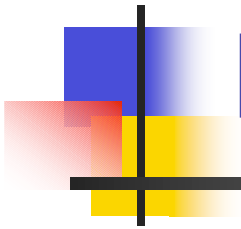
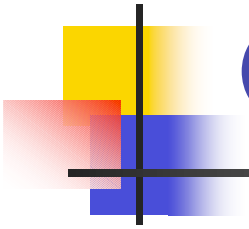


Models of Computation

John E Savage

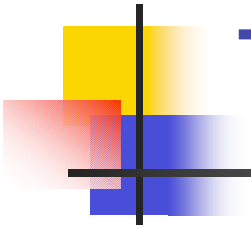
Computer Science
Brown University

CBSSS 2004
July 16, 2004



Overview

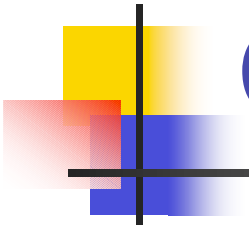
- Descriptions of tasks
- Computers and Computation
- Serial and Parallel Models
  - Gates and circuits
  - Finite-state machines
  - Turing machines
  - Systolic arrays and cellular automata
  - PRAM



Tasks Descriptions

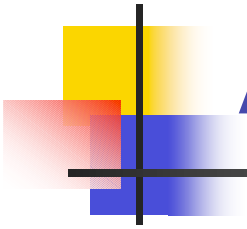
- Procedural – how to do it
- Functional – what is to be done
  - Starting point

- A functional task description has many different procedural implementations.



Computation

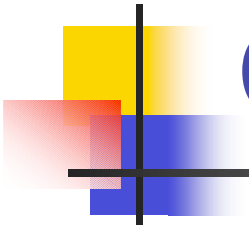
- Determining something by mathematical or logical methods.
  - WordNet Dictionary



A Computer

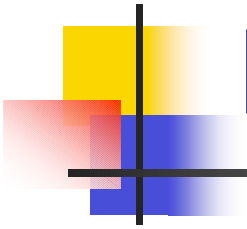
- Physical device that implements a computation.
- Digital computer: inputs, outputs and internal values drawn from finite sets
- Analog computer: inputs, outputs and internal values may assume real values

- We only examine digital computers models.



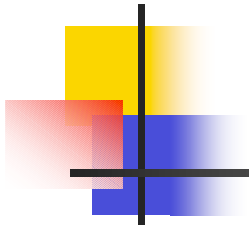
Computational Models

- Abstract essential features of computers
- Memoryless Models – logic circuits
- Models with memory – state-based
  - Bounded memory models
  - Unbounded memory models
- Serial and parallel models



Logic Gate

- Task: to combine truth values
  - AND gate: output is *true* only when both inputs are *true* and otherwise *false*.
  - OR gate: output is *true* when either input is *true* and otherwise *false*.
  - NOT gate: output is *true* when input *false* and vice versa.



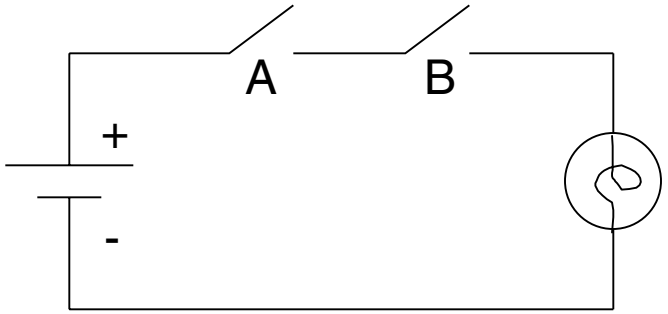
Functional Gate Descriptions

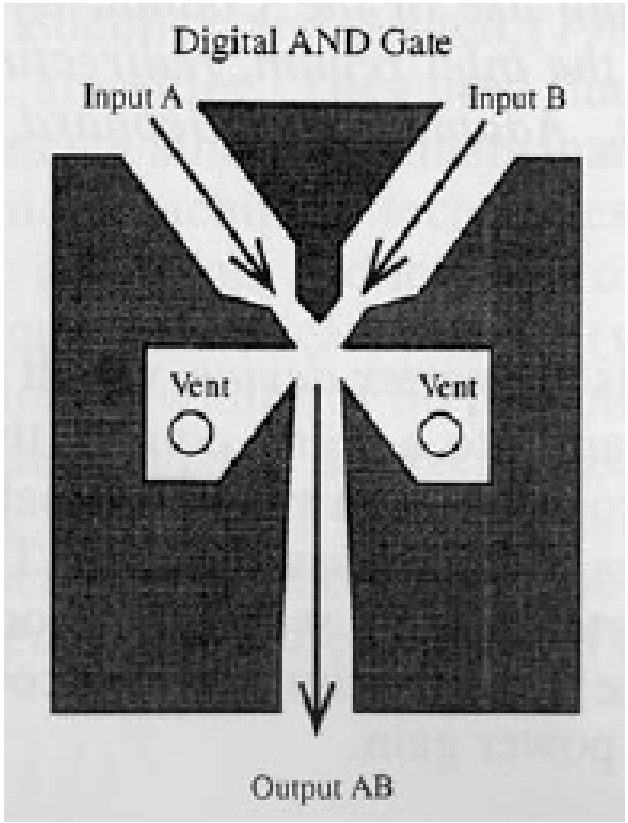
OR AND EXCLUSIVE OR

- How many 2-input Boolean functions are there?

Implementing Logic Gates

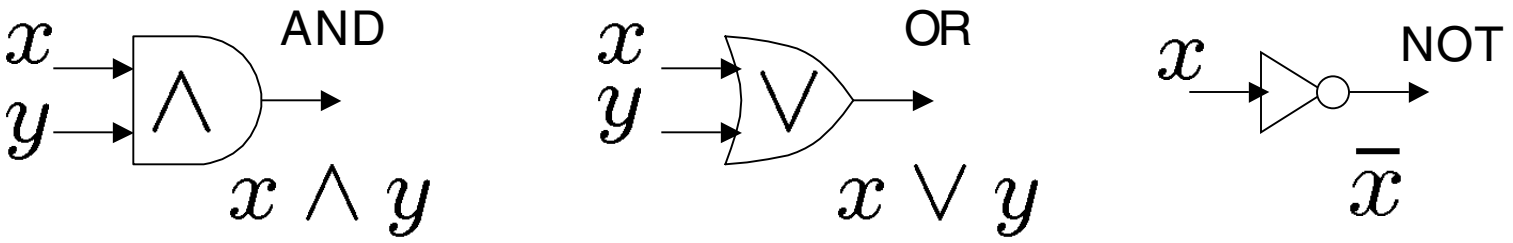
- The AND gate has many realizations.





- How about OR?

Logic Gate Symbols

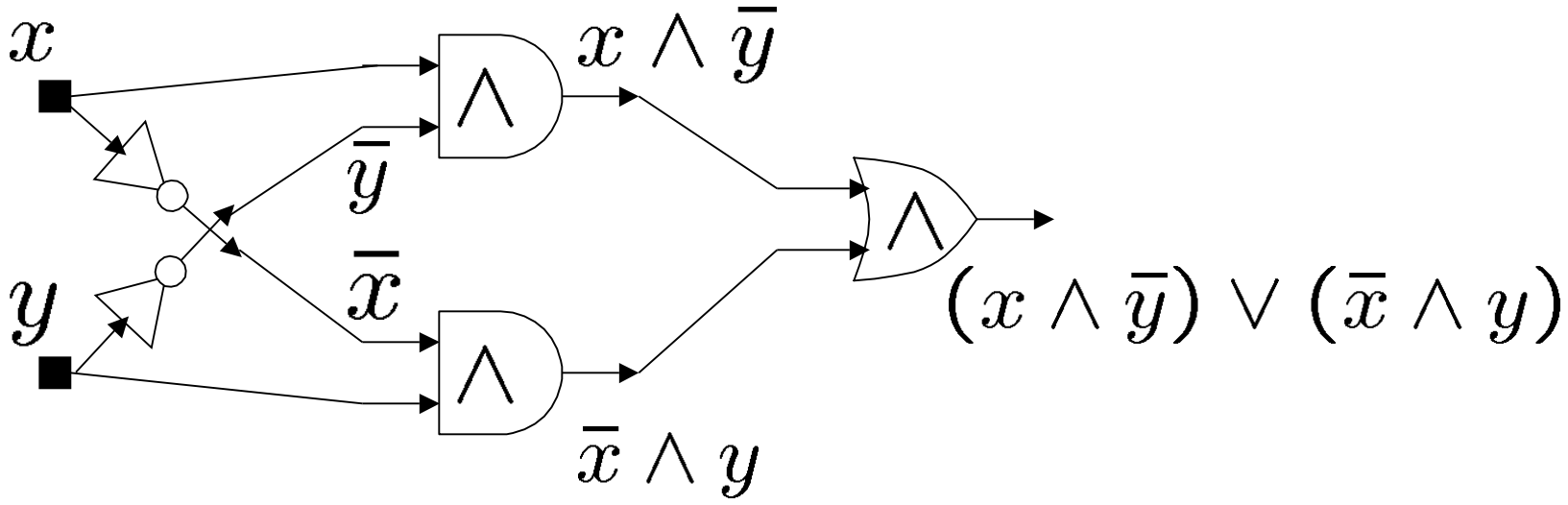


- Transistors are electronic switches that may be used to realize gates.

Logic Circuits

- A **circuit** is a directed acyclic graph in which vertices carry logic gate labels.

EXCLUSIVE OR



Boolean Functions

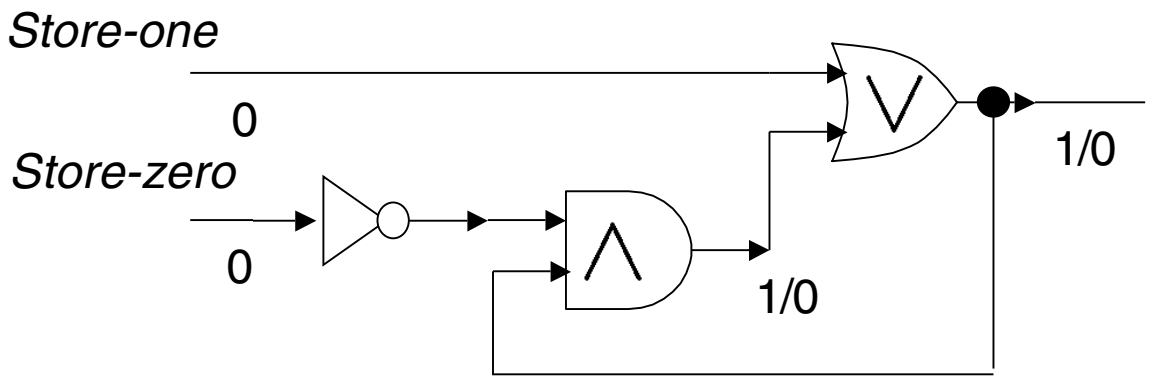
- EXCLUSIVE OR $x \oplus y = (x \wedge \bar{y}) \vee (\bar{x} \wedge y)$

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

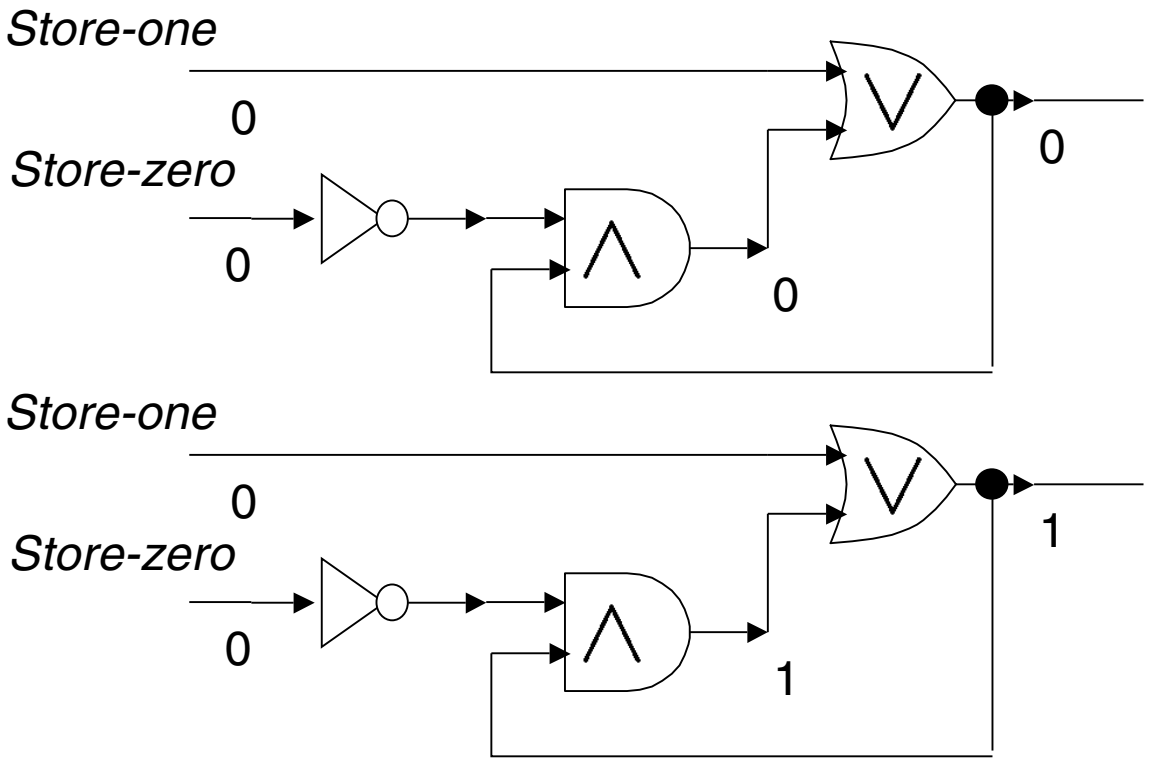
- Each output of each circuit is described as a **boolean function** of the circuit inputs.

Circuits with Memory

- Circuit with feedback holds its state.

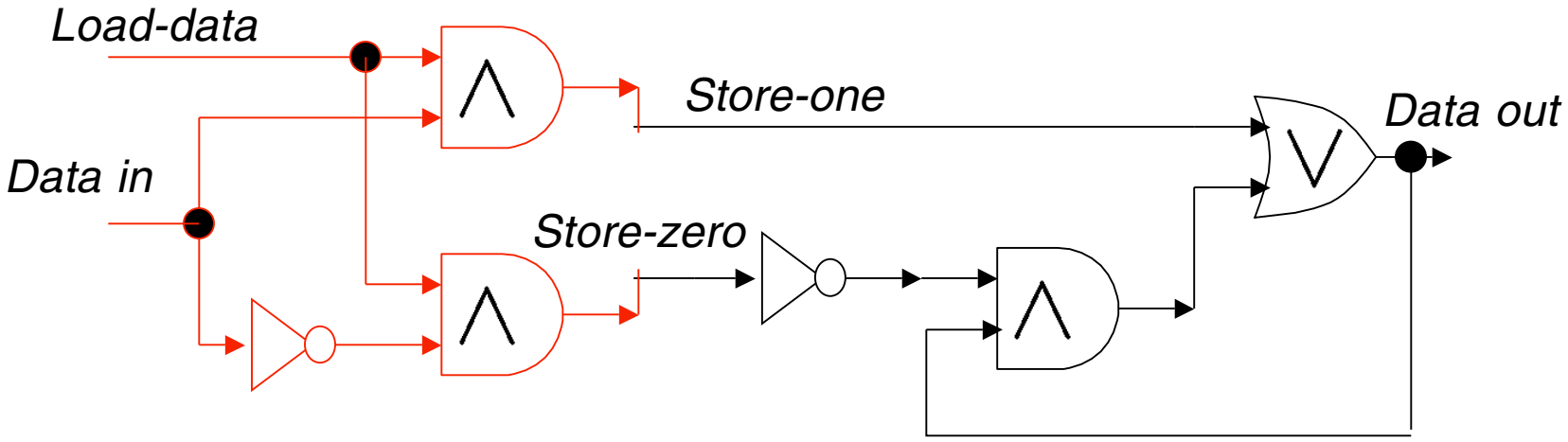


Circuits with Memory



- To change state to 1 (0), apply short 1/0 pulse to *Store-one* (*Store-zero*).

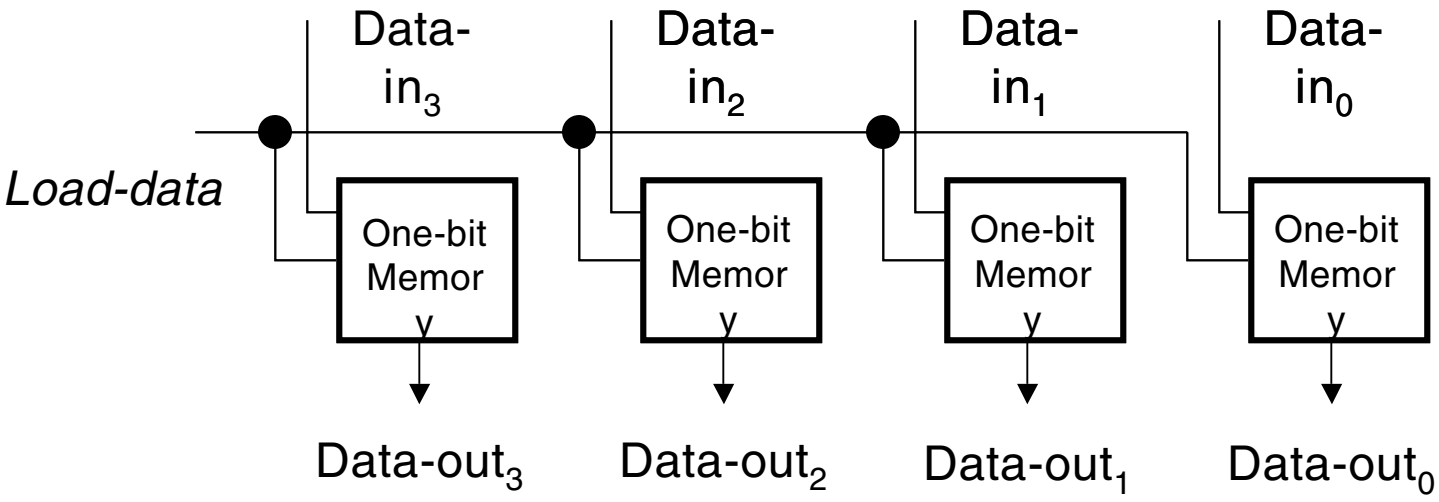
One-bit Clocked Memory



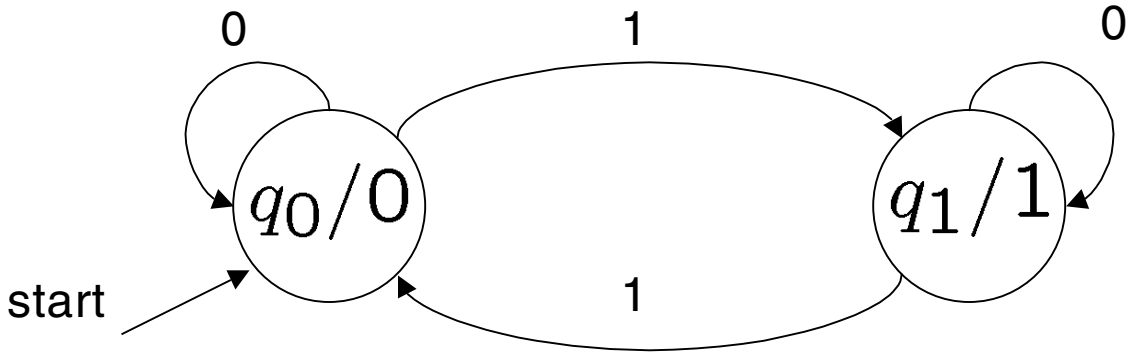
- *Load-data* (clock) must be high long enough for data in feedback loop to become stable.

Registers

- Multiple one-bit clocked memories.



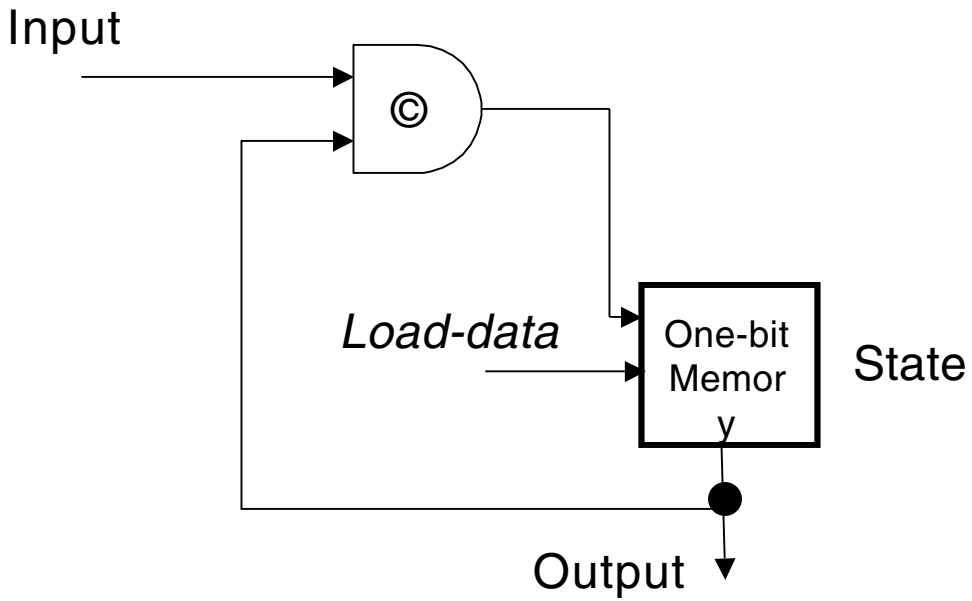
Memory-Based Models – Finite-State Machine (FSM)



"Parity" finite-state machine

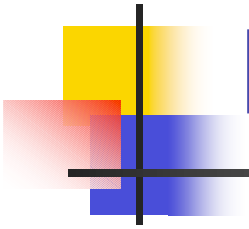
- q_0 is start state.
- $q_1/1$ indicates that output is 1 in q_1
- Arcs are labeled by input symbols.

Implementation of FSM as Clocked Sequential Circuit



- Input must be stable while *Load-data* is high.

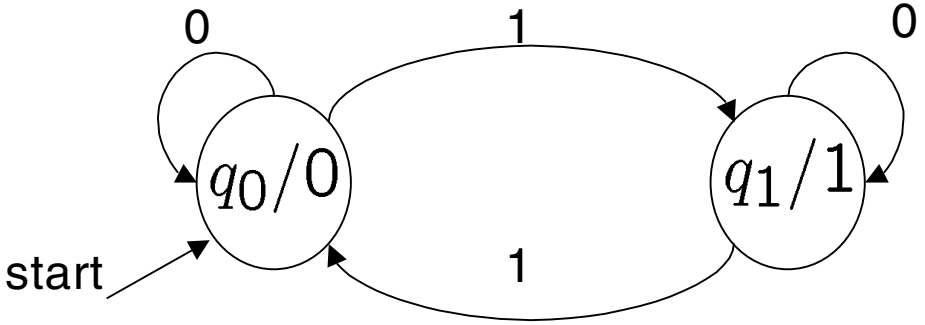
Functional Description of FSMs



- FSM is a 6-tuple $M = (\Omega, \Psi, Q, \delta, \lambda, s)$
  - Ω/Ψ = input/output alphabets,
  - Q is set of states,
  - $\delta : Q \times \Omega \rightarrow Q$ is next-state function,
  - $\lambda : Q \rightarrow \Psi$ is output function.
- Next state determined by current state and input.
- Output determined by current state.

FSM Example

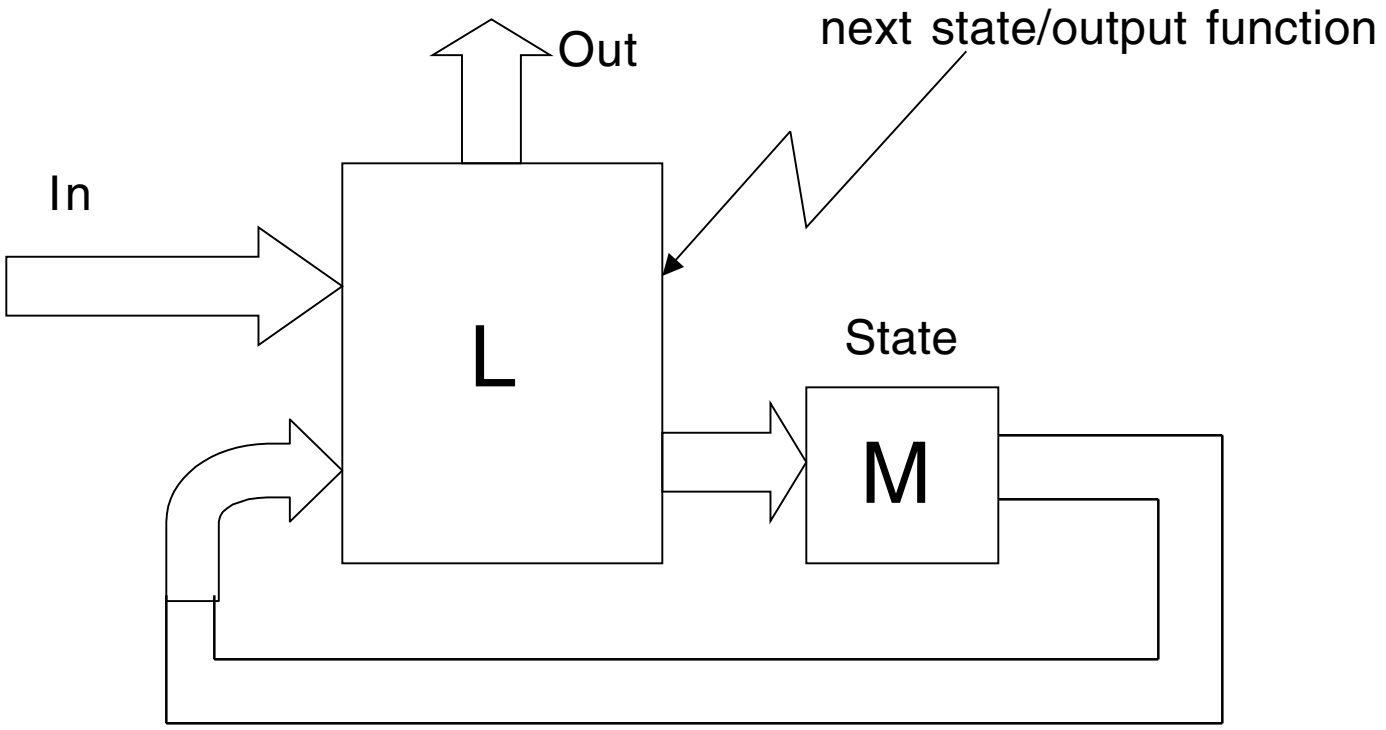
- $\Omega = \Psi = \{0,1\}$, $Q = \{q_0, q_1\}$

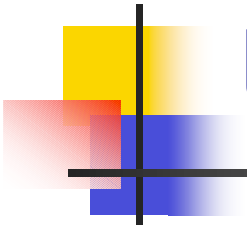


q	x	$\delta(q,x)$
q_0	0	q_0
q_0	1	q_1
q_1	0	q_1
q_1	1	q_0

q	$\lambda(q)$
q_0	0
q_1	1

FSM Model



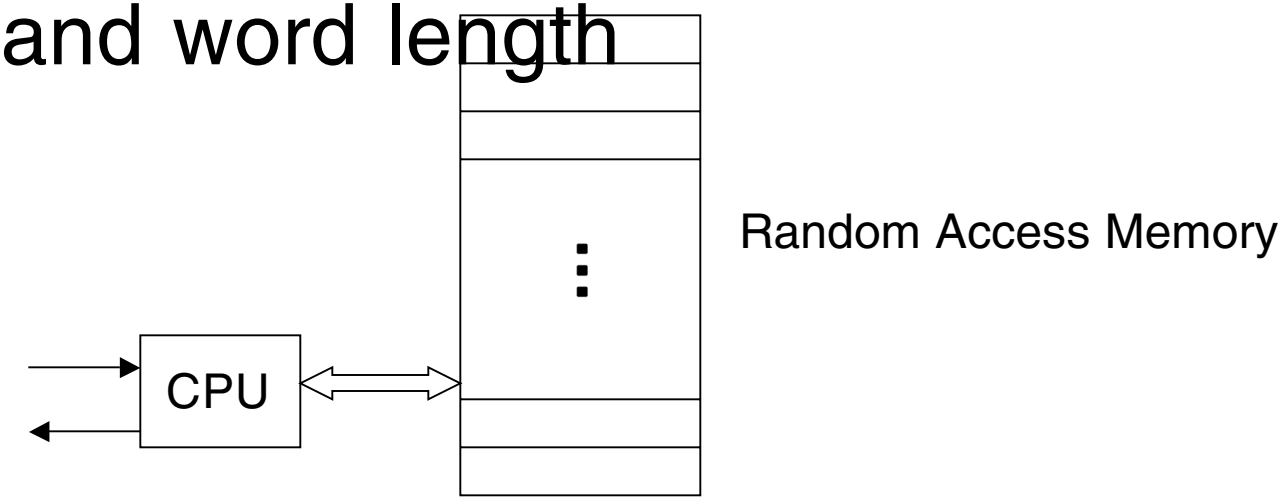


Unbounded State Machines

- Random Access Machine
  - CPU (FSM) connected to random access memory (each is word addressable)
- Turing machine
  - FSM controls read/write/movement on tape
- Cellular Automaton
  - 2D array of identical FSMs that communicate with neighbors.

Random Access Machine

- Implements fetch/execute cycle in CPU
- Has small set of instructions
- Potentially unbounded number of words and word length



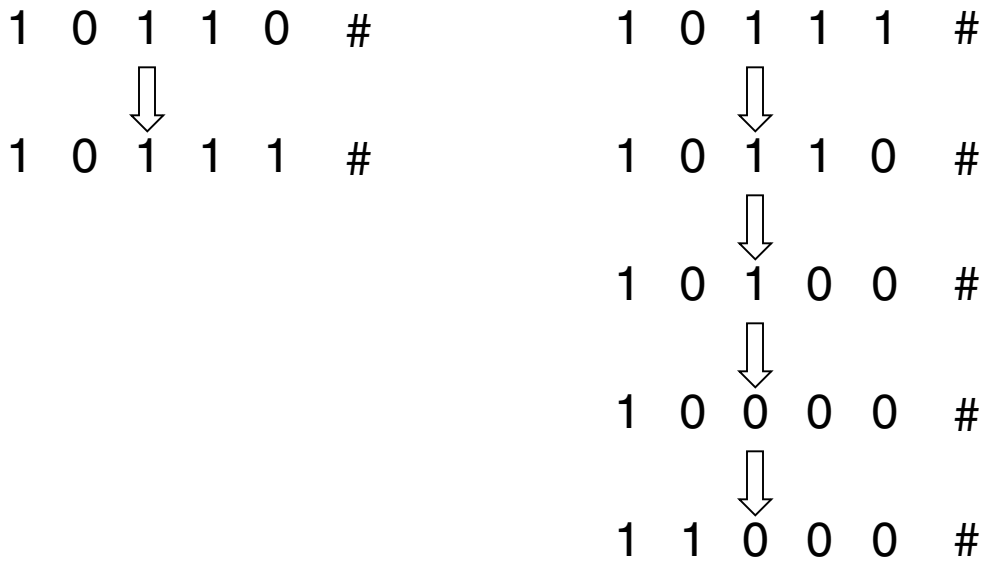
Turing Machine (TM)

- Control unit is an FSM:
  - Input is value under tape head.
  - FSM changes state and generates output (*c, m*).
  - *c* is written under head.
  - *m* directs head to move one cell *left* or *right*.
  - If next state is *h*, TM halts.

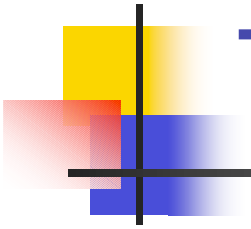
Functional Description of Turing Machine

- TM is a 6-tuple M = (Γ, β, Q, δ, s, *h*).
  - Γ is tape alphabet not containing blank β.
  - Q is set of states.
  - δ : Q £ (Γ [{β}) α (Q[{*h*})£ (Γ [{β}) £ (L,R) is next state/output function.
  - s & *h* are initial and halt states.
- Sometimes β is replaced by #

Adding One to a Binary Number



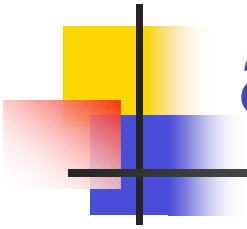
- Head initially over rightmost letter.



Turing Machine to Add One

- If head is initially over rightmost non-blank symbol, this program adds one to the binary string.

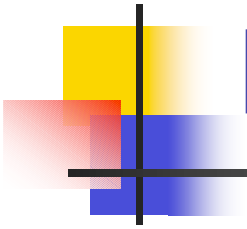
Turing Machine Conventions and Variations



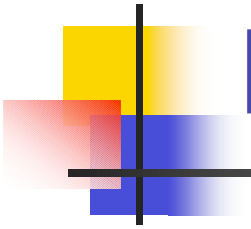
- Input string left-adjusted on otherwise blank tape.
- If computation halts, result is a string left-adjusted on otherwise blank tape.

- Tapes may be double-ended and/or have multiple tracks.
  - Do these changes increase power of TM?

Simulations of Turing Machines

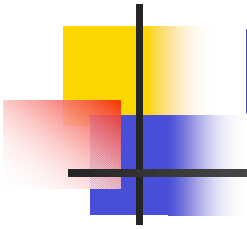


- See *The Most Complex Machine* by David Eck
  http://math.hws.edu/TMCM/java/index.html



Parallel Machine Models

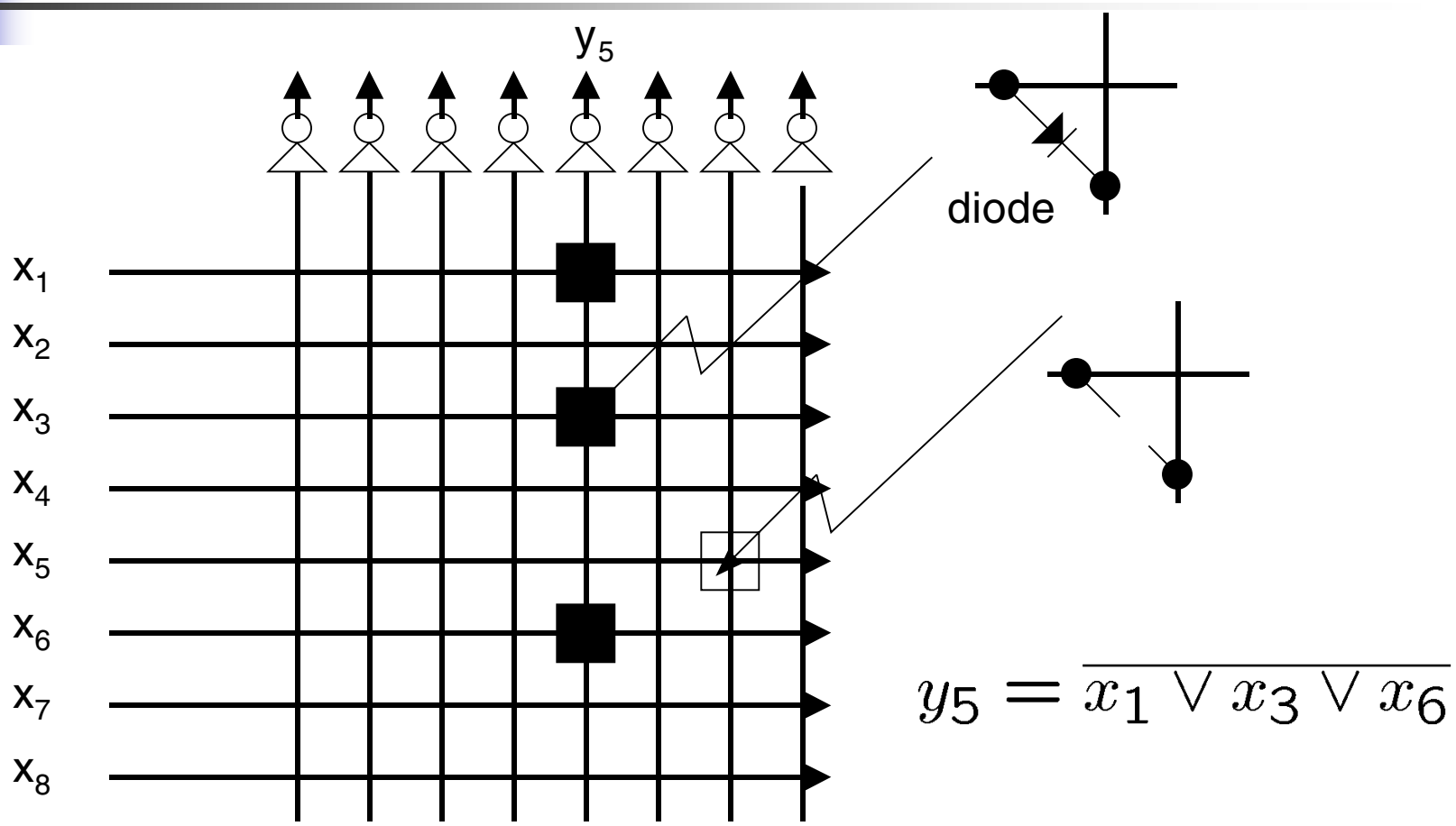
- Logic Circuits - memoryless
- Inter-connected serial machines.
  - Can be synchronous or asynchronous
  - Finite or infinite in number
- Types of interconnections
  - Unstructured, e.g. Internet
  - Structured, e.g. 1D and 2D arrays, hypercubes
- Models can be concrete or abstract



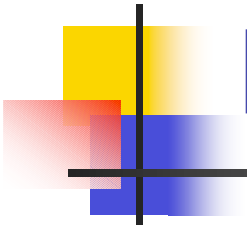
Logic Circuit

- A memoryless, parallel model
- Depth of circuit is measure of time
- Fan-in of gates generally bounded

The Programmable Crossbar Realizes "Wired NORs"

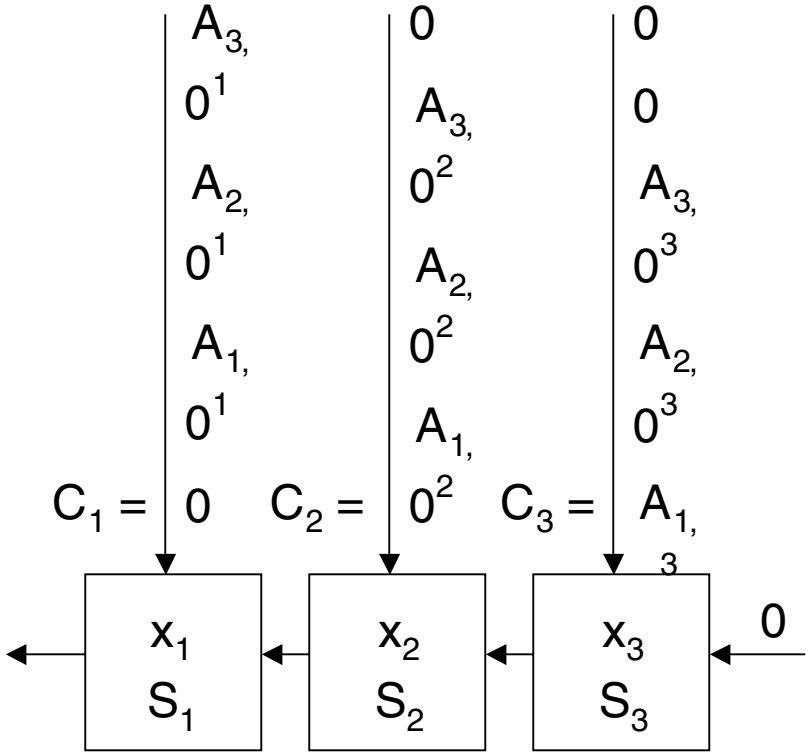


Asynchronous Parallel Machines



- Internet
- Clusters of processors on local network
- Processors connected via fast switch
- Many types of interconnection network

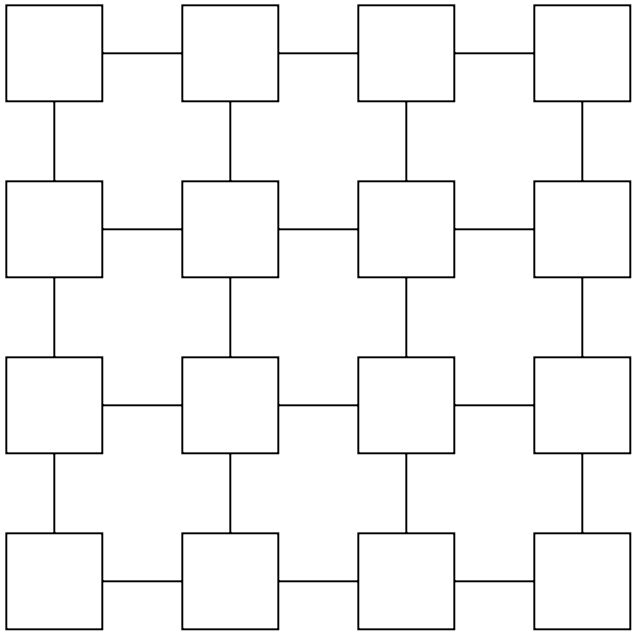
Systolic Array for Matrix-Vector Multiplication



$\mathbf{y} = A\,\mathbf{x}$

- S_i Ã S_{i+1} + x_i £ c_i

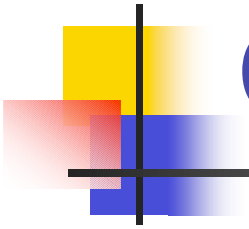
2D Mesh of Processors



Cells are FSMs

I/O on periphery

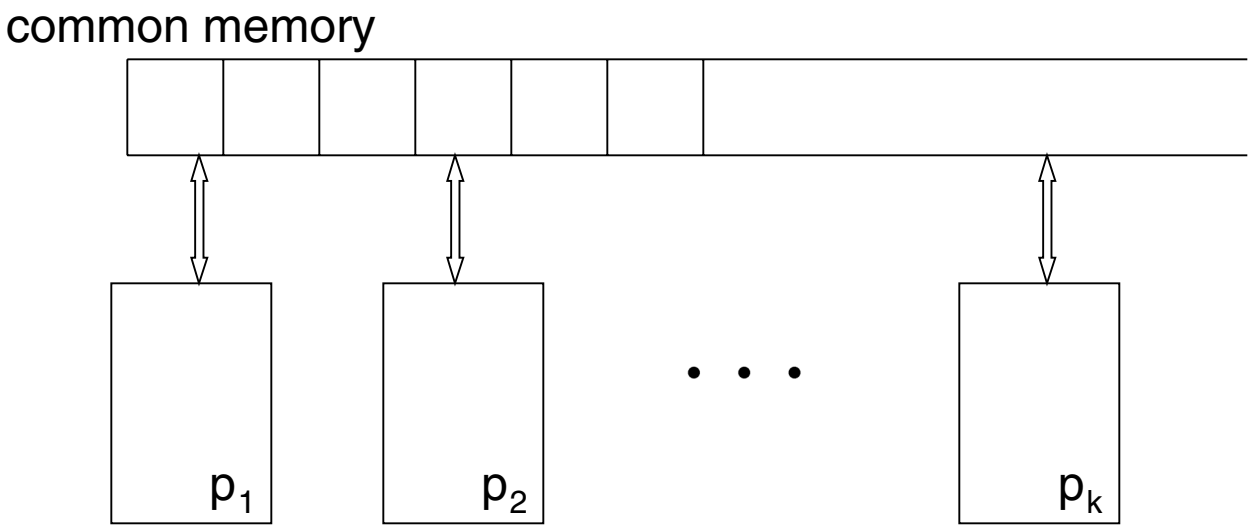
- How would you sort a set or multiply two matrices on this mesh?



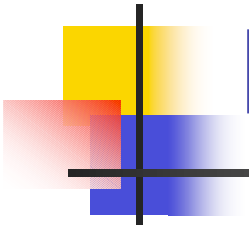
Cellular Automata

- Unbounded mesh of synchronous processors
- Game of Life uses simple cellular automaton
  - Each cell is alive or dead
  - State of cell depends on state of neighbors
- Very complex behavior is possible

Parallel Random Access Machine (PRAM)

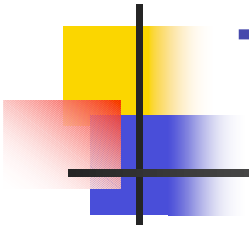


- RAMs have finite number of registers
- They read, compute, write in synch



PRAM Variants

- Processors have/do not have access to same memory location on same time step.
  - CRCW – concurrent read, concurrent write
  - CREW – concurrent read, exclusive write
  - ERCW – exclusive read, concurrent write
  - EREW – exclusive read, exclusive write
- Try to compute an arbitrary Boolean function in two steps on CRCW PRAM.



Topics for Next Lecture

- Are some problems not computable?
- How powerful are the various models?
- If we invent some new method of computation, can we solve problems that were previously unsolvable?
- Can some computational models solve problems more quickly than others?