

## Sequential Logic Circuits – Part II (Realizing Sequential Circuits)

CIT 595  
Spring 2008

CIT 595

1

## Sequential Circuits

- Output depends on stored information (current state) and may be on current inputs
- Built out of combinational logic and one or more memory/storage elements
- Examples seen so far: n-bit Register and m x n Memory
- Like combinational logic, we want a systematic way of analyzing/designing a sequential circuit

CIT 595

2

## Describing Behavior of Sequential Circuits

- With 1-bit storage, 2 possible states (values) that can be stored i.e. 0 or 1
  - With n-bits, there are  $2^n$  possible states
- The amount of information to be stored is finite
  - So total number of different possible *states* the information can be in also finite
- Thus sequential circuit is also known as **Finite State Machine(FSM)**
- The behavior of the circuit can be described using FSM model

CIT 595

3

## Finite State Machine (FSM)

- A Finite State Machine is an abstract model consisting:
  - *Finite set of states*
  - The *transitions* between those states
  - Along with the *actions* to be performed while in those state or during transitions
- The *state* reflects input changes from the time system started to present moment

CIT 595

4

## FSM Terminology

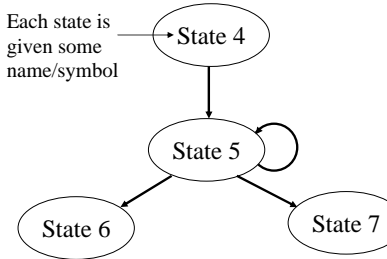
- **State Diagram:** Illustrates the form and function of a state machine
  - Usually drawn as a circle-and-arrow diagram
- **State:** is a unique configuration of information in a machine
- **Current/Present State:** configuration in which the machine is currently in
- **Next State:** The state to which the state machine makes the next transition, determined by the inputs present
- **Transition Arc:** A change from present state to next state

CIT 595

5

## State Diagram, State, Next State, Transition Arc

Each state is given some name/symbol



- For any given state, there is a **finite number of possible next states**

- One of the possible next states becomes the new present state, depending on the inputs present
  - For digital system, state changes on the clock cycle

Note: From this diagram it can be deduced that if the present state is State 5, then the previous state was either State 4 or 5 and the next state must be either 5, 6, or 7.

CIT 595

6

## Two Kinds of FSM Model

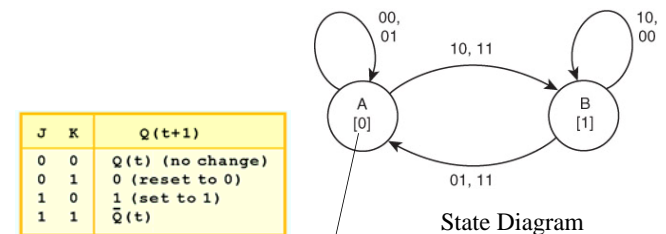
- Both machine model types stated below
  - Follow the basic characteristics of state machines
  - But differ in the way that outputs are produced
- **Moore Machine**
  - Outputs are independent of the inputs i.e. dependent on present state only
- **Mealy Machine**
  - Outputs are function of the present/current state and the present inputs

CIT 595

7

## Moore FSM

- Each state is associated with the output of the machine
  - Outputs are only dependent upon current state
- E.g. JK Flip-Flop as Moore Machine



If in state A then  
Output 0

CIT 595

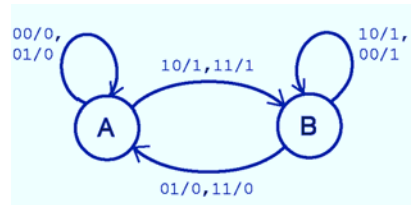
8

## Mealy FSM

- Outputs are produced as the machine makes a transition from one state to another
  - Outputs are dependent on current state and current input

J	K	Q(t+1)
0	0	Q(t) (no change)
0	1	0 (reset to 0)
1	0	1 (set to 1)
1	1	$\bar{Q}(t)$

Transition arch with 00/0 means when in state A and inputs are JK are 00 then output is 0.



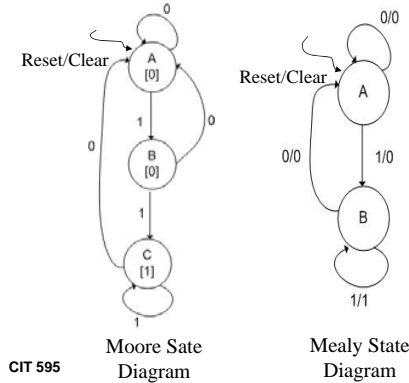
JK Flip-Flop represented as Mealy FSM

CIT 595

9

## Example: Sequence/Run Detector

- A binary sequence is transmitted 1-bit at a time. At one end of the line there is a sequential circuit that has to output a "1" when it sees **at least two subsequent 1s**. E.g. 01101110001011 etc..



Here "A" is starting state in both Models. Allows the FSM to be set to known state at beginning (indicated by "reset").

CIT 595

10

## FSM for Computer Hardware Application

- For actual hardware implementation
  - Requires memory element(s) to store state(s) and state is updated based on the present input with respect to clock
  - A block of combinational logic which determines the state transition
  - A second block of combinational logic that determines the output(s) of a FSM

CIT 595

11

## State Table

- Once you have conceptualized the problem in a state diagram (Moore or Mealy), you translate it to **State Table**
- Like Truth Table for Combinational Logic, a State Table enumerates
  - Inputs
  - Outputs with additional columns for current and next states of the sequential circuit/FSM

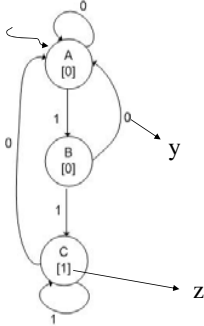
CIT 595

12

## State Table: Moore Sequence Detector

Input	Current State	Next State	Output
y	Q(t)	Q(t + 1)	z
0	A	A	0
1	A	B	0
0	B	A	0
1	B	C	0
0	C	A	1
1	C	C	1

Independent of y, only depends on current state



CIT 595

13

## State Encoding

- We need to convert letters A, B, C to particular binary combination of values such that they can be stored in flip-flops
  - Remember that our digital flip-flop is storing binary information
- Since there are 3 states, we can encode them in 2 bits
  - Implies that we have 2 Flip-Flops in sequential circuit
- Let A = 00, B = 01, C = 10
  - In this example we don't care about state 11

CIT 595

14

## State Table: Moore Sequence Detector (contd..)

y	Q(t)	Q(t + 1)	z
0	A	A	0
1	A	B	0
0	B	A	0
1	B	C	0
0	C	A	1
1	C	C	1



y	Q(t)		Q(t + 1)		z
	Q1	Q0	q1	q0	
0	0	0	0	0	0
1	0	0	0	1	0
0	0	1	0	0	0
1	0	1	1	0	0
0	1	0	0	0	1
1	1	0	1	0	1
0	1	1	x	x	x
1	1	1	x	x	x

Q1 Q0: current state variables

q1 q0: next state variables

x – don't care

CIT 595

15

## Finding the Output and Next State Function

- Draw up Kmaps, just like for combinational logic circuits, to come up the function for output and next state
- But before that:
  - Need to pick the *type of flip-flop* to be used
  - The most straight forward choice is D Flip-Flops
    - State of the flop is simply the inputs
  - Other than D Flip-Flop, need to do something extra – see an example later

CIT 595

16

### Kmap for Moore Sequence Detector (Using D Flip-Flop)

$q_1 = yQ_0 + yQ_1$   
 $= y(Q_0 + Q_1)$

$q_0 = y\bar{Q}_1\bar{Q}_0$

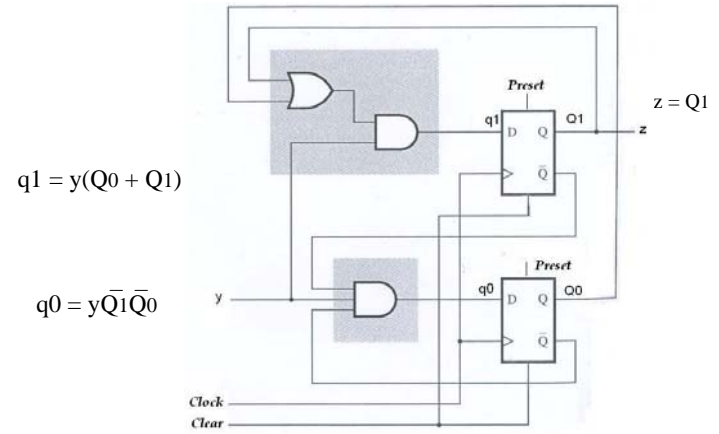
$z = Q_1$

y	Q(t) Q <sub>1</sub> Q <sub>0</sub>	Q(t + 1) q <sub>1</sub> q <sub>0</sub>	z
0	0 0	0 0	0
1	0 0	0 1	0
0	0 1	0 0	0
1	0 1	1 0	0
0	1 0	0 0	1
1	1 0	1 0	1
0	1 1	x : x	x
1	1 1	x : x	x

CIT 595

17

### Logic Diagram of Moore Sequence Detector



CIT 595

18

### State Table: Mealy Sequence Detector

y	Q(t)	Q(t + 1)	z
0	A	A	0
1	A	B	0
0	B	A	0
1	B	B	1

→ Dependent on y and current state

Inputs: y - 1 bit input, Q(t)- Current State of the Circuit  
 Outputs: z - output, Q(t + 1) - Next State of the Circuit

CIT 595

19

### State Table: Mealy Sequence Detector (contd..)

y	Q(t)	Q(t + 1)	z
	Q	q	
0	A	A	0
1	A	B	0
0	B	A	0
1	B	B	1

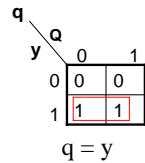
**Q = current state variables**  
**q = next state variables**

Need only 1 flip-flop to represent 2 states.  
 Let A = 0 and B = 1.

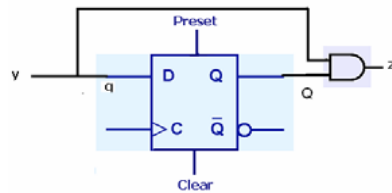
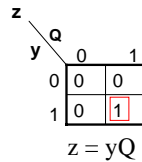
CIT 595

20

### Kmap for Mealy Sequence Detector (Using D Flip-Flop)



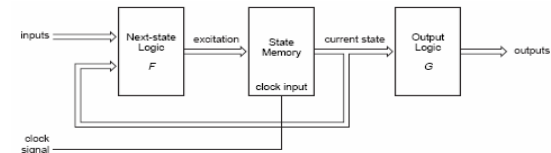
y	Q(t) Q	Q(t + 1) q	z
0	0	0	0
1	0	1	0
0	1	0	0
1	1	1	1



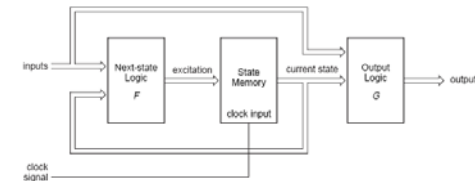
CIT 595

21

### General Mealy and Moore Machine Structure



General Moore Machine



General Mealy Machine

CIT 595

22

### Moore vs. Mealy

- Moore and Mealy FSMs can be functionally equivalent
- Mealy FSM has richer description and usually requires smaller number of states
  - Hence reduces the number of memory elements it needs
- Mealy FSM computes outputs as soon as inputs change
  - Mealy FSM responds one clock cycle sooner than equivalent Moore FSM

CIT 595

23

### For Flip-Flops other than D Flip-Flops

- We need to find the function for next state based on the required inputs to Flip-Flop
  - E.g. If JK FF then what should JK in order to make values  $q_1q_0$  to be the next state in sequence detector example?

(a) JK Flip-Flop				(b) SR Flip-Flop			
Q(t)	Q(t+1)	J	K	Q(t)	Q(t+1)	S	R
0	0	0	X	0	0	0	X
0	1	1	X	0	1	1	0
1	0	X	1	1	0	0	1
1	1	X	0	1	1	X	0

Excitation Table

CIT 595

24

### Sequence Detector Using JK Flip-Flop

(a) JK Flip-Flop

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

y	Q(t) Q <sub>1</sub> Q <sub>0</sub>	Q(t+1) q <sub>1</sub> q <sub>0</sub>	For J <sub>1</sub> K <sub>1</sub>	For J <sub>0</sub> K <sub>0</sub>	z
0	0 0	0 0	0 x	0 x	0
1	0 0	0 1	0 x	1 x	0
0	0 1	0 0	0 x	x 1	0
1	0 1	1 0	1 x	x 1	0
0	1 0	0 0	x 1	0 x	1
1	1 0	1 0	x 0	0 x	1
0	1 1	x x	x x	x x	x
1	1 1	x x	x x	x x	x

J<sub>1</sub>

y	Q <sub>1</sub> Q <sub>0</sub>	00	01	11	10
0	0	0	x	x	x
1	0	1	x	x	x

J<sub>1</sub> = yQ<sub>0</sub>

K<sub>1</sub>

y	Q <sub>1</sub> Q <sub>0</sub>	00	01	11	10
0	x	x	x	1	1
1	x	x	x	0	0

K<sub>1</sub> =  $\bar{y}Q_1$

J<sub>0</sub>

y	Q <sub>1</sub> Q <sub>0</sub>	00	01	11	10
0	0	x	x	x	0
1	1	x	x	x	0

J<sub>0</sub> = y $\bar{Q}_1$

K<sub>0</sub>

y	Q <sub>1</sub> Q <sub>0</sub>	00	01	11	10
0	x	1	x	x	x
1	x	1	x	x	x

K<sub>0</sub> = 1

CIT 595

### Summary of Steps for Designing Sequential Circuits

- Translate the state diagram into *State Table*
- Decide the kind of *Flip-Flop* you want to use in order to determine next state function
- Draw the Kmaps to determine the function
- Draw the logic circuit based on Kmap expression
  - i.e. with flip-flops and combinational logic

CIT 595 26

### Traffic Light Example

- A blinking traffic sign is controlled by a switch
  - No lights if switch is off
  - If switch is on then
    - 1 & 2 on
    - 1, 2, 3, & 4 on
    - 1, 2, 3, 4, & 5 on
    - All off
    - repeat as long as switch is turned on

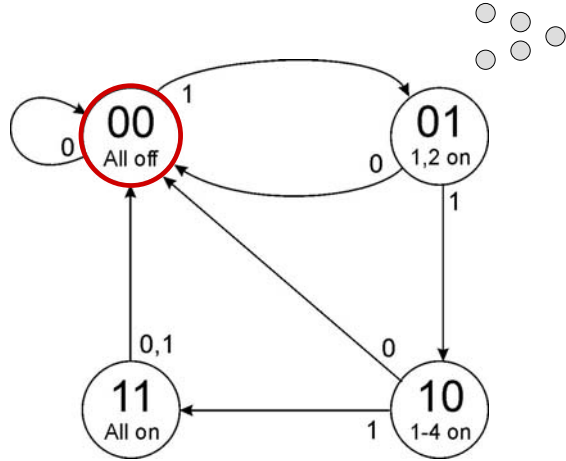
CIT 595 27

### Traffic Sign State Diagram – Moore Machine

Transition on each clock cycle

CIT 595 28

Traffic Sign State Diagram: State 00

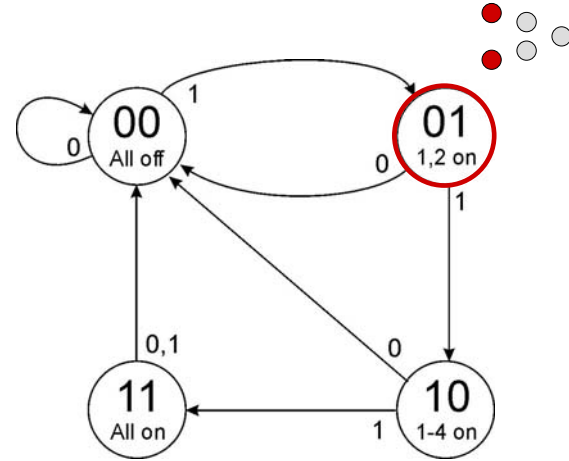


CIT 595

29

Transition on each clock cycle

Traffic Sign State Diagram: State 01

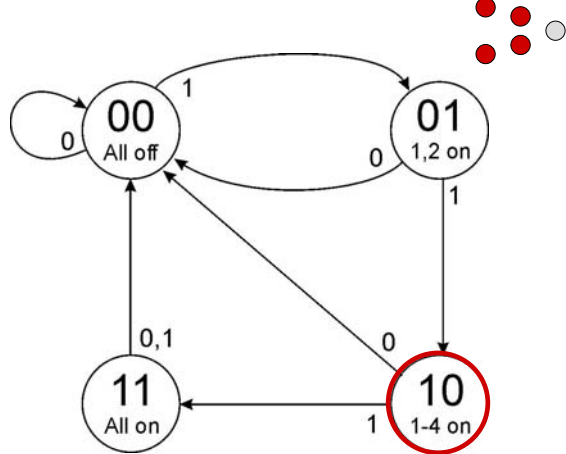


CIT 595

30

Transition on each clock cycle

Traffic Sign State Diagram: State 10

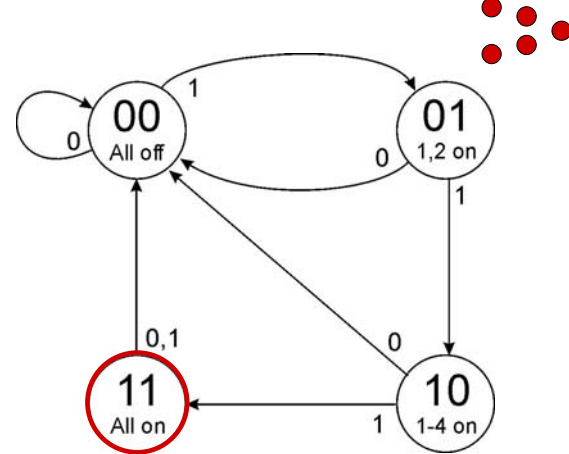


CIT 595

31

Transition on each clock cycle

Traffic Sign State Diagram: State 11

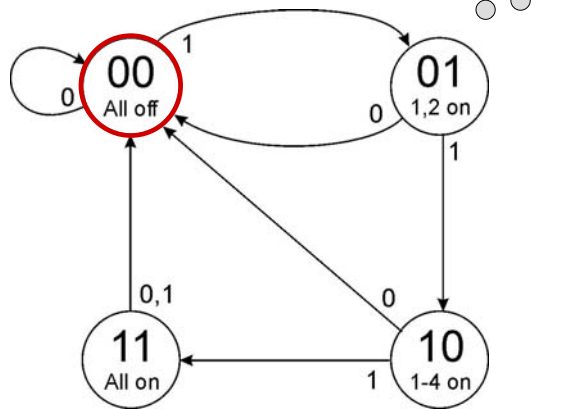


CIT 595

32

Transition on each clock cycle

## Traffic Sign State Diagram: State 00



CIT 595

33

Transition on each clock cycle

## Traffic Sign State Tables

Outputs depend only on current state:  $Q_1, Q_0$  (depend on state and input)

$Q_1$	$Q_0$	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:  $q_1, q_0$  (depend on state and input)

S	$Q_1$	$Q_0$	$q_1$	$q_0$
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

CIT 595

34

## Traffic Sign Kmap

$q_1$	S	$Q_1 Q_0$	00	01	11	10
0	0		0	0	0	0
1	1		0	1	0	1

$$q_1 = S \bar{Q}_1 Q_0 + S Q_1 \bar{Q}_0$$

$q_0$	S	$Q_1 Q_0$	00	01	11	10
0	0		0	0	0	0
1	1		1	0	0	1

$$q_0 = S \bar{Q}_0$$

Y	$Q_1$	$Q_0$
0	0	1
0	0	0
1	1	1

$$Y = Q_1$$

Z and X are straight forward:

$$Z = Q_1 + Q_0$$

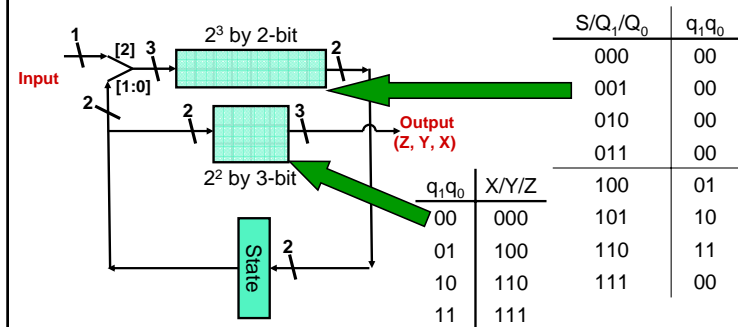
$$X = Q_1 Q_0$$

CIT 595

35

## Programmable State Machines

- What if we want to change the pattern of the sign?
  - An alternative state machine implementation
  - Use a memory indexed by state number



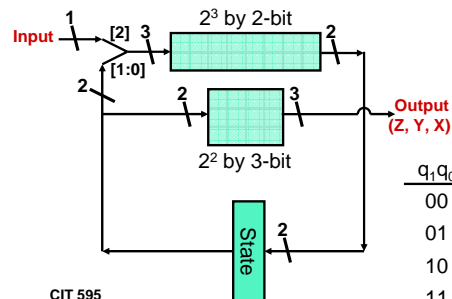
CIT 595

36

## Programmable State Machines

- Change to a two-state pattern:

- All off
  - All on
- State: 00 → State: 10 → State: 00



S/Q <sub>1</sub> /Q <sub>0</sub>	q <sub>1</sub> q <sub>0</sub>
000	00
001	--
010	00
011	--
100	10
101	--
110	00
111	--

q <sub>1</sub> q <sub>0</sub>	X/Y/Z
00	000
01	--
10	111
11	--

CIT 595

37

## Other uses of Finite State Machines

- Machine respond to set of events by generating predictable responses based on the history of prior events (current state)
- Besides logic design, broadly used computer science
  - E.g. Compiler and Programming languages Theory
  - Known as Deterministic Finite Automata (DFA)
  - E.g. FSM for accepting variable name in programming language

CIT 595

38

## Designing Digital Circuits

- Designing Digital Circuits:
  - Analysis** explores the relationship between a circuits inputs and its outputs
  - Synthesis** creates logic diagrams using the values specified in a truth/stable table
- Digital systems designers must also be mindful of the physical behaviors of electronic components
  - E.g. Propagation delays
    - Occur between the time when a circuit's inputs are energized and when the output is accurate and stable

CIT 595

39

## Designing Digital Circuits (contd..)

- Digital designers rely on specialized software
  - After few states Mealy and Moore Models become cumbersome
  - Thus, software is an enabler for the construction of better hardware
- E.g. Verilog is Hardware Descriptive Language (HDL) where you can synthesize hardware components as if you writing a high-level language

CIT 595

40

## Example of Verilog Language

```
module toplevel(clock,reset);
input clock;
input reset;
reg flop1; //declare a 1-bit memory unit
reg flop2;
always @ (posedge reset or posedge clock) begin
if (reset)
begin
flop1 <= 0;
flop2 <= 1;
end
else
begin
flop1 <= flop2;
flop2 <= flop1;
end
end
endmodule
```

CIT 595

41

## Designing Digital Circuits (contd..)

- To implement a simple, specialized algorithm and have execution speed as fast as possible
  - Hardware solution is often preferred
- This is the idea behind *embedded systems*, which are small special-purpose computers
  - Carry out limited set of tasks with the domain of a larger system
  - E.g. microwaves, aircrafts & automobile controls, cell phones (???)
- Embedded systems use configurable or custom made hardware

CIT 595

42

## Aside: Potential Project Topic

- Embedded Systems
  - Software Tools for digital circuit designs e.g. Verilog, VHDL
  - Programmable (Configurable) Logic Devices and the companies that make them
  - Embedded Operating System e.g. Windows CE
  - Selling hardware designs as Open Source Hardware (HDL code sharing)
    - What about IP (intellectual property)?

CIT 595

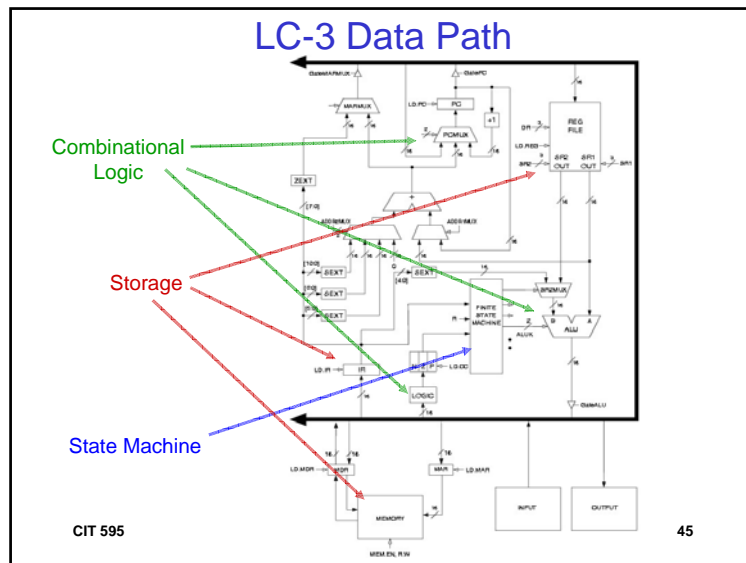
43

## From Logic to Data Path

- The data path of a computer is all the logic used to process information
  - See the data path of the LC-3 on next slide
- **Combinational Logic**
  - Decoders help convert instructions into control signals
  - Multiplexers help select inputs and outputs
  - ALU (Arithmetic and Logic Unit) perform operations on data
- **Sequential Logic**
  - Memory for storage
  - State machine (Control Unit)

CIT 595

44



### Looking Forward...

- We've touched on basic digital logic
  - Gates
  - Storage (latches, flip-flops, memory)
  - Combinational & Sequential Circuits
- Seen some simple circuits
  - Shifter, mux, decoder, adder/subtractor, 2-bit ALU
  - Register, 4 x 3 Memory, Sequence Detector
  - Hard-coded traffic sign state machine
  - Programmable traffic sign state machine
- Up next: Putting it all together - a computer as a (simple?) state machine

CIT 595 46