments for the flip-flops $Q_0Q_1Q_2Q_3$:
S0, 1000; S1, 0100; S2, 0010; S3, 0001;

S0 = X1' + X4' + X2

S1 = X4 + X1

S2 = X2' + X3'

S3 = X3



b. Provide Verilog code to implement the above state graph. Note that the Z1 and Z2 outputs, which depend only on the 4-bit State register, must NOT be delayed by a clock cycle with respect to the current value of State. An additional sheet of paper is provided for your convenience:

```
module prob5_moore (input clk, rst, X1, X2, X3, X4, output Z1, Z2);
reg [3:0] State;
```

prob5-moore.v

```
endmodule
```