


CS 135: Computer Architecture I


Instructor: Prof. Bhagi Narahari
 Dept. of Computer Science
 Course URL: www.seas.gwu.edu/~bhagiweb/cs135/



Summary of Combinational Logic

- **Combinational device/circuit:** any circuit built using the basic gates
- **Expressed as**
 - Truth table
 - Digital circuit
 - Boolean function
- **Any boolean function can be expressed as two level function**
- **Minimization procedure: Karnaugh Map**
 - Try to minimize the number of gates, and inputs to gates, in a two level circuit


CS 135



Combinational vs. Sequential

- **Combinational Circuit**
 - always gives the same output for a given set of inputs
 - ex: adder always generates sum and carry, regardless of previous inputs
- **Sequential Circuit**
 - stores information
 - output depends on stored information (state) plus input
 - so a given input might produce different outputs, depending on the stored information
 - **example: vending machine**
 - Current total increases when you insert coins
 - output depends on previous state
 - useful for building “memory” elements and “state machines”

CS 135



Sequential Logic

- **Build a device, using combinational logic devices, to **store** a value**
 - RS Latch (also called SR Latch)
 - concept of memory
- **Methodology behind design of sequential logic circuits**
 - Finite State Machines
 - Example of Vending machine
- **Combine sequential and combinational logic devices to “assemble” a simple processor!**

CS 135

Sequential Circuits – Storage Elements

- We need to design a device capable of storing information
 - Store a bit value
- Build it using the devices we have thus far
 - Use “feedback” circuit
- To be useful, sequential device needs mechanism for setting its state
- R-S Latch:
 - Output = previous value
 - Or Set output to new value (0 or 1), and hold this new value till next “write” into the device

CS 135

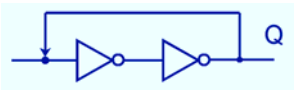
Feedback Circuits

- Stable circuit
 - Output point of circuit retains value indefinitely
- Unstable circuit
 - State that remains constant only for a duration of a few gate delays

CS 135

Feedback Circuits

- To retain their state values, sequential circuits rely on *feedback*.
- Feedback in digital circuits occurs when an output is looped back to the input.
- A simple example of this concept is shown below.
 - If Q is 0 it will always be 0, if it is 1, it will always be 1. Why?



CS 135

Feedback Circuits

- What if we had three gates in the circuit ?

CS 135

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.

S: 1, B: 0, A: 1, R: 1
a: 1, b: 0

S: 1, B: 1, A: 0, R: 1
a: 0, b: 1

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

CS 135

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!*

CS 135

Clocked Flip-Flops/Circuits

- Subsystem in a computer consists of a large number of combinational and sequential devices
 - Each sequential device is like an SR latch which is in one of two states
 - As machine executes its cycle, the states of all sequential devices change with time
- To control large collection of devices in an orderly (synchronized) fashion, machine maintains a **clock**
 - Requires all devices to change their states at the same time
 - Clock generates sequence of pulses

CS 135

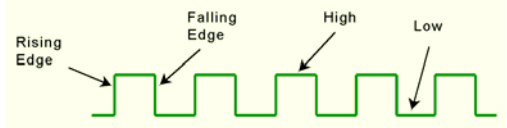
Clock

- As the name implies, sequential logic circuits require a means by which events can be sequenced.
- State changes are controlled by clocks.
 - A “clock” is a special circuit that sends electrical pulses through a circuit.
- Clocks produce electrical waveforms such as:

CS 135

Clocked Circuits

- State changes occur in sequential circuits only when the clock ticks.
- Circuits can change state on (a) the rising edge or falling edge – *edge triggered circuits*, or (b) when the clock pulse reaches its highest voltage level – *level triggered*.
 - Time between pulses is the period of the clock



Rising Edge Falling Edge High Low

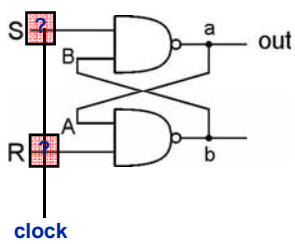
CS 135

Clocked RS Latch – Flip Flop

- Every sequential device has a Clock (CK) input in addition to its other inputs
 - Device designed to respond to inputs only during a clock pulse
- Where do we add clock to the RS Latch ?
 - Clocked RS Latch = RS Flip Flop
- Shield NAND gates (latch) from effect of S,R except when clock is high
 - When clk is low the inputs to the NAND gates are both 1 regardless of S,R
 - No change in latch output value
 - When clk is high, values of R and S pass to the NAND gates – i.e., latch

CS 135

Clocked Latches – Flip Flop



CS 135

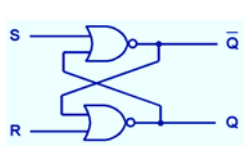
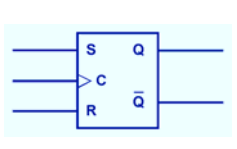
SR Flip Flop: Reverse of RS

- More common latch is the SR latch
 - Have 11 input as undefined
 - 00 holds state
 - Set S=1 to set latch to 1
 - Set R=1 to set latch to 0
- Replace NAND gates by NOR gates!

CS 135

SR Latch/Flip-flop

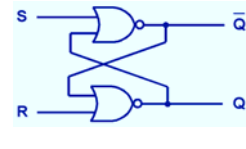
- the most basic sequential logic components, the SR flip-flop.
 - The "SR" stands for set/reset.

CS 135

SR Latch Behaviour

- The behavior of an SR flip flop is described by a characteristic table.
- $Q(t)$ means the value of the output at time t . $Q(t+1)$ is the value of Q after the next clock pulse.



S	R	$Q(t+1)$
0	0	$Q(t)$ (no change)
0	1	0 (reset to 0)
1	0	1 (set to 1)
1	1	undefined

CS 135

SR Flip Flop

- The SR flip-flop actually has three inputs: S, R, and its current output, Q.
- Thus, we can construct a truth table for this circuit, as shown at the right.
- Notice the two undefined values. When both S and R are 1, the SR flip-flop is unstable.

Present State			Next State
S	R	$Q(t)$	$Q(t+1)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	undefined
1	1	1	undefined

CS 135

Concept of Memory and Write to Memory

- We want ONE input to the latch
- We want to control when to "update" value in the latch (output)
- When write is enabled, we want Output = Value of Input
 - Else output = old value of output

CS 135

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

CS 135

D Flip Flop Behaviour

- Typical modeling of D flip flop simplifies things
 - Input D=1 then at clock pulse, output of flip flop becomes 1
 - Input D=0 then at clock pulse, output becomes 0

CS 135

D Flip Flop

- The D flip-flop is the fundamental circuit of computer memory.
 - D flip-flops are usually illustrated using the block diagram shown below.
- Modify latch to make it a clocked flip flop
- The characteristic table for the D flip-flop is shown at the right.

D	Q (t+1)
0	0
1	1

CS 135

Next... Storage Devices

- Temporary storage in a computer ?
 - Where are variables stored before being sent to the arithmetic unit for operations on them?
- Register
 - Can we build an n-bit register using latches?
- What about “main” memory
- Disk
 - Later...

CS 135

Register

- A register stores a multi-bit value.
 - We use a collection of D-latches, all controlled by a common WE.
 - When WE=1, n-bit value D is written to register.

CS 135

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*

$A = \overset{15}{0}1010011010101\overset{0}{1}$
 $A[14:9] = 101001$ $A[2:0] = 101$

- May also see $A\langle 14:9 \rangle$, especially in hardware block diagrams.

CS 135

Memory

- We know how to store m-bit number in a register
- How about many m-bit numbers ?
 - Bank of registers?
- How to fetch a specific m-bit number?
 - addressing

CS 135

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)

$k = 2^n$
 locations

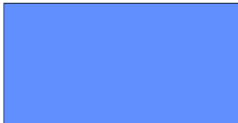
Addressability:
 number of bits per location
 (e.g., byte-addressable)

m bits

CS 135

Memory

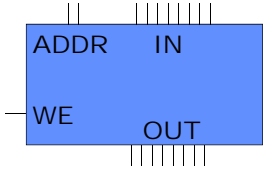
- Looking from the outside, what do we need?



CS 135

Memory

- Looking from the outside, what do we need?



CS 135

Memory - 1

A large number of addressable fixed size locations

- Address Space**

n bits allow the addressing of 2^n memory locations.

- Example: 24 bits can address $2^{24} = 16,777,216$ locations (i.e. 16M locations).
- If each location holds 1 byte (= 8 bits) then the memory is 16MB.
- If each location holds one word (32 bits = 4 bytes) then it is 64 MB.

CS 135

Memory - 2

- Addressability**

- Computers are either **byte** or **word** addressable - i.e. each memory location holds either 8 bits (1 byte), or a full standard word for that computer (16 bits for the LC-3, more typically 32 bits, though now many machines use 64 bit words).
- Normally, a whole word is written and read at a time:
 - If the computer is **word** addressable, this is simply a single address location.
 - If the computer is **byte** addressable, and uses a multi-byte word, then the word address is conventionally either that of its most significant byte (**big endian** machines) or of its least significant byte (**little endian** machines).

CS 135

Memory

- Given address, fetch contents at that address
 - Select or enable one of many locations

ADDR IN
 WE OUT

CS 135

Building a Memory

- Each bit
 - is a gated D-latch
- Each location
 - consists of w bits (here $w = 1$)
 - $w = 8$ if the memory is byte addressable
- Addressing
 - n locations means $\log_2 n$ address bits (here 2 bits \Rightarrow 4 locations)
 - decoder circuit translates address into 1 of n locations

CS 135

Memory Example

A 2^2 by 3 bits memory:


- two address lines: $A[1:0]$
- three data lines: $D[2:0]$
- one control line: WE

One gated D-latch

CS 135

Reading a location in memory

CS 135




More Memory Details

- This is no longer the way actual memory is implemented.
 - fewer transistors, much more dense, relies on electrical properties
- But the logical structure is very similar.
 - address decoder
 - word select line
 - word write enable
- Two basic kinds of **RAM** (Random Access Memory)
 - **Static RAM (SRAM)**
 - fast, maintains data as long as power applied
 - **Dynamic RAM (DRAM)**
 - slower but denser, bit storage decays – must be periodically refreshed

Also, non-volatile memories: ROM, PROM, flash, ...


CS 135




Course Adminis-trivia

- **Lab this week and next is Project 1 completion**
 - **Project 1 code**
 - Dlc 'code checker' tells you if rules violated and if there is a syntax error
 - Review sequential logic
- **HW3, Teamwork 3, Teamwork 4 posted**
 - Solutions to HW1,2 and Quiz 1,2 posted
 - Work on Teamhomework 4 in class next class
- **Quiz 3 next Tuesday**
- **Exam 1: October 7**
- **Reading:**
 - You should have completed reading Chapters 1,2,3
 - Look up Univ. Maryland notes linked from lectures webpage

CS 135



Design of Sequential Logic Circuits



Summary of Digital Logic

- **Combinational logic**
- **Storage elements**
 - R-S Latches and D-latch
 - Concept of memory: address space and addressability
- **Sequential circuits – next!**
 - Recall the vending machine example
 - Progress from state (current total) to new state (new total)

CS 135

Compare and Contrast

- **Combinational Logic Circuits**
 - Make decisions
 - Same inputs always produce same output
 - Depends on what is happening now
- **Sequential Logic Circuits**
 - Make decisions and store information
 - Output depends on inputs AND state
 - Depends on what has happened in the past as well as what is happening now

CS 135

A Vending Machine

- **Input valid coins:**
 - Q (25cents) D (10) or N (5)
- **Keep track of current total**
 - Is it 75 cents or more ?
- **When it reaches 75 or more:**
 - Generate output
- **States of the machine ?**

CS 135

Finite State Machines

- The behavior of sequential circuits can be expressed using characteristic tables or finite state machines (FSMs).
 - FSMs consist of a set of nodes that hold the states of the machine and a set of arcs that connect the states.
 - Directed graph to represent a FSM
- Moore and Mealy machines are two types of FSMs that are equivalent.
 - They differ only in how they express the outputs of the machine.
 - Moore machines place **outputs on each node/state**
 - Associate an output with each state
 - Mealy machines present their **outputs on the transitions.**

CS 135

State Machine

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

CS 135

Sequential Logic Circuits - State

- The concept of state
 - the **state** of a system is a “**snapshot**” of all relevant elements at a moment in time.
 - a given system will often have only a **finite number of possible states**.
 - For many systems, we can define the rule which determines under what conditions a system can **move from one state to another**.

CS 135

Example

- the game of tic-tac-toe has only a certain number of possible dispositions of Xs and Os on the 3x3 grid.
- A given game of tic-tac-toe will progress through a subset of these possible states (until someone wins) - i.e. it traverses a specific path through “state space”, one move at a time.

CS 135

Sequential Logic Circuits

The diagram shows a box labeled "State Machine". On the left, an arrow labeled "Inputs" points into the box. On the right, an arrow labeled "Outputs" points out of the box. Inside the box, there are two sub-components: a yellow box labeled "Combinational Logic Circuit" and a light blue box labeled "Storage Elements". The "Combinational Logic Circuit" has an arrow pointing to the "Storage Elements". The "Storage Elements" has an arrow pointing back to the "Combinational Logic Circuit", forming a feedback loop. Additionally, the "Storage Elements" has an arrow pointing to the "Outputs".

- The output is a function of the current input and the previous state
- It is computed by the *combinational logic circuit*
- The state is stored in the *storage element*
- The new state is also a function of the previous state and the current input
- This can work only if we make **transitions from one state to another** at well-defined times - this is why they are called sequential circuits.

CS 135

Finite State Machines

- Many systems meet the following five conditions:
 - A finite number of **states**
 - A finite number of **external inputs**
 - A finite number of **external outputs**
 - An explicit specification of all allowed **state transitions**
 - An explicit specification of the rules for each **external output value**
- a microprocessor is a perfect candidate for description as a FSM.

CS 135

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.

CS 135

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.

CS 135

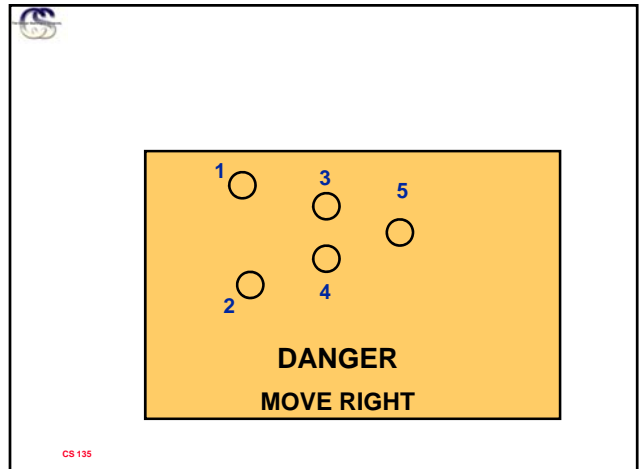
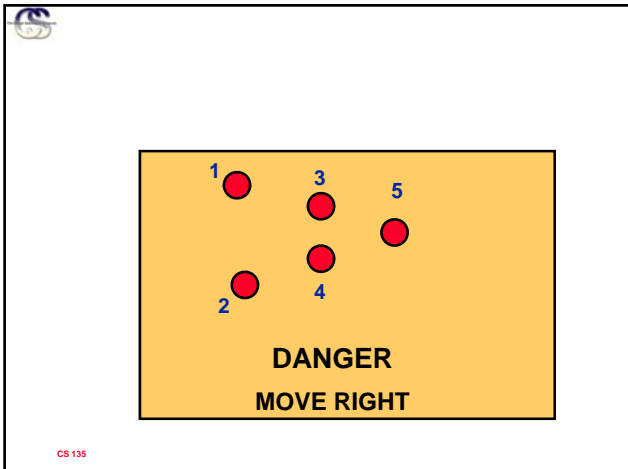
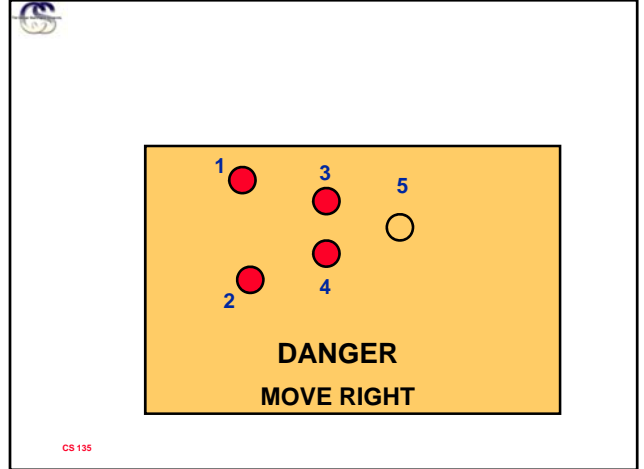
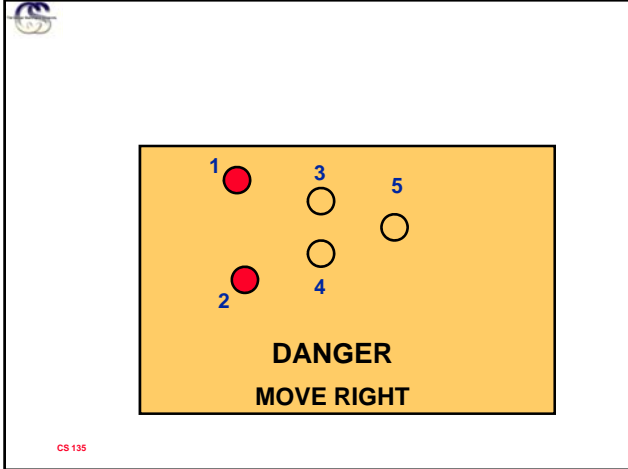
Storage

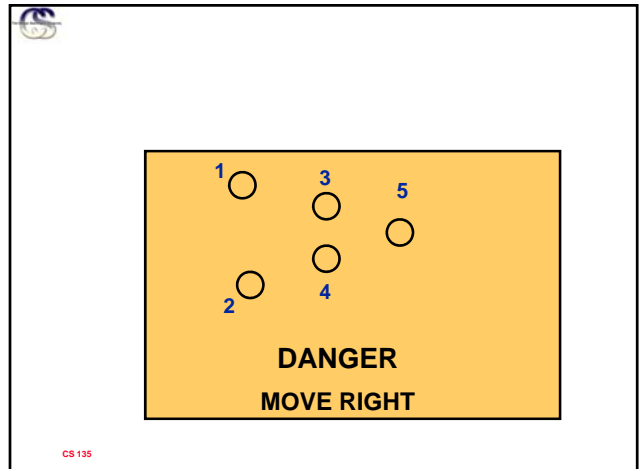
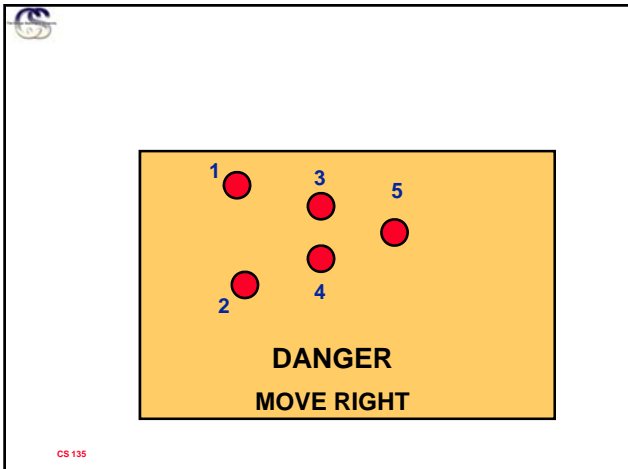
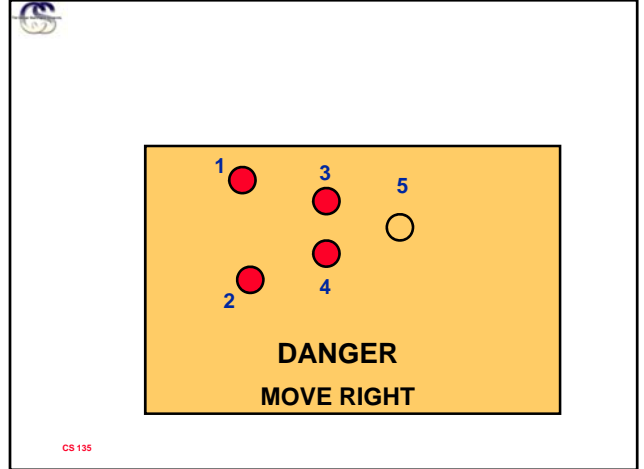
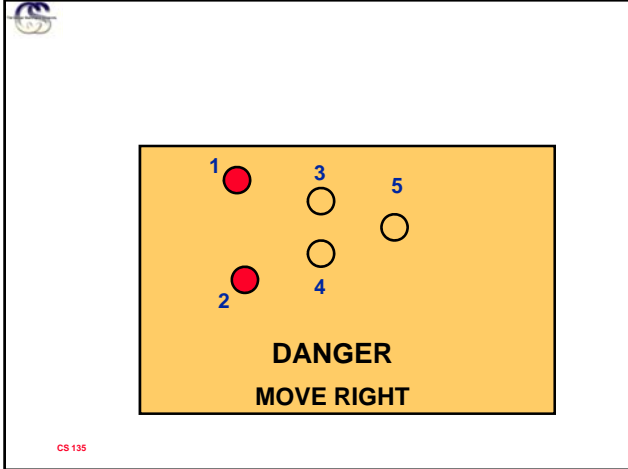
- Each master-slave flip-flop stores one state bit.
- The number of storage elements (flip-flops) needed is determined by the number of states (and the representation of each state).
 - Each bit can be 0 or 1 = 2 states
 - N bits can represent 2^N states
- Example: If a FSM has 12 states, then the circuit needs $\log_2 12 = 4$ storage elements.
 - Fewer the states, less hardware needed
 - Concept of Minimization of States for a given FSM

CS 135

Example: Blinking Traffic Sign

CS 135

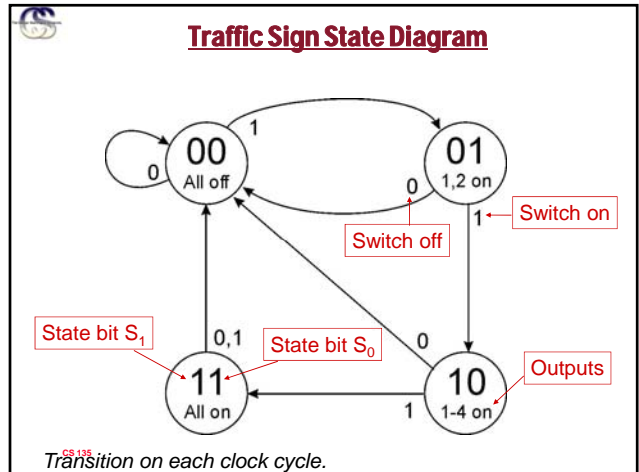




Example: Traffic Sign

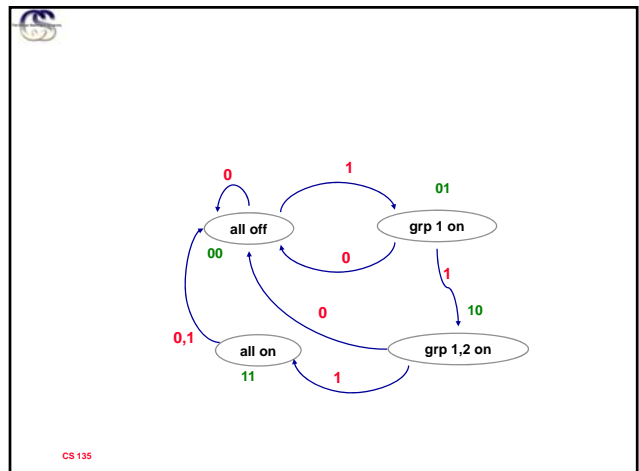
- A blinking traffic sign: How many states
- 4 states
 - No lights on
 - 1 & 2 on
 - 1, 2, 3, & 4 on
 - 1, 2, 3, 4, & 5 on
 - (repeat as long as switch is turned on)
- How many bits to represent the 4 states
- S_1S_0
 - With S_1S_0 values: 00, 01, 10, 11

CS 135



- Note we really have 3 groups of lights to be controlled = 3 control lines X,Y,Z
 - Group 1: Lights 1 and 2; controlled by Z
 - If Z=1 then Group 1 lights (1 and 2) are switched on
 - Group 2: lights 3 & 4; controlled by Y
 - Group 3: Light 5; controlled by X

CS 135



Finite State Machine Example - 2

- **When is group 1 on?**
 - in states 01, 10 and 11 - but only when the switch IN is on!
- **Logic expressions for X, Y, Z**
 - Depends on S_0 and S_1 and Input is on
 - If Input is off then X, Y, Z are at 0
- **can you come up with a logic expression for next state values of S_0 and S_1 ?**
 - Depends on current values of S_0 and S_1 and Input is on
 - Input off then both bits are set to 0 since next state is 00
 - Next state value of S_0 denoted S_0' = 1 if current state is 00 or current state 10 and $In=1$
- **When do we switch to the next state?**
 - the two bits of $S[1:0]$ are updated at every clock cycle
 - we have to make sure that the new state does not propagate to the combinational circuit input until the next clock cycle.

Traffic Sign Truth Tables

Outputs
(depend only on state: $S_1 S_0$)

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)

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.

CS 135

- $Z = (\bar{S}_1 S_0 + S_1 \bar{S}_0 + S_1 S_0) \cdot In$
- $Y = (S_1 \bar{S}_0 + S_1 S_0) \cdot In$
- $X = (S_1 S_0) \cdot In$

CS 135

Traffic Sign Logic

CS 135

From Logic to Processor Data Path

- The data path of a computer is all the logic used to process information.
 - Eg. data path of the LC-3.
- **Combinational Logic**
 - Decoders -- convert instructions into control signals
 - Multiplexers -- select inputs and outputs
 - ALU (Arithmetic and Logic Unit) -- operations on data
- **Sequential Logic**
 - State machine -- coordinate control signals and data movement
 - Registers and latches -- storage elements

CS 135

LC-3 Data Path


CS 135

CS 135

What Next ?


- Next topic: The von Neumann model of computer architecture
 - Basic components
 - How instructions are processed
 - The LC3 computer and instruction set

CS 135

 **Recall: what are Computers meant to do ?**

- We will be solving problems that are describable in English (or Greek or French or Hindi or Chinese or ...) and using a box filled with electrons and magnetism to accomplish the task.
 - This is accomplished using a system of well defined (sometimes) transformations that have been developed over the last 50+ years.

CS 135


 **Problem Transformation - levels of abstraction**

The desired behavior: the application

Natural Language
Algorithm
Program
Machine Architecture
Micro-architecture
Logic Circuits
Devices

The building blocks: electronic devices

CS 135

 **Putting it all together**

- The goal:
 - Turn a theoretical device - Turing's Universal Computational Machine - into an actual computer ...
 - ... interacting with data and instructions from the outside world, and producing output data.
- Smart building blocks:
 - We have at our disposal a powerful collection of combinational and sequential logic devices.
- Now we need a master plan ...

CS 135