Homework #3

(Deadline: 11:59PM PDT, Sunday, May 21, 2017)

Name (Last, First): Miller, Joseph

Student Id #: 504-744-848

INSTRUCTIONS

This homework is to be done individually. You may use any tools or refer to published papers or books, but may not seek help from any other person or consult solutions to prior exams or homeworks from this or other courses (including those outside UCLA). You're allowed to make use of tools such as Logisim, WolframAlpha (which has terrific support for boolean logic) etc.

You must submit all sheets in this file based on the procedure below. Because of the grading methodology, it is much easier if you print the document and answer your questions in the space provided in this problem set. It can be even easier if you answer in electronic form and then download the PDF. Answers written on sheets other that the provided space will not be looked at or graded. Please write clearly and neatly - if we cannot easily decipher what you have written, you will get zero credit

SUBMISSION PROCEDURE: You need to submit your solution online at Gradescope (https://gradescope.com/). Please see the following guide from Gradescope for submitting homework. You'd need to upload a PDF and mark where each question is answered.
http://gradescope-static-assets.s3-us-west-2.amazonaws.com/help/submitting_hw_guide.pdf

Problem #1

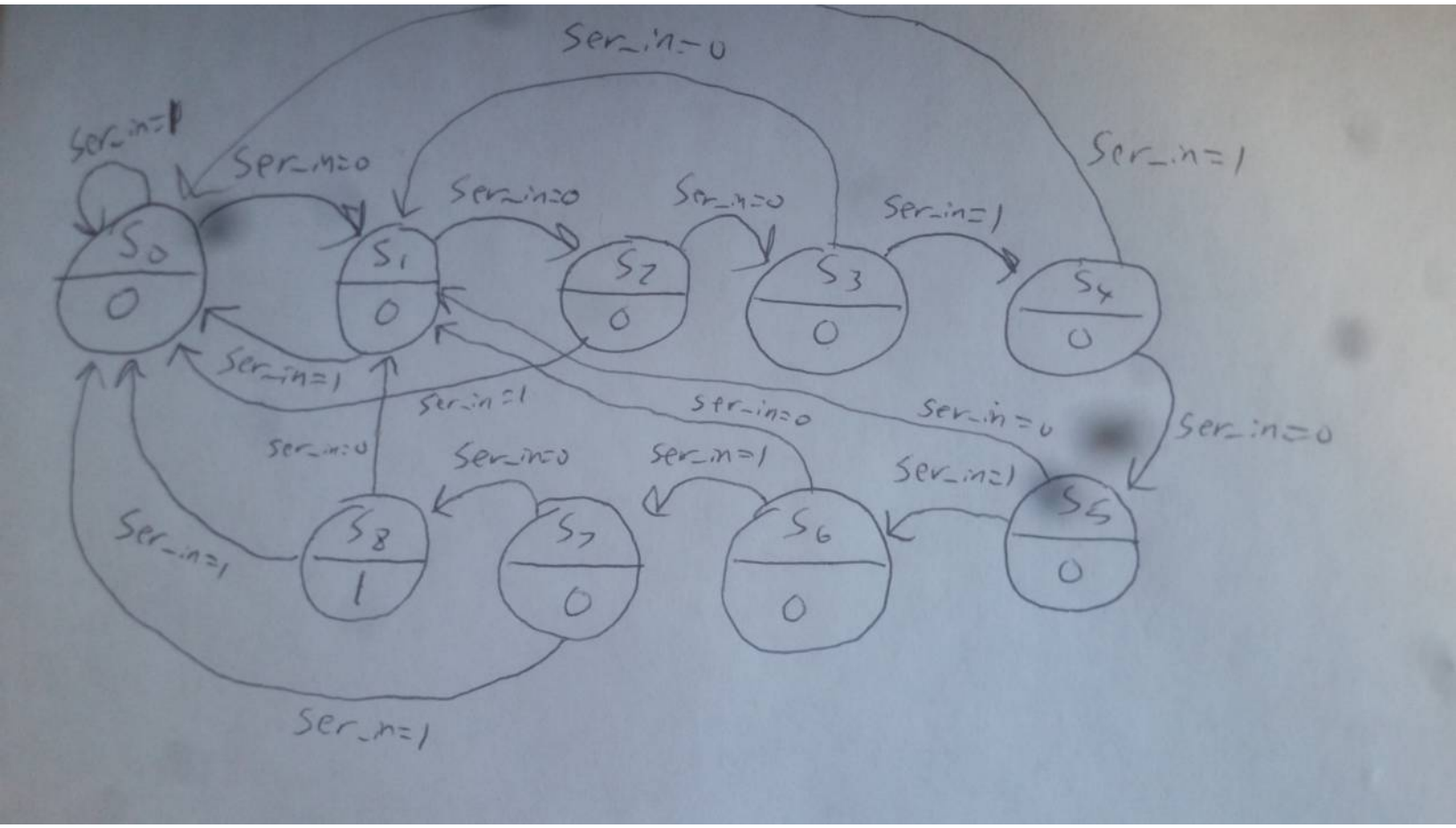
BSC is a point-to-point serial communication protocol back in the 1960's used by IBM (among others). It uses a handshake protocol similar to RS-232 or even later in TCP/IP. A transmission is recognized when a preamble sequence is detected. The preamble is the symbol, SYN, which corresponds to the hexadecimal sequence of 8'h16 (8'b00010110) where the input stream is big-endian. You are to design a sequential state machine that accepts the 1-bit input, *ser_in*, and one output, *match=1'b1* (asserted for 1 clock cycle), when the sequence is identified.

(a) Draw a Moore state machine that implements the function. How many states do you need?

(b) Draw a Mealy state machine that implements the function. How many states do you need? How is the Mealy FSM different from the Moore FSM (in terms of when the *match* signal is asserted)?

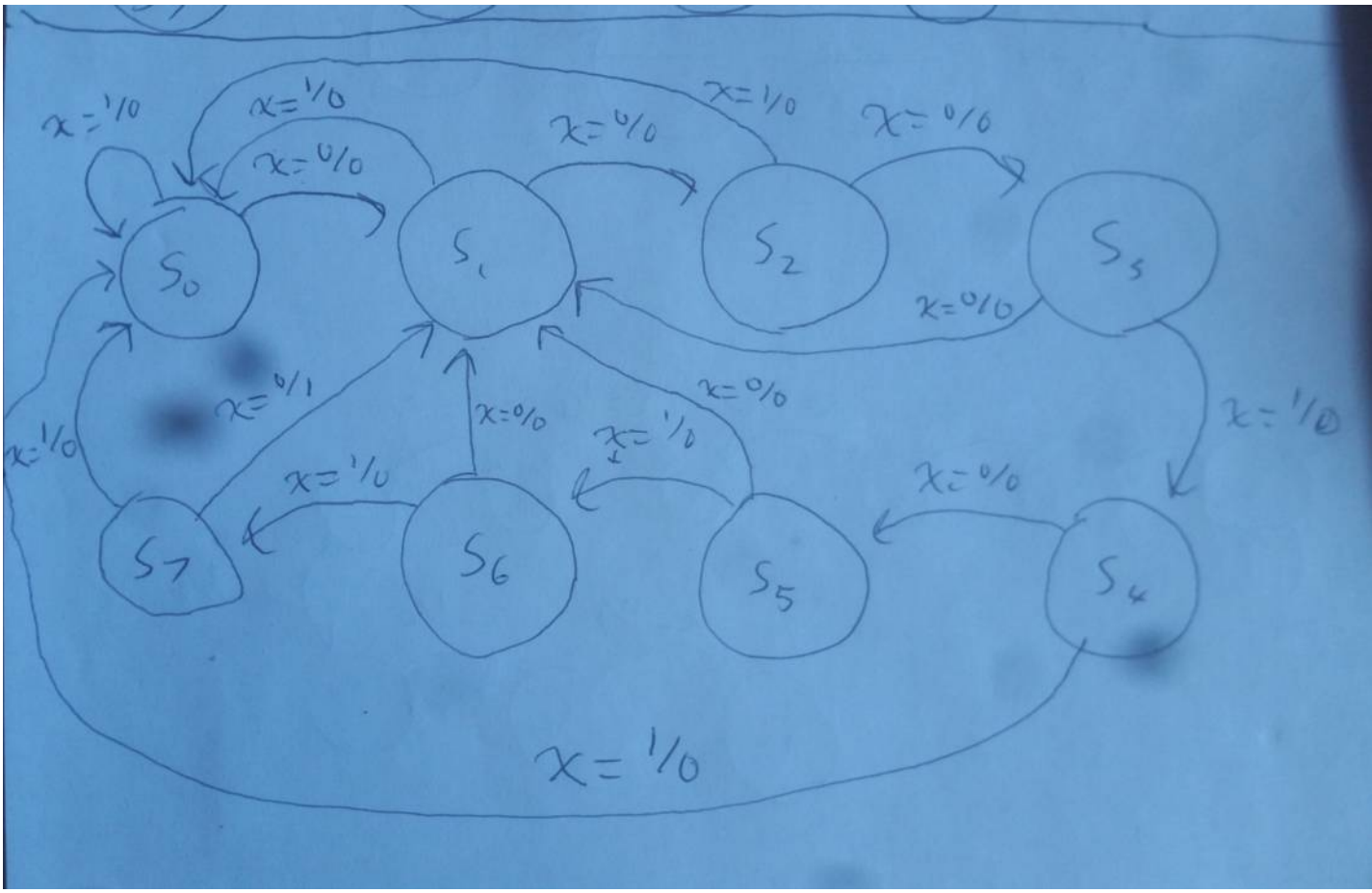
Answer the question for all parts in the space below.

a) For the Moore state machine we'll need 9 states. 1 will be used as the initial state and the final
8 will be used to signify having identified each of the 8 bits in the sequence 00010110.



(IMPORTANT: Keep this page in submission even if left unused)

b) Mealy machines change output as soon as the logic is done whereas a Moore machine will wait until the next clock edge to change. The Mealy design requires only 8 states for the
sign since it determines output based on the input as well as current state.



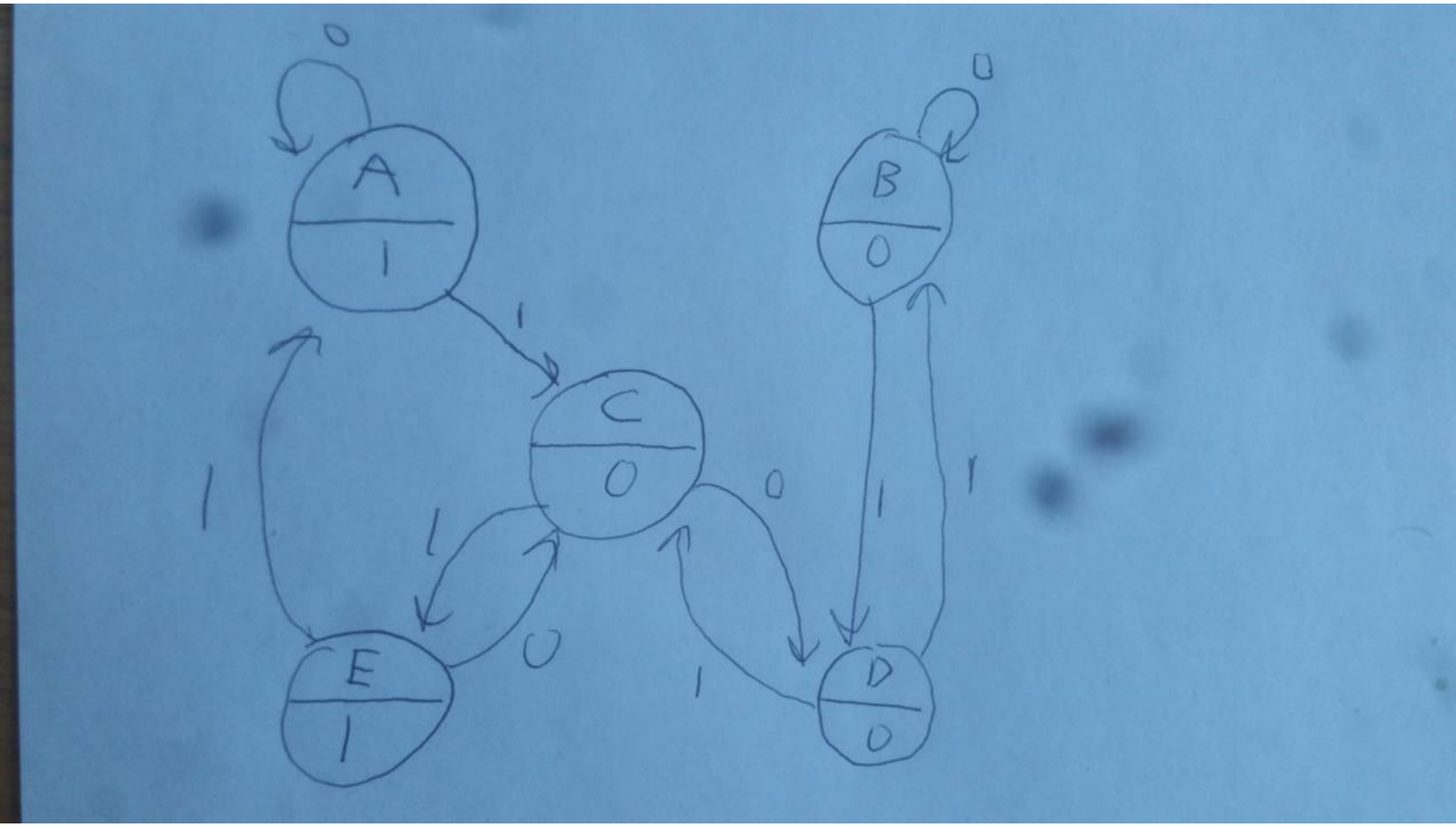
Problem #2

The following state transition table is provided.

Input	Current State	Output	Next State
0	A	1	A
1	A	0	C
0	B	0	B
1	B	0	D
0	C	0	D
1	C	1	E
0	D	0	C
1	D	0	B
0	E	0	C
1	E	1	A

(a) Draw the state diagram for the table.
(b) Is this a Mealy or Moore FSM?
(c) Use the following state assignment: A=001, B=010, C=101, D=110, E=100. Show the truth table for each of *next_state* bits and the *output* bit (4 signals).
(d) Write the logical equation for the MSB of the *next_state* bits. Use a K-Map to reduce the logic and write the logic in sum-of-products form.

Answer the question for all parts in the space below.



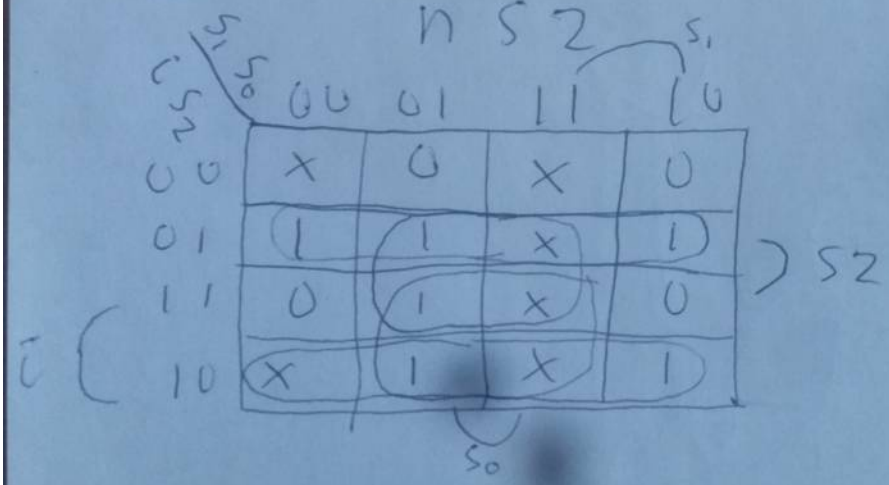
(IMPORTANT: Keep this page in submission even if left unused)

b) The diagram I drew is a Moore FSM since I noticed that the output was only dependent on what state the system was in. It could of course be drawn as a Mealy FSM by making the
outputs displayed alongside the inputs.

UC
(IN

S2 S1 S0

	State	i	NS2	NS1	NS0	Output
A	0 0 1	0	0	0	1	1
A	0 0 1	1	1	0	1	0
B	0 1 0	0	0	1	0	0
B	0 1 0	1	1	1	0	0
C	1 0 1	0	1	1	0	0
C	1 0 1	1	1	0	0	1
D	1 1 0	0	1	0	1	0
D	1 1 0	1	0	1	0	0
E	1 0 0	0	1	0	1	0
E	1 0 0	1	0	0	1	1

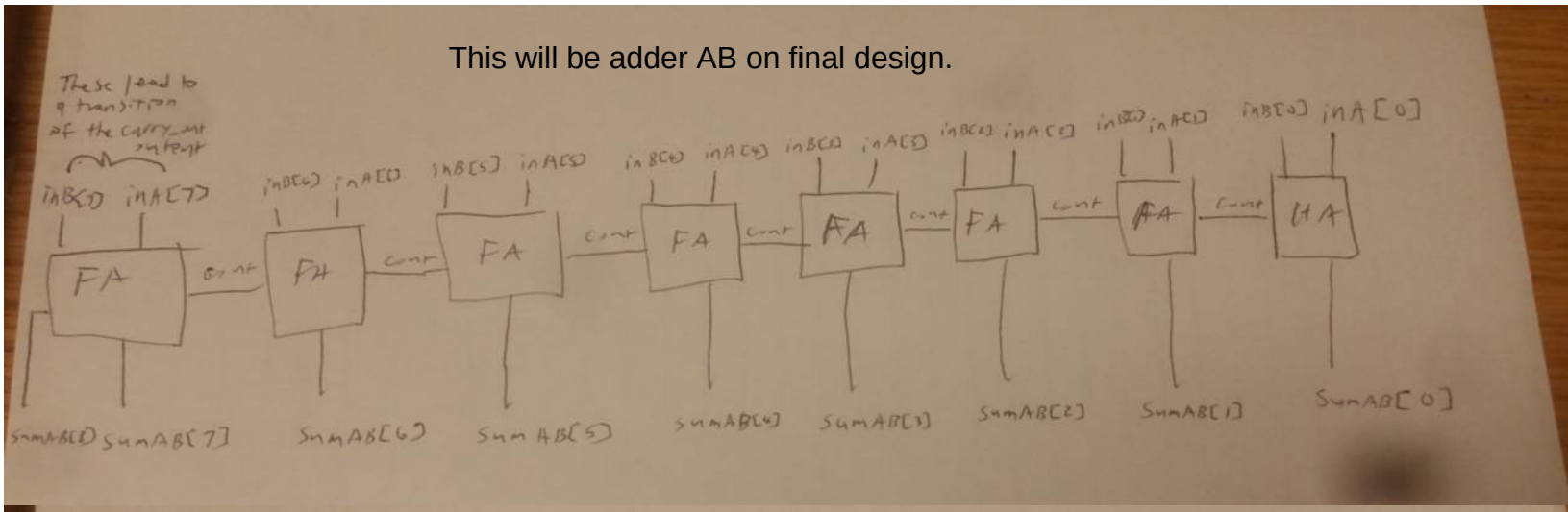


$NS2 = (S2 \wedge \neg i) \vee (S2 \wedge S0) \vee (S0 \wedge i) \vee (i \wedge \neg S2)$

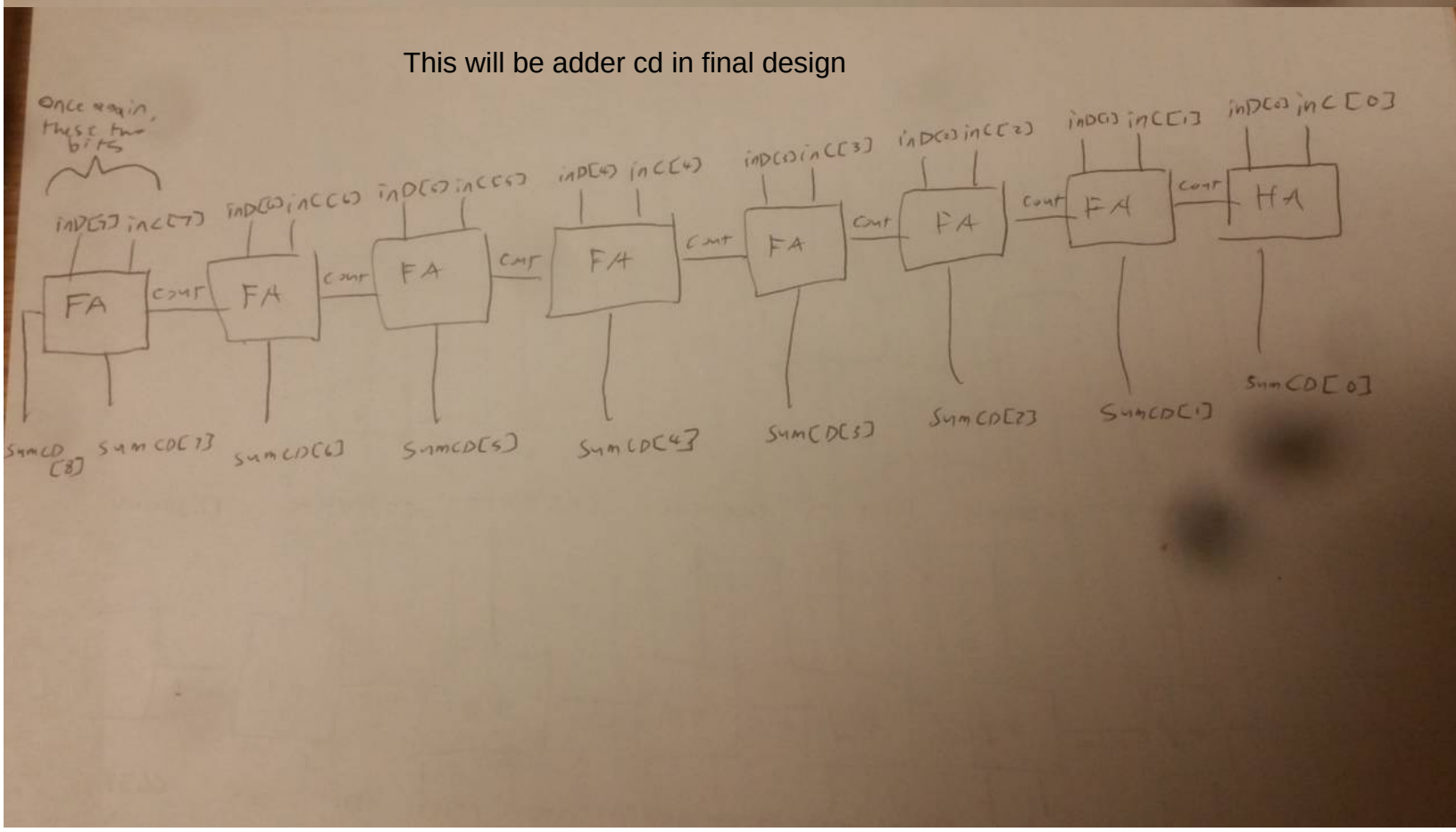
Problem #3

Using Full-Adders (FA) or Half-Adders (HA) as building blocks, design the logic that adds four 8-bit inputs, *inA*[7:0], *inB*[7:0], *inC*[7:0], and *inD*[7:0] for an output *sum*[8:0] and *carry_out*. Show on your design which input bit(s) lead to a transition of the *sum* or *carry_out* output (indicate which bit) that would traverses the most adder building blocks (HA or FA). Try to use the least number of FA or HA blocks in the design.

This will be adder AB on final design.

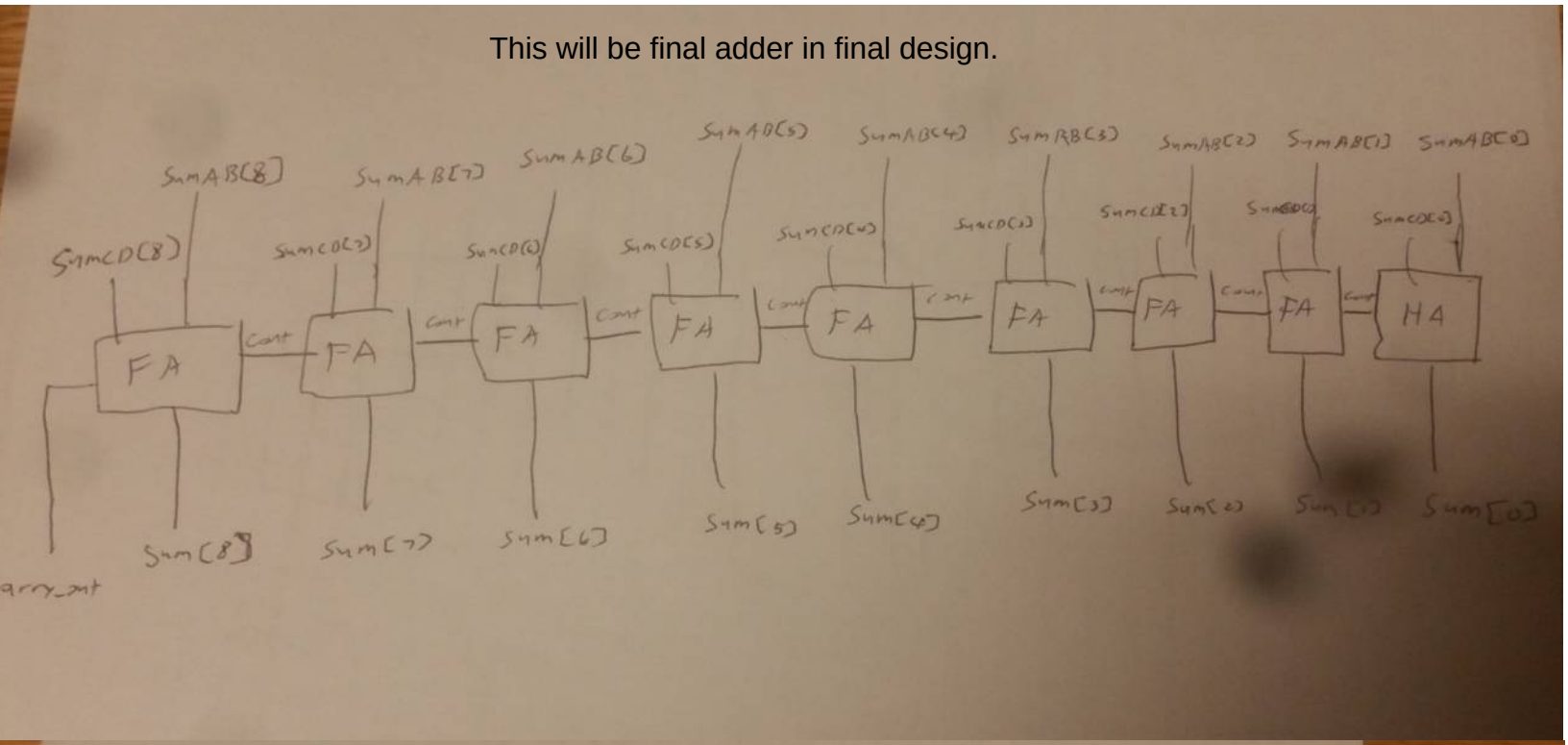


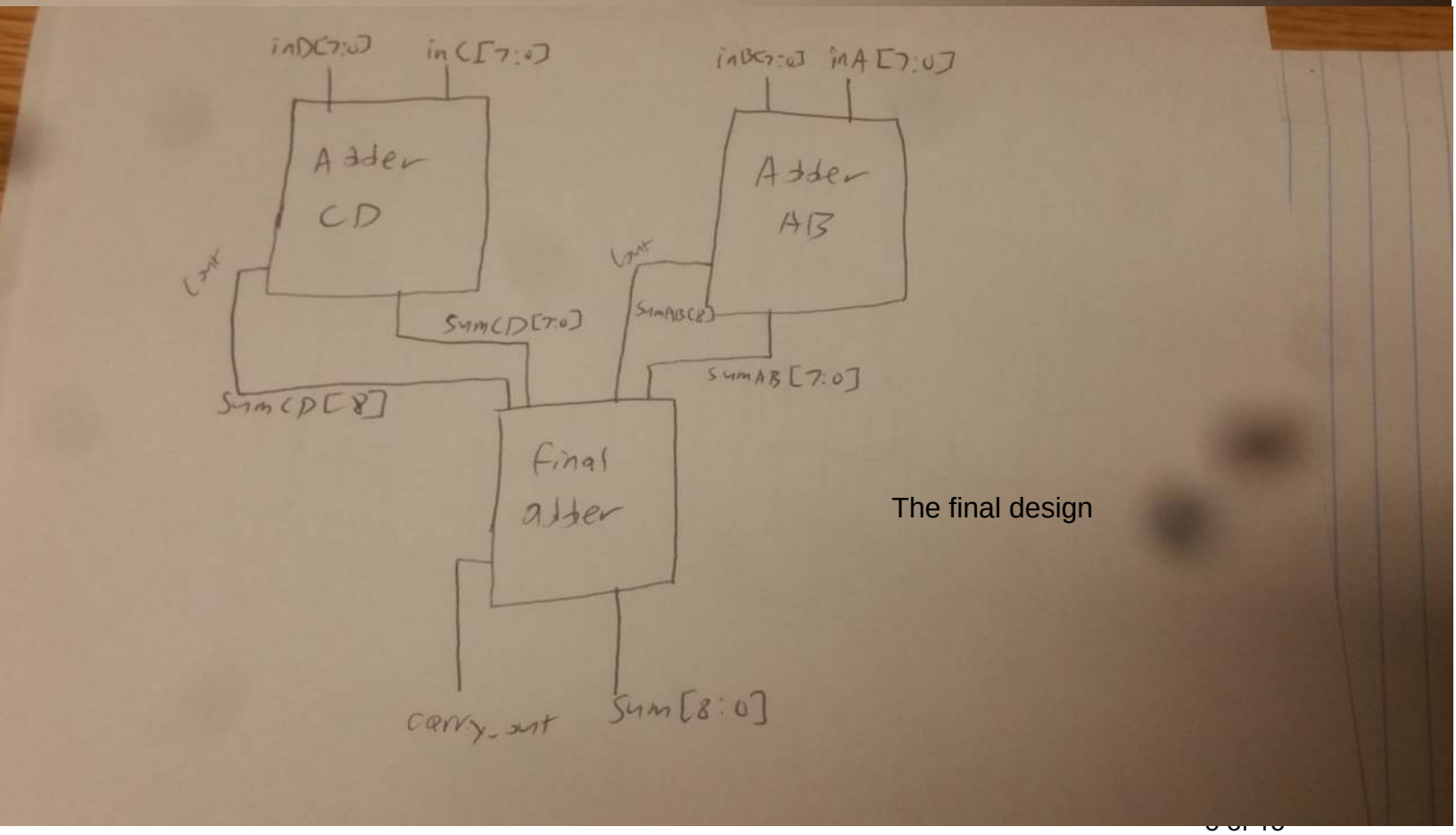
This will be adder cd in final design



(IMPORTANT: Keep this page in submission even if left unused)

This will be final adder in final design.

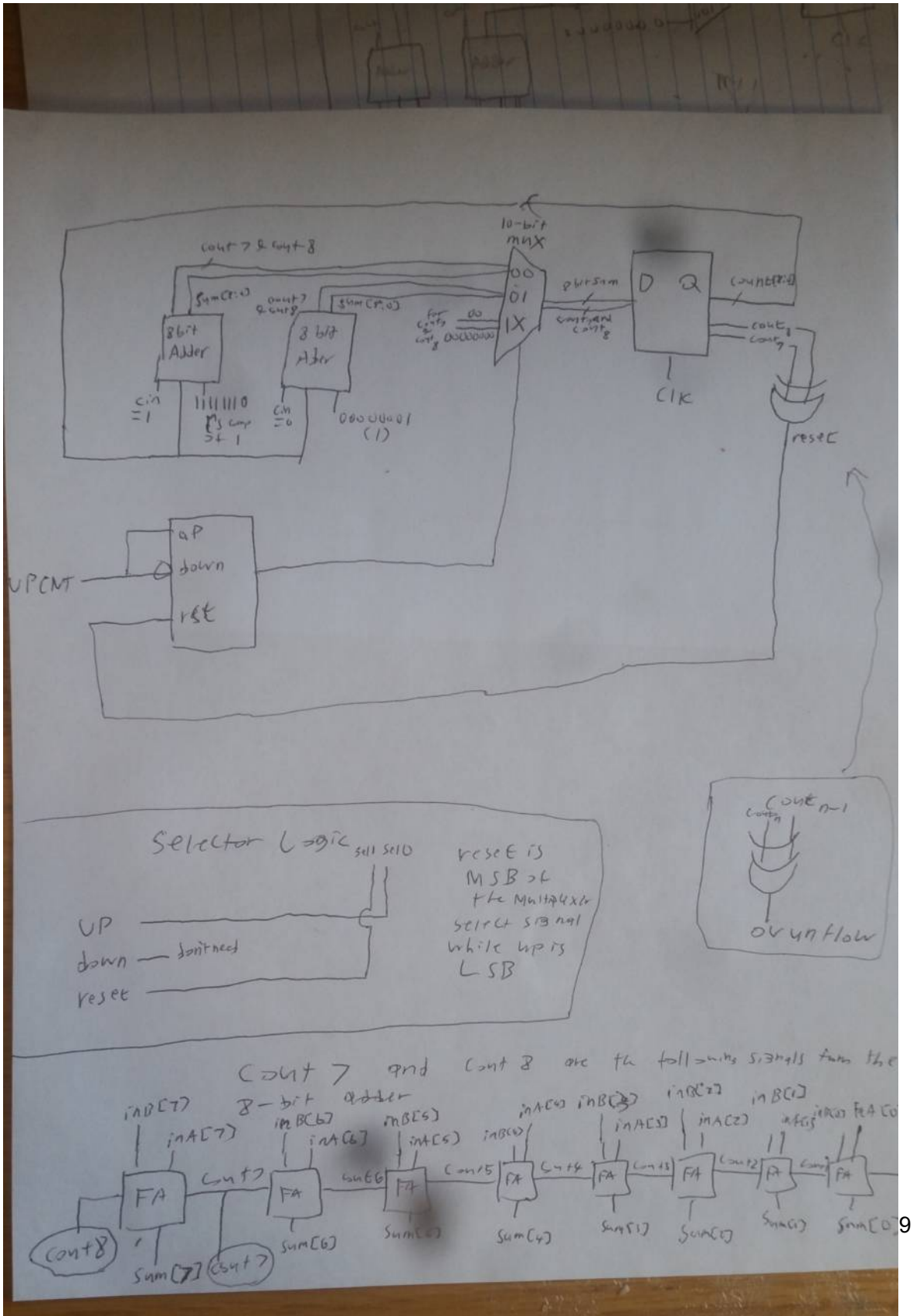




The final design

Problem #4

Design an up/down 8-bit counter using an 8-bit adder (ADD8) and logic gates that saturates when the count saturates to -128,127 when overflowing and resets to 8'b00000000. Input to indicate up or down is UPCNT=1b'1 for up and UPCNT=1b'0 for down.



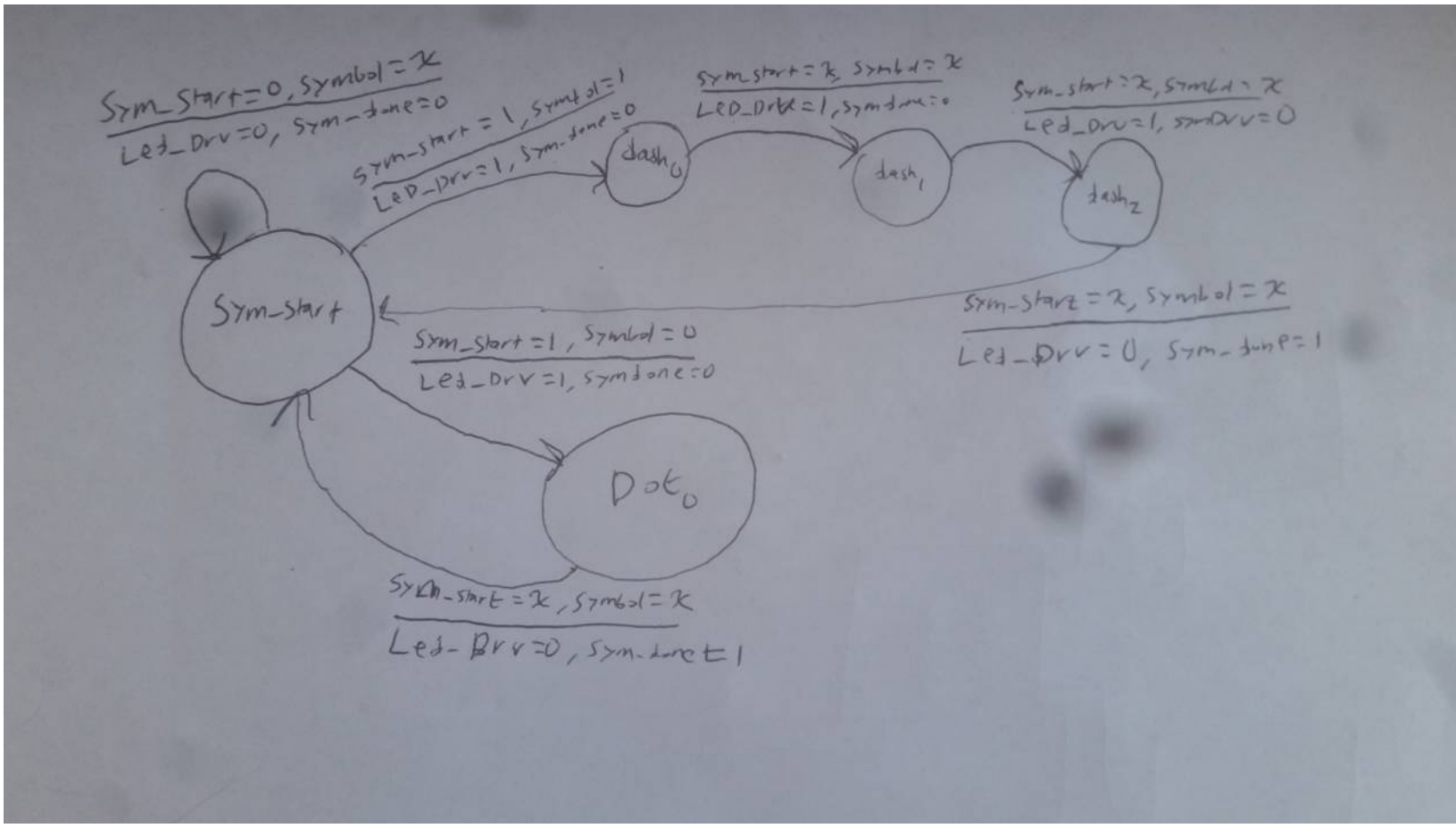
Problem #5

The following parts are steps to design a block that flashes SOS in Morse code on an LED by using a hierarchy of multiple FSMs. Note that part (a) will be used in Design Assignment #3 to implement a Morse encoder.

(a) Draw the state diagram of a Mealy FSM that handles either of the 2 Morse symbols, Dot and Dash. At the SYM_START (initial) state, if the input SYM_STRT=1'b1, the FSM transitions to either output a Dot or Dash depending on the input SYMBOL. Dash is outputted when SYMBOL=1'b1 and Dot is outputted when SYMBOL=1'b0. When outputting a Dash, the output signal LED_DRV is driven to 1'b1 for 3 cycles and then OFF for 1 cycle before starting the next symbol. Similarly, when outputting a Dot, the output signal LED_DRV is driven to 1'b1 for 1 cycle and then OFF for 1 cycle before starting the next symbol. When the LED turns off, a signal SYM_DONE=1'b1 is asserted for 1 cycle. The SYM_DONE should be 1'b0 otherwise.

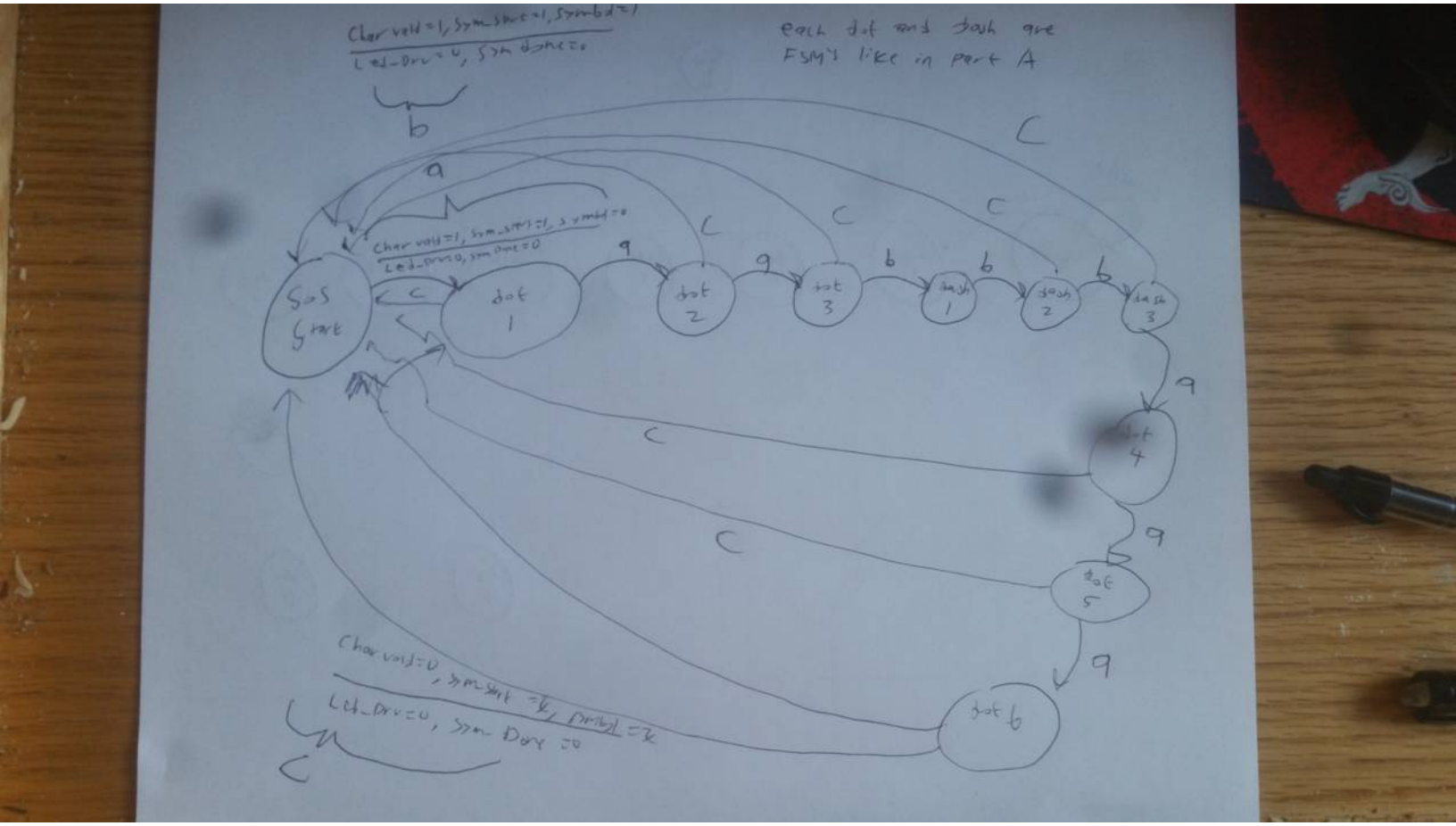
(b) Draw the state diagram of an FSM that generates 3 Dot, 3 Dash, then 3 Dot (SOS in Morse) using the FSM you designed in (a). Note that the state transition occurs whenever it receives SYM_DONE. A CHAR_VALD=1'b1 signal triggers this FSM when it is in its SOS_START state. Note that this FSM is a simpler version of one you will design in Design Assignment #3 where an arbitrary character will be encoded. However, this gives you an idea on how the FSM in (a) interacts with another FSM.

Answer the question for all parts in the space below.



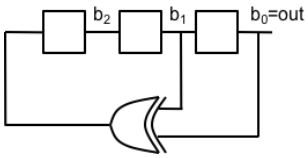
(IMPORTANT: Keep this page in submission even if left unused)

b)

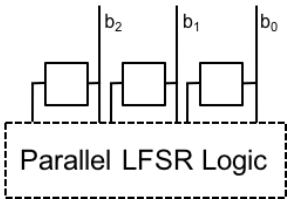


Problem #6

The following is a LFSR (linear feedback shift register). Note that it is often used to generate a pseudo-random number sequence because the generated sequence has very flat spectral properties implying good randomness.



(a) If the LFSR is reset to all ones, what is the sequence at the output?
(b) How many bits cycle through before the sequence repeats?
(c) What happens to the sequence if the LFSR is reset to all zeros?
(d) Instead of producing this sequence serially using the clock, f_A, you are tasked to produce this sequence in parallel, 3 bits at a time, with each cycle of clock, f_A. In this design, if a serializer running at $3*f_A$ shifts the 3 bits serially out, the same sequence as (a) would be produced at a higher rate. Design the parallel LFSR logic.



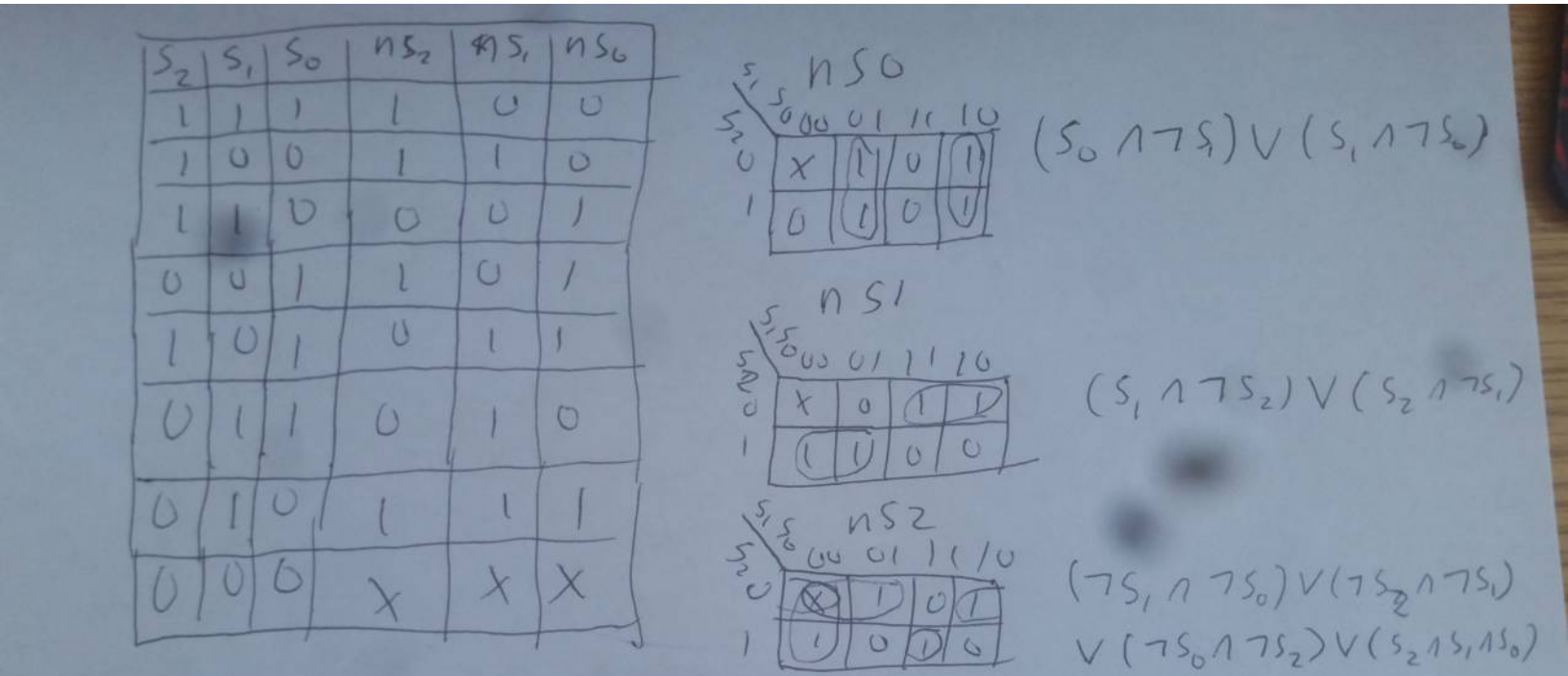
Answer the question for all parts in the space below.

a) 111, 011, 001, 100, 010, 101, 110 -> 111

b) it takes 7 cycles to repeat so 21 bits would shift before getting 111.

c) it will remain 000 forever since 0 xor 0 is 0.

d)



(IMPORTANT: Keep this page in submission even if left unused)