Wawrzynek & Weaver
Sp 2018

EECS 151/251A
Digital Design

MT 1

Print your name: ____________________________, ____________________________
(last) (first)

I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that any academic misconduct will be reported to the Center for Student Conduct, and may result in partial or complete loss of credit.

Sign your name: ____

You may consult two sheets of paper (double-sided) of notes. You may not consult other notes, textbooks, etc. Calculators, computers, and other electronic devices are not permitted.

Please write your answers in the spaces provided in the test. We will not grade anything on the back of an exam page.

You have 3 hours (minus 10 minutes) although we don't expect you to take more than half that time. There are 7 questions, of varying credit (105 points total). The questions are of varying difficulty, so avoid spending too long on any one question.

Do not turn this page until your instructor tells you to do so.

Question:	1	2	3	4	5	6	7	Total
Points:	10	16	10	10	25	10	24	105
Score:								

Problem 1 ***Design Tradeoffs*** **(10 points)**

For a given project you have a choice between an FPGA-based design and an ASIC design. For each question, state which would make a better solution and a 1-2 sentence why.

(a) (2 points) You want to be able to update the design?

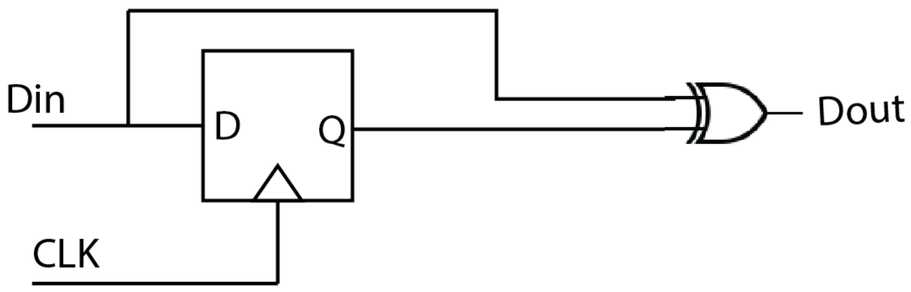
(b) (2 points) You want to minimize circuit board area?

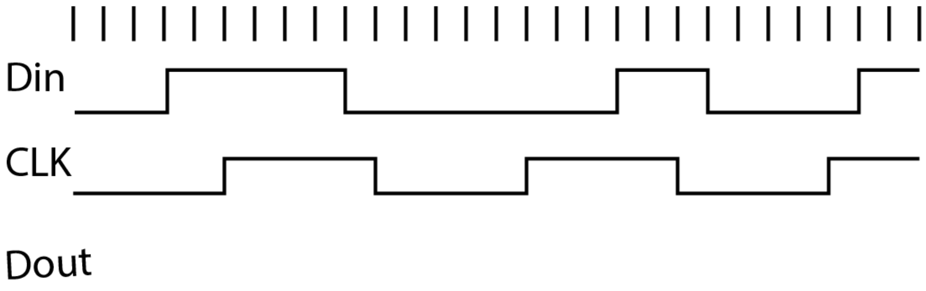
(c) (2 points) You want to minimize power?

(d) (4 points) Draw a rough diagram of total cost vs. volume for the both FPGA and ASIC designs below:

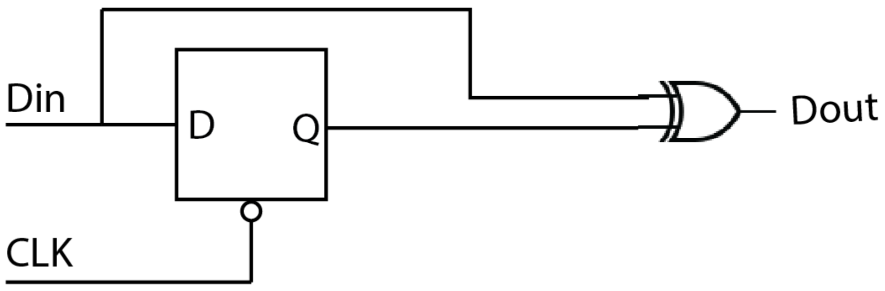
Problem 2 ***A matter of timing*** **(16 points)**

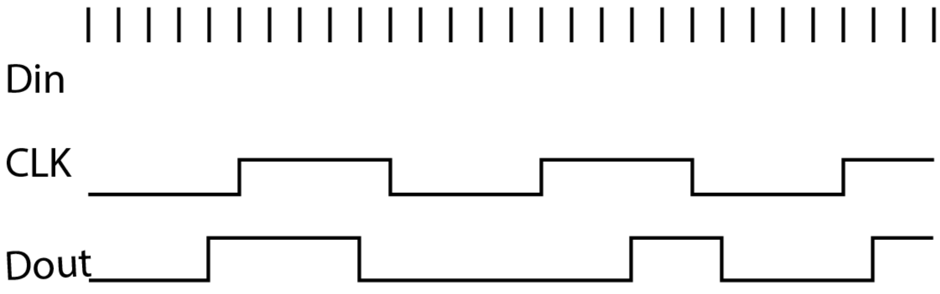
(a) (8 points) Consider the following circuit below. The clock period is 10ns and the propagation delay of the XOR is 2ns. For the flip-flop, the setup time is 1ns, the CLK→Q time is 1ns, and the hold time is 1ns. The flip-flop initially holds a 0. Draw the timing diagram for the output which matches the input.





(b) (8 points) **251A students only.** In the circuit below, assume the latch has no setup time, but it does have a 1ns propagation delay when it is transparent. Initially the latch holds a 0. Draw the timing diagram for the *input* which matches the output.





Problem 3 *Combinational Logic Design* (10 points)

Consider the function F defined by the truth-table below, where "-" indicates a don't care.

a	b	c	d	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	–
1	0	1	1	0
1	1	0	0	–
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

(a) (4 points) Find a minimized sum-of-products form (better answers have fewer literals).

(b) (4 points) Find a minimized products-of-sums form.

(c) (2 points) Find $\bar{F}$ in products-of-sums form.

a)

ab \ cd	00	01	11	10
00	1	0	0	1
01	1	1	0	0
11	–	0	0	0
10	1	0	0	–

$\bar{b}\bar{d} + \bar{a}b\bar{c}$

b)

ab \ cd	00	01	11	10
00	1	0	0	1
01	1	1	0	0
11	–	0	0	0
10	1	0	0	–

$(b + \bar{d})(\bar{a} + \bar{b})(\bar{b} + \bar{c}$

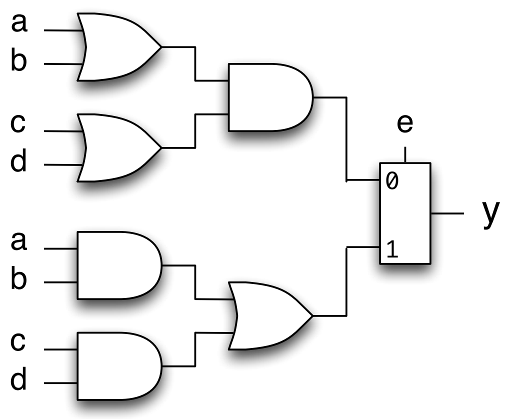
c) Use dual property:

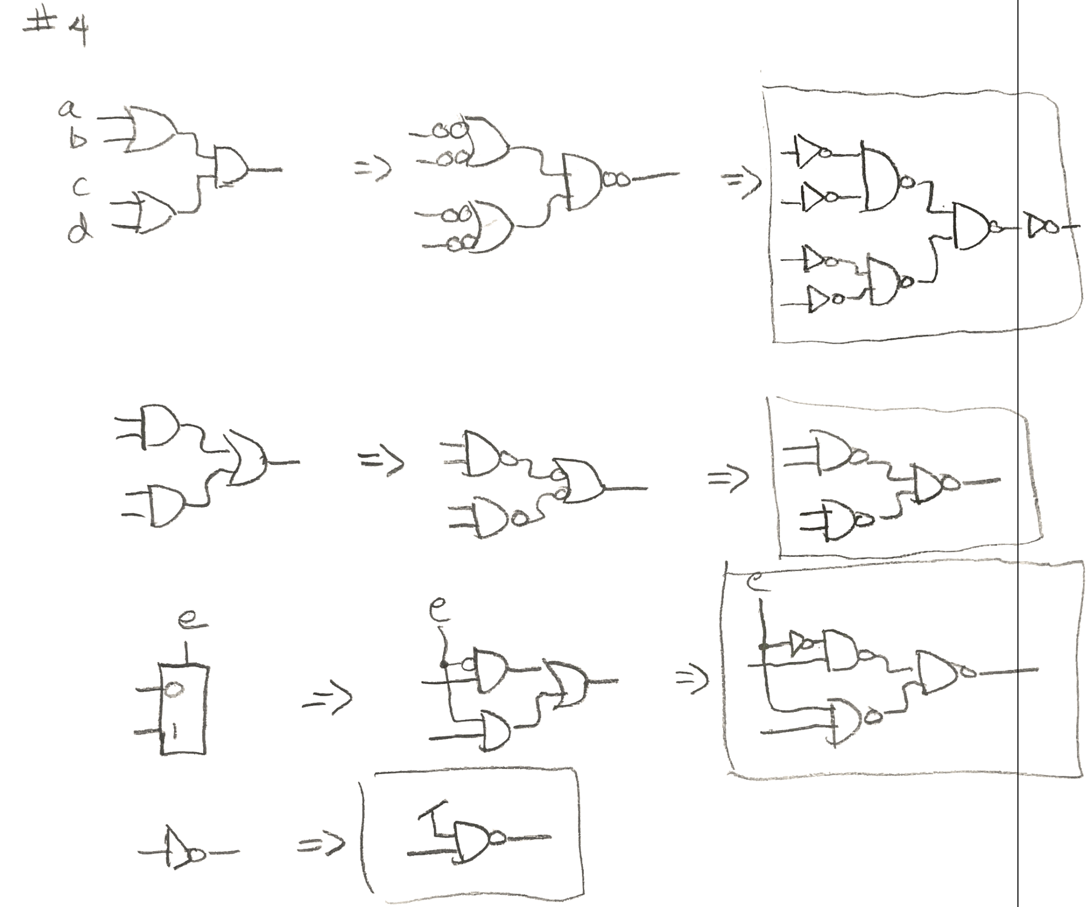
$(b + d)(a + \bar{b} + c)$

Solution:

Problem 4 *Logic Gates* **(10 points)**

Reimplement the combination logic circuit shown below *using as few NAND gates* as possible and *only* 2-input NAND gates.

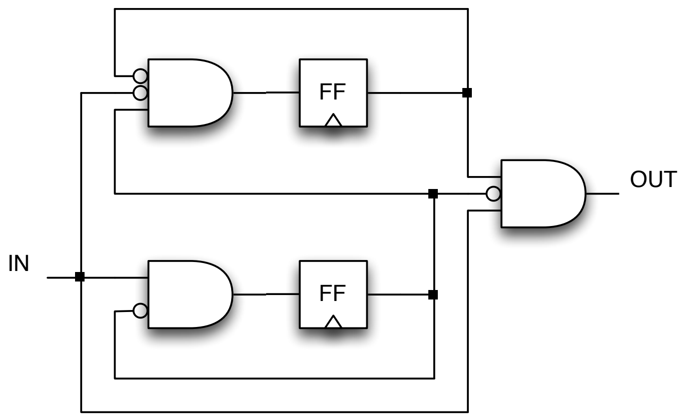




Solution:

Problem 5 *Finite State Machines* (25 points)

(a) (15 points) Draw the state transision diagram for the FSM implemented by the circuit show below. Assume that a reset signal initializes both flip-flops to 0.



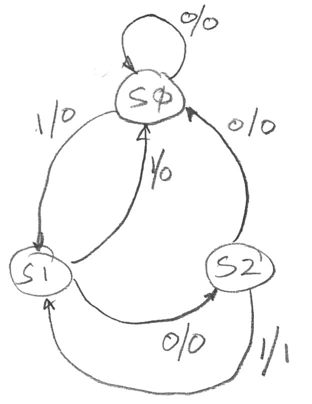
Assign bottom FF to be PS_0
top " " " PS_1

State Transition Table

PS_1 PS_0	IN	NS_1 NS_0	OUT
0 0	0	0 0	0
0 0	1	0 1	0
0 1	0	1 0	0
→ 0 1	1	0 0	0
1 0	0	0 0	0
1 0	1	0 1	1

State Transition Diagram

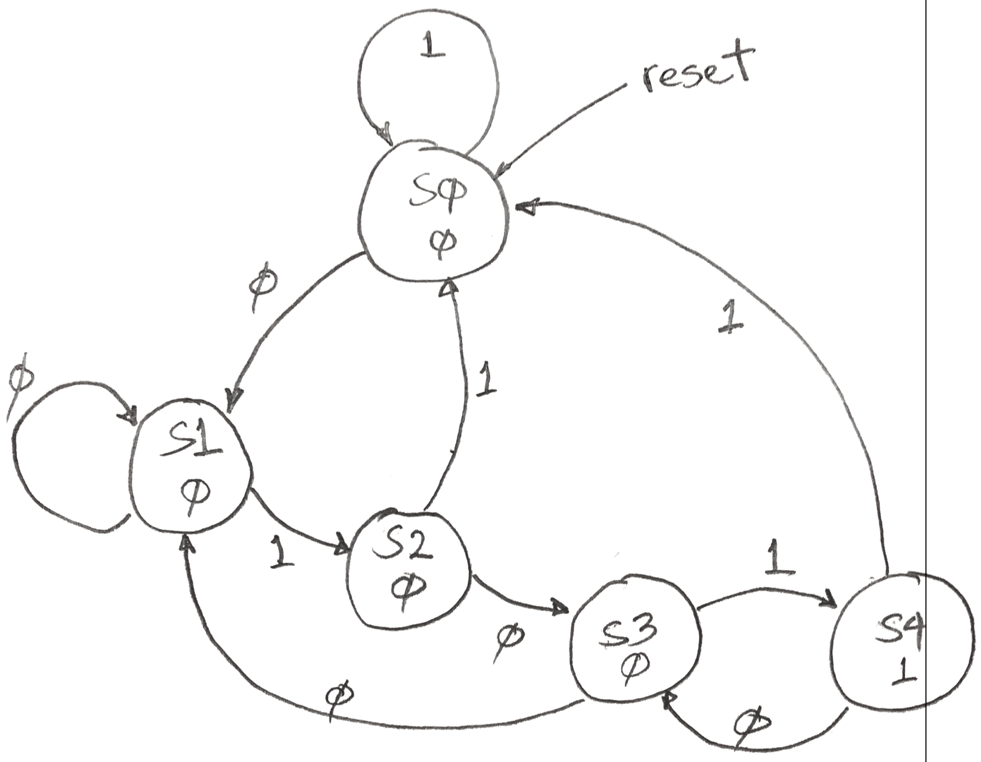
Let 00 = S0, 01 = S1, 10 = S2



Solution:

(b) (5 points) Draw the state transition diagram for a *Moore style* finite state machine with the following behavior: The FSM has a single 1-bit input and a single 1-bit output. A reset signal, RST, puts the FSM into an initial state. From there it looks for consecutive input occurrences of 01 and outputs a 1 after the second occurrence, and then on subsequent *contiguous* occurrences. For instance below is a typical input/output sequence:

input :	0	1	0	1	0	1	0	0	0	1	0	1	0	1	
output :		0	0	0	1	0	1	0	0	0	0	0	1	0	1



Solution:

(c) (5 points) **251A students only.** How many different Mealy machines with 3 states, 2 1-bit inputs, and 1 1-bit output exist? (Some machines might have the same behavior or be degenerate, but count them anyway.)

5.C Look at state transition table

PS	IN	NS	OUT
	00		
SØ	01		
	10		
	11		
	00		
S1	01		
	10		
	11		
	00		
	01		
S2	10		
	11		

12 rows

3 possibilities per row

2 possibilities per row

$$\boxed{3^{12} \cdot 2^{12}}$$

$531441 \cdot 4096 = 2{,}176{,}782{,}336$

$\approx 2^{19} \cdot 2^{12} = 2^{31}$

Solution:

Problem 6 ***CMOS logic*** **(10 points)**

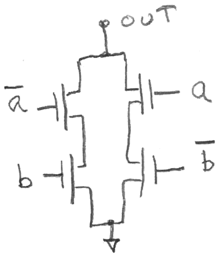
In the space below neatly draw the circuit diagram for a static CMOS implementation of a XOR or XNOR gate (your choice). Try to minimize the number of transistors.

#6 (easier in cmos)

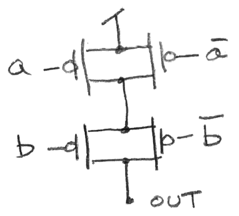
a b	XNOR
0 0	1
0 1	0
1 0	0
1 1	1

output $= \phi$ if $\bar{a}b + a\bar{b}$ is true

$\therefore$ pull down network is

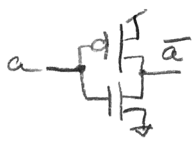


pull up network is the dual



also need 2 inverters

$\boxed{\text{12 transistors}}$



likewise for b

Solution:

Problem 7 *UART Sender* (24 points)

As mentioned in class, a UART (Universal Asynchronous Receiver/Transmitter) is a common logic block used to send or receive digital data over a serial line.

The sender is simpler than the receiver. The sender has four inputs: a bit clock `clk` equal to the data transmission rate, a reset `rst`, a parallel 8-bit data in `din`, and a 1-bit signal to indicate there is data to send `drdy`. It has a single 1-bit output `dout`.

When there is nothing to send the output is high. When `drdy` is 1, the sender captures the data to be sent and begins the sending process. To send data, the sender first outputs a 0 for a full bit clock (the "start bit"), then outputs each bit of data, starting at the lowest bit (D0), and finally outputs a 1 for a full clock cycle (the "stop bit").

Your implementation must be a Moore machine: `drdy` can glitch and you *must not* allow `dout` to glitch as a result. Additionally, if `drdy` is asserted while sending data you *must* ignore it: you must only accept `drdy` when idle or when you are sending the stop bit.

(a) (8 points) Draw the state transition diagram. For shorthand, you can use `d[0]` to `d[7]` as outputs for the data bits that are received when `drdy` is asserted when in idle state or when sending the stop bit.

(b) (12 points) Complete the following code for a Verilog implementation of a UART sender. You can create additional internal variables if you desire. You also don't need to implement your state transition diagram directly, but can instead use clever logic to simplify things.

```
module uart_send(clk, rst, din, drdy, dout) begin
  input clk, rst, drdy;
  input [7:0] din;
  output dout;

  reg [8:0] shifter;
  reg [3:0] ctr;
  reg dout;
  reg active;

  always @* begin
     dout = shifter[0]
     active = (ctr > 0 && ctr < 9))
  end

  always @(posedge clk) begin
     shifter <= rst ? (9'b111111111):((drdy && !active) ? {din[7:0],1'b0} :{1'b1,
     ctr <= rst ? 0 : (drdy || active ? ctr + 1 : 0)
  end
```

(c) (4 points) Assuming a straight forward synthesis mapping. How many flip-flops will need to be synthesized for your design?