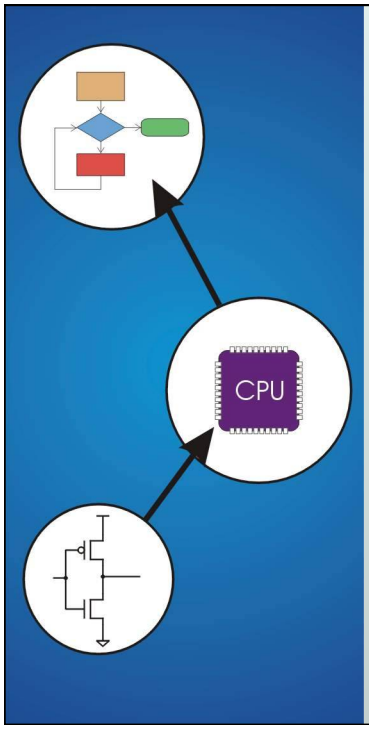
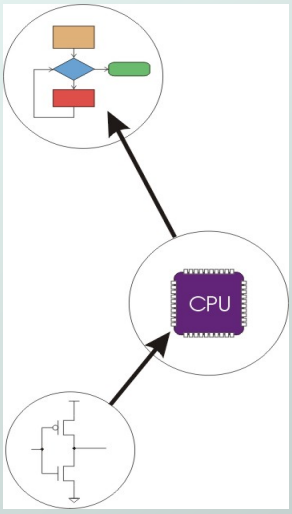


Chapter 3
Digital Logic Structures

Original slides from Gregory Byrd, North Carolina State University

Modified slides by C. Wilcox, S. Rajopadhye
Colorado State University

Computing Layers



Problems

Algorithms

Language

Instruction Set Architecture

Microarchitecture

Circuits

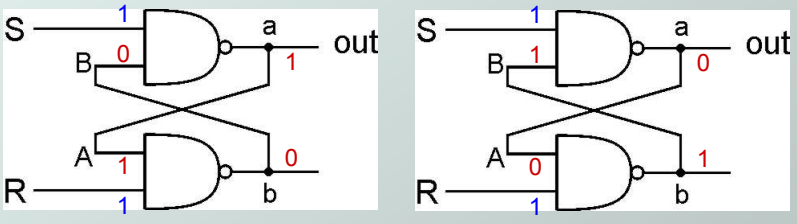
Devices

Combinational vs. Sequential

- **Combinational Circuit**
  - does not store information, always gives the same output for a given set of inputs
    - *example*: adder always generates sum and carry, regardless of previous inputs
- **Sequential Circuit**
  - stores information, output depends on stored info (state) plus input
  - so a given input might produce different outputs, depending on the stored information
  - useful for building "memory" elements and "state machines"
    - *example*: ticket counter

R-S Latch: Simple Storage Element

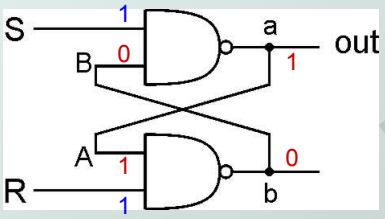
- R is used to "reset" or "clear" the element – set it to zero.
- S is used to "set" the element – set it to one.



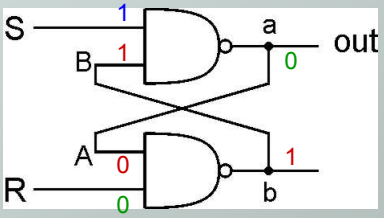
- If both R and S are one, output could be either zero or one.
  - "quiescent" state -- holds its previous value
  - if a is 1, b is 0, and vice versa

Clearing the R-S latch

- Suppose we start with output = 1, then change R to zero.



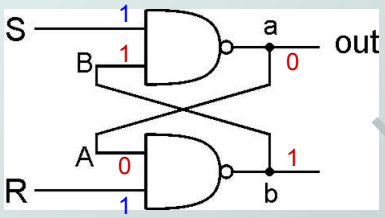
Output changes to zero.



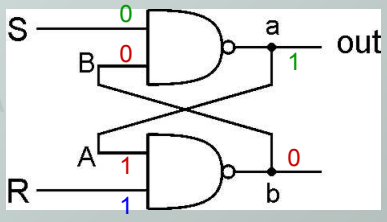
Then set R=1 to "store" value in quiescent state.

Setting the R-S Latch

- Suppose we start with output = 0, then change S to zero.



Output changes to one.



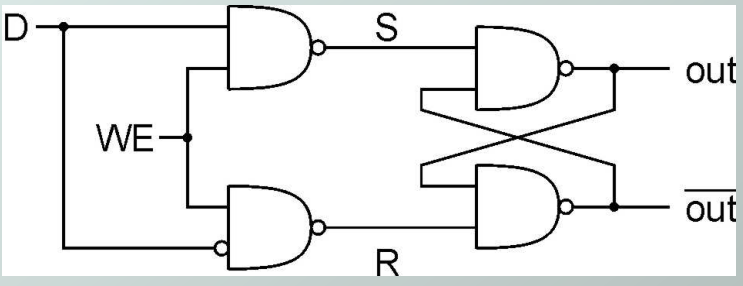
Then set S=1 to "store" value in quiescent state.

R-S Latch Summary

- **R = S = 1**
  - hold current value in latch
- **S = 0, R=1**
  - set value to 1
- **R = 0, S = 1**
  - set value to 0
- **R = S = 0**
  - both outputs equal one
  - final state determined by electrical properties of gates
  - ***Don't do it!***

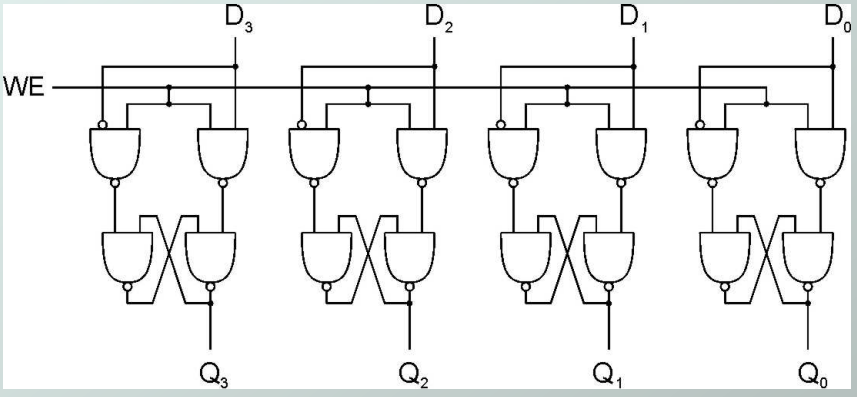
Gated D-Latch

- Two inputs: D (data) and WE (write enable)
  - when **WE = 1**, latch is set to **value of D**
    - S = NOT(D), R = D
  - when **WE = 0**, latch holds **previous value**
    - S = R = 1



Register

- A register stores a multi-bit value.
  - We use a collection of D-latches, all controlled by a common WE.



Representing Multi-bit Values

- Number bits from right (0) to left (n-1)
  - just a convention -- could be left to right, but must be *consistent*
- Use brackets to denote range:
  D[l:r] denotes bit **l** to bit **r**, from *left* to *right*

15 0

A = **0101001101010101**

A[14:9] = **101001**

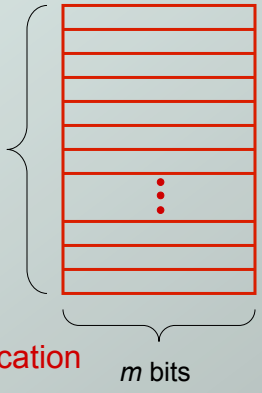
A[2:0] = **101**

- May also see A**<14:9>**, especially in hardware block diagrams.

Memory

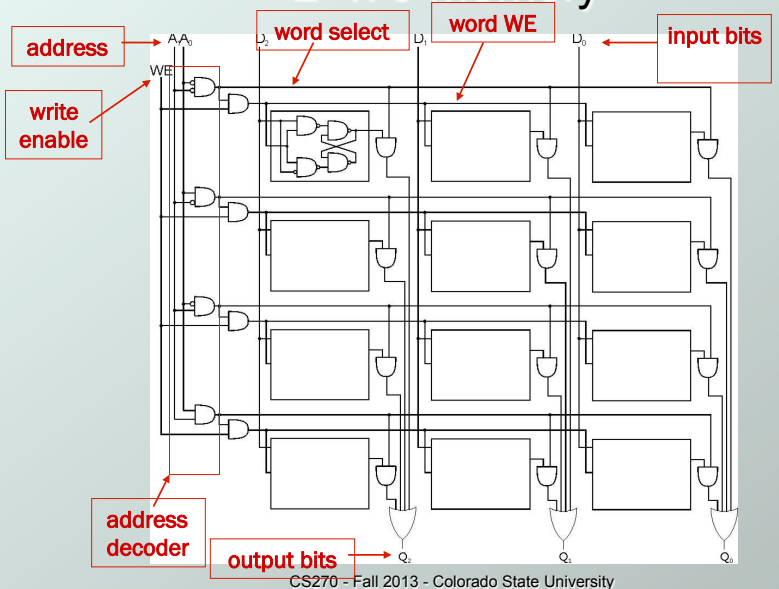
- Now that we know how to store bits, we can build a memory – a logical $k \times m$ array of stored bits.

Address Space:
number of locations
(usually a power of 2)



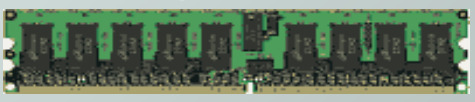
Addressability:
number of bits per location
(e.g., byte-addressable)

2^2 x 3 Memory



More Memory Details

- Not the way actual memory is implemented!
  - fewer transistors, denser, relies on electrical properties
- But the logical structure is very similar.
  - address decoder, word select line, word write enable
- Random Access Memory: 2 different types
  - **Static RAM** (SRAM)
    - fast, used for caches, maintains data when powered
  - **Dynamic RAM** (DRAM)
    - slower but denser, storage decays, must be refreshed
- Non-Volatile Memory: **ROM, PROM, Flash**

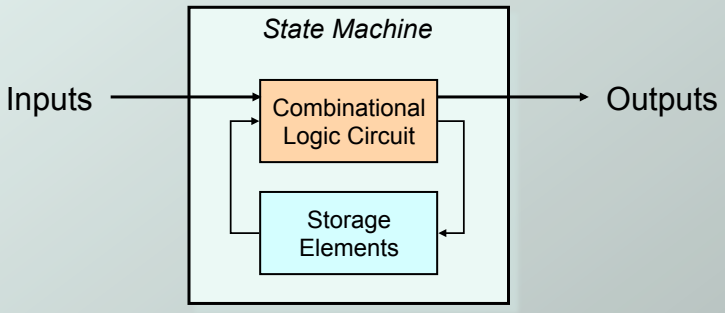


Memory Bandwidth

- Bandwidth is the rate at which memory can be read or written by the processor.
- Approximately equal to the memory bus size times the speed at which the memory is clocked.
- Examples of bandwidth (from Wikipedia):
  - Phone line, Modem, up to 5.6KB/s
  - Digital subscriber line, ADSL, up to 128KB/s
  - Wireless networking, 802.11g, up to 17.5MB/s
  - Peripheral connection, USB 2.0, 60MB/s
  - Digital video, HDMI, up to 1.275GB/s
  - Computer bus, PCI Express, up to 25.6GB/s
  - Memory chips, SDRAM, up to 52GB/s

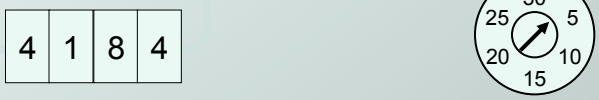
State Machine

- Another type of sequential circuit
  - Combines combinational logic with storage
  - "Remembers" state, and changes output (and state) based on **inputs** and **current state**



Combinational vs. Sequential

- Two types of "combination" locks



Combinational
Success depends only on the values, not the order in which they are set.

Sequential
Success depends on the sequence of values (e.g, R-13, L-22, R-3).

State

- The state of a system is a snapshot of all the relevant elements of the system at the moment the snapshot is taken. Examples:
  - The state of a basketball game can be represented by the scoreboard: number of points, time remaining, possession, etc.
  - The state of a tic-tac-toe game can be represented by the placement of X's and O's on the board.

State of Sequential Lock

Our lock example has four different states, labelled A-D:

A: The lock is not open, and no relevant operations have been performed.

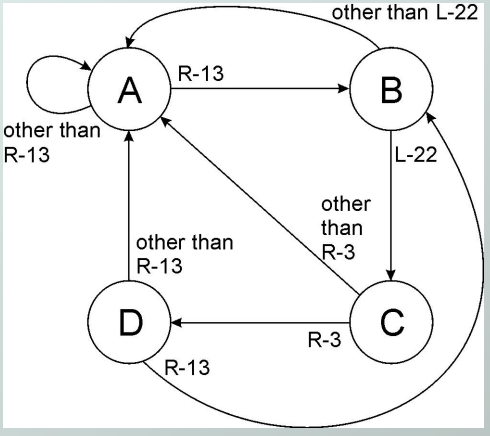
B: The lock is not open, and the user has completed the R-13 operation.

C: The lock is not open, and the user has completed R-13, followed by L-22.

D: The lock is open.

State Diagram

- Shows **states** and **actions** that cause a **transition** between states.



Finite State Machine

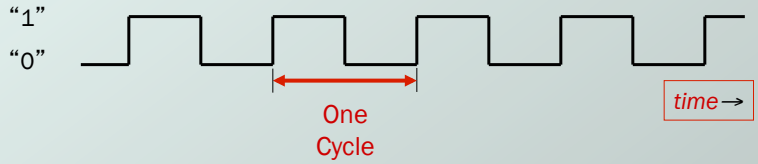
- A system with the following components:

1. A finite number of **states**
2. A finite number of external **inputs**
3. A finite number of external **outputs**
4. An explicit specification of all **state transitions**
5. An explicit specification of what determines each external **output value**

- Often described by a state diagram.
  - Inputs trigger state transitions.
  - Outputs are associated with each state (or with each transition).

The Clock

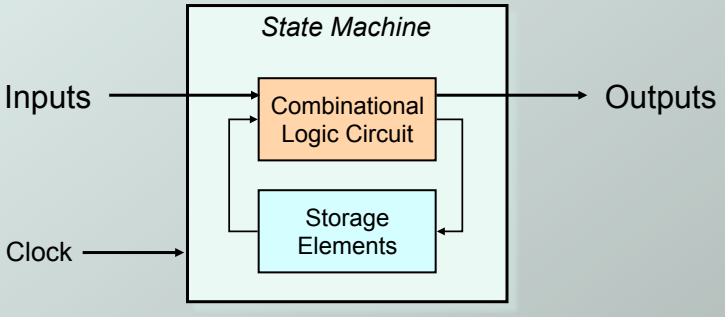
- Frequently, a **clock circuit** triggers transition from one state to the next.



- At the beginning of each clock cycle, state machine makes a transition, based on the current state and the external inputs.
  - **Not always required. In lock example, the input itself triggers a transition.**

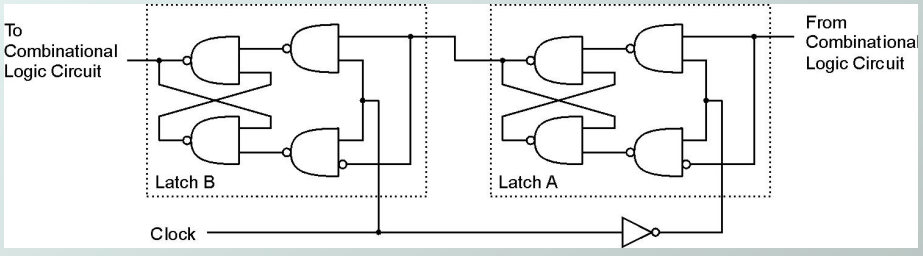
Implementing a Finite State Machine

- **Combinational logic**
  - Determine outputs and next state.
- **Storage elements**
  - Maintain state representation.



Storage: Master-Slave Flipflop

- A pair of gated D-latches, to isolate *next* state from *current* state.



During 1st phase (clock=1), previously-computed state becomes *current* state and is sent to the logic circuit.

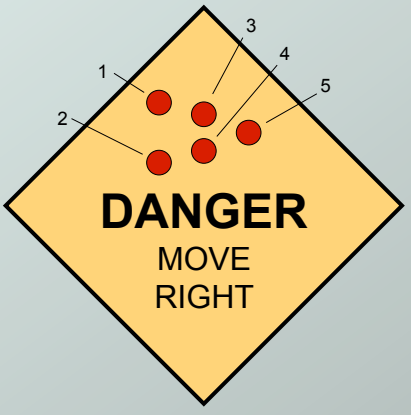
During 2nd phase (clock=0), *next* state, computed by logic circuit, is stored in Latch A.

Storage

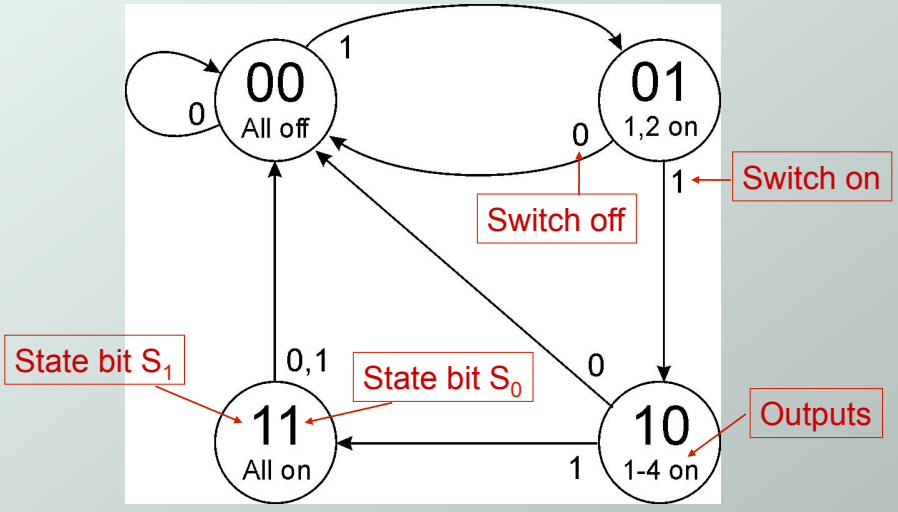
- Each master-slave flipflop stores one state bit.
- The number of storage elements (flipflops) needed is determined by the number of states (and the representation of each state).
- Examples:
  - Sequential lock
    - Four states – two bits
  - Basketball scoreboard
    - 7 bits for each score, 5 bits for minutes, 6 bits for seconds,1 bit for possession arrow, 1 bit for half, …

Complete Example

- A blinking traffic sign
  - No lights on
  - 1 & 2 on
  - 1, 2, 3, & 4 on
  - 1, 2, 3, 4, & 5 on
  - (repeat as long as switch is turned on)



Traffic Sign State Diagram



Transition on each clock cycle.

Traffic Sign Truth Tables

Outputs
(depend only on state: S_1S_0)

Lights 1 and 2
Lights 3 and 4
Light 5

S_1	S_0	Z	Y	X
0	0	0	0	0
0	1	1	0	0
1	0	1	1	0
1	1	1	1	1

Next State: S_1' S_0'
(depend on state and input)

Switch

In	S_1	S_0	S_1'	S_0'
0	X	X	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0

Whenever In=0, next state is 00.

Traffic Sign Logic