Problem 1 (10 points)

i.

ii. (2 points) Select which of the following always outputs 0

a. x^1
b. x^x
c. ~(x^x)
d. x^0

iii.

Problem 2 (10 points)

What are the minimum required number of AND, OR gates required to implement F as a sum-of-products. Assume you have inputs and their complements available as inputs to your circuit.

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

a) One 2-input gate
b) One 3-input gate
c) Two 2-input gates
d) Two 3-input gates
e) One 2-input gate and one 3-input gate
f) Three 2-input gates
g) Three 3-input gates
h) Two 2-input gates and one 3-input gate
i) None of the above

Problem 3 (10 points)

Let's use the subscript to denote the base number system. That is: $(1011)_2$ implies binary representation, while $(B)_{16}$ implies the same number in hexadecimal (base-16).

i. (3 points) Select all the options which are strictly smaller than $(136)_{10}$

   a. $(1000\ 0100)_2$
   b. $(89)_{16}$
   c. $(88)_{16}$
   d. $(1000\ 1000)_2$
   e. None of the above

ii. (5 Points) If all these values values are encoded as two's complement, select the true statement:

   a. C7 + BB = $(7E)_{16}$
   b. C7 + BB = $(126)_{10}$
   c. C7 + BB = $(82)_{16}$
   d. None of the above

iii. (2 points) If these hexadecimal numbers represent two's complement encoded values, circle the largest value

   a. C4
   b. C6
   c. E8
   d. 70

Problem 4 (10 points)

(6 points) Each of the numbers below is represented in two's complement, each variable, w, x,y,z can be any hexadecimal value independently of each other.

i.

```
  (B3X1)16
+ (47WZ)16
________
```

ii.

```
  (7YCD)16
+ (0110)16
________
```

a) i. and ii. both can result in overflow

b) i. can result in overflow, but not ii.

c) i. cannot result in overflow but ii. can

d) neither i. nor ii. can possibly result in an overflow

(4 points) Consider integers X[15:0] and Y[15:0] that result in SUM[15:0] when added. If we know the values of X[15:14] and Y[15:14] as well as SUM[15] we can always determine if there's an overflow:

a. For signed integers only, not for unsigned integers

b. For both signed and unsigned integers

c. For unsigned integers only not for signed integers

d. We cannot **always** determine if there's been an overflow in either case

Problem 5 (10 points)

1. (3 points) If you had to make a 2:1 multiplexor (2 inputs, 1 select bit) using only NAND gates. How many NAND gates would you need?

   a. 3
   b. 4
   c. 5
   d. 6
   e. 7
   f. 8
   g. None of the above

2. (3 points) If you had to make a 2:1 Decoder (2 outputs, 1 select bit) using only NOR gates. How many NOR gates would you need?

   a. 3
   b. 4
   c. 5
   d. 6
   e. 7
   f. 8
   g. None of the above

3. (4 points) A Majority gate, with output f and inputs a, b, and c is given by **f = a&b | a&c | b&c**. If you could only use 2- and 3-input NAND gates, how many such gates would you need to implement this function?

   a. 3
   b. 4
   c. 5
   d. 6
   e. 7
   f. 8
   g. None of the above

Problem 6 (10 points)

i. (2 points) Consider a two-input XOR gate with two inputs and one output. Given the ability to set the inputs to constants (e.g., 1/0). Is it possible to construct a NAND gate using an infinite supply of such two-input XOR gates?

a. Yes

b. No

ii. (4 points) Given only NAND gates, what is the minimum number of NAND gates needed to implement a two-input XOR.

a. 3

b. 4

c. 5

d. 6

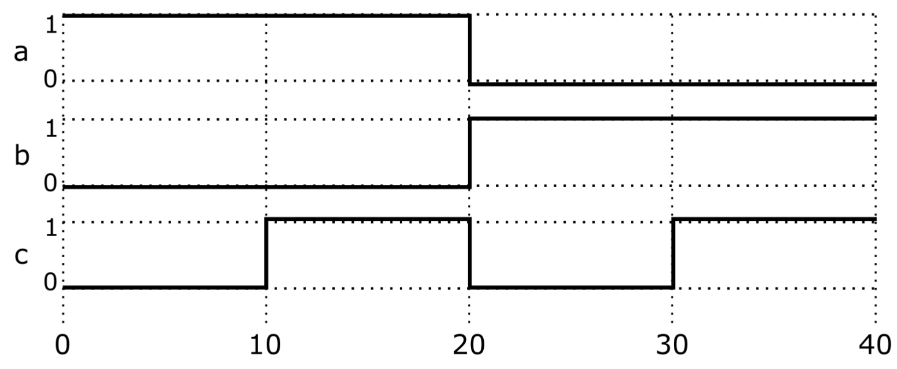
e. 7

f. 8

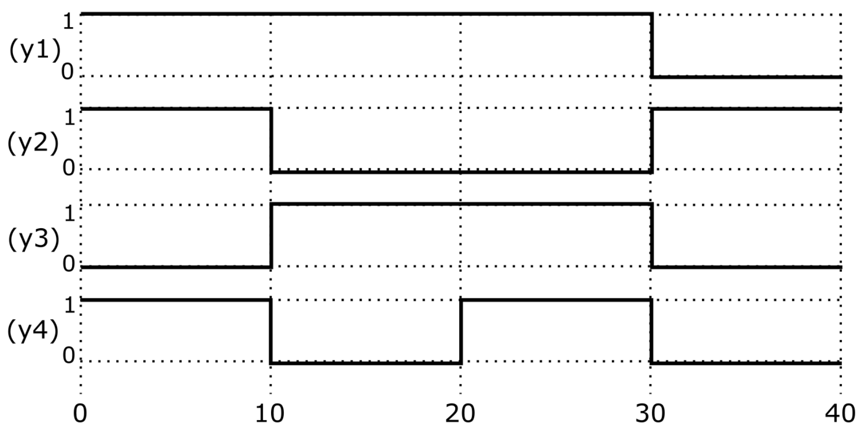
iii. (4 points) Below is the Verilog code for two modules, <u>thing</u> and <u>unknown</u>:

```
module thing (
   input    a,
   input    b,
   output   y
   );

   assign y = a & ~b | b & ~a;

endmodule

module unknown (
   input    a,
   input    b,
   input    c,
   output   y
   );

   wire w;

   thing thing1 (
      .a (a),
      .b (b),
      .y (w)
   );

   thing thing2 (
      .a (w),
      .b (c),
      .y (y)
   );

endmodule
```

The following waveforms are applied to the inputs of unknown:



Which of the following is the correct waveform for output y of module unknown?



a. y1
b. y2
c. y3
d. y4

Problem 7 (10 points)

i. (3 points) Given a Karnaugh map for a 4-input function, the maximum number of product terms that can ever remain after minimization are:

a. 6

b. 7

c. 8

d. 9

e. Impossible to determine without knowing the entries

ii. (3 points) Select all the correctly minimized expressions for the K-Map below

ab \ cd	00	01	11	10
00		1	1	1
01				
11				
10	1			1

a. (~a&~b&d) \| (a&~b&~d)\|(~a&~b&c)

b. (~a&~b&~d) \| (~a&~b&~d)\|(~b&c&~d)

c. (~a&~b&d) \| (a&~b&~d)\|(~b&c&~d)\|(~a&~b&c)

d. none of the above

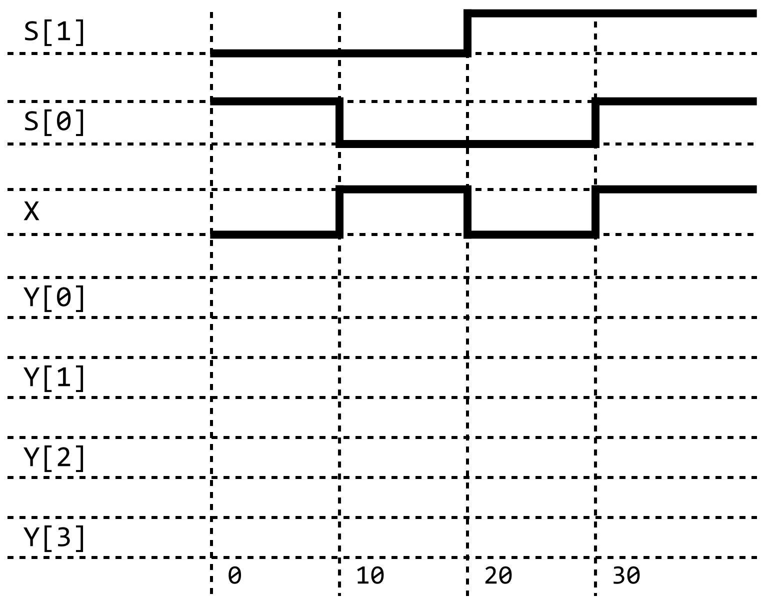
Problem 9 (10 points)

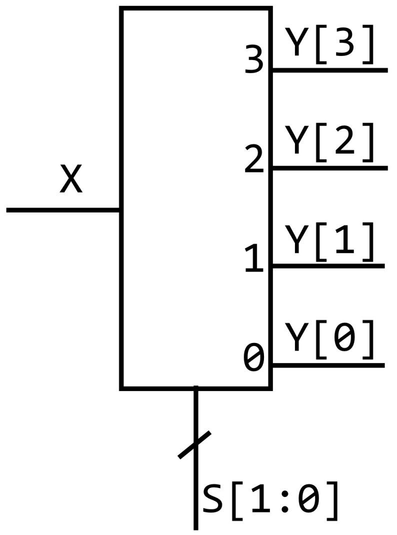
A De-Multiplexor or a DEMUX takes an input (X) and places it on one of 2^n outputs (Y[2^n-1:0]) based on an n-bit selection input (S[n-1:0]).

i. (4 points) Partial verilog code is provided to correctly describe a DEMUX.

```
module demux(
input X,
input S,
output [1:0] Y
);

…

endmodule
```

ii. The input to the DEMUX looks is provided below





Answer some questions about a testbench for the DEMUX.

iii. (2 points) Select all the true statements about the testbench for this

a. S should be registers
b. S should be wires
c. X should be a wire
d. Y should be registers

iv. (2 points) Consider a testbench that replicates the waveform above, after the “initial begin” --- you type

a. `S[1]=0; S[0]=0; X=0;`
b. `#10 S[1]=0; S[0]=1; X=0;`
c. `S[1]=0; S[0]=1; X=0;`
d. `#10 S[1]=0; S[0]=0; X=0;`
e. `S[1]=0; S[0]=0; Y=0;`
f. `#10 S[1]=0; S[0]=0; Y=0;`
g. `#10 S[1]=0; S[0]=1; Y=0;`
h. `S[1]=0; S[0]=1; Y=0;`