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, 50
   6) Possesion: 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?

   (100*100)*4*100*4*101*2*901 = 2912032000000.

   2^41 < 2912032000000 < 2^42 so we need 42 bits

   (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?

   1) 7 x 2 bits

   2) 2 bits

   3) 7 bits

   4) 2 bits

   5) 7 bits

6) 1 bit

7) 4 bits for minutes 6 bits for seconds

Total 43 bits

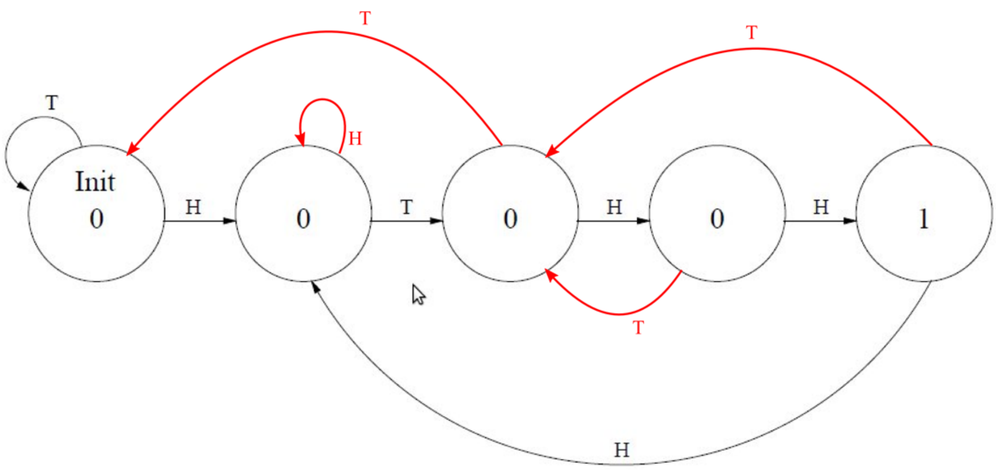
(c) Why might the method of part b be a better way to specify the state than the method of part a?

The assignments in (b) are easier to decode

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

    a) Complete the state diagram of the finite state machine that will do this for any input sequence of any length

Figure
4



For 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 8^{th} coin toss (H) is part of two HTHH sequences.

b) If this state machine is implemented with a sequential logic circuit how many state variables will be needed?

3 bits

3. (3.37)

If a particular computer has 8 byte addressability and a 8 bit address space, how many bytes of memory does that computer have?

Number of byes = address space x adressability. 2^8 x 2^3 = 2^11 = 2048 bytes

4. (3.33)

Using Figure 3.21 on page 78 in the book, the diagram of the, 2^2-by-3-bit memory.

a. To read from the third memory location, what must the values of *A[1:0]* and *WE* be?

To read from the third location `A[1:0]` should be 10, to read from memory the `WE` bit should be 0. To write to memory the WE bit must be 1.

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?

To address 60 locations you need 6 bits of address line, which means your MAR is 6 bits . However since we did not change the number of bits stored at each location the addressability is still 3 bits

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?

You need 6 bits for part b, which can address 64 different locations so you could add 4 more locations and not have to increase the width of the program counter.

5. 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.

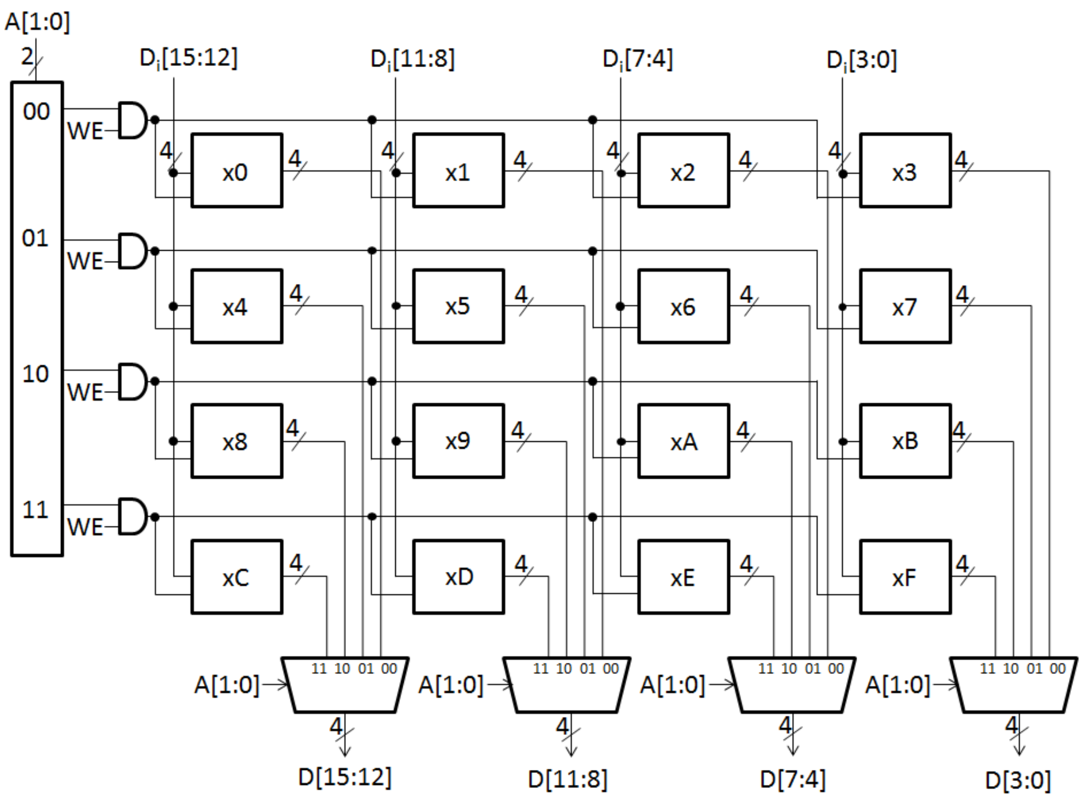


Figure 3: 2^2-by-16 bit memory

a. What is the address space of this memory?

2^2=4 memory locations.

b. What is the addressability of this memory?

16 bits.

c. What is the total size in bytes of this memory?

8 bytes.

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]	Di[15:0]	D[15:0]	Read/Write
0	01	xFADE	**x4567**	**Read**
1	10	xDEAD	**xDEAD**	**Write**
0	**00**	xBEEF	x0123	Read
1	11	**xFEED**	xFEED	Write

6. (4.8)
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*?

   255 opcode, 8 bits are required to represent the OPCODE

b. What is the minimum number of bits required to represent the Destination Register *(DR)*?

   120 registers, 7 bits to represent the DR

c. What is the maximum number of *UNUSED* bits in the instruction encoding?

   3 registers and 1 opcode, 3x7 + 8 = 29 bits. So there are 3 ununsed bits

7. *A State Diagam*

We wish to invent a two-person game, which we will call XandY that can be played on the computer. Your job in this problem is contribute a piece of the solution.

The game is played with the computer and a deck of cards. Each card has on it one of four values (X, Y, Z, and N). Each player in turn gets five attempts to accumulate points. We call each attempt a round. After player A finishes his five rounds, it is player B's turn. Play continues until one of the players accumulates 100 points. Your job today is to ONLY design a finite state machine to keep track of the STATE of the current round. Each round starts in the intial state, where X=0 and Y=0. Cards from the deck are turned over one by one. Each card transitions the round from its current state to its next state, until the round terminates, at which point we'll start a new round in the initial state.

The transistions are as follows:

X: The number of X's is incremented, producing a new state for the round.

Y: The number of Y's is incremented, producing a new state for the round.

Z: If the number of X's is less than 2, the number of X's is incremented, producing a new state for the round. If the number of X's is 2, the state of the current round does not change.

N: Other information on the card gives the number of points accumulated. N also terminates the current round.

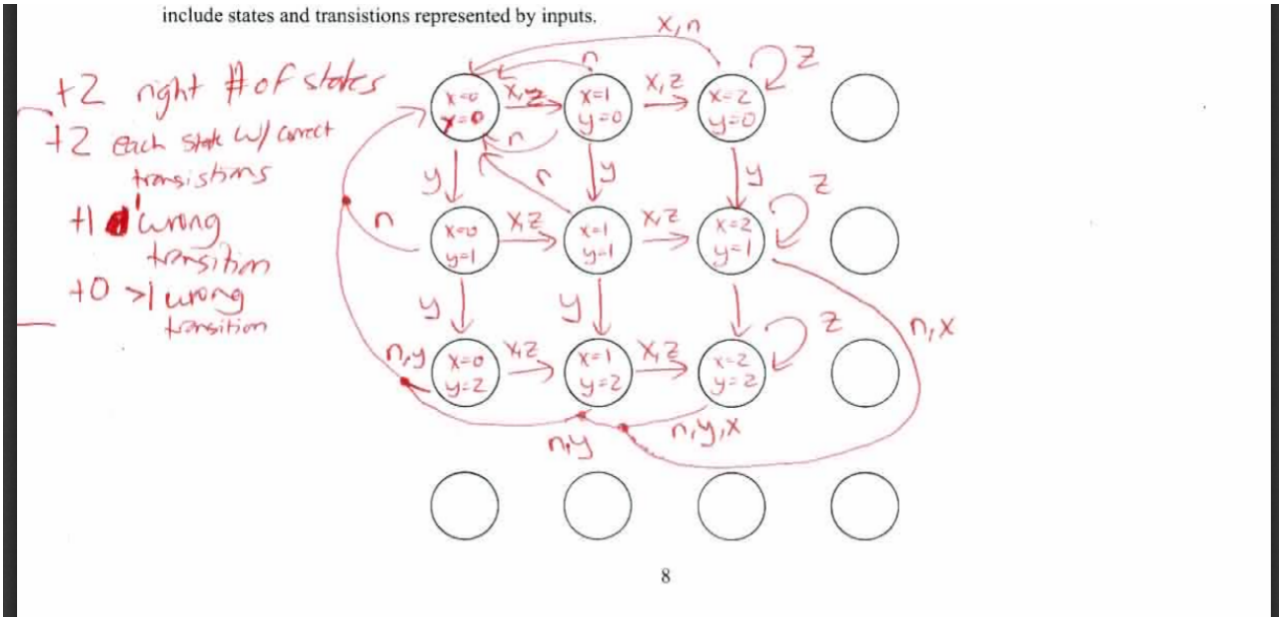
Important rule: If the number of X's or Y's reaches a count of 3, the current round is terminated and another round is started. When a round starts, its state is X=0, Y=0.

Hint: Since the number of X's and Y's specify the state of the current round, how many possible states are needed to describe the state of the current round.

Hint: A state cannot have X=3, because then the round would be finished, and we would have started a *new* current round.

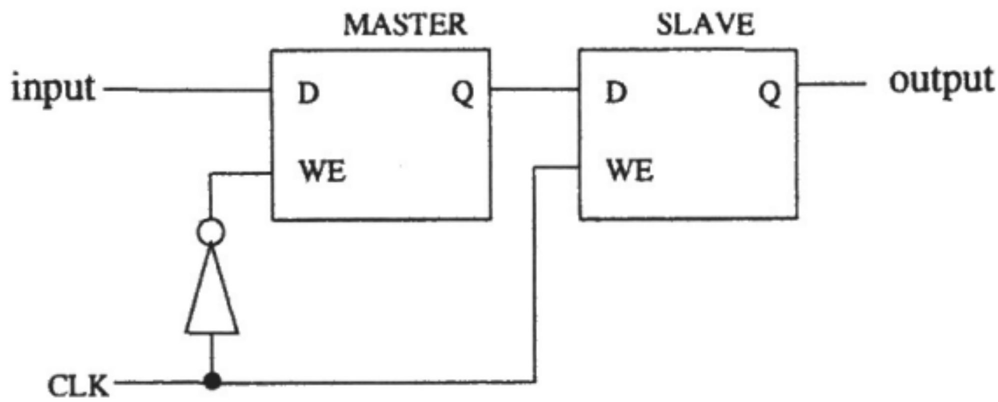
On the diagram below, label each state. For each state draw an arrow showing the transition to the next state that would occur for each of the four inputs. (We have provided sixteen states. You will not need all of them. Use only as many as you need).

Note, we did not specify outputs for these states. Therefore, your state machine will not include outputs. It will only include states and transistions represented by inputs.

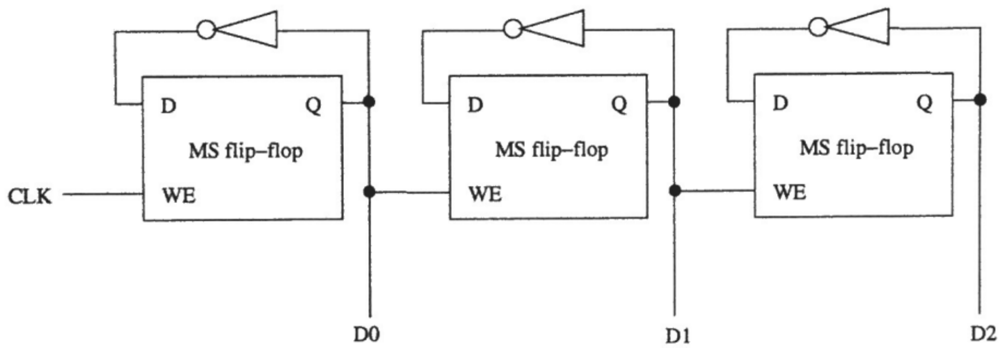


8. *Trying Out Flip-Flops*

The we introduced in class is shown below.



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 flipflops.



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)

	cycle 0	cycle 1	cycle 2	cycle 3	cycle 4	cycle 5	cycle 6	cycle 7
D2	0	1	1	1	1	0	0	0
D1	0	1	1	0	0	1	1	0
D0	0	1	0	1	0	1	0	1

D2: inverts on positiv "edge" of D_1

D1: ← inverts on positive "edge" of D_0

D0: ← inverts on Positive "edge" of clock

"edges" in bold

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

D_2, D_1, D_0 act as a decrementing counter