Digital Logic Design

Chapter 5: Sequential Circuits Design

Sequential Logic

Logic That Remembers

Sequential Logic

- Memory added to Combinational Logic
- Memory elements: Output is held in One of Two Stable States
  - Latches
  - Flip Flops
- Identified by the presence of Feedback
- Output depends on Inputs & State of the Circuit

Sequential Logic Circuits

- Classification
  - Synchronous: External timing signal determines output updates
  - Asynchronous: Outputs are updated following changes in state or inputs after propagation delays
- Structural Classification:
  - Moore: Output only derived from Flip Flops
  - Mealy: Output is combinational function of Flip Flops and inputs

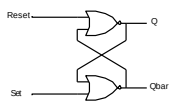
Hazards and Sequential Logic

- In synchronous sequential circuits, most of the glitches that may occur do not cause problems because they occur in the part of the clock cycle where they do not affect the flip-flops.
- However, in asynchronous sequential circuits changes occurring at any time can affect the signals on the feedback loops and cause the circuit to enter in an incorrect state.

Latches

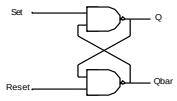
- Binary Storage Elements
  - Asynchronous in nature: No Clock
  - Types include SR and D
  - NOR and NAND implementations possible

Set	Reset	Q	Qbar	
1	0	1	0	Latch in Set State
0	0	1	0	
0	1	0	1	Reset State of the Latch
0	0	0	1	
1	1	0	0	Next State is Undefined



NAND Implementation of Latch

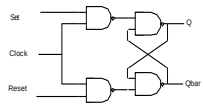
Set	Reset	Q	Qbar	
0	1	1	0	Latch in Set State
1	1	1	0	
1	0	0	1	Reset State of the Latch
1	1	0	1	
0	0	1	1	Next State is Undefined



The *Clocked* Latch

- A control input (called Clock) is added
  - Output changes only when this input is HIGH
  - Allows more control over the time when state changes are required

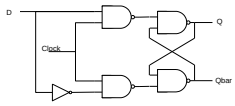
Clock	Set	Reset	Q	Qbar	
0	X	X	1	0	No change: Previous state held
1	0	0	1	0	
1	0	1	0	1	Reset State
1	1	0	1	0	Set State
1	1	1	1	1	Undefined



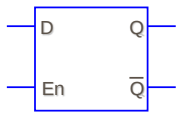
The *D* Latch

- Avoids ambiguous inputs
- Used as Data (*D*) storage device enabled by the clock
- Also called a Transparent Latch

Clock	D	Q	
0	X	0	No change:
1	0	0	Reset State
1	1	1	Set State



D Latch Symbol

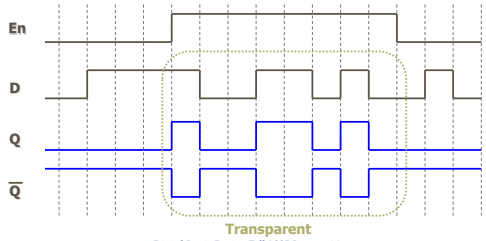


En	D	Q	$\overline{Q}$
0	X	NC	NC
1	0	0	1
1	1	1	0

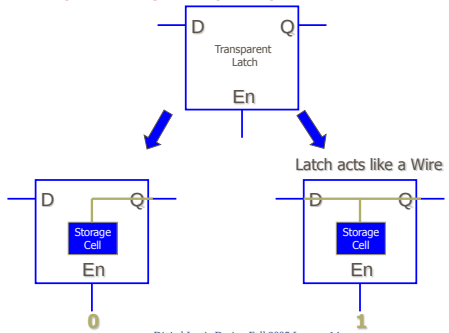
NC: No Change

Latch is Transparent

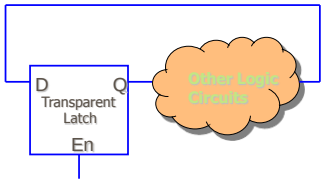
- D Latch is called “transparent” or “level sensitive”
- Output follows input instantaneously



Transparency Property

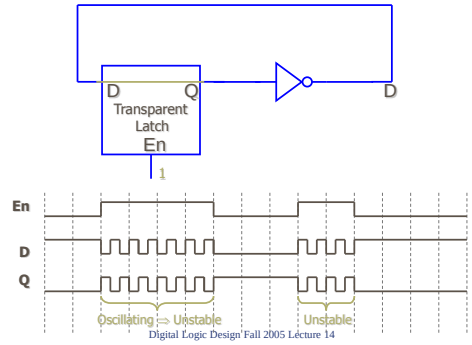


Problem of Transparency

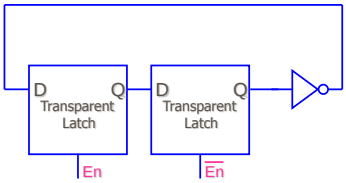


- A momentary input change tunnels through the latch and the entire circuitry
- What problem this can cause?

Problem of Transparency



Eliminating Transparency

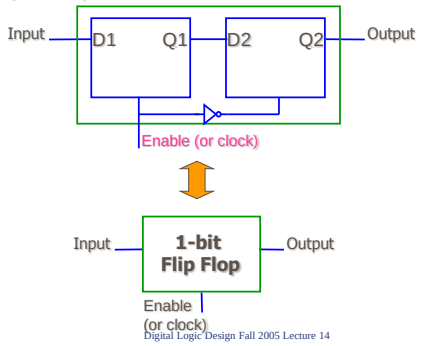


- Separating the input and output, so they are independently controlled
- Only open one gate at a time to avoid tunneling

The Flip Flop

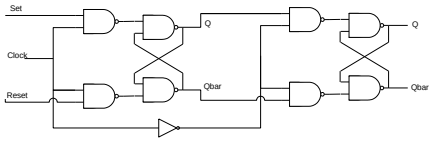
- Two ways to design F/F
- Construction Methods
  - Master Slave
  - Edge-Triggered
- Master Slave F/F
  - Two Cascaded Latches and an Inverter
    - First Latch is enabled during High time of the Clock
    - Second Latch is enabled during Low time of Clock
- Edge Triggered Latch
  - *Outputs can change only at the Transition of Clock*
  - *Hi-to-Low (falling) and Low-to-High (rising) Edge*
- Designed for Synchronous Logic
  - Time Window for Data to Pass to the Output is Extremely Short

Flip-Flop (F/F)

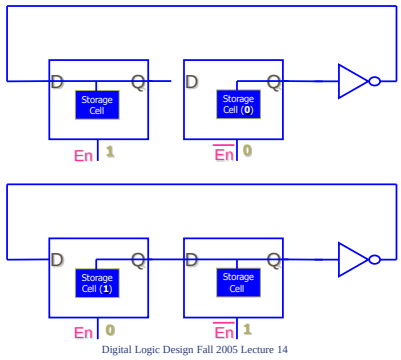


The Master-Slave SR Flip Flop

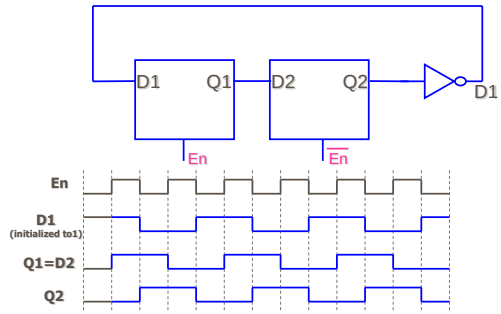
- Two Cascaded Latches and an Inverter
  - First Latch is enabled during High time of the Clock
  - Second Latch is enabled during Low time of Clock



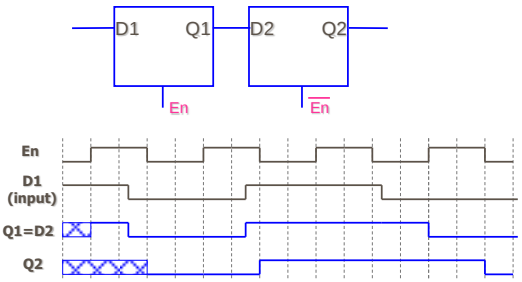
Behavior of Master–Slave FF



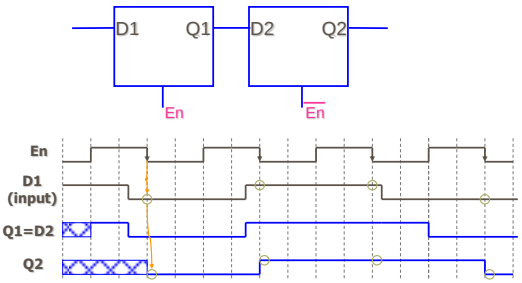
Behavior of Master–Slave FF



Behavior of Master–Slave FF



Behavior of Master–Slave FF



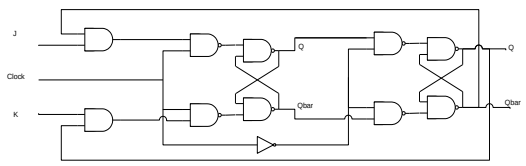
The JK Flip Flop

- SR Flip Flop or Latch has *One Prohibited* input State
- JK overcomes the SR limitation of both inputs being asserted simultaneously
- The Behavior is described by the following Table
  - Both J and K being TRUE cause a TOGGLE of the State

J	K	Next State
0	0	Q
0	1	0
1	0	1
1	1	Q′

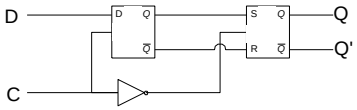
The JK Flip Flop

- Two AND gates added to the SR FF
  - The current state of the FF allows the RIGHT input to pass through

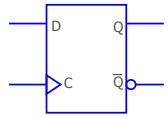


D-type ↓ Edge Triggered Flip Flop

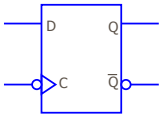
- Cascade of a D and an SR Latch



Flip Flops Symbols



Positive Edge Triggered D Flip Flop



Negative Edge Triggered D Flip Flop

Important FF Parameters

- Setup Time: Minimum time that the input e.g. *D* must be present and stable before the clock edge
- Hold Time: Minimum time that the input e.g. *D* must be held stable after the clock edge
  - A violation of Setup or Hold Time will result in unpredictable outputs
- Propagation Delay or Clock to Q delay:
  - The time after the clock edge to the output being stable in the new state

The Characteristic Table

- Defines the Logical Properties of the Flip Flop
- Similar to the Truth Table for Combinational Circuits

S	R	Next State Q(t+1)	
0	0	Q(t)	Hold
0	1	0	Reset
1	0	1	Set
1	1	?	Undefined

J	K	Next State Q(t+1)	
0	0	Q(t)	Hold
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Toggle

D	Next State Q(t+1)	
0	0	Reset
1	1	Set

T	Next State Q(t+1)	
0	Q(t)	Maintain state
1	Q'(t)	Toggle

Sequential Circuit Analysis

- Label All Flip Flop Outputs (arbitrarily)
- Write Equations describing combinational logic leading to all Flip Flop Inputs
- Determine the FF Outputs from Characteristic Table
- Complete the **State Table**
  - Present State: Inputs: Next State: Outputs
- Circuit with *m* FF and *n* inputs need 2^{m+n} entries in the **State Table**

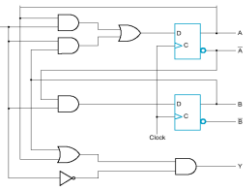
State Table

- The table shows the present states, inputs, next states and outputs
- One Dimensional (present state and inputs combined) Moore Model
- Two Dimensional (present state left column and inputs tabulated across the top) Mealy Model

Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

$A(t+1) = D_A = AX + B\bar{X}$

$B(t+1) = D_B = \bar{A}X$

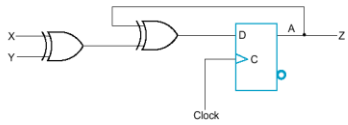


$Y = A\bar{X} + B\bar{X}$

2-D State Table

Present state		Next state				Output	
		X = 0		X = 1		X = 0	X = 1
A	B	A	B	A	B	Y	Y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

Logic Diagram and State Table for $D_A = A \oplus X \oplus Y$

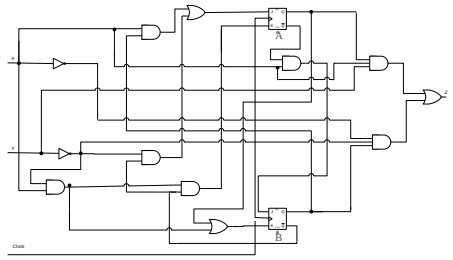


(a)

Present state	Inputs		Next state	Output
A	X	Y	A	Z
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(b) State table

Sequential Circuit Analysis: Example



Sequential Circuit Analysis: Example

1. Find the Flip Flop Input Equations

$J_A = BX + \bar{B}\bar{Y}$ $\quad K_A = \bar{B}X\bar{Y}$

$J_B = \bar{A}X$ $\quad K_B = A + X\bar{Y}$

2. Find the Output Equations

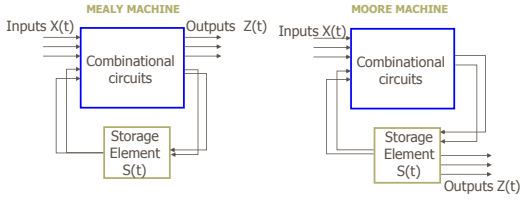
$Z = AXY + B\bar{X}\bar{Y}$

3. Determine the Next State of the Flip Flops from Characteristic Tables
4. Fill the State Table
5. Draw the State Diagram

State and State Diagram

- A **state** represents the machine snapshot at a given clock period
- A clock is typically used to **synchronize** the state transition
- A graph consists of a set of
  - Circles:
    - Each represents a **state**
    - Use double circle to represent the **initial state**
  - Directed arc: each represents a state transition
  - Inputs/outputs
- Mealy machine:
  - Label **input/output** along each arc
- Moore machine:
  - Label **input** along each arc
  - Label **output** inside the circle (i.e. state)

Mealy and Moore Machines



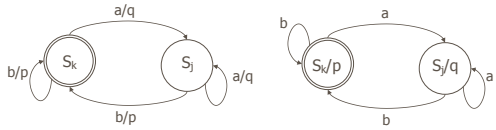
$Z(t) = \lambda\{S(t), X(t)\}$

Output is dependent on state and input

$Z(t) = \lambda\{S(t)\}$

Output is dependent on state only

State Diagrams



A Mealy machine example

A Moore machine example

Example:

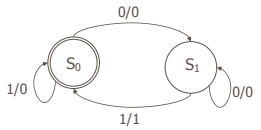
State: $S(t) \in \{S_k, S_j\}$

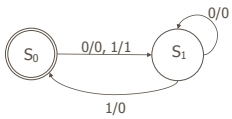
Inputs: $X(t) \in \{a, b\}$

Outputs: $Z(t) \in \{p, q\}$

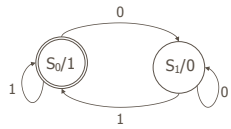
Initial state: $S(0) = S_k$

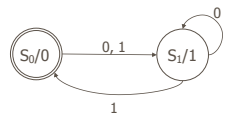
State Diagram Examples (Mealy)





State Diagram Examples (Moore)





Design Example: Sequence Recognizer

- A sequential circuit that recognizes the occurrence of a particular bit sequence
- Input: $X(t) \in \{0, 1\}$
- Output: $Z(t) \in \{0, 1\}$

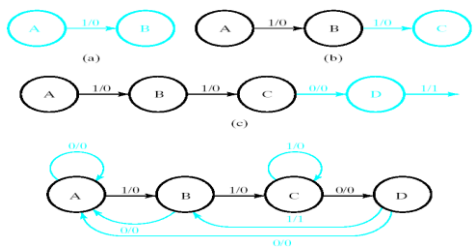
$$Z(t) = \begin{cases} 1, \text{if } X(t-3,t) = 1101 \\ 0, \text{Otherwise} \end{cases}$$

Sequence Recognizer

$$Z(t) = \begin{cases} 1, \text{if } X(t-3,t) = 1101 \\ 0, \text{Otherwise} \end{cases}$$

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
X(t)	1	0	0	1	0	1	1	0	1	0	1	1	0	1	1	0	1
Z(t)	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1

State Diagram & State Table

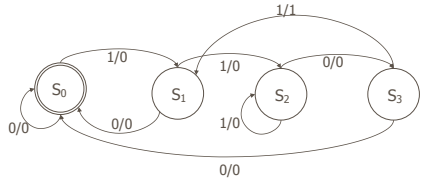


Present State	Next State X = 0	Next State X = 1	Output Z X = 0	Output Z X = 1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

Sequence Recognizer

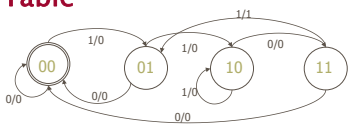
$$Z(t) = \begin{cases} 1, \text{if } X(t-3,t) = 1101 \\ 0, \text{Otherwise} \end{cases}$$

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
X(t)	1	0	0	1	0	1	1	0	1	0	1	1	0	1	1	0	1
Z(t)	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1



A Mealy Machine

State Table



Present State P1	Present State P0	Input X	Next State N1	Next State N0	Output Z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	1	1	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	1	1

Logic Circuits Design Steps

- Obtain state diagram or state table as requirement
- Assign binary codes to states
- Generate a Boolean function for
  - Each external output
  - Each state encoded bit
- Simplify the Boolean functions
- Draw a D F/F (or register) for each state encoded bit
- Draw logic circuits for
  - External outputs
  - Each inputs of state encoded bits
  - Input of state encoded bits = the next state
  - Output of state encoded bits = the current state

Logic Circuits Design

Present State P1	Present State P0	X	Next State N1	Next State N0	Z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	1	1	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	1	1

N1

X \ P1P0	00	01	11	10
0	0	0	0	1
1	0	1	0	1

$N1 = X\overline{P}_1P_0 + P_1\overline{P}_0$

N0

X \ P1P0	00	01	11	10
0	0	0	0	1
1	1	0	1	0

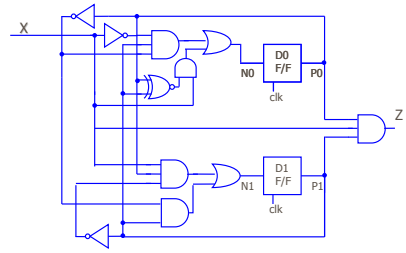
$N0 = X\overline{P}_1\overline{P}_0 + XP_1P_0 + \overline{X}P_1\overline{P}_0$

$= X(\overline{P1 \oplus P0}) + \overline{X}P_1\overline{P}_0$

$Z = XP_1P_0$

Logic Circuits Design

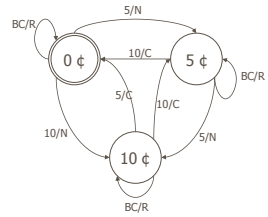
$N0 = X(\overline{P1 \oplus P0}) + \overline{X}P_1\overline{P}_0$ $\quad N1 = X\overline{P}_1P_0 + P_1\overline{P}_0$ $\quad Z = XP_1P_0$



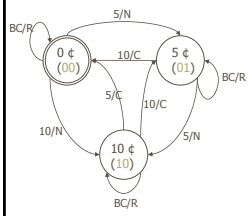
Vending Machine State Machine

- Dispense a Coke when depositing 15 ¢
- Inputs
  - 5 = a nickel
  - 10 = a dime
  - BC = bad coin (including quarters in this example)
- Outputs
  - R = reject
  - C = coke
  - N = no coke

State Diagram



State Table



5: 00 N: 00
10: 01 C: 01
BC: 10 R: 10

Present State (0¢, 5¢, 10¢)		Input (5¢, 10¢ , BC)		Next State (0¢, 5¢, 10¢)		Output (C, N, R)	
P1	P0	X1	X0	N1	N0	C1	C0
0	0	0	0	0	1	0	0
0	0	0	1	1	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	1	1	0
1	0	0	0	0	0	0	1
1	0	0	1	0	1	0	1
1	0	1	0	1	0	1	0

Logic Circuits Design

Present State		Input		Next State		Output	
P1	P0	X1	X0	N1	N0	C1	C0
0	0	0	0	0	1	0	0
0	0	0	1	1	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	1	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	1	1	0
1	0	0	0	0	0	0	1
1	0	0	1	0	1	0	1
1	0	1	0	1	0	1	0
X	X	1	1	X	X	X	X
1	1	X	X	X	X	X	X

N1

P1P0 \ X1X0	00	01	11	10
00	0	1	X	0
01	1	0	X	0
11	X	X	X	X
10	0	0	X	1

$$N1 = P0\overline{X1X0} + \overline{P1P0}X0 + P1X1$$

N0

P1P0 \ X1X0	00	01	11	10
00	1	0	X	0
01	0	0	X	1
11	X	X	X	X
10	0	1	X	0

$$N0 = \overline{P1P0}\,\overline{X1}\,\overline{X0} + P1X0 + P0X1$$

Logic Circuits Design

Present State		Input		Next State		Output	
P1	P0	X1	X0	N1	N0	C1	C0
0	0	0	0	0	1	0	0
0	0	0	1	1	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	1	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	1	1	0
1	0	0	0	0	0	0	1
1	0	0	1	0	1	0	1
1	0	1	0	1	0	1	0
X	X	1	1	X	X	X	X
1	1	X	X	X	X	X	X

C1

P1P0 \ X1X0	00	01	11	10
00	0	0	X	1
01	0	0	X	1
11	X	X	X	X
10	0	0	X	1

$$C1 = X1$$

C0

P1P0 \ X1X0	00	01	11	10
00	0	0	X	0
01	0	1	X	0
11	X	X	X	X
10	1	1	X	0

$$C0 = P1\overline{X1} + P0X0$$

Logic Circuits for Vending Machine

$$N1 = P0\overline{X1X0} + \overline{P1P0}X0 + P1X1$$
$$N0 = \overline{P1P0}\,\overline{X1X0} + P1X0 + P0X1$$

$$C1 = X1$$
$$C0 = P1\overline{X1} + P0X0$$