CSCI 2570 Introduction to Nanocomputing

Historical Context for Computing

John E Savage

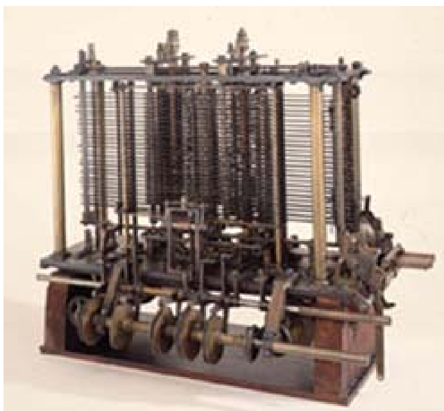
A Brief History of Computing and Computer Technologies

- Let's look at some of the key signposts in the development of computer technology.

- Let's briefly examine models of computation

Early Computers

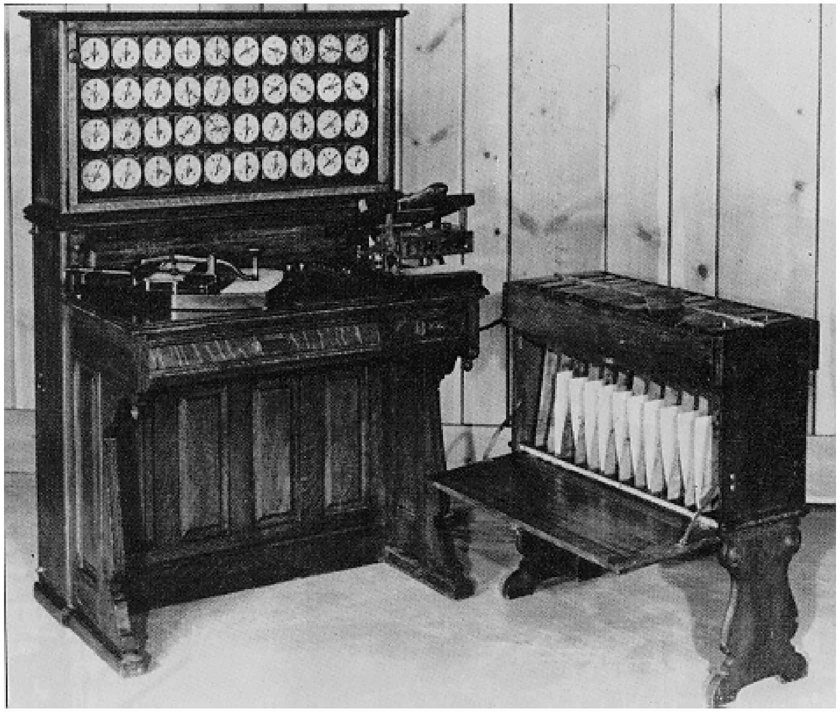
- Jacquard Loom – 1746
  - Punched cards control weaving



- Babbage's Analytical Engine – 1834
  - Mechanical computer, punched-card data input
  - Mill is shown above
  - Arithmetic done in base 10.

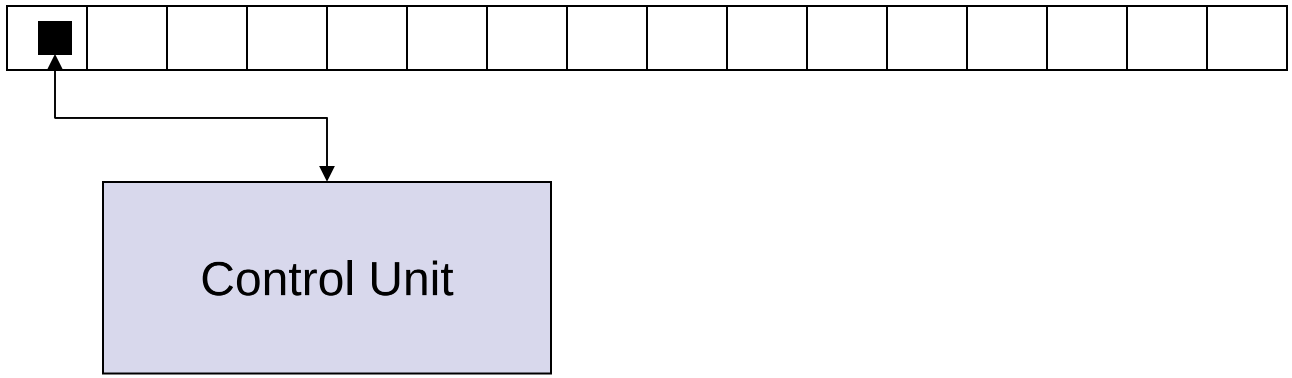
Early Computers

- Hollerith electric tabulator/sorter
  - Punched-card sorter – collated 1890 census data that was forecast to take more than 10 years!



Computers in the 20th Century

- Turing machine
  - Two-way tape for data input and storage and finite-state machine for reading/writing on tape.



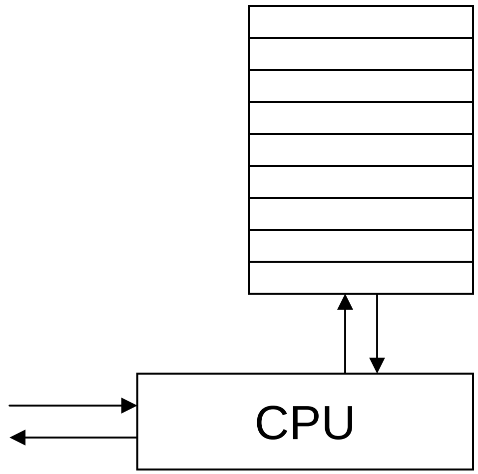
- Demonstrated impossibility of certain computations.

20^{th} Century *Programmable* Computers

- Atanasoff (1940) – linear eqn. solver, tube-based
- Zuse’s Z3 (1941) – relay-based computer
- Colossus (1943) – broke Enigma code, tube-based
- Mark I (1944) – general-purpose, relay-based
- ENIAC (1946) – general-purpose, tube-based
- Thousands of “computers” existed in 1940s

Computers in the 20th Century

- The von Neumann model



- Stored programs
- Fetch-execute cycle

The Computer Revolution Begins

- Transistor invented at Bell Labs in 1947
  - Semiconductor switch – replaced vacuum tube.

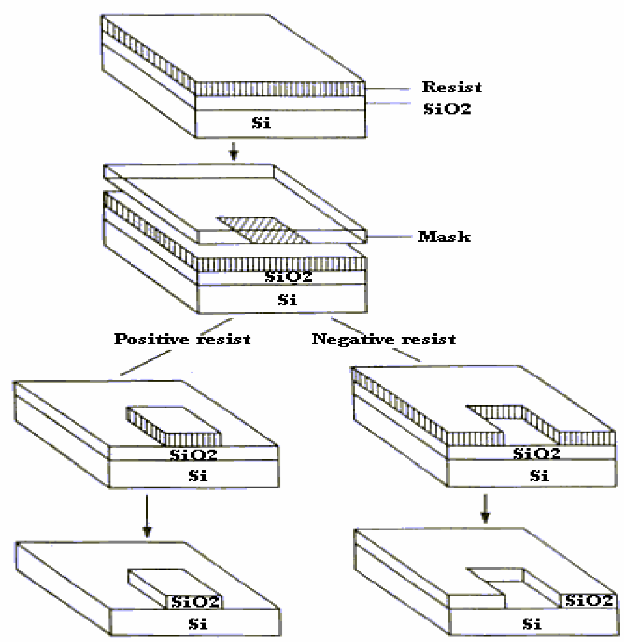
- By 1958 IBM was selling the 7070, a transistor-based computer.

The Integrated Circuit

- Integrated circuits invented independently in 1959 by Jack Kilby and Robert Noyce
  - Transistors and wires combined on a chip through photolithography.
  - *"What we didn't realize then was that the integrated circuit would reduce the cost of electronic functions by a factor of a million to one, nothing had ever done that for anything before"* - Jack Kilby

Photolithography

- This is the process of transferring a pattern to the surface of a chip using light.



The VLSI Revolution

- Intel 4004 CPU placed on a chip – 1969
- By late 1970s very complicated chips were being assembled.
- New challenges were encountered:
  - Specifying large chip designs simply
  - Simulating the electronics
  - Laying out chips
  - Designing area efficient algorithms
  - Understanding tradeoffs through analysis

VLSI Emerges as an Academic Area in Late 1970s

- **Introduction to VLSI** published by Carver Mead and Lynn Conway in 1980.
- Large chip designs now had to be specified
  - Hardware design languages invented
- Complicated electronics needed to be simulated.
  - Electronic simulators, such as Spice, developed
- Gates and memory cells needed to be placed
  - Computer-aided design emerges
- Area-efficient algorithms and theory
  - VLSI layouts and AT^2 lower bounds developed

The VLSI Model

- Wires have width, gates have area.
  - The **feature size** of a VLSI technology is the size of the smallest feature (wire width/separation)

- The area of gates is comparable to the square of feature size
  - The area occupied by wires often dominates the area of gates.

The VLSI Crisis

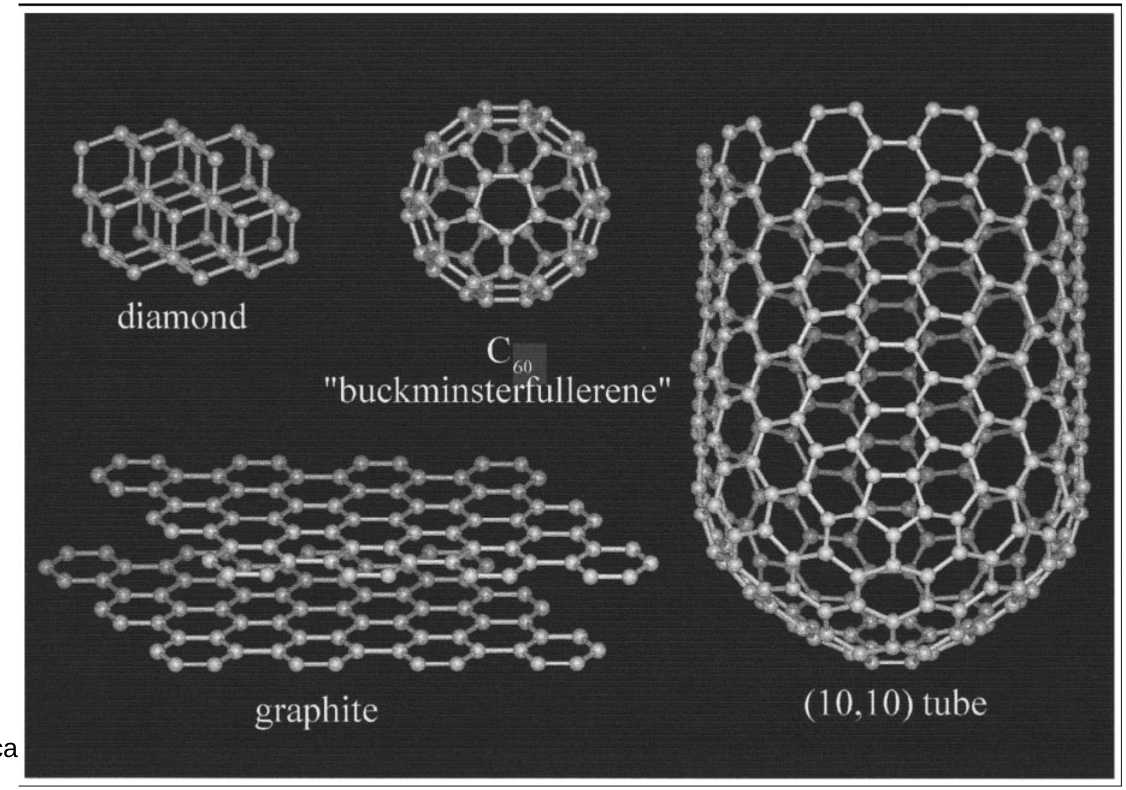
- Moore's Law – doubling of # transistors/chip every 18 months – coming to an end.
- Chip factories now cost $3-5 billion to construct!
- Devices are so small that electronic models are no longer accurate; expensive redesign needed to meet systems requirements.

What's Next?

- Nanotechnology of course!
- Nanotechnology is a broad term that includes biological elements, molecular electronics, and quantum computing.
- We give an overview of these technologies but focus primarily on the systems issues arising from nano-electronics.

Emergence of Nanotechnology

- Bucky balls (C_{60}) discovered at Rice in 1985
- Iijima discovered carbon nanotubes in 1991



Properties of Nanotechnologies

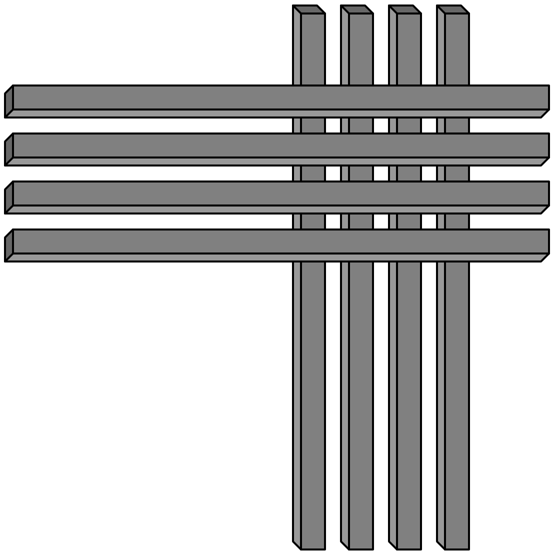
- Methods of assembly are either very slow and precise or fast and non-deterministic.

- Fast assembly is good at creating fairly regular structures.

- There is hope that through DNA templating non-regular structures will be possible

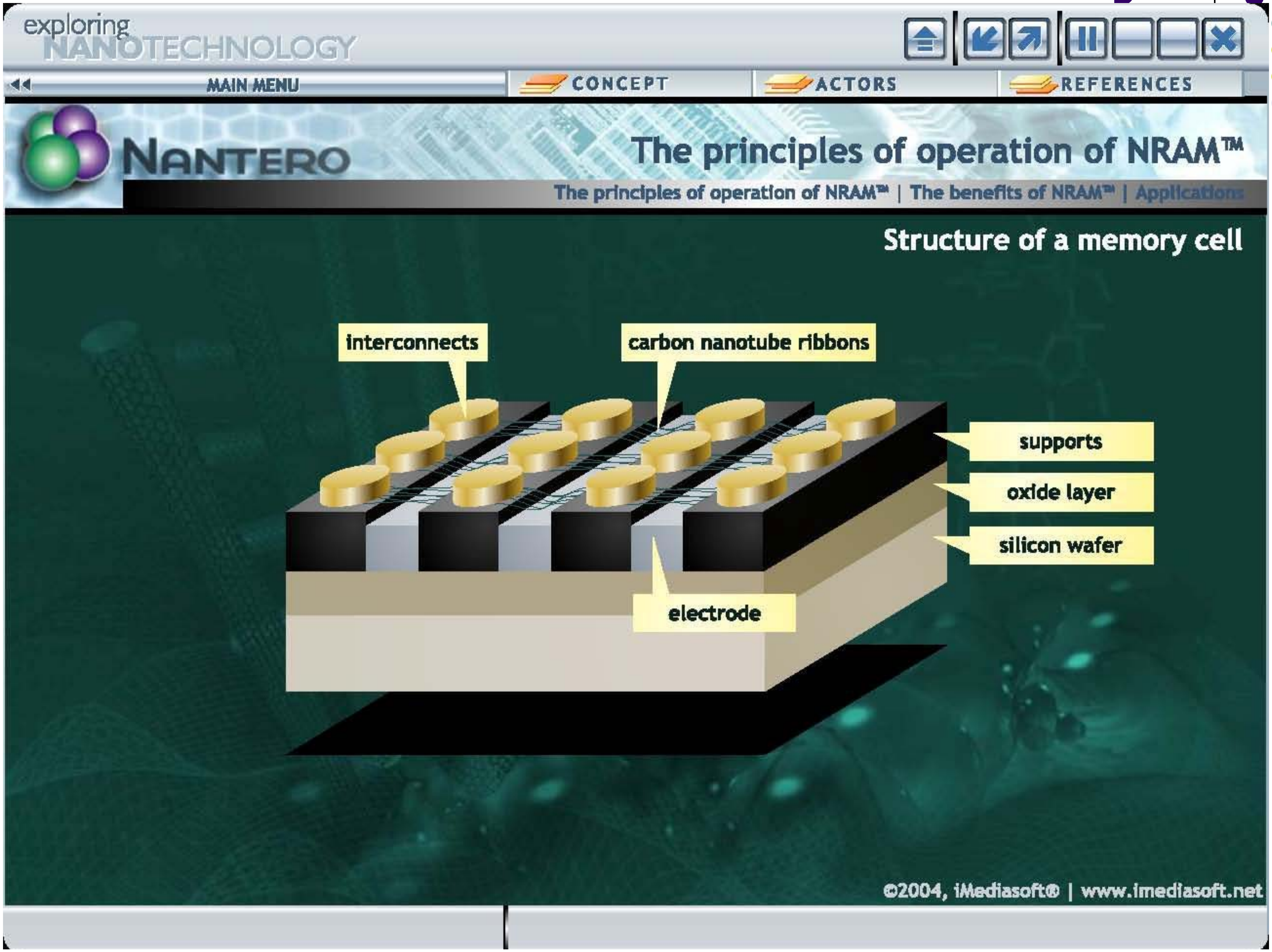
The Crossbar – A Promising Nanotechnology

- Two sets of parallel wires with switches at their intersections.



- Crossbars are used as routers and memories today.

Mechanical Crossbar Memory



NRAM – Nonvolatile RAM Crossbars of Carbon Nanotubes

- Electrostatic attraction used to make contacts, repulsion breaks them.
- Nantero's claims: (play the movie)
  - Permanently nonvolatile memory
  - Speed comparable to DRAM/SRAM
  - Density comparable to DRAM
  - Unlimited lifetime
  - Immune to soft errors
  - Will replace all existing forms of bulk memory!
- No behavioral models yet presented

Many Other Examples of Computational Nanotechnology

- Crossbars realized with silicon nanowires (NWs).
- Many issues concerning controlling NWs with mesoscale wires (MWs).
- Reliable computation with unreliable elements.

Goals of the US National Nanotechnology Initiative

- Maintain a world-class research and development program aimed at realizing the full potential of nanotechnology;
- Facilitate transfer of new technologies into products for economic growth, jobs, and other public benefit;
- Develop educational resources, a skilled workforce, and the supporting infrastructure and tools to advance nanotechnology; and,
- Support responsible development of nanotechnology.

Introduction to Formalized Models of Computation

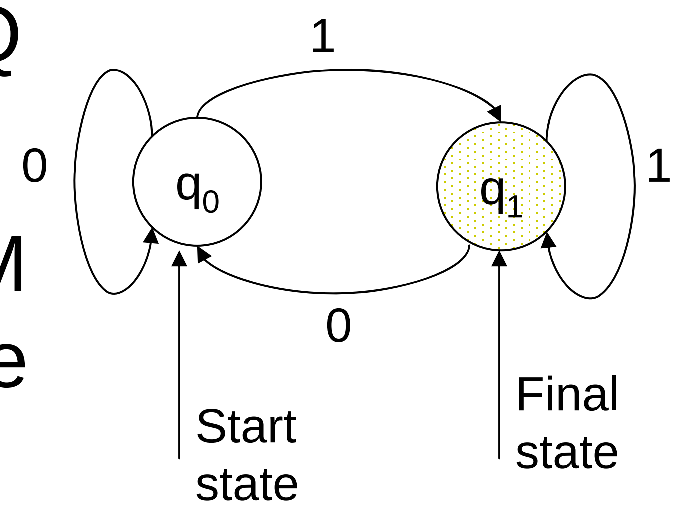
- Logic circuits
- Finite state machines (FSAs)
  - Deterministic and non-deterministic
- Turing machines
  - Containing one or more potentially infinite tapes
  - Deterministic and non-deterministic
- Languages
- NP-complete problems.

Logic Circuits

- Feasibility of two-level logic leads to computation of binary functions.
- Binary function $f : \Sigma^n \rightarrow \Sigma^m$ defined by table.
- Can be realized with AND, OR, NOT
  - {NAND} is another "complete basis"
- Challenging to find small circuits
  - Most functions $f : \Sigma^n \rightarrow \Sigma$ have circuit size $O(2^n/n)$
  - Practical circuits have size $O(n)$ to $O(n^3)$.

Finite-State Machine (Σ, Q, δ, F)

- Bounded number of states Q.
- Input in Σ takes machine from a state to a state, $\delta: Q \times \Sigma \rightarrow Q$
- Some states are final (in F).
- "Accepted" strings move FSM from start state to a final state
- The FSM "recognizes" the language of accepted strings.



Languages

- A language is a set of strings over an alphabet.

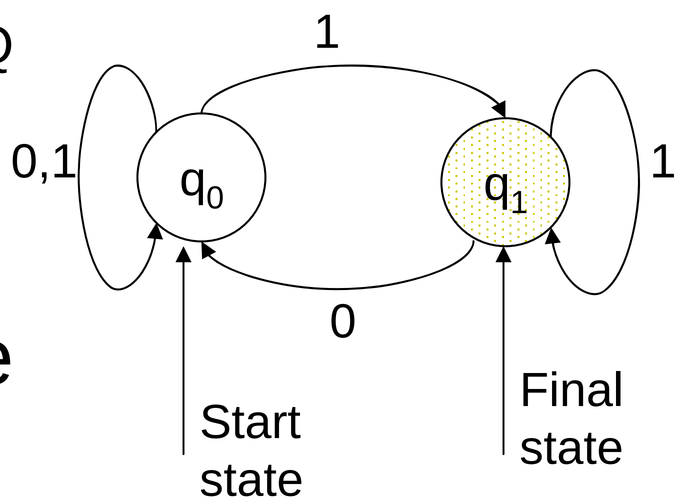
- Examples:
  - {0, 00, 000, …}
  - {1, 01, 10, 100, 010, 001, 0001, …, 1101, … } (odd number of 1s)

Limits on Language Acceptance

- Are there languages that cannot be accepted by an FSM?
- How about $\{0^n1^n\}$?
- What is the property of FSMs that prevents them from “counting?”

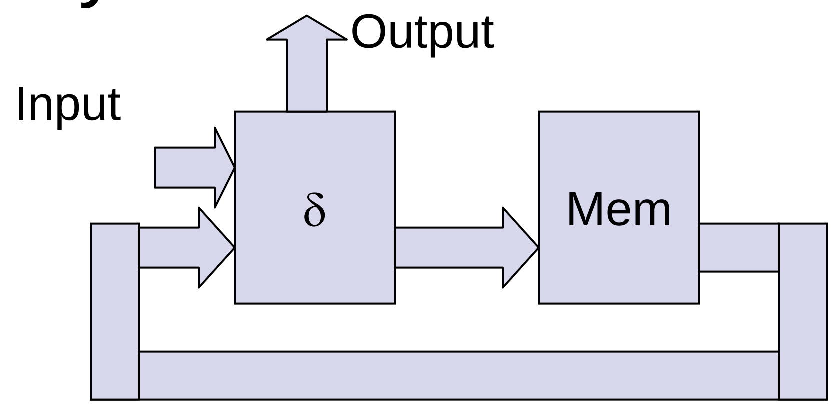
Nondeterministic Finite-State Machine (Σ,Q,δ,F)

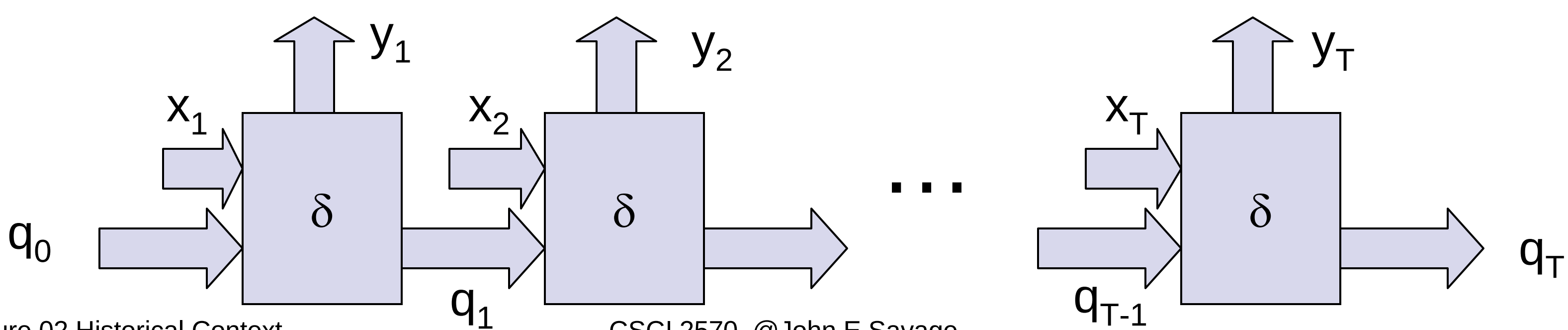
- Possibly more than one successor state, $\delta: Q\times \Sigma\rightarrow 2^Q$
- Addition of "hidden" input removes nondeterminism
- Hidden inputs form certificate for acceptance of a string.
- The languages recognized by NFSMs and FSMs are the same. Why?



Circuits and FSMs

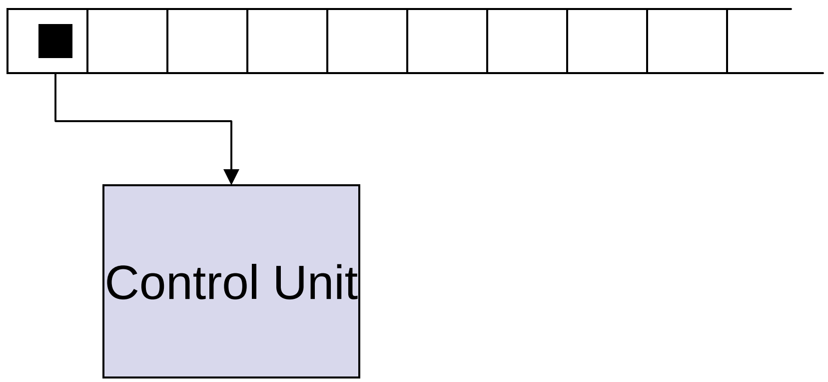
- If an FSM executes T cycles, can it be simulated by a circuit?





Turing Machines

- A Turing machine is an FSM or NFSM control unit connected to one or more potentially infinite tapes.



- Is the power of a TM enhanced by having more tapes?

Language Acceptance by TMs

- A string placed on the otherwise blank input tape of a TM is accepted if its control unit enters a final state.
  - This applies to both FSM and NFSM control units.

- Can a Turing machine accept $\{0^n1^n\}$? How?

- The time to accept a string on a TM is the number of steps taken by its control unit.
  - Time will depend on the number of tapes.

The Classes P and NP

- The class P is the set of languages accepted by deterministic TMs in polynomial time.
- The class NP is the set of languages accepted by nondeterministic TMs in polynomial time.

Reductions

- Reducing to a previously solved problem.
  - Given a solution (program), use it to solve a new problem.
  - E.g. Use a squaring program to multiply integers.

- If problem P is reduced to problem Q (the program for Q is used to solve P), can Q be easier than P?
- If P is hard, can Q be easy?

Polynomial-Time Reductions

- Given problem P, we transform it using a polynomial time algorithm into problem Q.
- If Q can be done in polynomial time, so can P
- If P requires more than polynomial time, so does Q.

The Class of NP-Complete Decision Problems

- A problem Q is NP-complete if
  - Q is in NP,and
  - Every problem in NP can be reduced to Q by a deterministic polynomial-time algorithm

- Example – 3-Satisfiability
  - *Instance:* A set of clauses in three variables, e.g. $(x_2+x_3+\bar{x}_5)$
  - *Yes Instance:* All the clauses can be satisfied (made True) by some choices for the variables.

NP-Complete Problems

- Thousands of problems have been shown to be NP-complete.
- If one of them can be shown to require more than polynomial time, all require more than polynomial time.
- If one of them can be shown to be done in polynomial time, all of them can.