Department of Electrical and Computer Engineering

The University of Texas at Austin

EE 306, Fall 2023
Problem Set 3
Due: **October 9th**, before class
Yale N. Patt, Instructor
TAs: Chester Cai, Kayvan Mansoorshahi, Sophia Jiang, Jaeyoung Park, Varun Arumugam, Anna Guo, Nadia Houston, Asher Nederveld, Edgar Turcotte

Instructions: You are encouraged to work on the problem set in groups and turn in one problem set for the entire group. **The problem sets are to be submitted on Gradescope**. Only one student should submit the problem set on behalf of the group, but everyone should create a gradescope account and be tagged on the homework.

1. We want to make a state machine for the scoreboard of the Texas vs. Oklahoma Football game. The following information is required to determine the state of the game:

1. Score: 0 to 99 points for each team
2. Down: 1, 2, 3, or 4
3. Yards to gain: 0 to 99
4. Quarter: 1, 2, 3, 4
5. Yardline: any number from Home 0 to Home 49, Visitor 0 to Visitor 49, or 50
6. Possession: Home, Visitor
7. Time remaining: any number from 0:00 to 15:00, where m:s (minutes, seconds)

a. What is the minimum number of bits that we need to use to store the state required?
b. Suppose we make a separate logic circuit for each of the seven elements on the scoreboard, how many bits would it then take to store the state of the scoreboard?
c. Why might the method of part b be a better way to specify the state than the method of part a?

2. Shown below is a partially completed state diagram of a finite state machine that takes an input string of *H* (heads) and *T* (tails) and produces an output of 1 every time the string HTHH occurs.

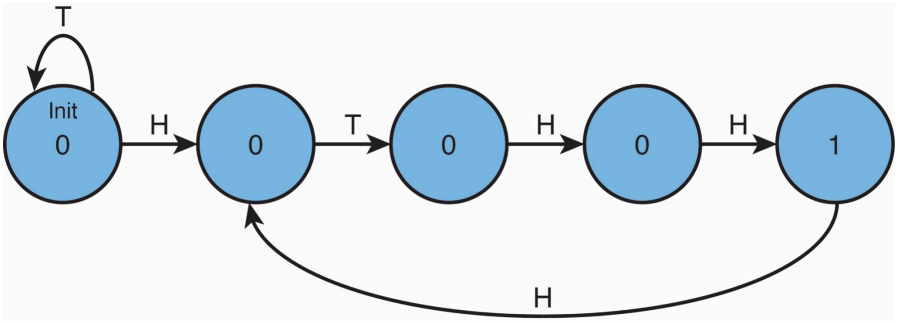


Figure 1.

Fore example, if the input string is:

```
H, H, H, H, H, T, H, H, T, H, H, H, H, H, T, H, H, T
```

The output would be:

```
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0
```

Note that the 8th coin toss (H) is part of two HTHH sequences.

a. Complete the state diagram of the finite state machine that will do this for any input sequence of any length.
b. If this state machine is implemented with a sequential logic circuit how many state variables will be needed? (Recall, the number of state variables is the same as the number of bits needed to represent all of the states.)

3. (Adopted from 3.37 in the textbook) If a particular computer has 8 byte addressability and a 4 bit address space, how many bytes of memory does that computer have?

4. ***Elevator Problem Revisited.***

Recall the elevator controller problem on Problem Set 2. You were asked to design the truth table for an elevator controller such that the option to move up or down by one floor is disabled. If there is a request to move only one floor or to move zero floors, the elevator should remain on the current floor. For this problem, you will design the state machine for the sequential logic circuit for an elevator controller which performs the same operation. You can assume that the building the elevator is in has 4 floors. The input to the state machine is the next requested floor. There will be a state for each floor

the elevator could be on. Draw a finite state machine that describes the behavior of the elevator controller. How many bits are needed for the inputs?

5. (Adopted from 3.39 in the textbook) Using Figure 3.21 on page 78 in the book, the diagram of the, 2^2-by-3-bit memory.

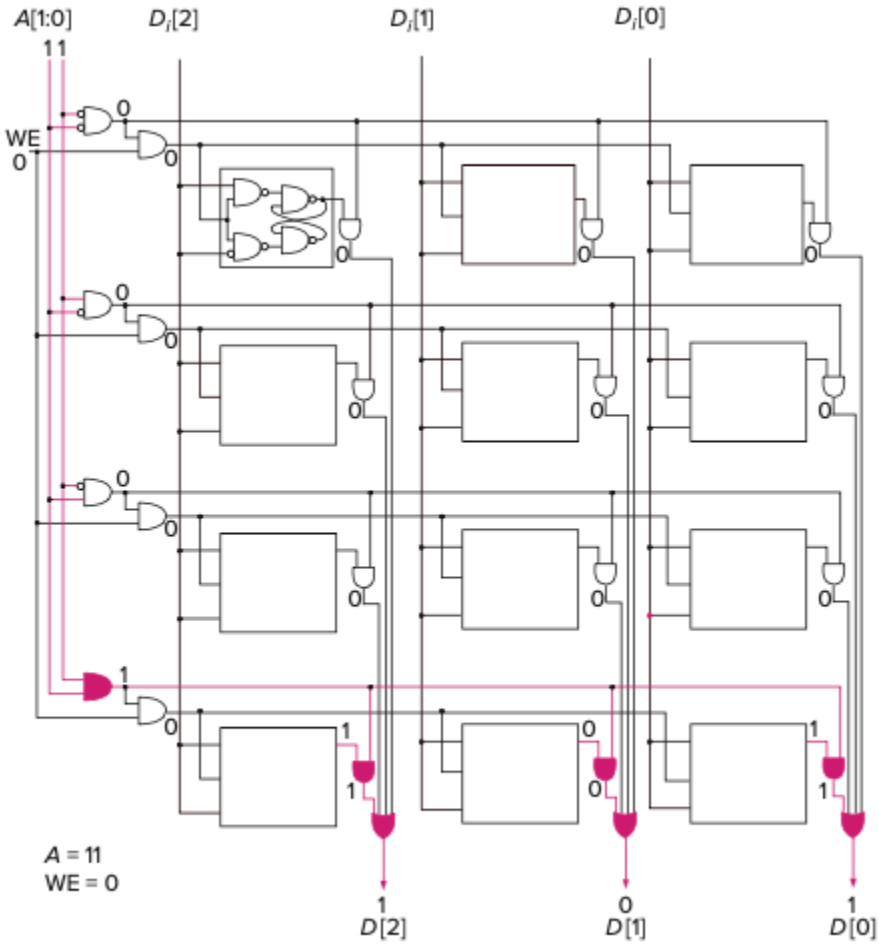


Figure 3.21 Reading location 3 in our 2^2-by-3-bit memory.

a. To read from the fourth memory location, what must the values of A[1:0] and WE be?
b. To change the number of locations in the memory from 4 to 60, how many address lines would be needed? What would the addressability of the memory be after this change was made?
c. Suppose the width (in bits) of the program counter is the minimum number of bits needed to address all 60 locations in our memory from part (b). How many additional memory locations could be added to this memory without having to alter the width of the program counter?

6. The figure below is a diagram of a 2^2-by-16-bit memory, similar in implementation to the memory of Figure 3.21 in the textbook. Note that in this figure, every memory cell represents **4 bits** of storage instead of **1 bit** of storage. This can be accomplished by using 4 Gated-D Latches for each memory cell instead of using a single Gated-D Latch. The hex digit inside each memory cell represents what that cell is storing prior to this problem.

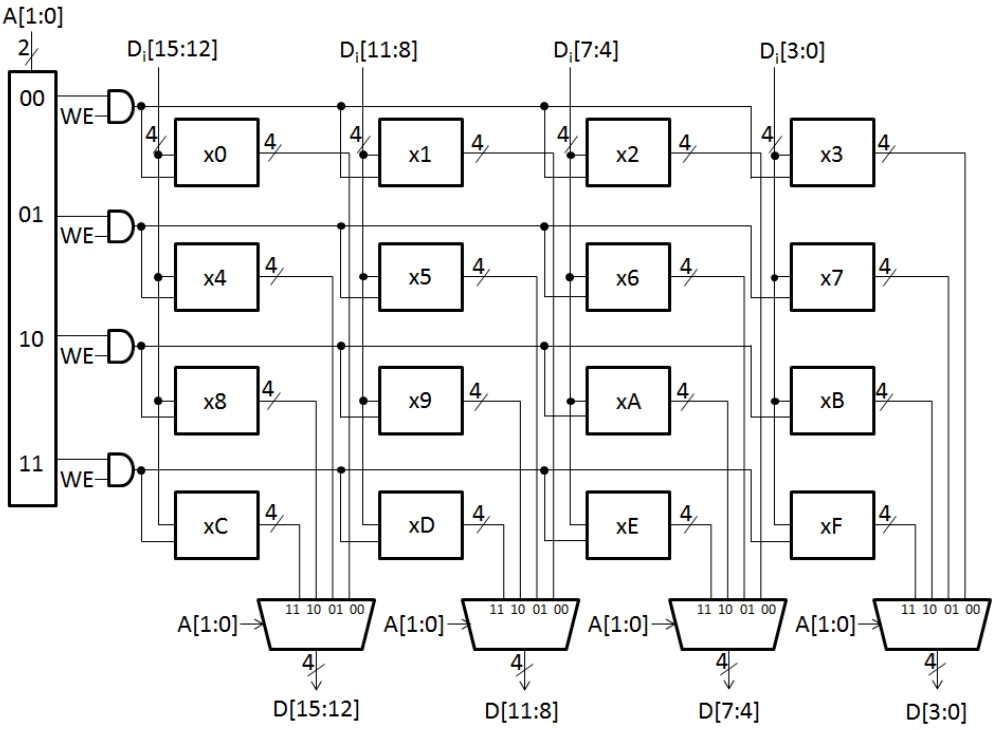


Figure 2. 2^2-by-16 bit memory

a. What is the address space of this memory?
b. What is the addressability of this memory?
c. What is the total size in bytes of this memory?
d. This memory is accessed during four consecutive clock cycles. The following table lists the values of some important variables **just before the end of the cycle** for each access. Each row in the table corresponds to a memory access. The read/write column indicates the type of access: whether the access is reading memory or writing to memory. Complete the missing entries in the table.

WE	A[1:0]	D_i[15:0]	D[15:0]	Read/Write
0	01	xFADE		
1	10	xDEAD		
		xBEEF	x0123	Read
	11		xFEED	Write

7. (Adopted from 3.47 in the textbook) The Eta Kappa Nu (HKN) office sells sodas for 35 cents. Suppose they install a soda controller that only takes the following three inputs: nickel, dime, and quarter. After you put in each coin, you push a pushbutton to register the coin. If at least 35 cents has been put in the controller, it will output a soda and proper change (if applicable). Draw a finite state machine that describes the behavior of the soda controller. Each state will represent how much money has been put in (*Hint:* There will be seven of those states). Once enough money has been put in it, the controller will go to a final state where the person will receive a soda and proper change (*Hint:* There are five such final states). From the final state, the next coin that is put in will start the process again, contributing to the next purchase.

8. Suppose that an instruction cycle of the LC-3 has just finished and another one is about to begin. The following table describes the values in select LC-3 registers and memory locations.

Register	Value
IR	x3001
PC	x3003
R0	x3000
R1	x3000

R2	x3002
R3	x3000
R4	x3000
R5	x3000
R6	x3000
R7	x3000
Memory Location	**Value**
x3000	x62BF
x3001	x3000
x3002	x3001
x3003	x62BE

For each phase of the new instruction cycle, specify the values that PC, IR, MAR, MDR, R1, and R2 will have *at the end* of the phase in the following table.

	PC	IR	MAR	MDR	R0	R1	R2	R3	R4	R5	R6	R7
Fetch												
Decode												
Evaluate Address												
Fetch Operands												
Execute												
Store Result												

Hint: Notice that values of memory locations x3000, and 3003 can be interpreted as LDR instructions.

9. (Adopted from 4.8 in the textbook) Suppose a 32-bit instruction has the following format:

OPCODE	DR	SR1	SR2	UNUSED

If there are 255 opcodes and 120 registers, and every register is available as a source or destination for every opcode,

a. What is the minimum number of bits required to represent the OPCODE?
b. What is the minimum number of bits required to represent the Destination Register (DR)?
c. What is the maximum number of UNUSED bits in the instruction encoding?

10. ***Trying Out Flip-Flops.***

The flip-flop we introduced in class is shown below.

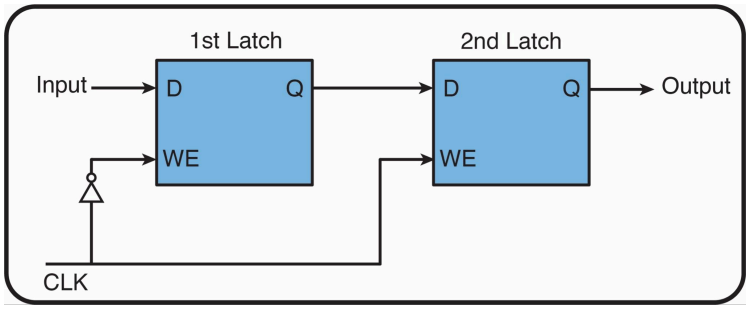
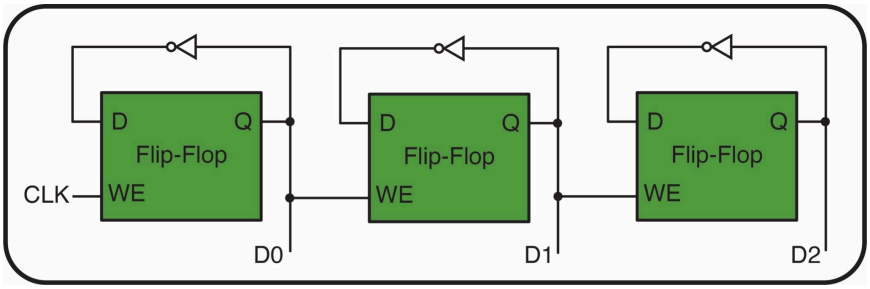
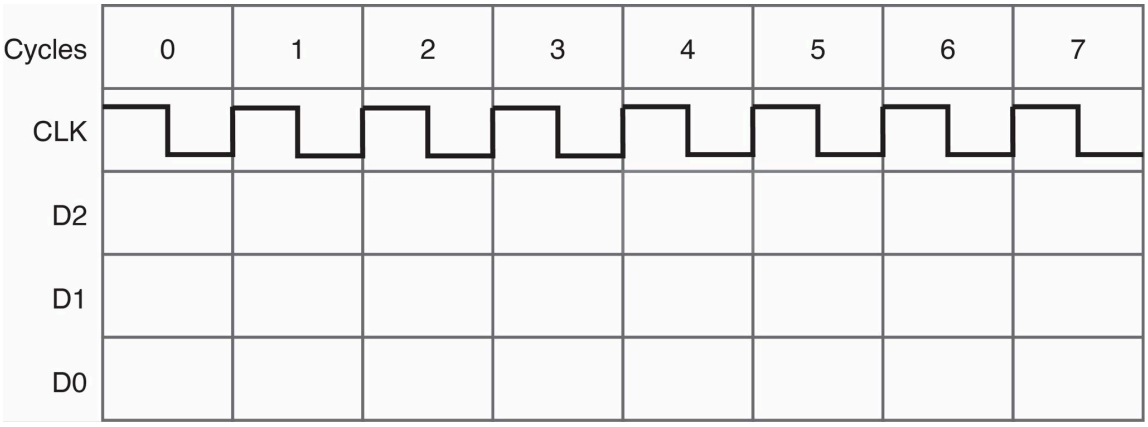


Figure 3. The two-gated-latch flip-flop.

Note that the input value is visible at the output after the clock transitions from 0 to 1. Shown below is a circuit constructed with three of these flip-flops.



Your job: Fill in the entries for D2, D1, D0 for each of clock cycles shown: (In Cycle 0, all three flip-flops hold the value 0). (Just fill in 0 or 1 in empty squares, you do not need to draw a timing diagram)



In 10 words or less, what is this circuit doing?

11. Write a program in LC-3 machine language that places a 1 into R0 if the value in R1 is identical to the value in R2, and places a 0 into R0 if the values in R1 and R2 are different.

12. Write the LC-3 assembly code for the following program, then explain what it does (in 20 words or fewer).

```
x5920
x907F
x1021
x1002
x0801
x1921
xF025
```

13. Write the LC-3 assembly code for the following program, then explain what it does (in 20 words or fewer).

```
x5020
x5A61
x0A01
x1021
xF025
```

14. An LC-3 program is stored in memory locations x4000 to x4005. Note that the branch instruction in memory location x4002 has an unspecified PCoffset9, denoted as **x**.

Address	Instruction
x4000	0101 000 000 1 00000
x4001	0001 000 000 1 00001
x4002	0000 011 **x**
x4003	0001 000 000 1 00010
x4004	0001 000 000 1 00011
x4005	1111 0000 0010 0101

The program starts executing with PC = x4000.

Your job: In the table below, for each value of X, answer the question: “Does the program halt?” (Yes or No). If your answer is “Yes”, answer the question: “What value is stored in R0 immediately after the instruction at x4004 completes execution?” If your answer is “No”, put a dash in the column labeled “Value stored in R0”.

x	Does the program halt?	Value stored in R0
000000010		
000000001		
000000000		
111111111		
111111110		