

Digital Systems Design I Notes

Anthony
Fiorito

Binary Multiplication

Binary Multipliers

$$A = a_1 a_0$$

$$B = b_1 b_0$$

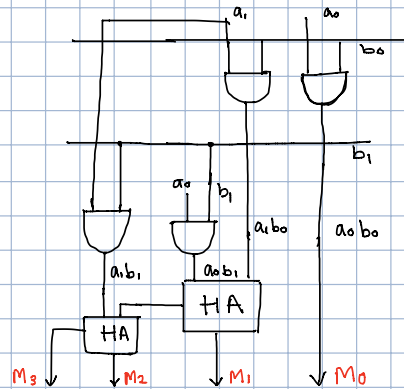
$$\begin{array}{r} a_1 a_0 \\ b_1 b_0 \\ \hline a_1 b_0 \ a_0 b_0 \\ a_1 b_1 \ a_0 b_1 \end{array}$$

$$\begin{array}{r} 11 \\ 10 \\ 00 \\ \hline 110 \end{array} \rightarrow 6_{10}$$

Multiplication process:

1. multiply

2. add



Example:

Design a 4 bit by 4 bit multiplier for two unsigned binary numbers A, B

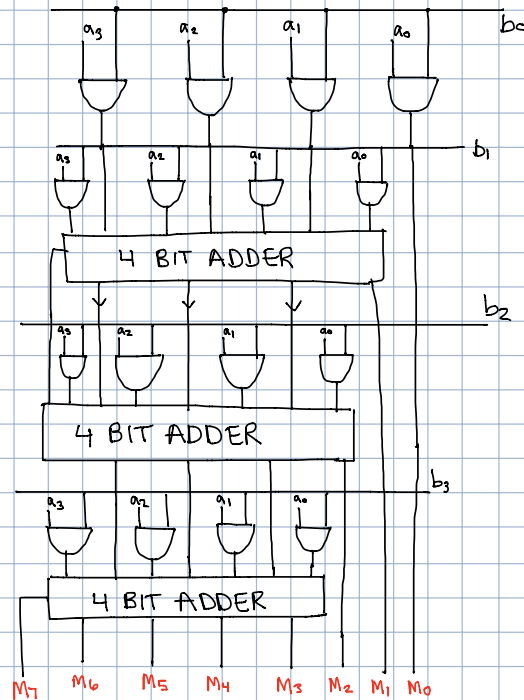
$$A = a_3 a_2 a_1 a_0$$

$$B = b_3 b_2 b_1 b_0$$

$$\begin{array}{r} a_3 \ a_2 \ a_1 \ a_0 \\ b_3 \ b_2 \ b_1 \ b_0 \\ \hline a_3 b_0 \ a_2 b_0 \ a_1 b_0 \ a_0 b_0 \\ a_3 b_1 \ a_2 b_1 \ a_1 b_1 \ a_0 b_1 \\ a_3 b_2 \ a_2 b_2 \ a_1 b_2 \ a_0 b_2 \\ a_3 b_3 \ a_2 b_3 \ a_1 b_3 \ a_0 b_3 \end{array}$$

4 bit Adder
4 bit Adder
4 bit Adder

M7 M6 M5 M4 M3 M2 M1 M0

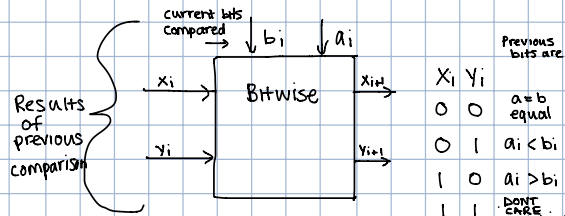
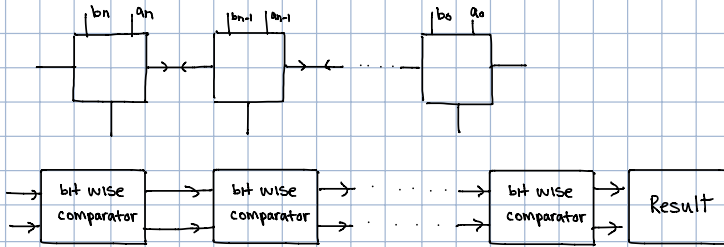


Example:

$\begin{matrix} \text{MSB} & & & & \text{LSB} \\ A & 1 & 0 & 0 & 1 \\ B & 0 & 1 & 1 & 1 \end{matrix}$

Compare A & B
 if $a_3 > b_3 \rightarrow A > B$

Iterative Method



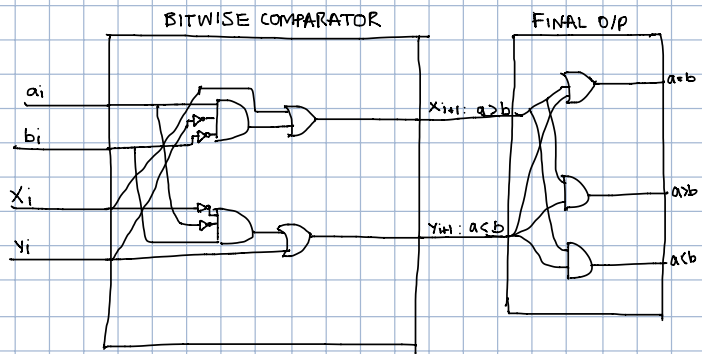
X_i	Y_i	a_i	b_i	X_{i+1}	Y_{i+1}
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

$X_i \backslash Y_i$	00	01	11	10
00	0	4	X	1
01	1	5	X	9
11	8	7	X	11
10	1	6	X	10

$X_{i+1} = X_i + Y_i \cdot a_i \cdot b_i$

$X_i \backslash Y_i$	00	01	11	10
00	0	1	X	8
01	1	5	X	9
11	8	7	X	11
10	1	6	X	10

$Y_{i+1} = Y_i + \bar{X}_i \cdot \bar{a}_i \cdot b_i$



Tutorial

4.12 Half Sub X-Y

X	Y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

$$B_{out} = x'y$$

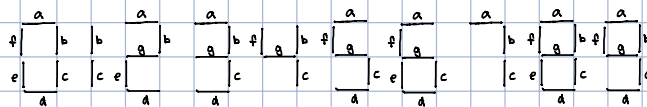
$$D = x \oplus y$$

Full Sub X-y-Bin

X	Y	B _{in}	B _{out}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$D = x \oplus y \oplus z$$

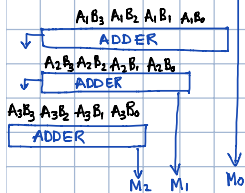
4.9



$$a = A'C + A'BD + B'C'D$$

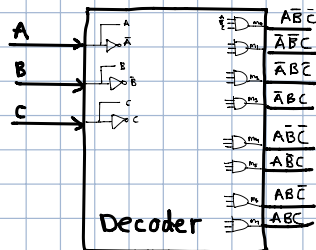
A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0						
0	1	0	1	1						
0	1	1	0	1						
0	1	1	1	1						
1	0	0	0	1						
1	0	0	1	1						
1	0	1	0							
1	0	1	1							
1	1	0	0							
1	1	0	1							
1	1	1	0							
1	1	1	1							

4.20 B₃ B₂ B₁ B₀ Kbits JxK AND
A₃ A₂ A₁ A₀ Jbits (J-K) Adder
A₀B₃ A₀B₂ A₀B₁ A₀B₀ M_(next)



Decoder

MSB
A₂A₁A₀



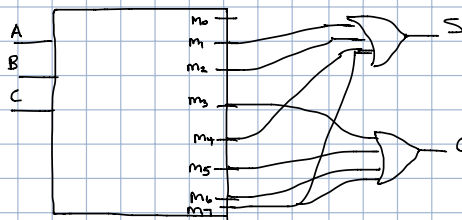
Use for addressing
minterm generation
SOP-AND-OR

Example: implement a full adder using a decoder & extra logic

A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$$C = \Sigma(3, 5, 6, 7)$$

$$S = \Sigma(1, 2, 4, 7)$$



Encoders

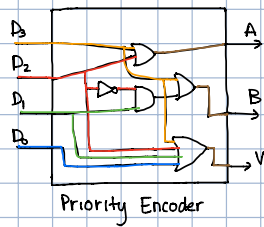
$$3 \rightarrow 2^3 \rightarrow 8$$

n	$\rightarrow$	2^n
-----	---------------	-------

Priority Encoder

$$Z^n \rightarrow m$$

Example: design a priority encoder for 4 bit binary code to 2 bits. Input: D_3, D_2, D_1, D_0 — A, B



	D ₃	D ₂	D ₁	A	B	V (Valid)
0	0	0	0	X	X	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	1
4	1	0	0	1	0	1
5	1	0	1	0	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	1
8	1	0	0	0	1	1
9	1	0	1	0	1	1
10	1	1	0	0	1	1
11	1	1	1	0	1	1
12	1	0	0	1	1	1
13	1	0	1	1	1	1
14	1	1	0	1	1	1
15	1	1	1	1	1	1

$\frac{A_2 A_1}{A_0}$	00	01	11	10
00	X	4	12	8
01		5	13	9
11		7	15	11
10		6	14	10

$$A = D_2 + D_3$$

$\frac{D_2}{D_1}$

	00	01	11	10
00	X		1	1
01		4	1	1
11	1	5	1	1
10	1	3	7	1

$$B = \overline{D_2} D_1 + D_3$$

Multiplexers

2-1 Multiplexer : 2 → 1 MUX

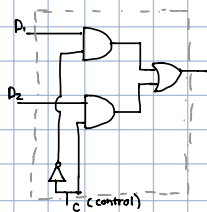
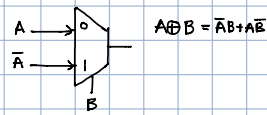
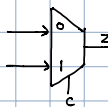
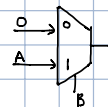


Diagram of a 2-to-1 multiplexer (MUX) with inputs D_1 and D_2 , control input C (labeled "control"), and output Z . The output is given by the equation:

$$Z = \bar{C}D_1 + CD_0$$

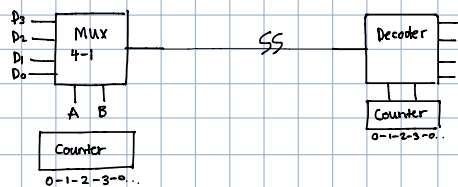


$$A \oplus B = \bar{A}B + A\bar{B}$$



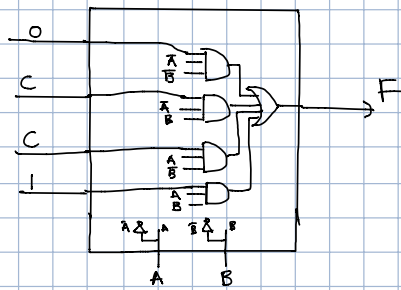
$$A \cdot B = AB$$

Example: transmission of 4 bits of information over a long distance



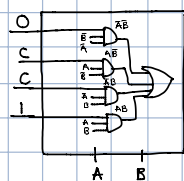
Example: Implement the following function using 4-1 mux

AE	00	01	11	10
0			1	
1		1	1	1

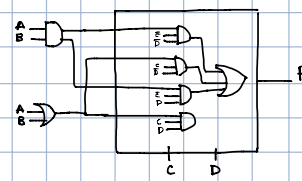


Example: Implement function F given by the k-map on a 4-1 mux you may use minimum extra logic

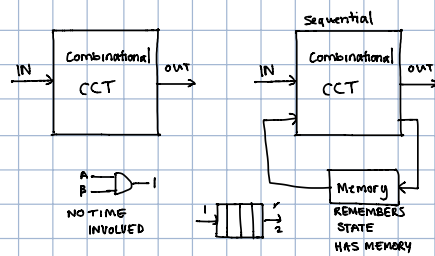
AB \ CD	00	01	11	10
00	0	1	1	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1



AB \ CD	00	01	11	10
00	0	1	1	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

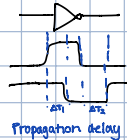
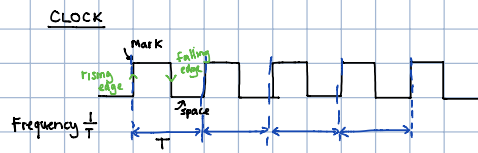


Sequential Circuits



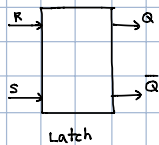
2 types:

- Synchronous CCT (depends on clocks)
- Asynchronous CCT (Depends on events)



Flip Flop

The RS Flip Flop



stores one bit of information 1,0

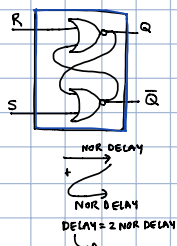
Q is the output $\bar{Q}$ is the complement of Q

$Q = 1 \rightarrow$ the latch is set

$\bar{Q} = 1 \rightarrow$ the latch is reset

set $S = 1, R = 0 \rightarrow$ FF is set

reset $S = 0, R = 1 \rightarrow$ FF is reset



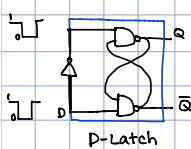
S	R	Q	Q+1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

↑ Set ↑ reset ↑ present state ↑ next state

Q	Q+1
0	0
0	1
1	0
1	1

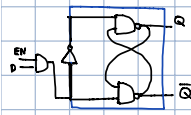
$Q+1 = S + Q\bar{R}$
CHARACTERISTIC EQUATION OF RS LATCH

Transition Table



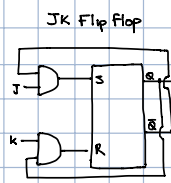
D	Q	Q+1
0	0	0
0	1	1
1	0	0
1	1	1

$Q+1 = D$
Transparent Latch



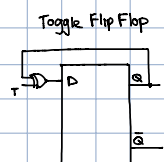
Controlled D-Latch

LEVEL SENSITIVE



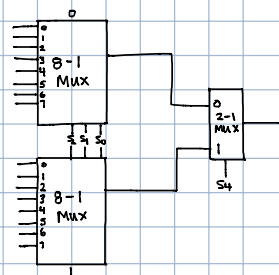
$Q+1 = J\bar{Q} + \bar{K}Q$

connect J to K: Toggle FF



$Q+1 = T \oplus Q$
 $= T\bar{Q} + \bar{T}Q$

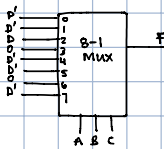
Tutorial



4.32

a) $F(A,B,C,D) = \sum(0,2,5,8,10,14)$

A	B	C	D	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0



b) $F(A,B,C,D) = \pi(2,6,11)$

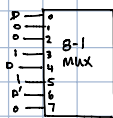
$F = (A + B' + C + D')(A + B + C' + D')(A + B' + C + D)$
 $F' = ABC'D + AB'CD + A'B'C'D'$
 $F' = \sum(13,11,8)$
 $F = \sum(0,1,2,3,4,5,6,7,9,10,12,14,15)$

4.33 4-1 MUX

A	B	C _{in}	S	C _{out}
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0



4.34



$$I_1 = I_2 = I_3 = 0$$

$$I_3 = I_5 = 1$$

$$I_0 = I_4 = D$$

$$I_6 = D'$$

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

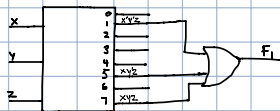
A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

4.28

$$F_1 = (y' + x)z = y'z + xz = xy'z + x'y'z + xyz + xy'z$$

$$F_2 = y'z' + x'y + yz'$$

$$F_3 = (x + y)z$$



Level Sensitive

SR Flip Flop

D Flip Flop

JK Flip Flop

T Flip Flop

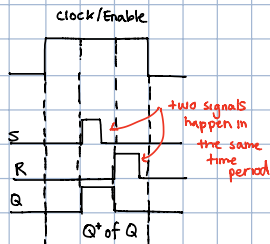
Characteristic Equation

$$Q^+ = S + \bar{R}Q$$

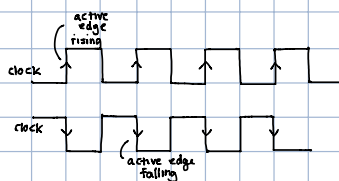
$$Q^+ = D$$

$$Q^+ = J\bar{Q} + \bar{K}Q$$

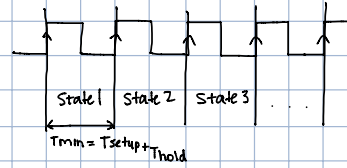
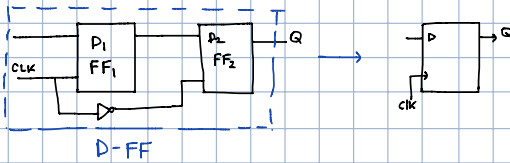
$$Q^+ = \bar{T}Q + T\bar{Q}$$



Edge Sensitive Flip Flops (real flip flops)



Master-Slave Flip Flop



When clock is high

$D \rightarrow Q_1$

and is set at input D_2

Data may change but it is not propagated to the output of D-FF

The time required for the data to get to input of FF_2 is called

Setup time

When the clock is low

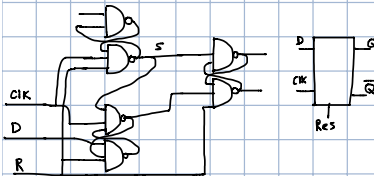
• D_1 is closed - no more changes can get through to FF_1 . The last data will wait at the gate FF_2

• Now FF_2 is inactive because the clk edge has arrived. Data is let in and appears at the output of the D-FF

* During one period of the clock only one Data is transmitted to the output and that occurs at the -ve going edge of the clock

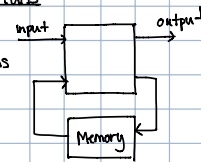
the time required for the data to go through FF_2 from the the edge of the clock is called **Hold Time**

Differential Flip Flop



Analysis of Synchronous Sequential Circuits

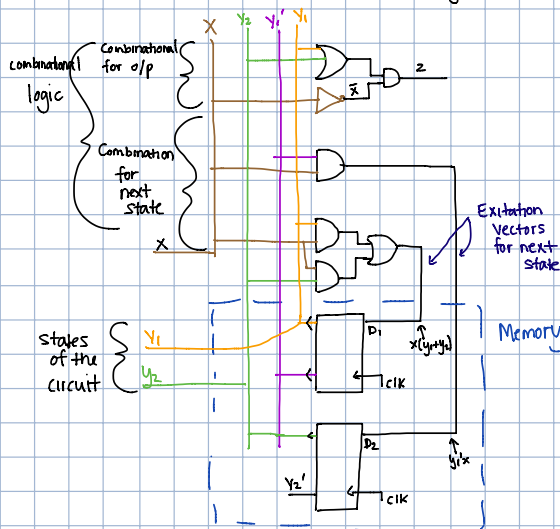
Determining for all possible combinations of inputs + present states what o/p we get and what are the next states



This is usually represented in a:

transition table or state table or state diagram.

1. Determine excitation vectors (input to the FF)
2. Determine the output
3. Determine the next states
4. Draw the transition table
5. Draw the state Table/Diagram



$$\begin{aligned} D_1 &= x(y_1 + y_2) \\ D_2 &= y_1'x \\ Z &= \bar{x}(y_1 + y_2) \end{aligned}$$

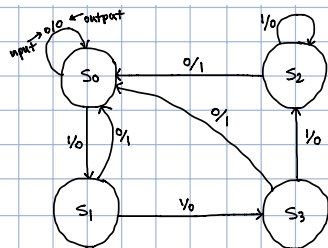
$$\begin{aligned} Q_1^+ &= D_1 \\ Q_2^+ &= D_2 \end{aligned}$$

Present State	Next state x=0	Next state x=1	output x=0	output x=1
$y_1 \ y_2$	$y_1^+ \ y_2^+$	$y_1^+ \ y_2^+$	Z	Z
00 - S_0	00	01	0	0
01 - S_1	00	11	1	0
10 - S_2	00	10	1	0
11 - S_3	00	10	1	0

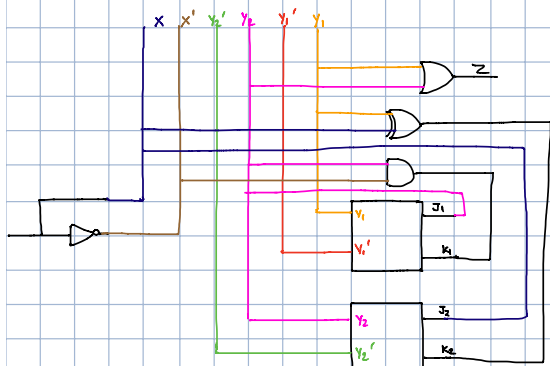
Transition Table

Present State	Next State		output z	
$x=0 \quad x=1$	$x=0 \quad x=1$	$x=0 \quad x=1$	$x=0 \quad x=1$	
S_0	$S_0 \quad S_1$	$0 \quad 0$		
S_1	$S_0 \quad S_2$	$1 \quad 0$		
S_2	$S_0 \quad S_2$	$1 \quad 0$		
S_3	$S_0 \quad S_2$	$1 \quad 0$		

State Table



State Diagram



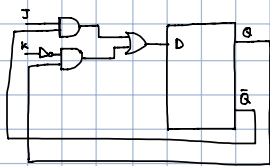
$$\begin{aligned}
 Z &= y_1 + y_2 \\
 J_1 &= y_2 \\
 K_1 &= \bar{x} y_2 \\
 J_2 &= x \\
 K_2 &= y_1 \oplus x \\
 Q^+ &= J\bar{Q} + KQ
 \end{aligned}$$

Present State	Input		Excitation 1		Excitation 2		output	Next State	
$y_1 \quad y_2$	x		$J_1 \quad K_1$		$J_2 \quad K_2$		Z	$y_1^+ \quad y_2^+$	
0 0	0		0 0		0 0		0	0 0	00 - S_0
0 0	1		0 0		1 1		0	0 1	01 - S_1
0 1	0		1 1		0 0		1	1 1	10 - S_2
0 1	1		1 0		1 1		1	1 0	11 - S_3
1 0	0		0 0		0 1		1	1 0	
1 0	1		0 0		1 0		1	1 1	
1 1	0		1 1		0 1		1	0 0	
1 1	1		1 0		1 0		1	1 1	

Present State	Next State		output z	
$x=0 \quad x=1$	$x=0 \quad x=1$	$x=0 \quad x=1$	$x=0 \quad x=1$	
S_0	$S_0 \quad S_1$			
S_1	$S_3 \quad S_2$			
S_2				
S_3				

Tutorial

$$Q^+(t+1) = J\bar{Q} + KQ$$

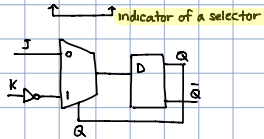


J	K	Q_P	Q_N
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

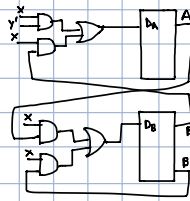
$J \quad K$	0 0	0 1	1 0	1 1
0 0	0	1	0	0
0 1	1	1	0	1
1 0	0	1	1	0
1 1	0	1	1	0

$$\begin{aligned}
 Q^+ &= J\bar{Q} + K'Q \\
 Q^{++} &= J\bar{Q} + KQ
 \end{aligned}$$

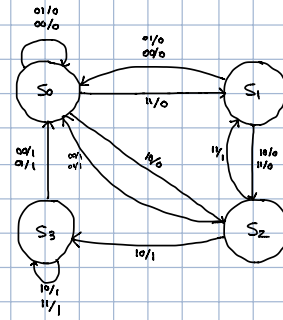
$$D = J\bar{Q} + K'Q$$



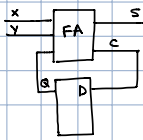
DFFA $A(t+1) = xy' + xB$
 DFFB $B(t+1) = xA + xB'$
 Inputs x, y $Z = A$
 Outputs z



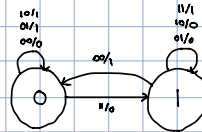
x	y	A	B	A'	B'	z
0	0	0	0	1	1	0
0	0	0	1	1	0	0
0	0	1	0	0	1	1
0	0	1	1	0	0	1
0	1	0	0	1	1	0
0	1	0	1	1	0	0
0	1	1	0	0	1	1
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	1	0	0
1	0	1	0	0	1	1
1	0	1	1	0	0	1
1	1	0	0	1	1	0
1	1	0	1	1	0	0
1	1	1	0	0	1	1
1	1	1	1	0	0	1



5.7



x	y	Q	Q'	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



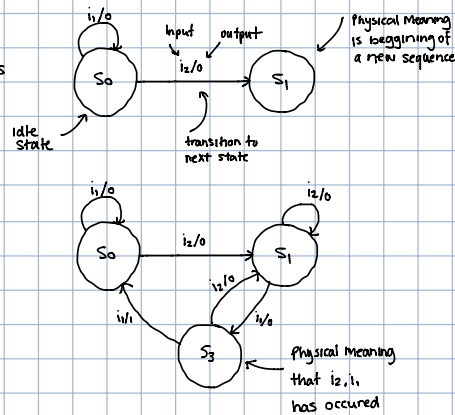
Design of Synchronous Sequential Circuits

Finite State Machines (FSM)



Depending on inputs
 we move to same state
 or other states, considering
 all input combinations

This specifications looks
 for the sequence:
 i_2, i_1, i_1

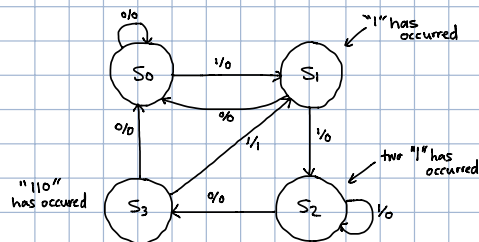


Example:

Derive the state diagram for a synchronous sequential circuit to recognize the sequence:

1101

Input: 1 0 1 1 0 0 1 1 0 1 1 0 1 0 0 0 1 0
 Output: 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18



Example:

Serial Adder

Two serial data inputs, design a circuit to add two numbers a, b and store results R in Z

