CMSC 313 Lecture 27

- Last Time: FSM simplification
- DigSim2 Questions??
  - ◇ Place your username on the paper submission
- Registers
- Memory Organization
- DRAM

DigSim Assignment 2: Finite State Machine Simplifications

Due: Tuesday December 9, 2003

Objective

The objective is to design and simplify a moderately complex a finite state machine.

Assignment

For this project you will design and implement in DigSim a Mealy finite state machine with one input bit x and one output bit z. The machine's output z is 1 whenever the input sequence …0111 or …1000 has been detected. The patterns may overlap. For example, if the machine is given input 0111000, the output would be 0001001. [Adapted from *Contemporary Logic Design*, Randy H. Katz, Benjamin/Cummings Publishing, 1994.]

In order to complete the project, you must accomplish the following tasks. Note that for this project, you will submit both your design work (on paper) and the DigSim implementation of the resulting finite state machine (via submit).

1. Draw a transition diagram for the finite state machine described above. Use the state reduction algorithm to minimize the number of states in your machine. Although, you might start with an FSM with as many as 15 states, at the end of the reduction process you should have 7 states in your machine. Name your states A, B, C, D, E, F and G. You may take shortcuts, but you must show your work.
2. Assign bit patterns to each state of the machine. Use the heuristics for state assignment presented in class (available on the course lecture topics web page). Do this step carefully as it will have a large effect on the complexity of your final circuit. *Hint:* follow the loops in the FSM and try to assign bit patterns such that in each step of the loop only one state bit changes. You might not be able to achieve this for *every* step of the loop, but you can try to have only one state bit change for *most* steps of the loop. Write down your final state assignment in the table given.
3. Fill in the state transition table for the finite state machine given below. In this table, s2, s1 and s0 are the state bits, x is the input bit and z is the output bit. Then s2', s1' and s0' are the next states to be stored in the flip flops.
4. Use the Karnaugh maps provided to simplify the Boolean formulas for a finite state machine to be implemented using D flip-flops. You will need Karnaugh maps for s2', s1', s0' and z.
5. Using the excitation table for J-K flip-flops, fill in the columns j2, k2, j1, k1, j0 and k0 in the truth table below. The columns j2 and k2 represent the settings to the inputs of the J-K flip-flop that will cause it to store the state bit s2'. The columns j1, k1, j0 and k0 are analogous for the J-K flip-flops used to store state bits s1 and s0.
6. Use the Karnaugh maps provided to simplify the Boolean formulas for the J and K inputs to each J-K flip flop. You will need Karnaugh maps for j2, k2, j1, k1, j0 and k0. The formula for the output z will be the same as in Step 4.
7. Decide whether you want to use D flip-flops or J-K flip-flops. Briefly justify your choice.
8. Implement the resulting circuit in DigSim. Hint*:* it is possible to implement this FSM using fewer than 14 gates. If your circuit requires many more gates, redo the simplification steps.

What to submit

In class on Tuesday December 9, turn in your finite state diagram, truth table, Karnaugh maps and the resulting Boolean formulas on paper. Make copies of these, if you still need them to implement your circuit in DigSim.

Save your circuit as you did in DigSim Assignment 1. Submit the circuit file using the Unix submit command as in previous assignments. The submission name for this assignment is: `digsim2`.

CMSC 313 Digsim Exercise 2

Name: ________________________________

Minimized 7-State Transition Diagram (show work)

State Assignment:

A	
B	
C	
D	
E	
F	
G	
unused	

Excitation Table for J-K Flip-Flops

Q	Q'	J	K
0	0	0	*d*
0	1	1	*d*
1	0	*d*	1
1	1	*d*	0

Truth Table:

	s2	s1	s0	x	s2'	s1'	s0'	z	j2	k2	j1	k1	j0	k0
0	0	0	0	0										
1	0	0	0	1										
2	0	0	1	0										
3	0	0	1	1										
4	0	1	0	0										
5	0	1	0	1										
6	0	1	1	0										
7	0	1	1	1										
8	1	0	0	0										
9	1	0	0	1										
10	1	0	1	0										
11	1	0	1	1										
12	1	1	0	0										
13	1	1	0	1										
14	1	1	1	0										
15	1	1	1	1										

Question:

Should you use D flip-flops or J-K flip-flops to implement this circuit? Why?

Karnaugh Maps for D Flip-Flops and the output

	s2 s1			s2		
	s0 x	00	01	11	10	
	00	0	4	12	8	
	01	1	5	13	9	x
s0	11	3	7	15	11	
	10	2	6	14	10	
			s1			

s2’ =

	s2 s1			s2		
	s0 x	00	01	11	10	
	00	0	4	12	8	
	01	1	5	13	9	x
s0	11	3	7	15	11	
	10	2	6	14	10	
			s1			

s1’ =

	s2 s1			s2		
	s0 x	00	01	11	10	
	00	0	4	12	8	
	01	1	5	13	9	x
s0	11	3	7	15	11	
	10	2	6	14	10	
			s1			

s0’ =

	s2 s1			s2		
	s0 x	00	01	11	10	
	00	0	4	12	8	
	01	1	5	13	9	x
s0	11	3	7	15	11	
	10	2	6	14	10	
			s1			

z =

Karnaugh Maps for J-K Flip-Flops

				s2		
	s2 s1 s0 x	00	01	11	10	
	00	0	4	12	8	
	01	1	5	13	9	x
s0	11	3	7	15	11	
	10	2	6	14	10	
			s1			

j2 =

				s2		
	s2 s1 s0 x	00	01	11	10	
	00	0	4	12	8	
	01	1	5	13	9	x
s0	11	3	7	15	11	
	10	2	6	14	10	
			s1			

k2 =

				s2		
	s2 s1 s0 x	00	01	11	10	
	00	0	4	12	8	
	01	1	5	13	9	x
s0	11	3	7	15	11	
	10	2	6	14	10	
			s1			

j1 =

				s2		
	s2 s1 s0 x	00	01	11	10	
	00	0	4	12	8	
	01	1	5	13	9	x
s0	11	3	7	15	11	
	10	2	6	14	10	
			s1			

k1 =

				s2		
	s2 s1 s0 x	00	01	11	10	
	00	0	4	12	8	
	01	1	5	13	9	x
s0	11	3	7	15	11	
	10	2	6	14	10	
			s1			

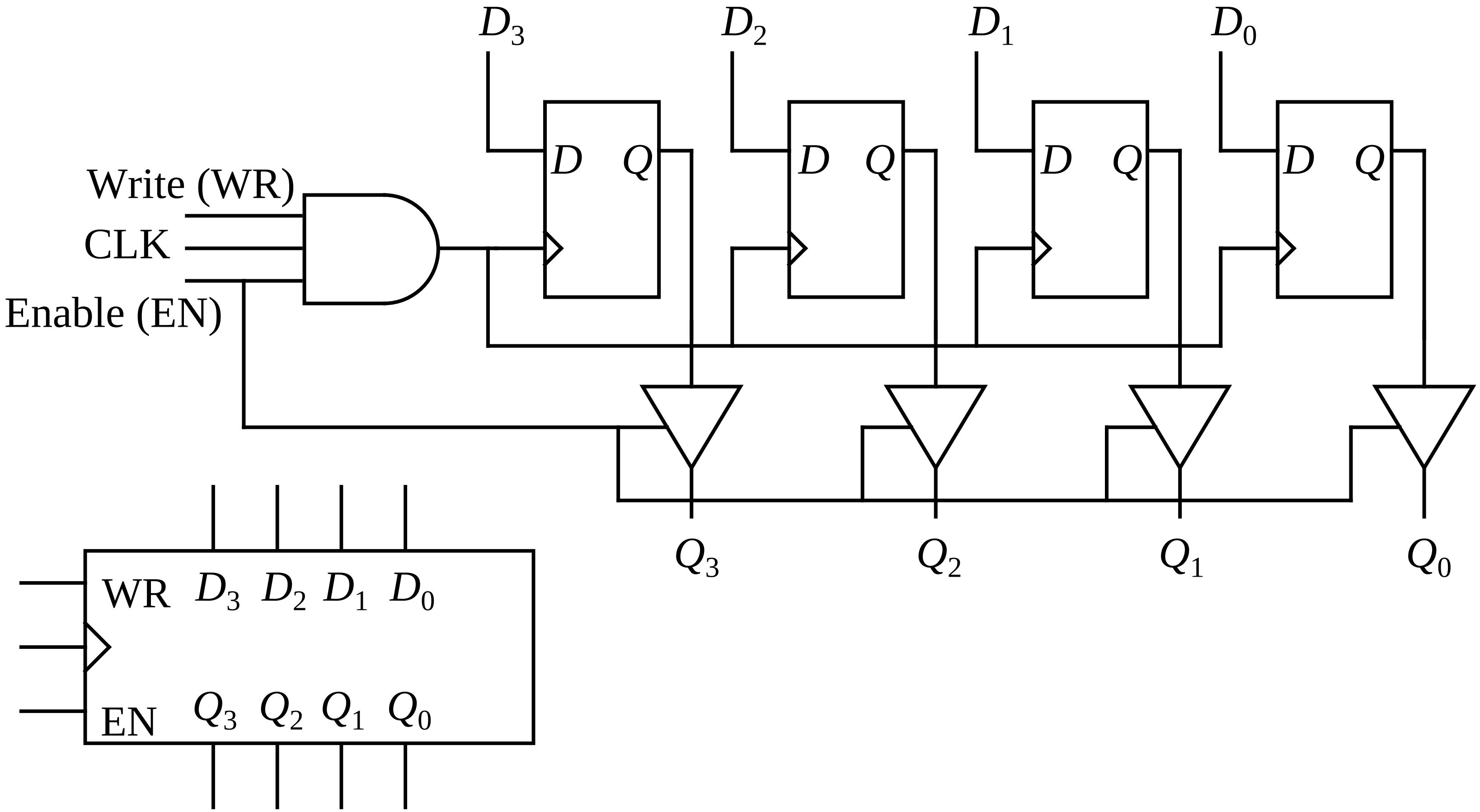
j0 =

				s2		
	s2 s1 s0 x	00	01	11	10	
	00	0	4	12	8	
	01	1	5	13	9	x
s0	11	3	7	15	11	
	10	2	6	14	10	
			s1			

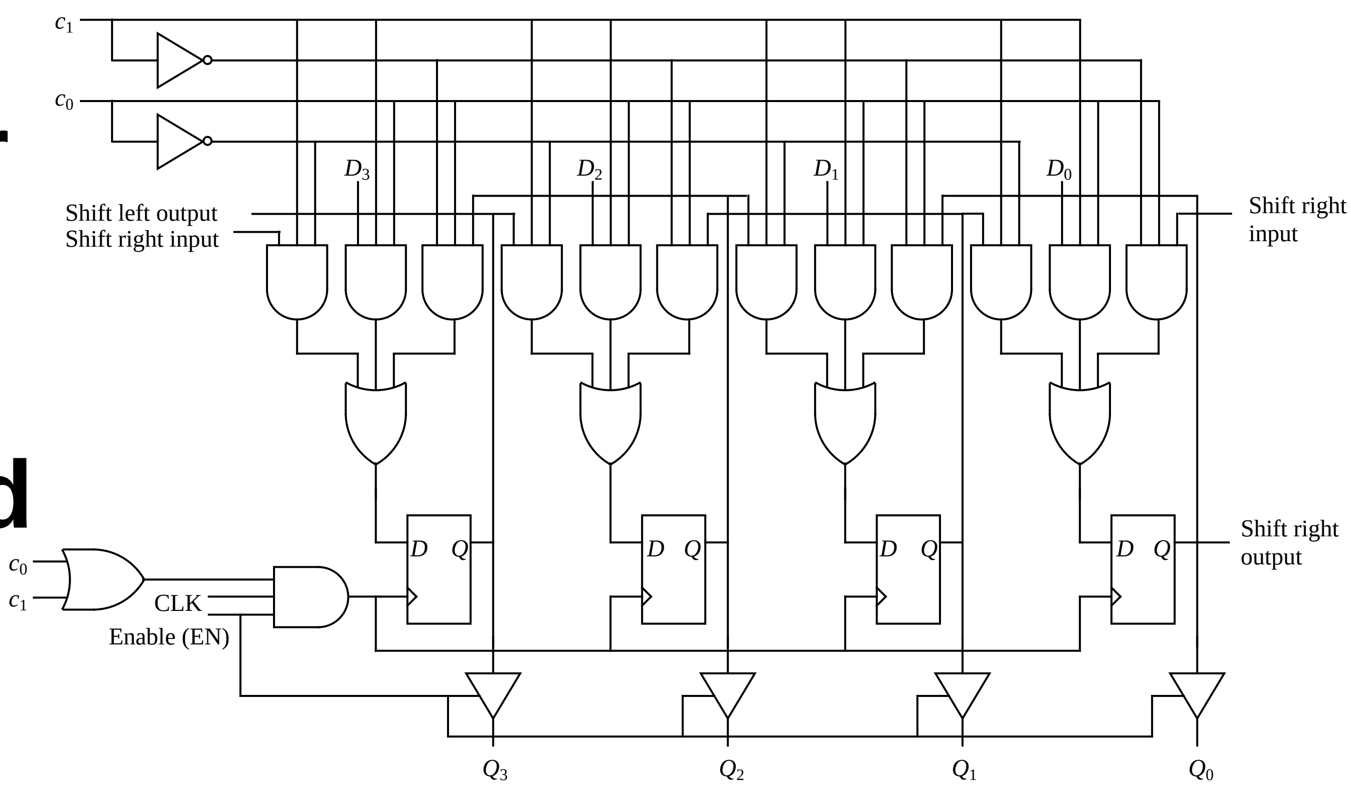
k0 =

Four-Bit Register

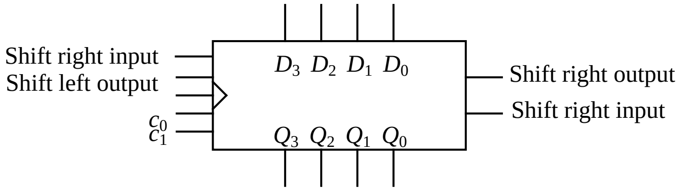
- Makes use of tri-state buffers so that multiple registers can gang their outputs to common output lines.



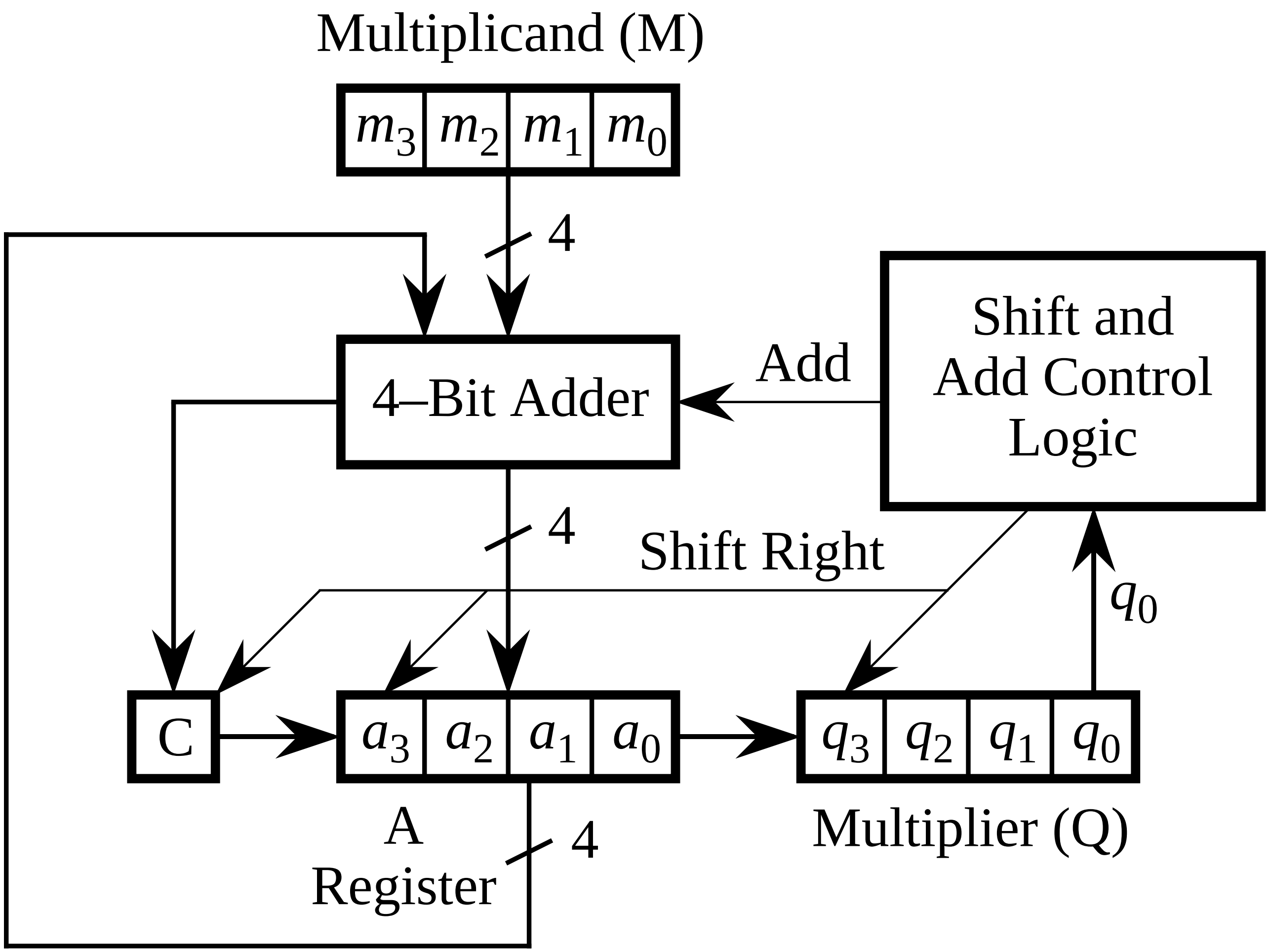
Left-Right Shift Register with Parallel Read and Write



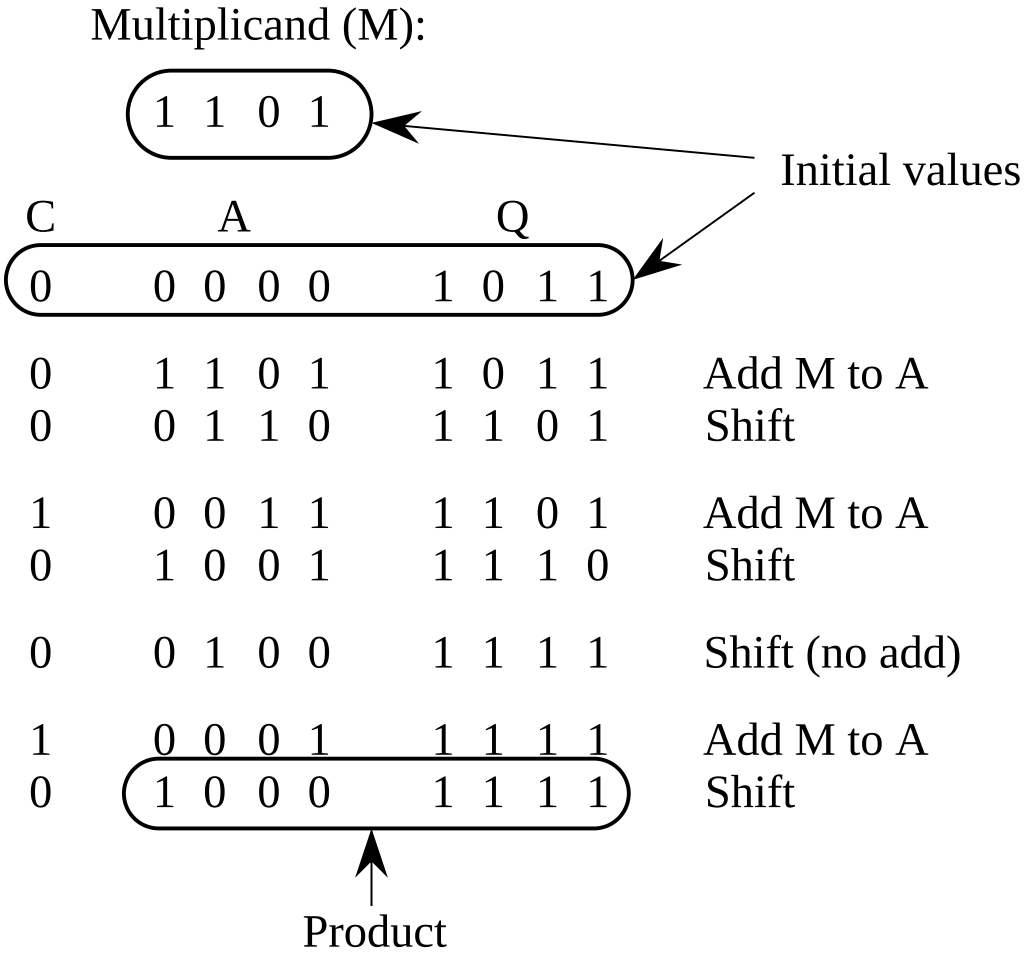
Control c_1 c_0		Function
0	0	No change
0	1	Shift left
1	0	Shift right
1	1	Parallel load



A Serial Multiplier



Example of Multiplication Using Serial Multiplier



Multiplicand (M):

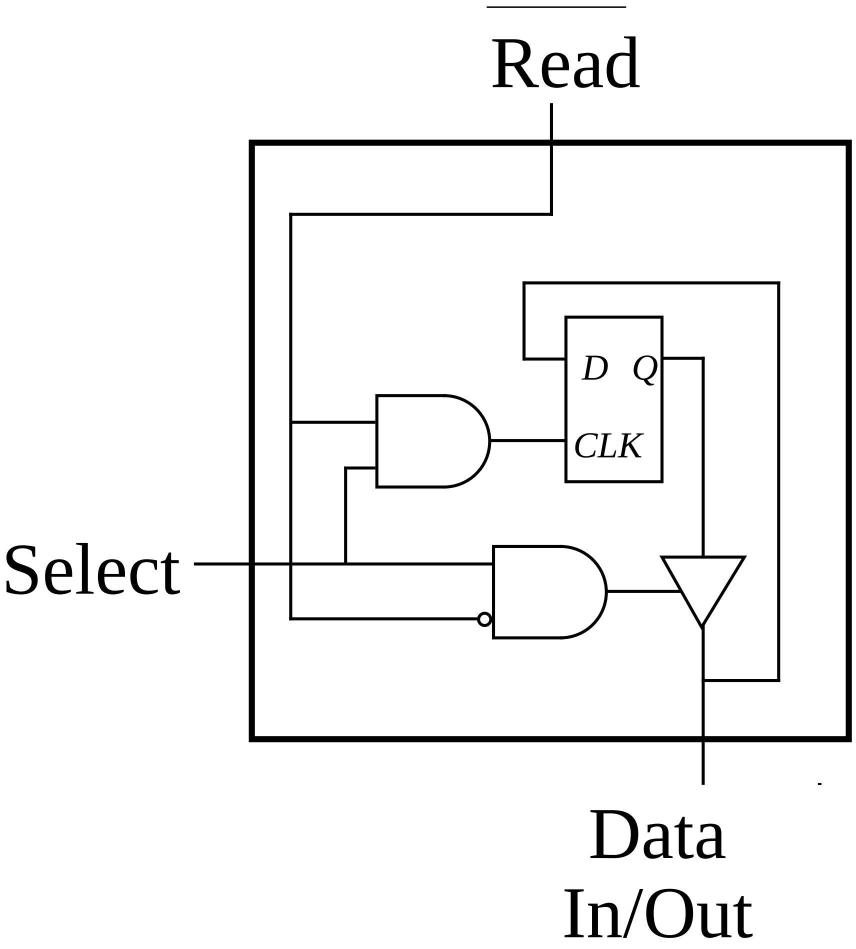
1 1 0 1

Initial values

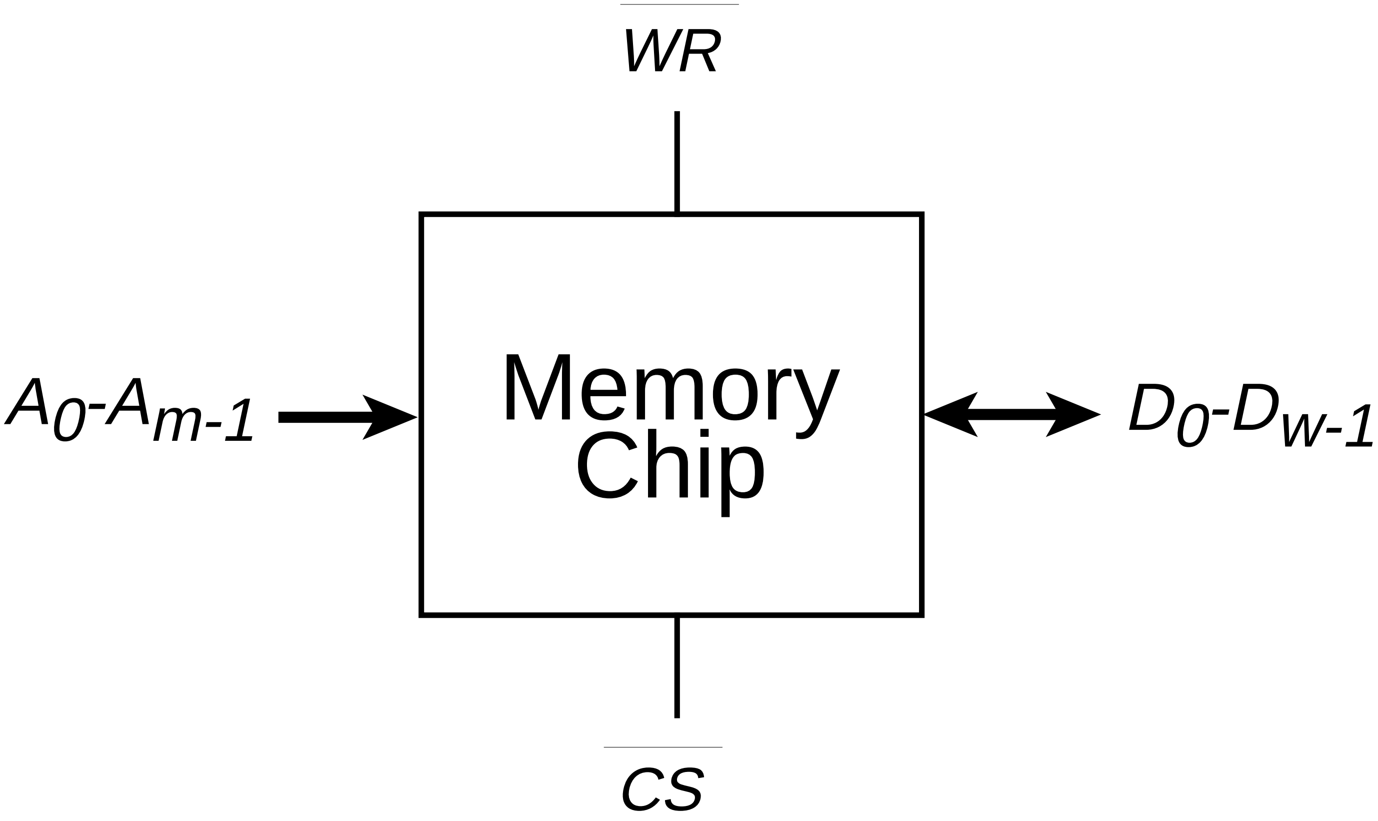
C	A	Q	
0	0 0 0 0	1 0 1 1	
0	1 1 0 1	1 0 1 1	Add M to A
0	0 1 1 0	1 1 0 1	Shift
1	0 0 1 1	1 1 0 1	Add M to A
0	1 0 0 1	1 1 1 0	Shift
0	0 1 0 0	1 1 1 1	Shift (no add)
1	0 0 0 1	1 1 1 1	Add M to A
0	1 0 0 0	1 1 1 1	Shift

Product

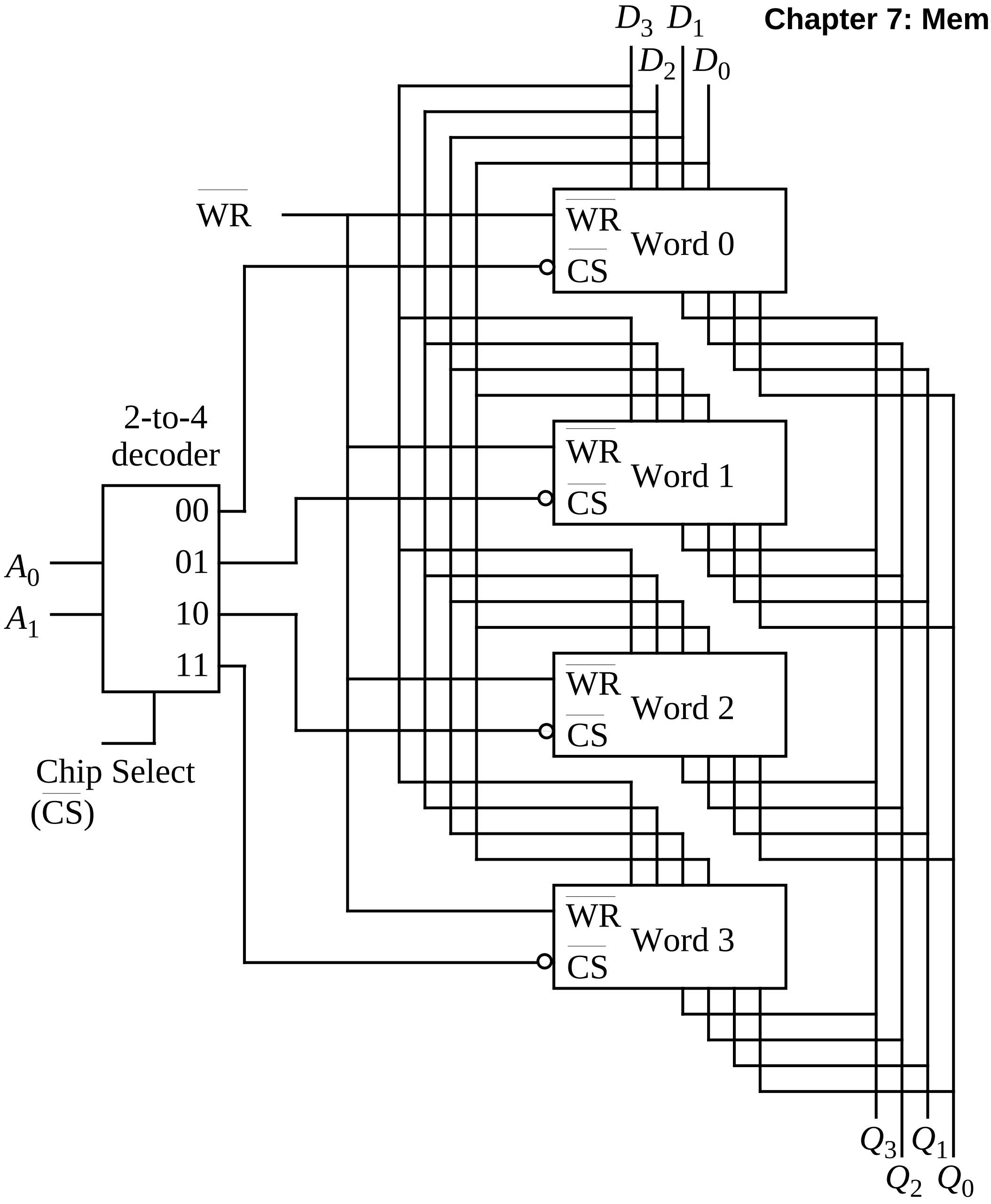
Functional Behavior of a RAM Cell



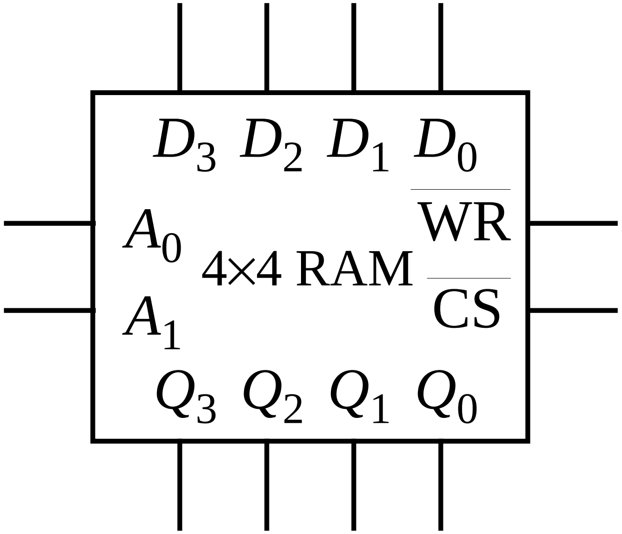
Simplified RAM Chip Pinout



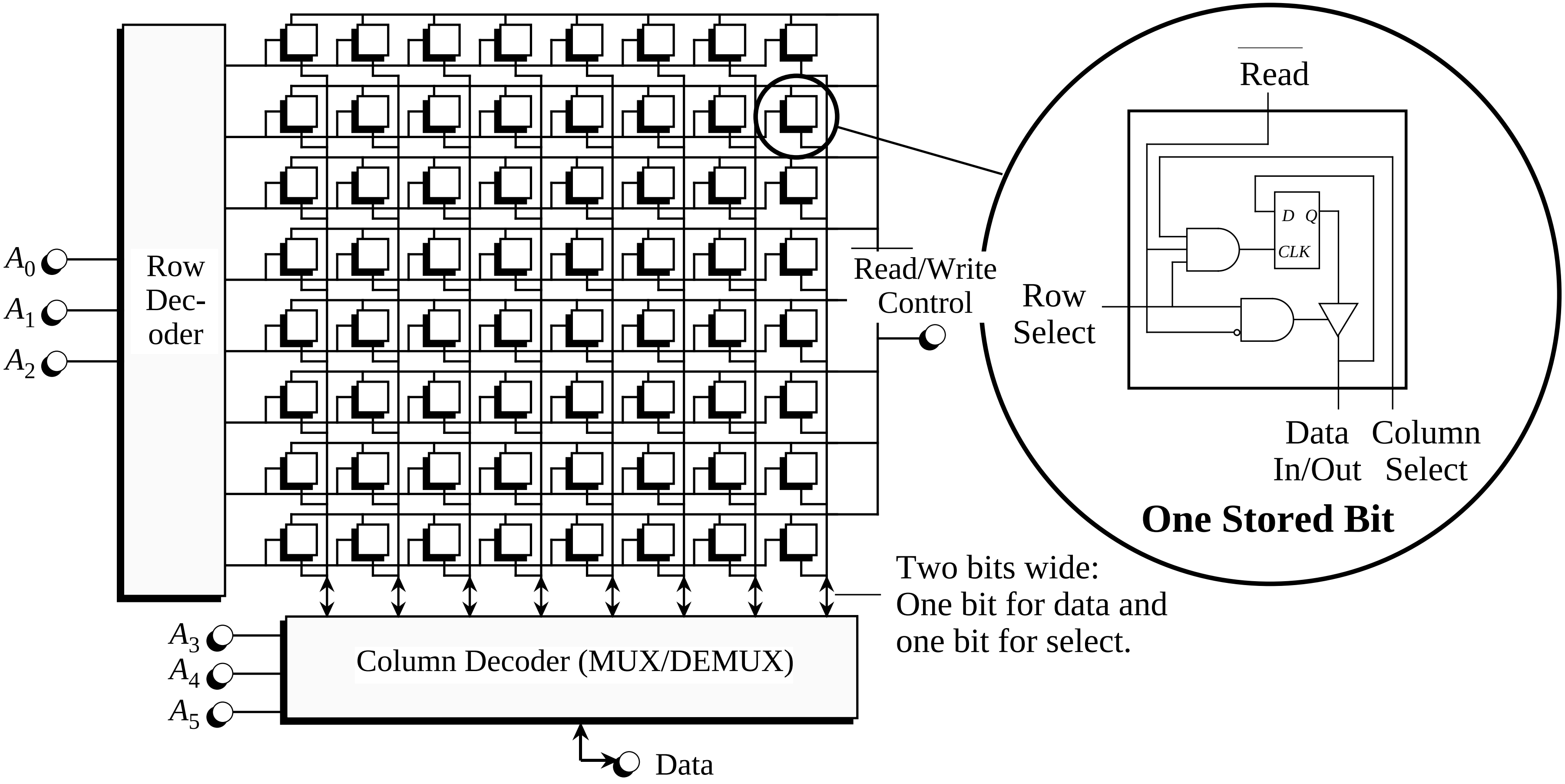
A Four-Word Memory with Four Bits per Word in a 2D Organization


D_3 D_1
D_2 D_0
$\overline{WR}$
2-to-4 decoder
00
01
10
11
A_0
A_1
Chip Select ($\overline{CS}$)
$\overline{WR}$ $\overline{CS}$ Word 0
$\overline{WR}$ $\overline{CS}$ Word 1
$\overline{WR}$ $\overline{CS}$ Word 2
$\overline{WR}$ $\overline{CS}$ Word 3
Q_3 Q_1
Q_2 Q_0

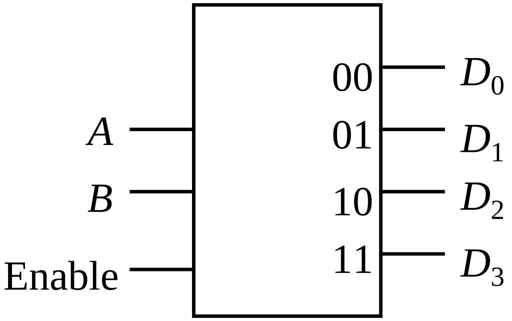
A Simplified Representation of the Four-Word by Four-Bit RAM



2-1/2D Organization of a 64-Word by One-Bit RAM



Decoder

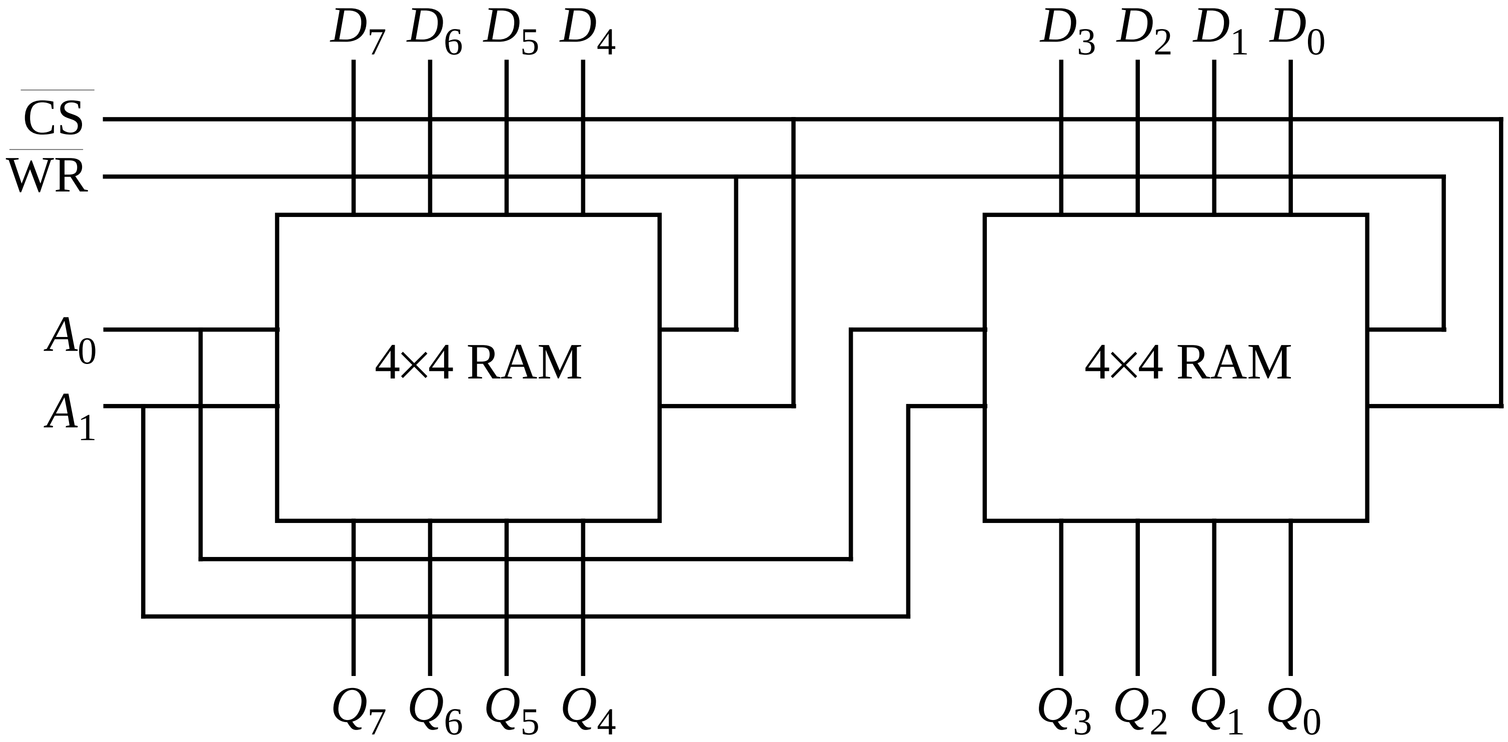


Enable = 1					
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

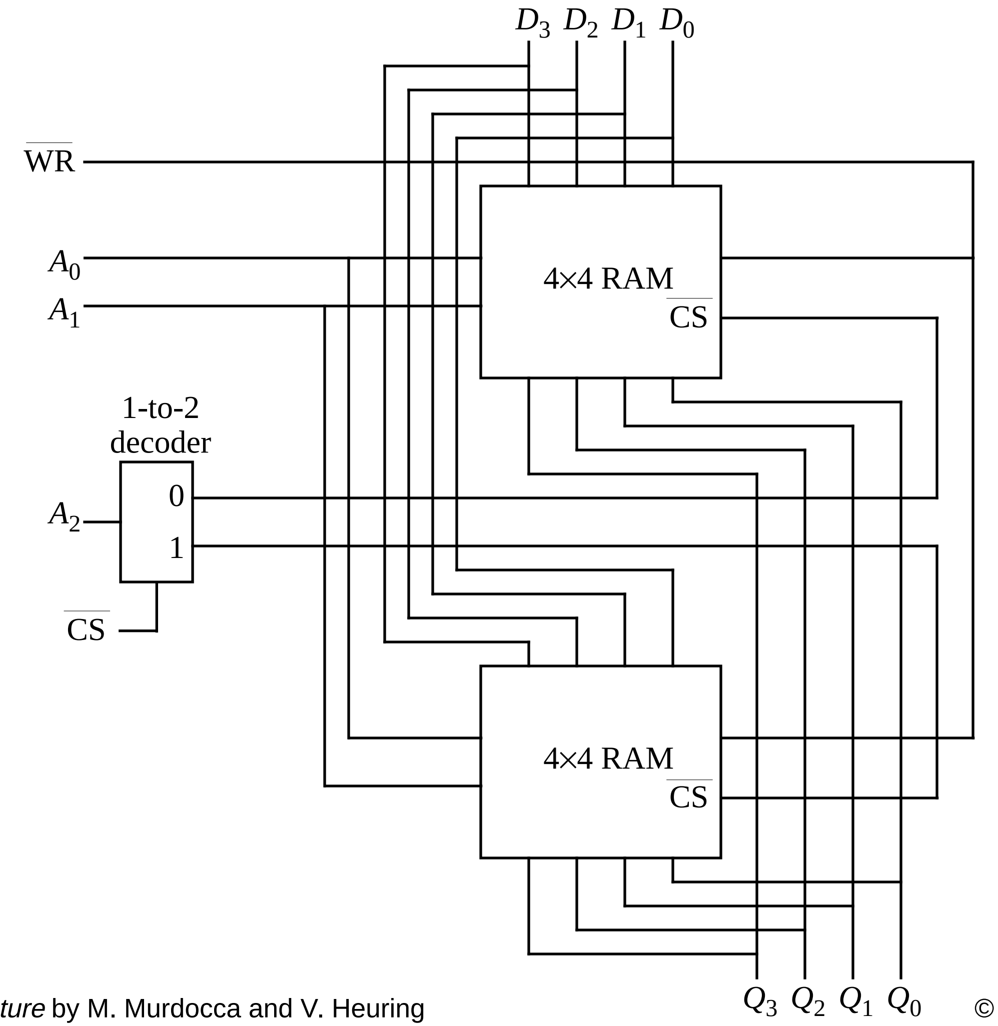
Enable = 0					
A	B	D_0	D_1	D_2	D_3
0	0	0	0	0	0
0	1	0	0	0	0
1	0	0	0	0	0
1	1	0	0	0	0

$D_0 = \overline{A}\,\overline{B}$ $D_1 = \overline{A}\,B$ $D_2 = A\,\overline{B}$ $D_3 = A\,B$

Two Four-Word by Four-Bit RAMs are Used in Creating a Four-Word by Eight-Bit RAM



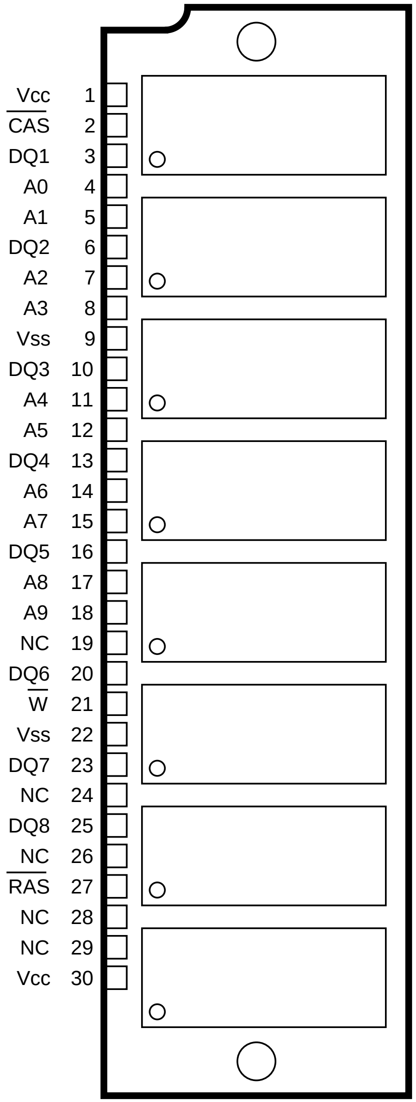
Two Four-Word by Four-Bit RAMs Make up an Eight-Word by Four-Bit RAM



Single-In-Line Memory Module

- **Adapted from(Texas Instruments, *MOS Memory: Commercial and Military Specifications Data Book*, Texas Instruments, Literature Response Center, P.O. Box 172228, Denver, Colorado, 1991.)**

PIN NOMENCLATURE	
A0-A9	Address Inputs
$\overline{\text{CAS}}$	Column-Address Strobe
DQ1-DQ8	Data In/Data Out
NC	No Connection
$\overline{\text{RAS}}$	Row-Address Strobe
V_{cc}	5-V Supply
V_{ss}	Ground
$\overline{\text{W}}$	Write Enable

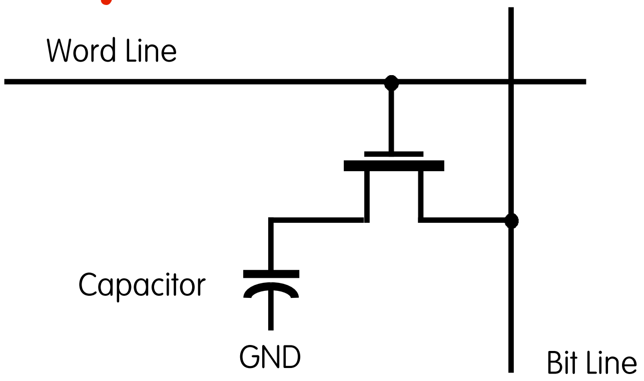


Types of Random Access Memory

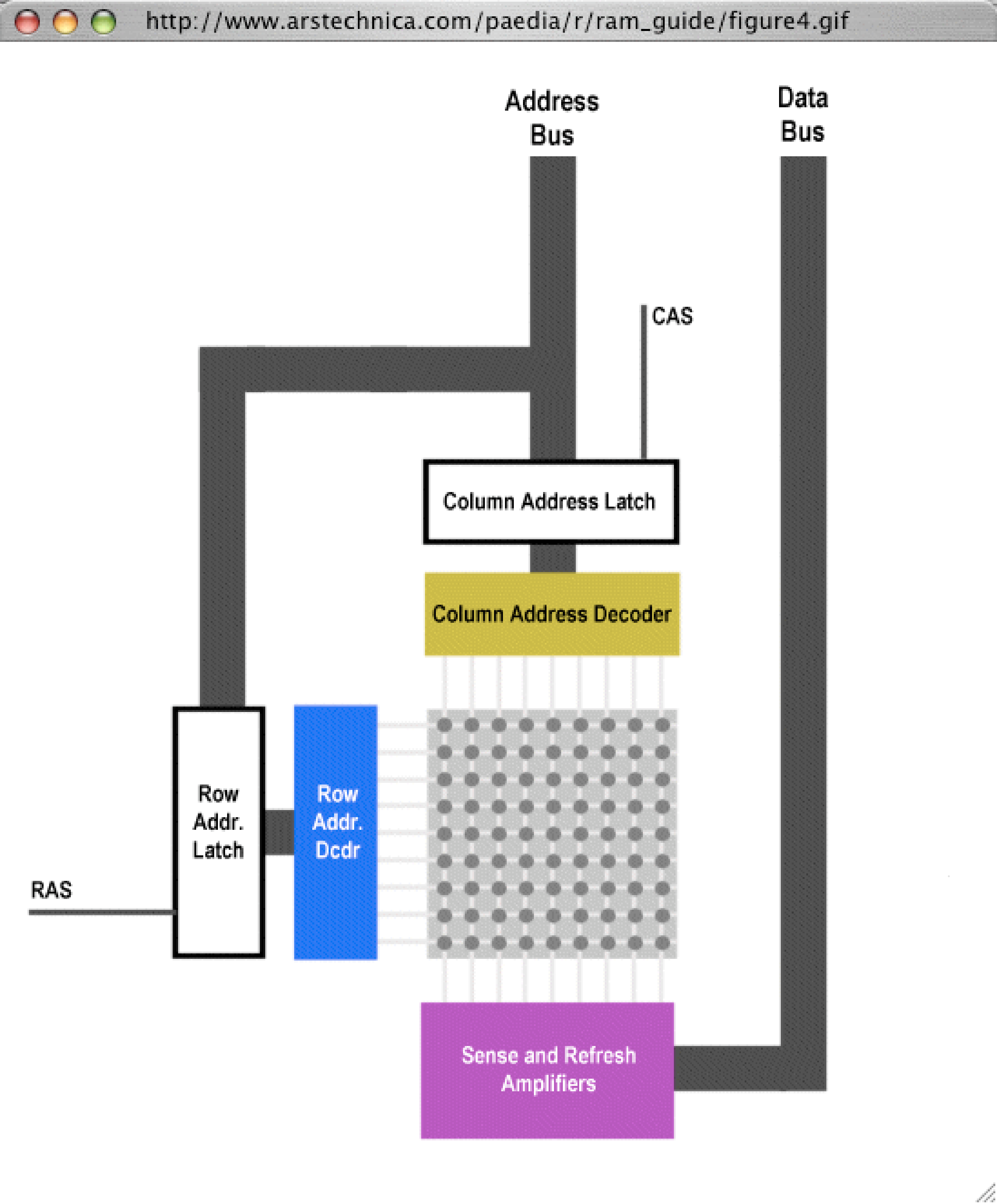
- **Static RAM (SRAM)**
    - ◇ Each bit is stored in a type of flip-flop
    - ◇ Typically takes four or six transistors per bit
    - ◇ Faster, but takes up more space in a chip
    - ◇ Retains information as long as power is supplied
    - ◇ Not to be confused with flash memory in digital cameras (EEPROMs)
- **Dynamic RAM (DRAM)**
    - ◇ Each bit is stored in a capacitor
    - ◇ Uses one capacitor and one transistor per bit
    - ◇ Slower, but takes up less space in a chip
    - ◇ Must be refreshed periodically (milliseconds), since the capacitor leaks

DRAM

- A DRAM memory cell



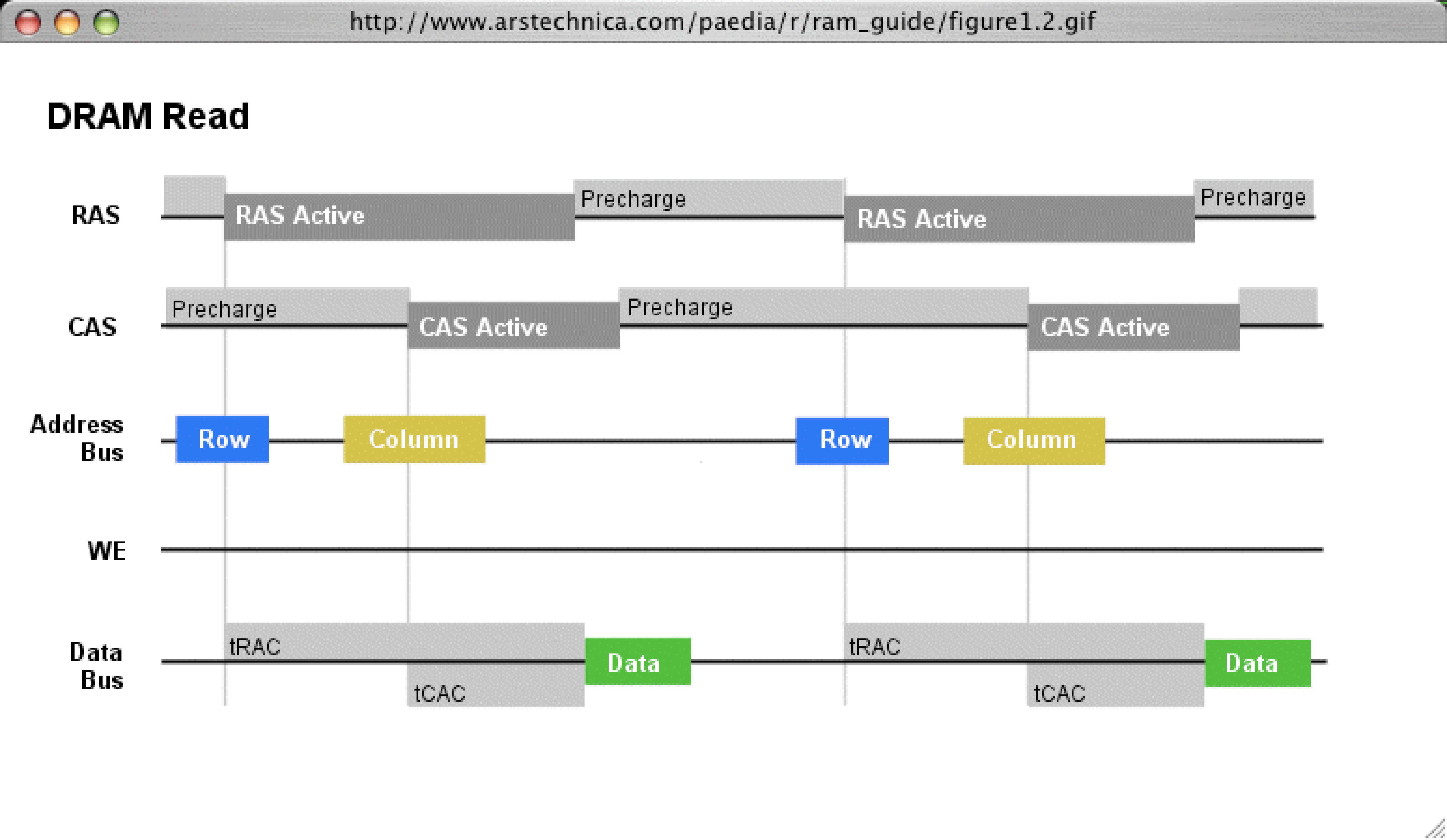
- Word line selects cell for reading or writing
- To write, the bit line is charged with logic 1 or 0
- To read, sensitive amplifier circuits detect small changes in bit line.
- Reading discharges the capacitor.



http://www.arstechnica.com/paedia/r/ram_guide/figure4.gif
Address Bus
Data Bus
CAS
Column Address Latch
Column Address Decoder
Row Addr. Latch
Row Addr. Dcdr
RAS
Sense and Refresh Amplifiers

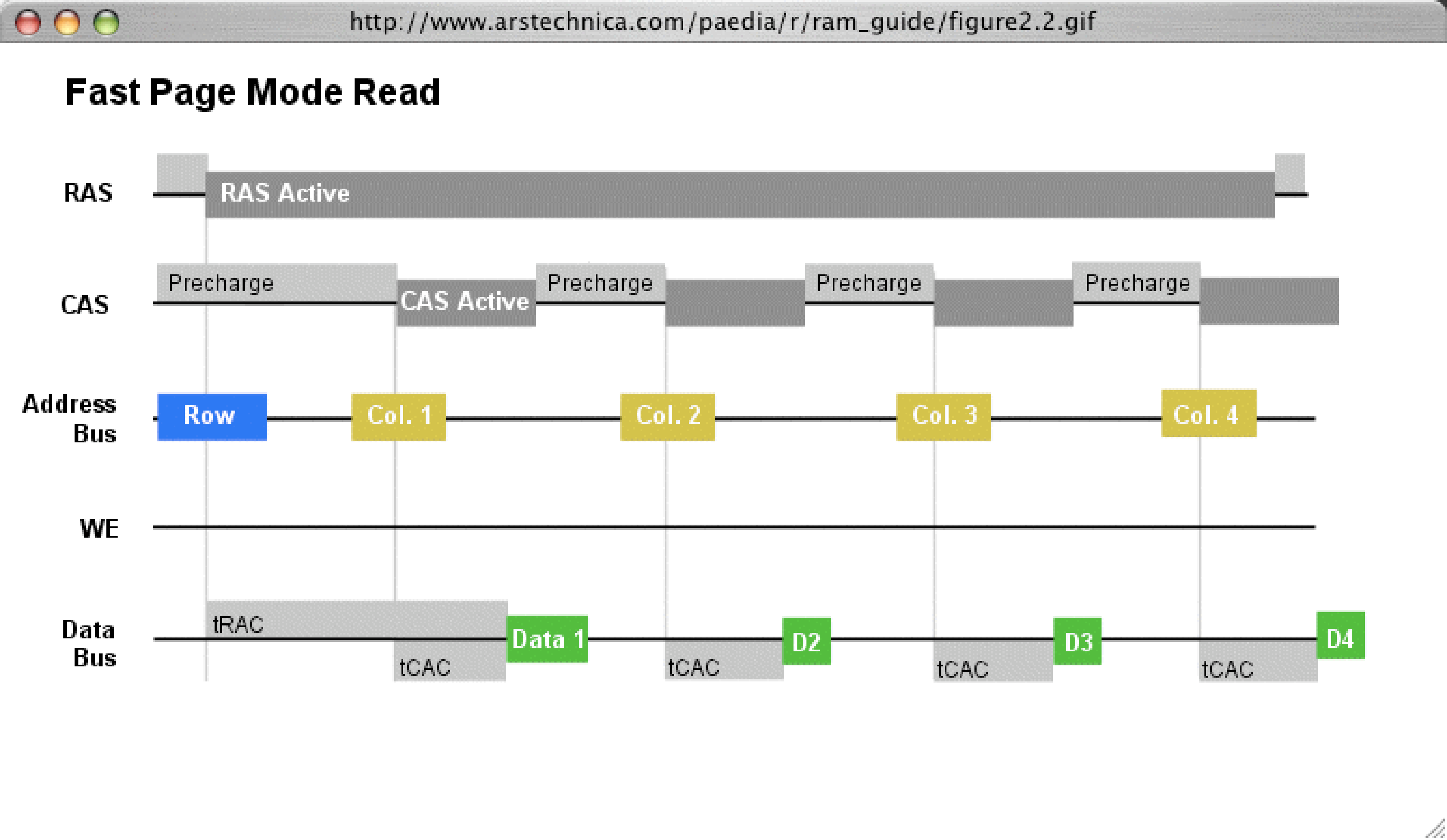
DRAM Read Cycle

1. Row address placed on the address bus.
2. Row Address Strobe (RAS) is asserted, allowing the row address to latch.
3. Row address decoder selects proper row.
4. Write Enable (WE) disabled.
5. Column address placed on the address bus.
6. Column Address Strobe (CAS) is activated, allowing the column address to latch.
7. Once the CAS signal has stabilized, sensing amplifiers places data from the selected row & column on data bus.
8. RAS and CAS deactivated. Cycle begins again.


http://www.arstechnica.com/paedia/r/ram_guide/figure1.2.gif
DRAM Read
RAS
RAS Active
Precharge
RAS Active
Precharge
CAS
Precharge
CAS Active
Precharge
CAS Active
Address Bus
Row
Column
Row
Column
WE
Data Bus
tRAC
tCAC
Data
tRAC
tCAC
Data

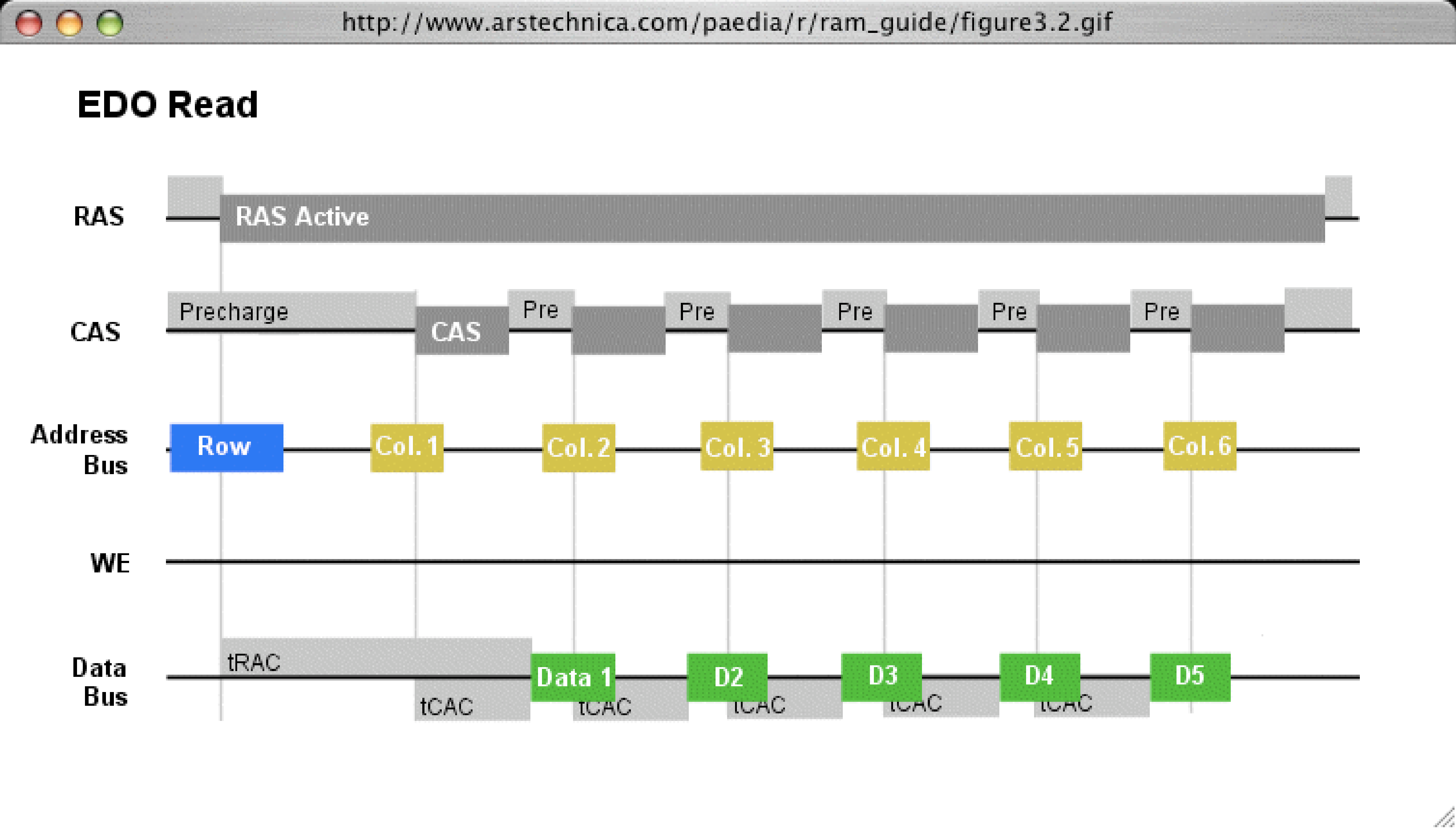
DRAM

- DRAM is asynchronous, ignores system bus clock.
  - ◇ tRAC = Row Access Time = delay from RAS assertion until data is ready
  - ◇ tCAC = Column Access Time = delay from CAS assertion until data is ready
- DRAM access is slooooow
- Each memory access must wait for time it takes to activate and deactivate RAS.
- Fast Page Mode (FPM) DRAM allows successive reads from the same row without deactivating RAS.
- Extended Data Out (EDO) DRAM overlaps CAS assertion and data reads.



http://www.arstechnica.com/paedia/r/ram_guide/figure2.2.gif

Fast Page Mode Read

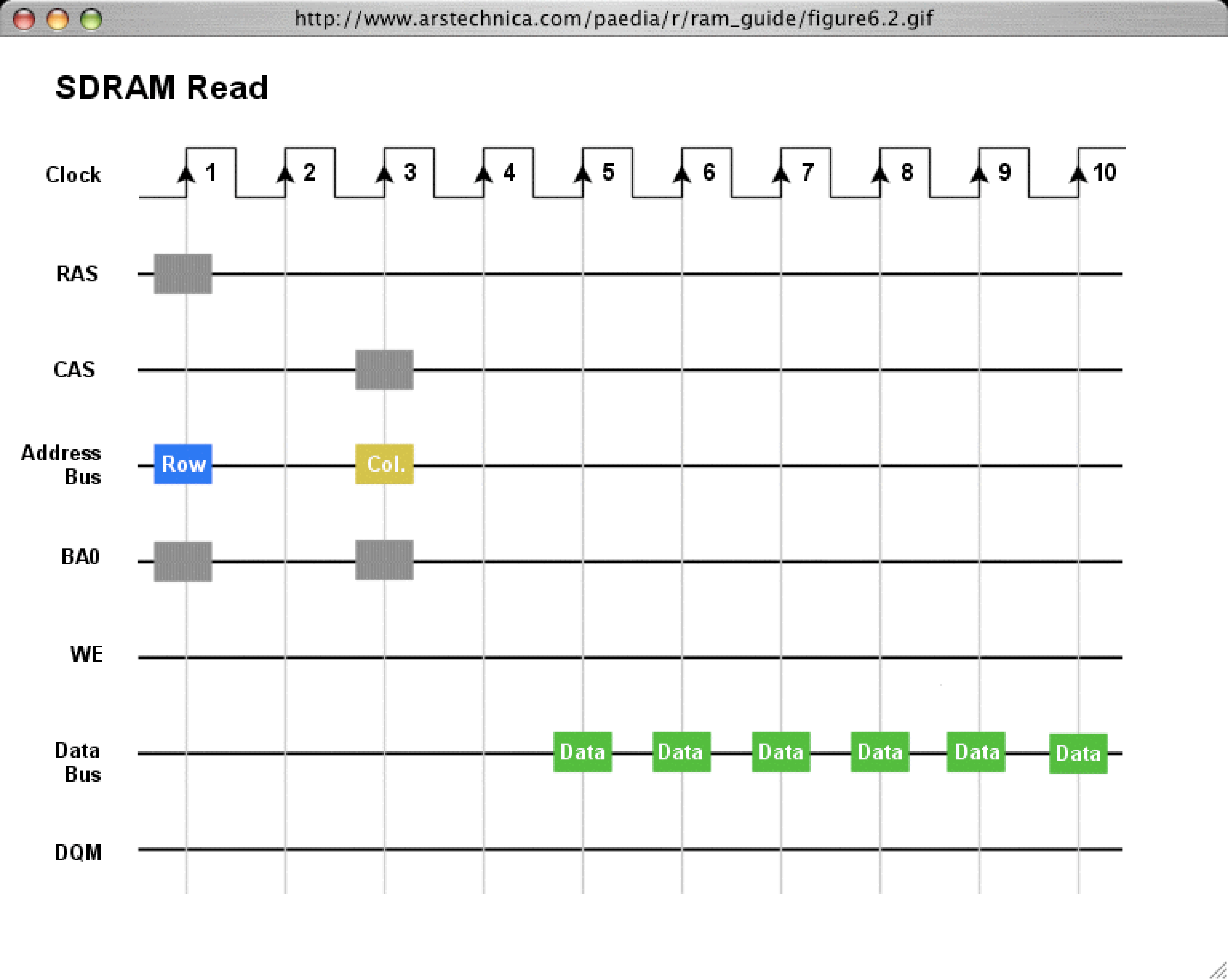


http://www.arstechnica.com/paedia/r/ram_guide/figure3.2.gif

EDO Read

Synchronous DRAM (SDRAM)

- Uses system bus clock.
- Current models run at 433MHz (still much slower than CPU).
- Burst mode allows fast successive reads from the same row. (Good way to read in a cache line!)
- Double Data Rate (DDR) SDRAM provides data on the positive and negative edges of the clock.


http://www.arstechnica.com/paedia/r/ram_guide/figure6.2.gif
SDRAM Read
Clock
1
2
3
4
5
6
7
8
9
10
RAS
CAS
Address Bus
Row
Col.
BA0
WE
Data Bus
Data
Data
Data
Data
Data
Data
DQM

Next Time

- Review of semester
- Final exam review