CPE/EE 422/522
Advanced Logic Design
L03

Electrical and Computer Engineering
University of Alabama in Huntsville

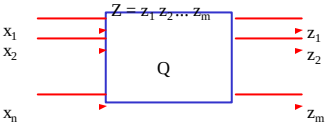
Outline

- What we know
  - Combinational Networks
    - Analysis, Synthesis, Simplification, Building Blocks, PALs, PLAs, ROMs
  - Sequential Networks: Basic Building Blocks
- What we do not know
  - Design: Mealy, Moore
  - Sequential Network Timing
  - Setup and hold times
  - Max clock frequency

Sequential Networks

- Have memory (state)
  - Present state depends not only on the current input, but also on all previous inputs (history)
  - Future state depends on the current input and state

$X = x_1\ x_2 \ldots\ x_n$
$Q = Q_1\ Q_2 \ldots\ Q_k$
$Z = z_1\ z_2 \ldots\ z_m$

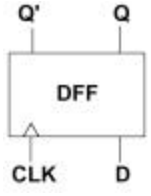


$Z(t) = F(X(t), Q(t))$

$Q(t^+) = G(X(t), Q(t))$

Flip-flops are commonly used as storage devices: D-FF, JK-FF, T-FF

Review: Clocked D Flip-Flop with Rising-edge Trigger



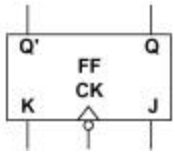
D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

$Q^+ = D$

Next state

The next state in response to the rising edge of the clock is equal to the D input before the rising edge

Review: Clocked JK Flip-Flop



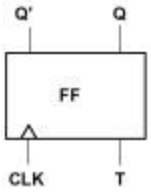
J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

$Q^+ = JQ' + K'Q$

Next state

JK = 00 => no state change occurs
JK = 10 => the flip-flop is set to 1, independent of the current state
JK = 01 => the flip-flop is always reset to 0
JK = 11 => the flip-flop changes the state $Q^+ = Q'$

Review: Clocked T Flip-Flop



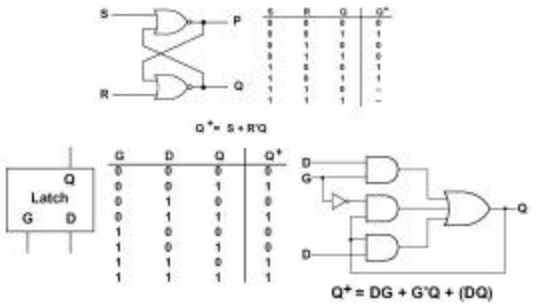
T	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

$Q^+ = QT' + Q'T = Q \oplus T$

Next state

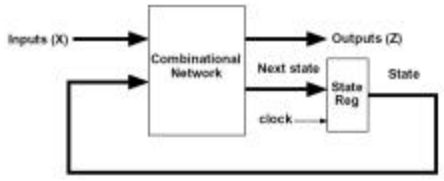
T = 1 => the flip-flop changes the state $Q^+ = Q'$
T = 0 => no state change

Review: S-R Latch, Transparent D-Latch



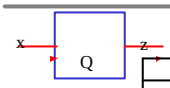
Mealy Sequential Networks

General model of Mealy Sequential Network



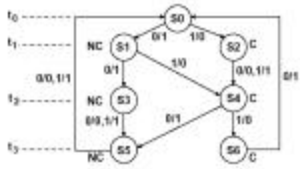
(1) X inputs are changed to a new value
(2) After a delay, the Z outputs and next state appear at the output of CM
(3) The next state is clocked into the state register and the state changes

An Example: 8421 BCD to Excess3 BCD Code Converter



X (inputs)				Z (outputs)			
t3	t2	t1	t0	t3	t2	t1	t0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

State Graph and Table for Code Converter



	NS		Z	
PS	X = 0	X = 1	X = 0	X = 1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	–	1	–

State Assignment Rules

I. States which have the same next state (NS) for a given input should be given adjacent assignments (look at the columns of the state table).

II. States which are the next states of the same state should be given adjacent assignments (look at the rows).

III. States which have the same output for a given input should be given adjacent assignments.

I. (1,2) (3,4) (5,6) (In the X=1 column, S_1 and S_2 both have NS S_4; in the X=0 column, S1 & S4 have NS S5, and S5 & S6 have NS S0)

II. (1,2) (3,4) (5,6) (S_1 & S_2 are NS of S_0; S_3 & S_4 are NS of S_1; and S_5 & S_6 are NS of S_4)

III. (0,1,4,6) (2,3,5)

Q2 Q3 \ Q1	0	1
00	S0	S1
01		S2
11	S5	S3
10	S6	S4

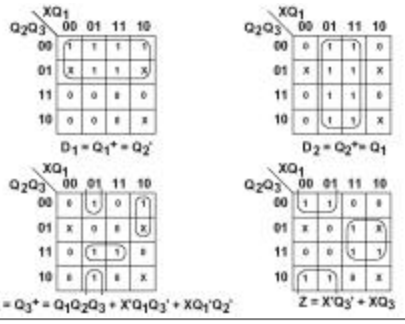
Transition Table

	NS		Z	
PS	X=0	X=1	X=0	X=1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	–	1	–

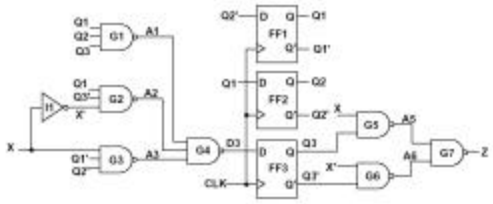
	Q1*Q2*Q3*		Z	
Q1Q2Q3	X=0	X=1	X=0	X=1
000	100	101	1	0
100	111	110	1	0
101	110	110	0	1
111	011	011	0	1
110	011	010	1	0
011	000	000	0	1
010	000	xxx	1	x
001	xxx	xxx	x	x

S0 = 000, S1 = 100, S2 = 101, S3 = 111, S4 = 110, S5 = 011, S6 = 010

K-maps



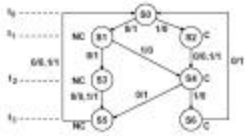
Realization

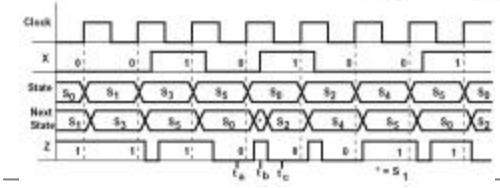


Sequential Network Timing

- Code converter
  - X = 0010_1001 =>
    Z = 1110_0011

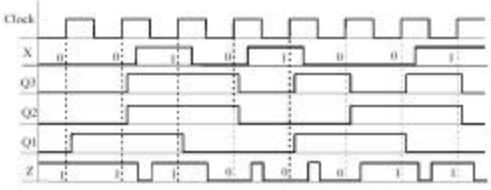
Changes in X are not synchronized with active clock edge =>
glitches (false output), e.g. at tb





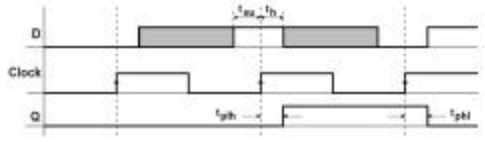
Sequential Network Timing (cont'd)

Timing diagram assuming a propagation delay
of 10 ns for each flip-flop and gate
(State has been replaced with the state of three flip-flops)



Setup and Hold Times

- For a real D-FF
  - D input must be stable for a certain amount of time before the active edge of clock cycle => *Setup time*
  - D input must be stable for a certain amount of time after the active edge of the clock => *Hold time*
- Propagation time: from the time the clock changes to the time the output changes



Manufacturers provide minimum tsu, th, and maximum tplh, tphl

Maximum Clock Frequency

$t_{c\ max}$ - Max propagation delay through the combinational network

$t_{p\ max}$ - Max propagation delay from the time the clock changes to the flip-flop output changes { = max(tplh, tphl)}

t_{ck} - Clock period

$$t_{c\ max} + t_{p\ max} \le t_{ck} - t_{su}$$

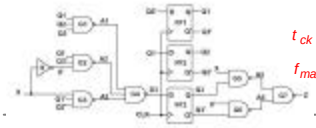
$$t_{ck} \ge t_{c\ max} + t_{p\ max} + t_{su}$$

Example:

$t_{p\ max} = 15\ ns$, $t_{su} = 5\ ns$, $t_{gate} = 15\ ns$

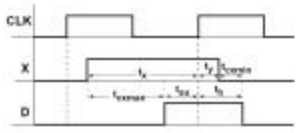
$$t_{ck} = 2 * 15 + 15 + 5 = 50\ ns$$

$$f_{max} = \frac{1}{50\ ns} = 20\ MHz$$



Hold Time Violation

- Occur if the change in Q fed back through the combinational network and cause D to change too soon after the clock edge



Hold time is satisfied if:

$t_{p\ min} + t_{c\ min} \geq t_h$

What about X?

Make sure that input changes propagate to the flip-flops inputs such that setup time is satisfied.

$t_x \geq t_{cx\ max} + t_{su}$

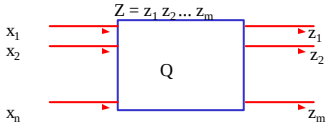
Make sure that X does not change too soon after the clock.
If X changes at time ty after the active edge, hold time is satisfied if

$t_y \geq t_h - t_{cx\ min}$

Moore Sequential Networks

Outputs depend only on present state!

$X = x_1\ x_2 \ldots\ x_n$
$Q = Q_1\ Q_2 \ldots\ Q_k$
$Z = z_1\ z_2 \ldots\ z_m$

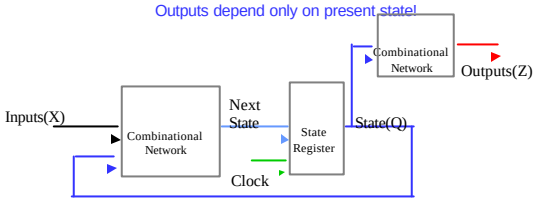


$Z(t) = F(Q(t))$

$Q(t^+) = G(X(t), Q(t))$

General Model of Moore Sequential Machine

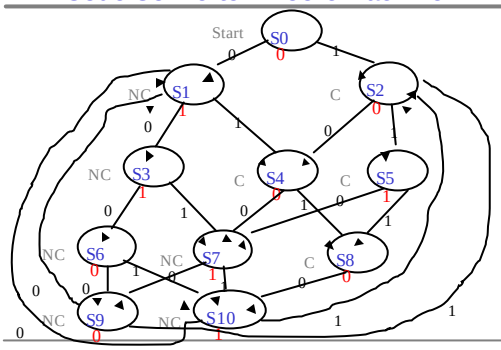
Outputs depend only on present state!



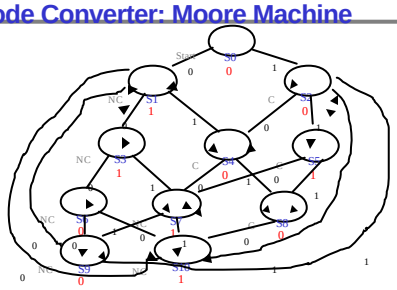
$X = x_1\ x_2 \ldots\ x_n$
$Q = Q_1\ Q_2 \ldots\ Q_k$
$Z = z_1\ z_2 \ldots\ z_m$

$Q(t^+) = G(X(t), Q(t))$
$Z(t) = F(Q(t))$

Code Converter: Moore Machine



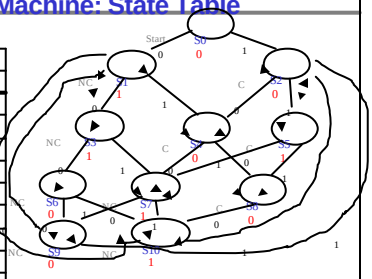
Code Converter: Moore Machine



Do we need state S0?
How many states does Moore machine have?
How many states does Mealy machine have?

Moore Machine: State Table

PS	NS		Z
	X=0	X=1	
S0	S1	S2	0
S1	S3	S4	1
S2	S4	S5	0
S3	S6	S7	1
S4	S7	S8	0
S5	S7	S8	1
S6	S9	S10	0
S7	S9	S10	1
S8	S10	-	0
S9	S1	S2	0
S10	S1	S2	1



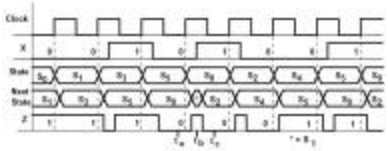
Note: state S0 could be eliminated (S0 == S9), if S9 was start state!

Moore Machine Timing

- X = 0010_1001 => Z = 1110_0011

Moore

Mealy



State Assignments

Guidelines to reduce the amount of combinational logic

I. States which have the same next state (NS) for a given input should be given adjacent assignments (look at the columns of the state table).

II. States which are the next states of the same state should be given adjacent assignments (look at the rows).

III. States which have the same output for a given input should be given adjacent assignments.

Rule I: (S0, S9, S10), (S4, S5), (S6, S7)
Rule II: (S1, S2), (S3, S4), (S4, S5), (S6, S7), (S7, S8), (S9, S10)
Rule III: (S0, S2, S4, S6, S8, S9) (S1, S3, S5, S7, S10)

S0 – 0010
S1 - 0111
....
S10 - 0100

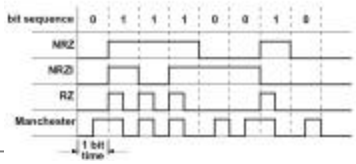
Q1Q2 / Q3Q4	00	01	11	10
00	S9	S10	S8	
01				S5
11		S1	S3	S4
10	S0	S2	S7	S6

PS	NS X=0	NS X=1	Z
S0	S1	S2	0
S1	S3	S4	1
S2	S4	S5	0
S3	S6	S7	1
S4	S7	S8	0
S5	S7	S8	1
S6	S9	S10	0
S7	S9	S10	1
S8	S10	-	0
S9	S1	S2	0
S10	S1	S2	1

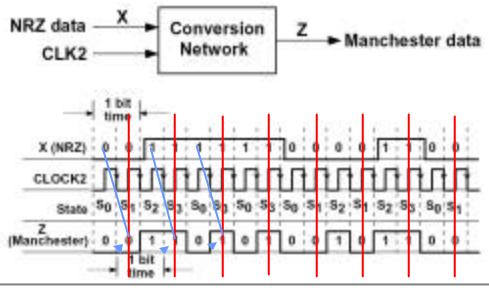
Moore Machine: Another Example

A Converter for Serial Data Transmission: NRZ-to-Manchester

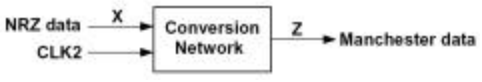
- Coding schemes for serial data transmission
  - NRZ: nonreturn-to-zero
  - NRZI: nonreturn-to-zero-inverted
    - 0 in input sequence – the bit transmitted is the same as the previous bit;
    - 1 in input sequence – transmit the complement of the previous bit
  - RZ: return-to-zero
    - 0 – 0 for full bit time; 1 – 1 for the first half, 0 for the second half
  - Manchester

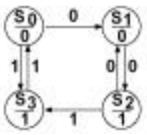


Moore Network for NRZ-to-Manchester



Moore Network for NRZ-to-Manchester

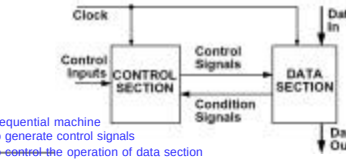




Present State	Next State X = 0	Next State X = 1	Present Output (Z)
S_0	S_1	S_3	0
S_1	S_2	–	0
S_2	S_1	S_3	1
S_3	–	S_0	1

Synchronous Design

- Use a clock to synchronize the operation of all flip-flops, registers, and counters in the system
  - all changes occur immediately following the active clock edge
  - clock period must be long enough so that all changes flip-flops, registers, counters will have time to stabilize before the next active clock edge
- Typical design: Control section + Data Section

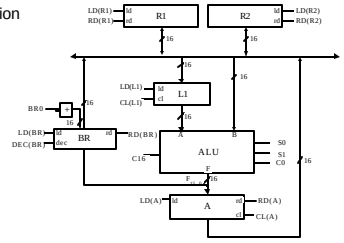


Data registers
Arithmetic Units
Counters
Buses, Muxes, ...

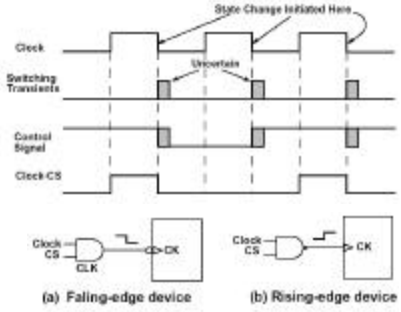
Sequential machine
to generate control signals
~~to control the~~ operation of data section

An Example

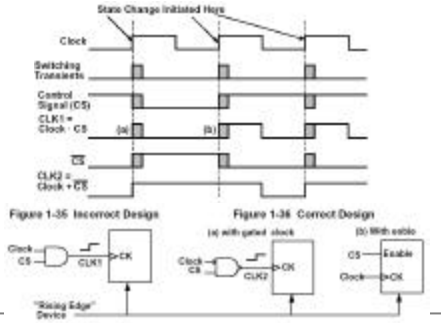
- Data section // s= n*(n+a) // R1=n, R2=a // R1=s
- Design flowchart for SMUL operation
- Design Control section
- S0 S1 F
  0 0 B
  0 1 B – C0
  1 0 B + C0
  1 1 A + B



Timing Chart for System with Falling-edge Devices



Timing Chart for System with Rising-edge Devices



Principles of Synchronous Design

- Method
  - All clock inputs to flip-flops, registers, counters, etc., are driven directly from the system clock or from the clock ANDed with a control signal
- Result
  - All state changes occur immediately following the active edge of the clock signal
- Advantage
  - All switching transients, switching noise, etc., occur between the clock pulses and have no effect on system performance

Asynchronous Design

- Disadvantage - More difficult
  - Problems
    - Race conditions: final state depends on the order in which variables change
    - Hazards
  - Special design techniques are needed to cope with races and hazards
- Advantages = Disadvantages of Synchronous Design
  - In high-speed synchronous design propagation delay in wiring is significant => clock signal must be carefully routed so that it reaches all devices at essentially same time
  - Inputs are not synchronous with the clock – need for synchronizers
  - Clock cycle is determined by the worst-case delay

To Do

- Read
  - Textbook chapters 1.6, 1.7, 1.8, 1.10, 1.11, 1.12