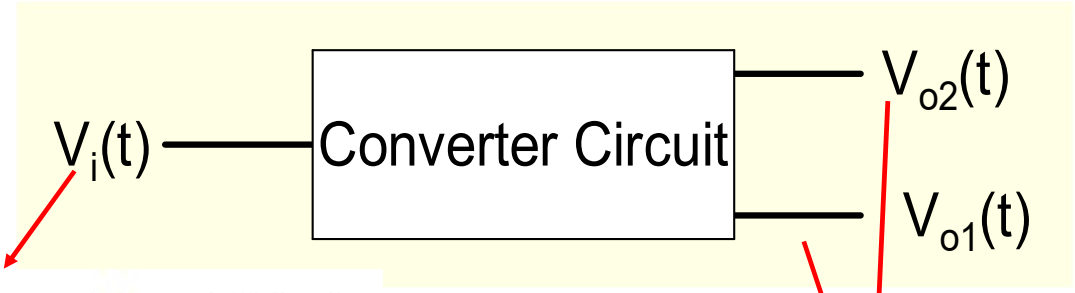
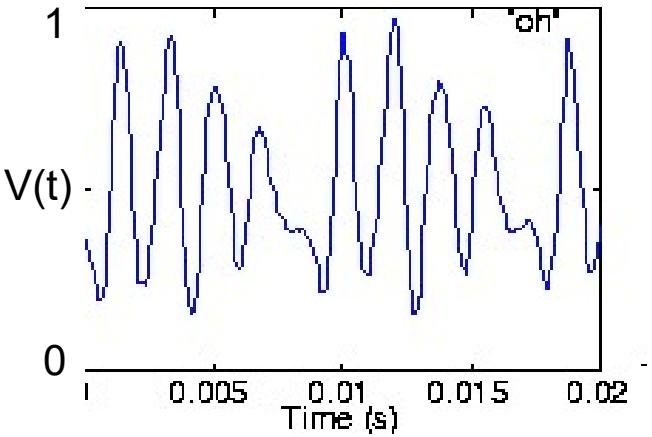
ESC201T : Introduction to Electronics

Lecture 31: Digital Circuits-1

B. Mazhari
Dept. of EE, IIT Kanpur

Analog vs. Digital Signal





Can take any value in the range 0-1V

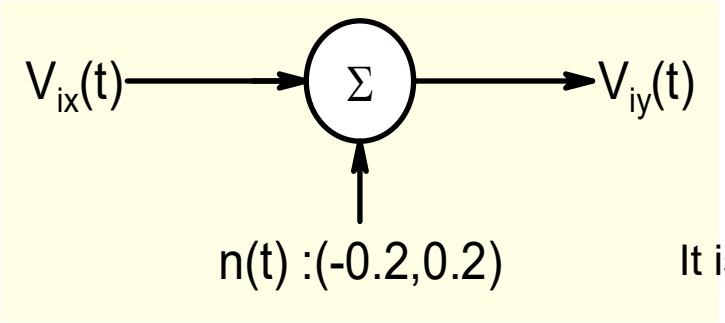
Can take only discrete values (0,1,2..9)

Analog Voltage (V)	Vo2 (V)	Vo1 (V)
0.45	4	5
0.75	7	5
0.93	9	3
0.932	9	3
0.934	9	3
0.936	9	4

Loss in accuracy

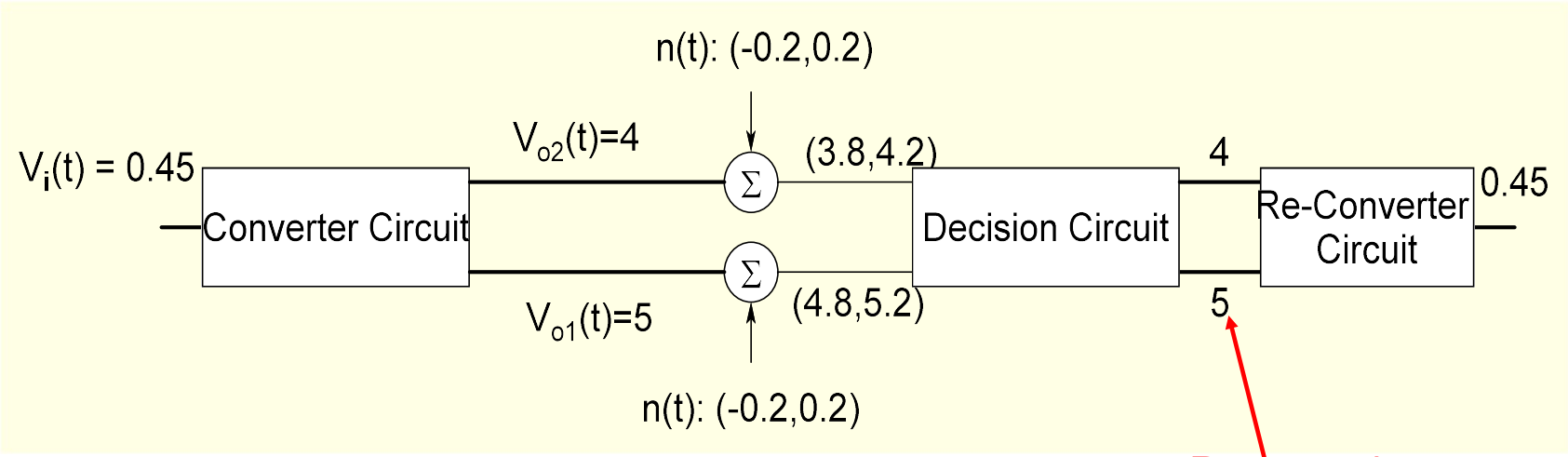
Advantages of using digital Signals

Robustness towards noise



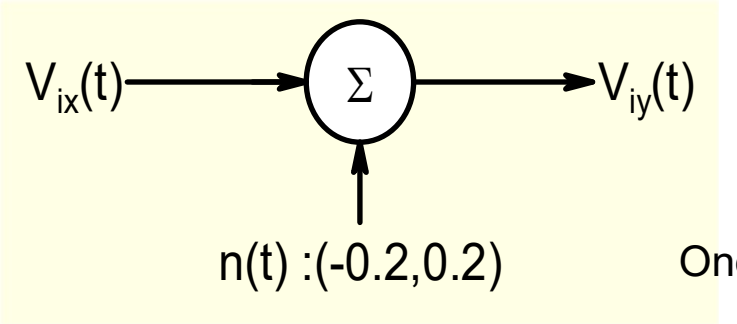
$$V_{ix}(t) = 0.45 \rightarrow V_{iy}(t) : (0.25, 0.65)$$

It is very difficult to recover the original signal



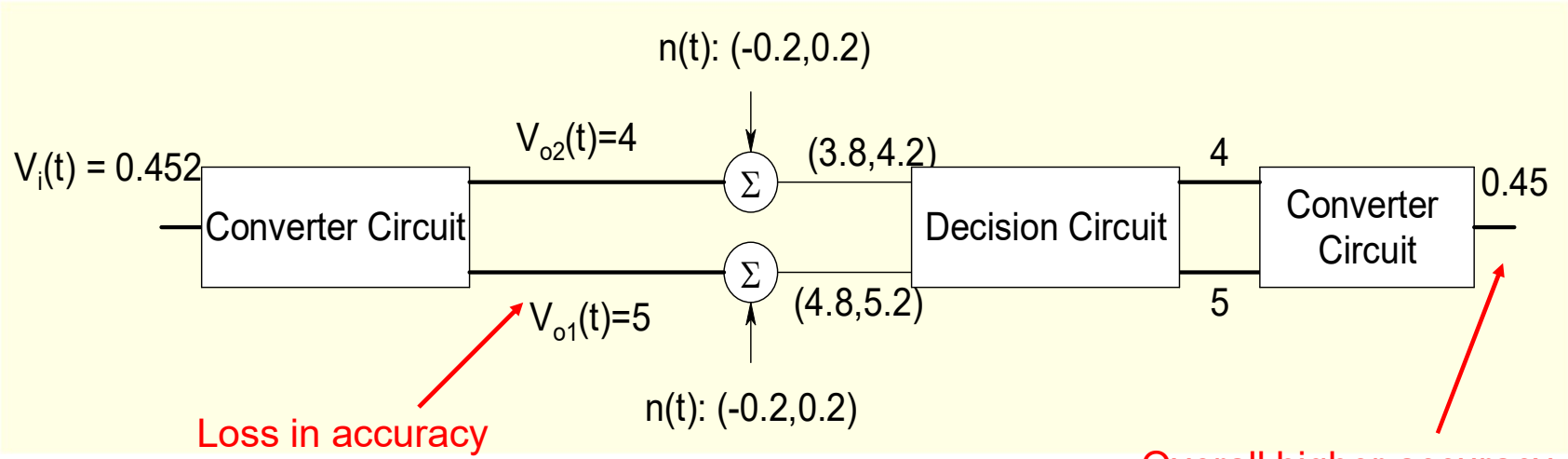
Advantages of using digital Signals

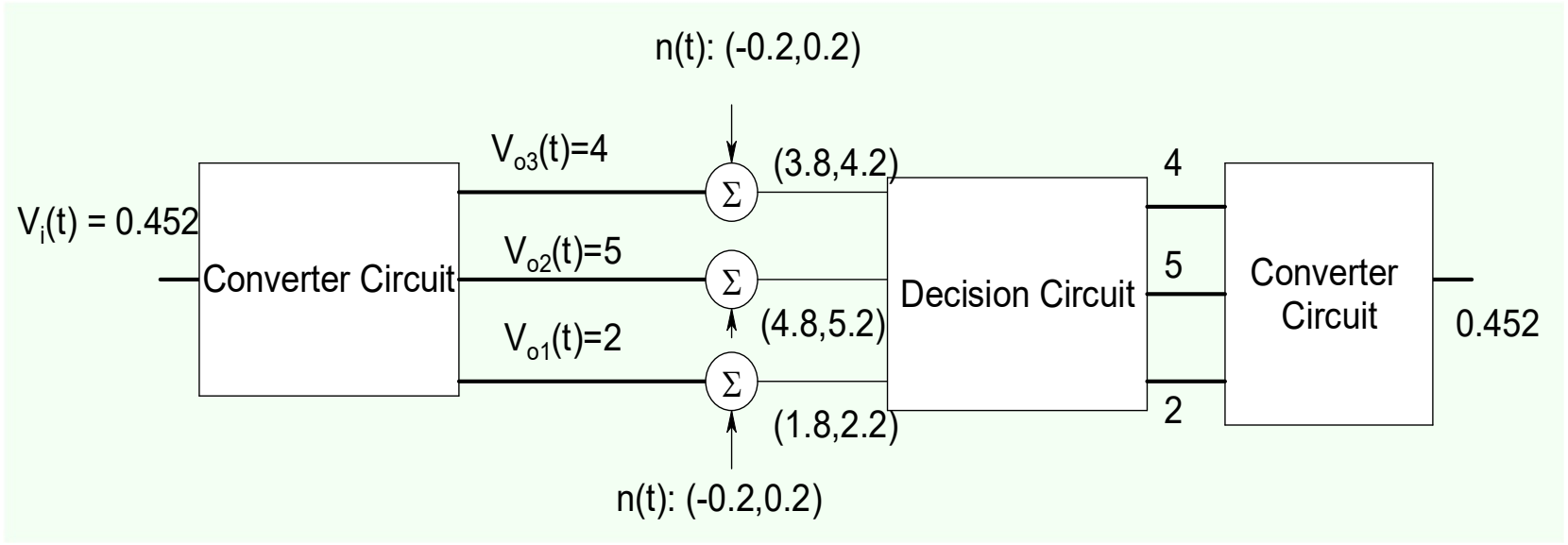
Robustness towards noise



$$V_{ix}(t) = 0.452 \rightarrow V_{iy}(t) : (0.252, 0.652)$$

One is uncertain about the first decimal place !!

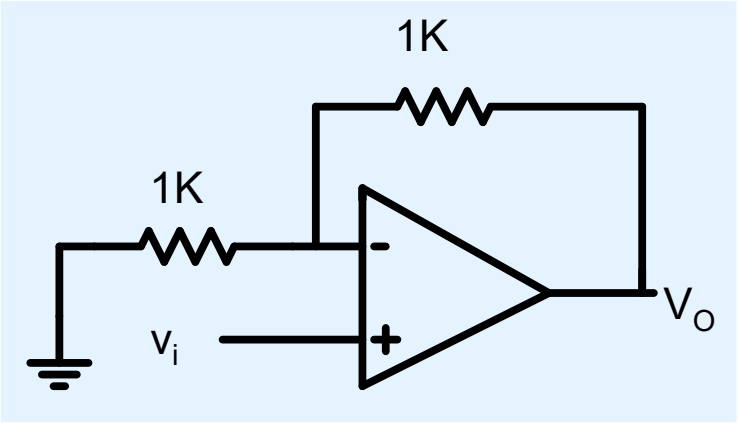




One can get the desired accuracy using larger number of digits

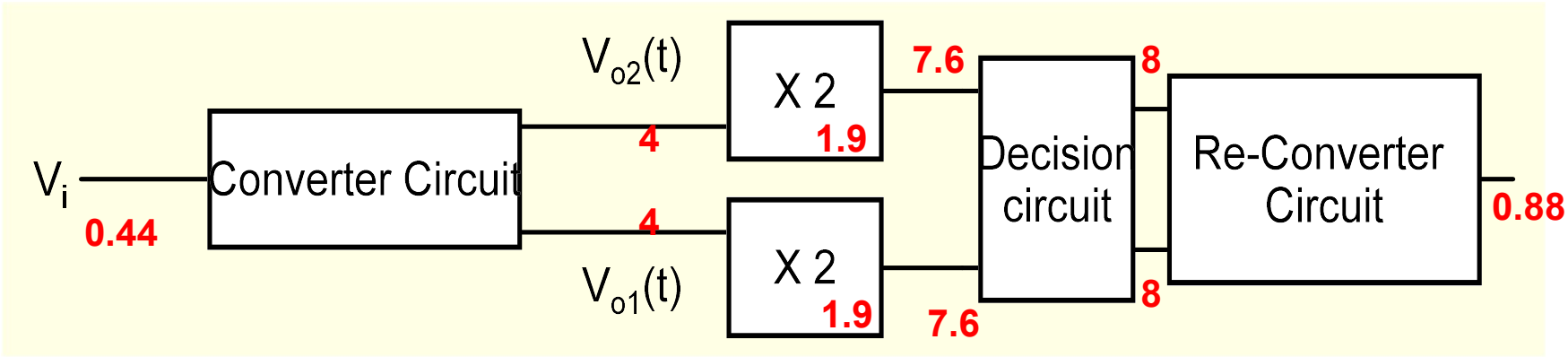
Accurate Processing ?

Suppose we would like to multiply a signal by a factor of 2



Because of tolerances etc, we would not get a gain of 2. Suppose the gain is 1.9.

For Vi = 0.44, we would get 0.836 instead of 0.88V.

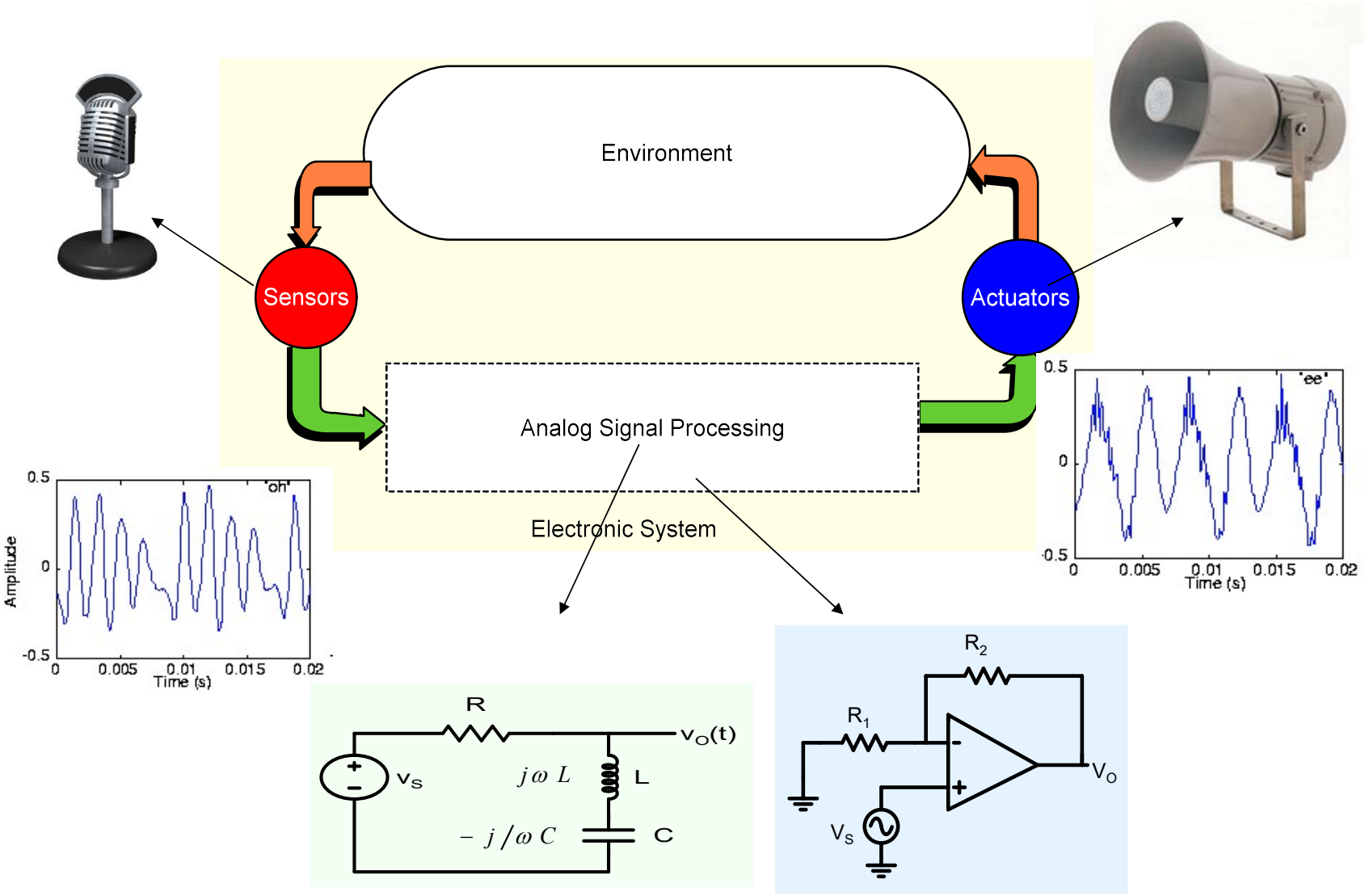


Processing of digital signals is often much simpler if numbers are represented properly !

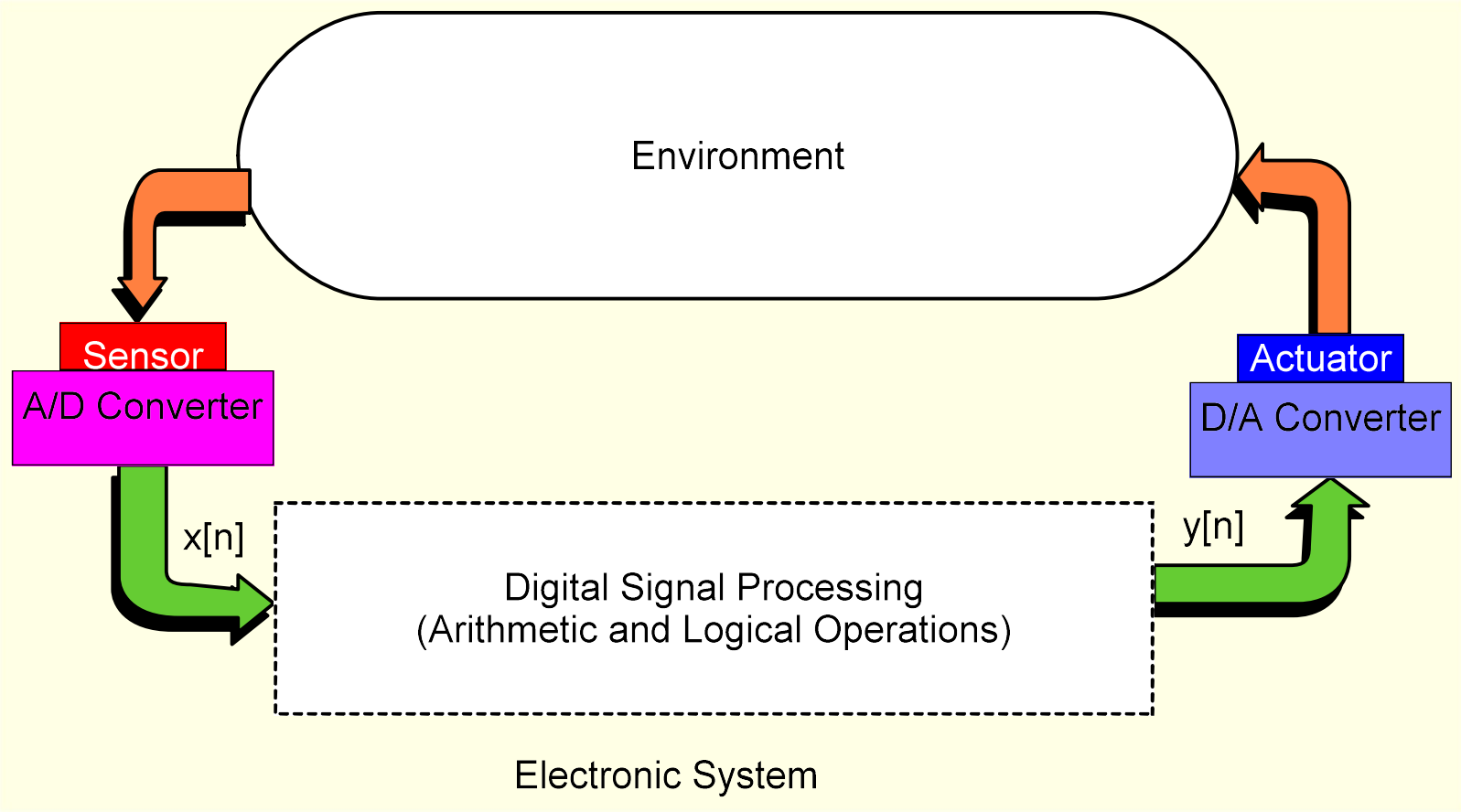
Digital circuits allow much more complex information processing

Deep Blue was a chess-playing computer developed by IBM. On May 11, 1997, the machine won a six-game match by two wins to one with three draws against world champion Garry Kasparov Kasparov accused IBM of cheating and demanded a rematch, but IBM declined and dismantled Deep Blue. Kasparov had beaten a previous version of Deep Blue in 1996….wikipedia

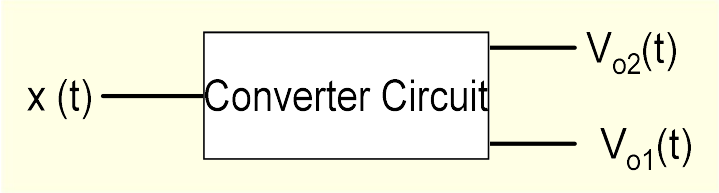
Analog signal Processing



Digital signal Processing

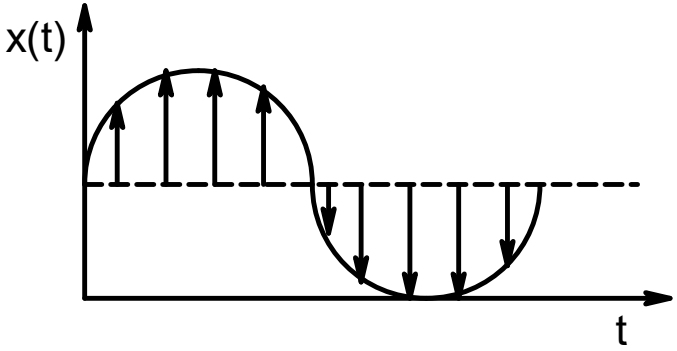


Converting signals into a sequence of numbers and vice-versa



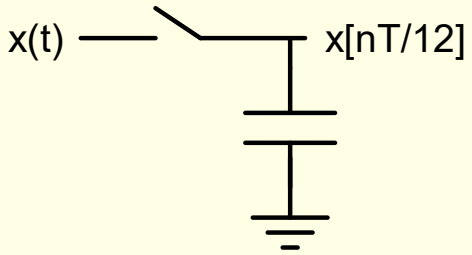
$$x(t) = 0.5 + 0.5 \times \sin(\frac{2\pi}{10^{-3}} t)$$

Step-1: Sampling

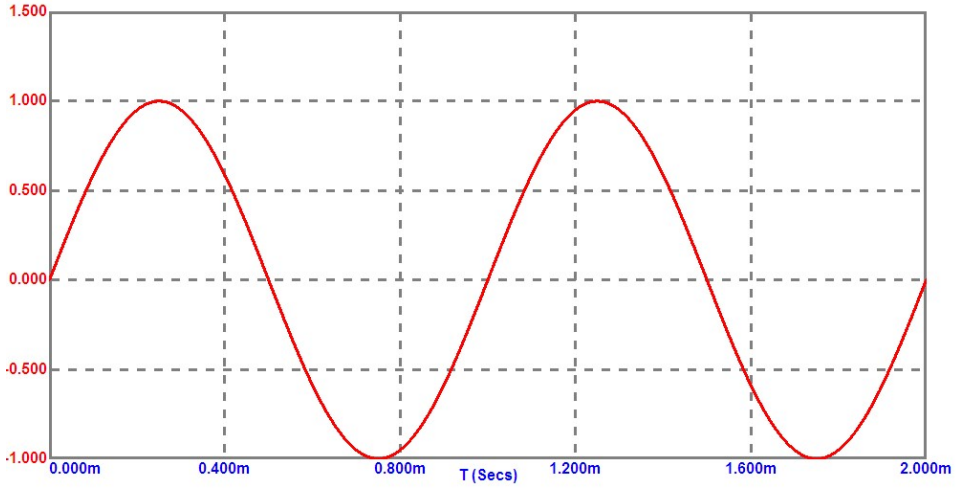


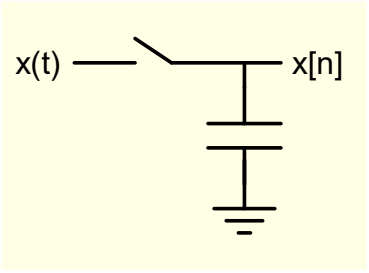
Sample at intervals of T/12

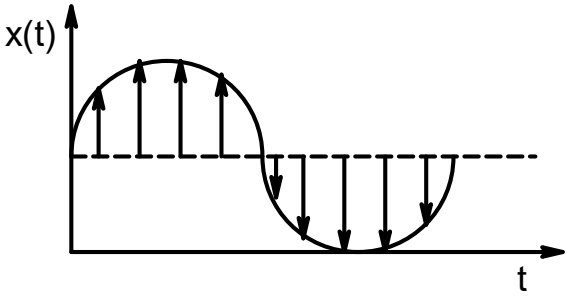
$$x[n\frac{T}{12}] = [0.5, 0.75, 0.93, 1, 0.93, 0.75, 0.5, 0.25, 0.067, 0, 0.067, 0.25, 0.5....]$$

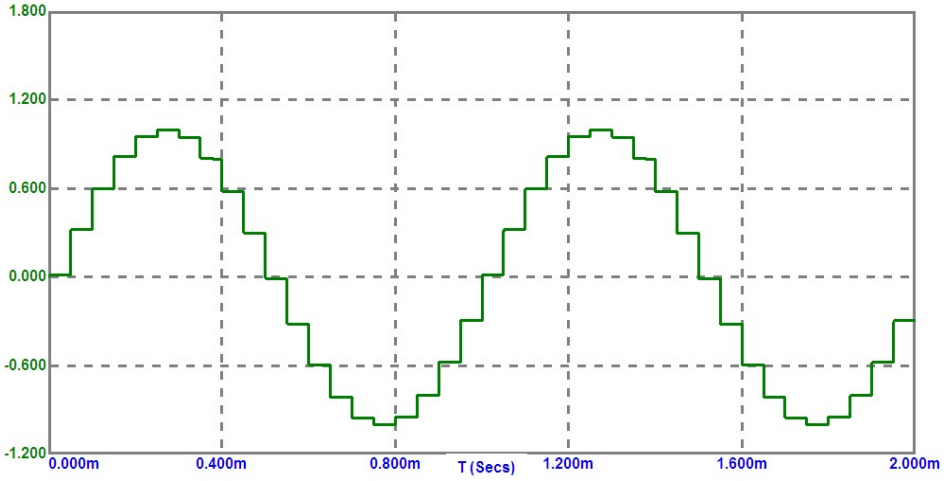


Example



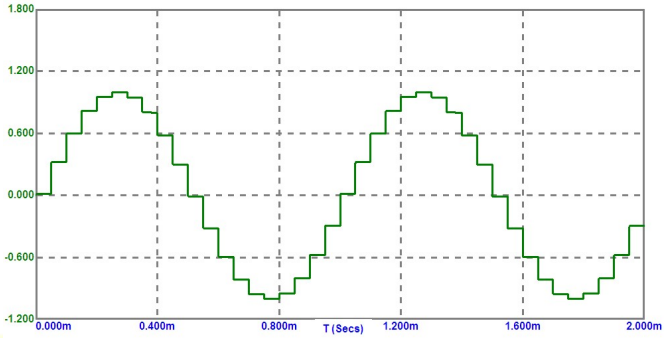


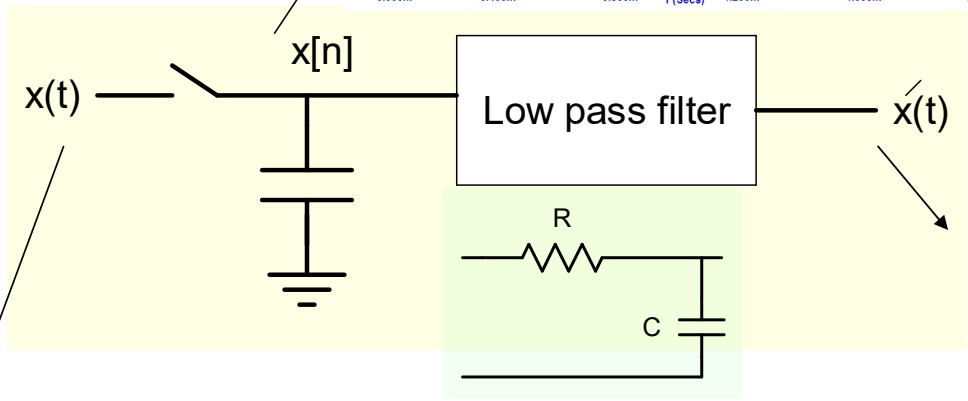


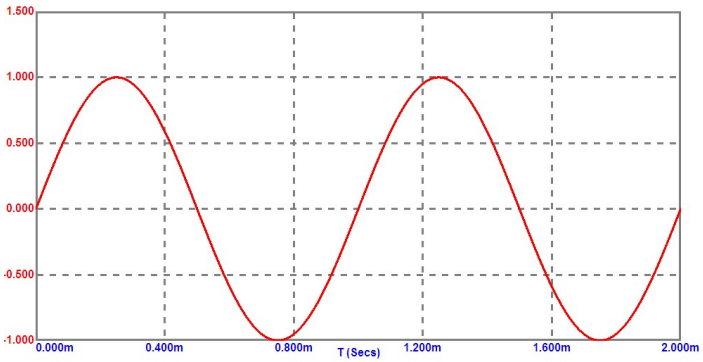


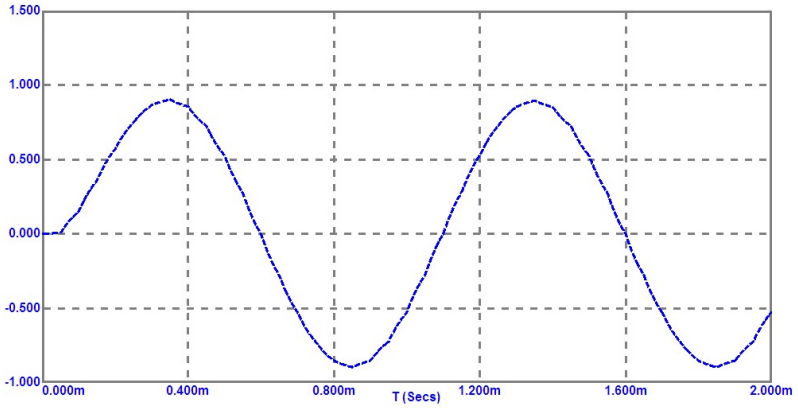
Sample and Hold

One can recover the original waveform by low pass filtering the sampled waveform

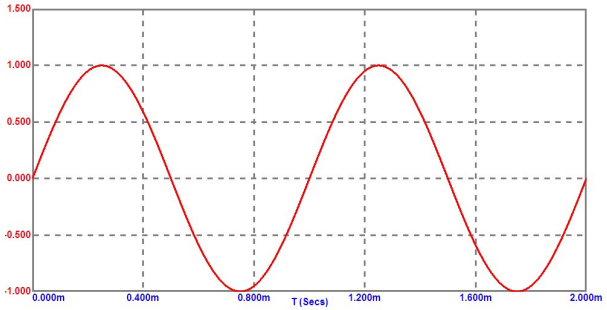


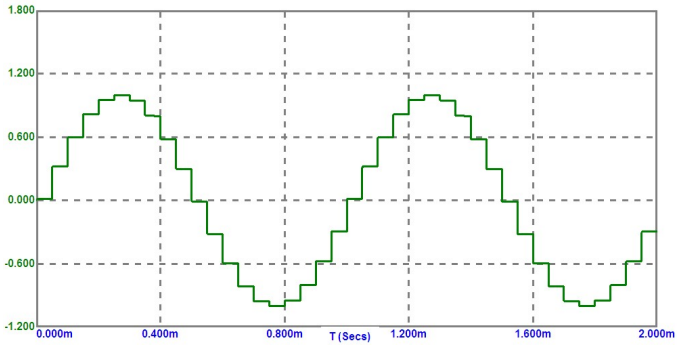




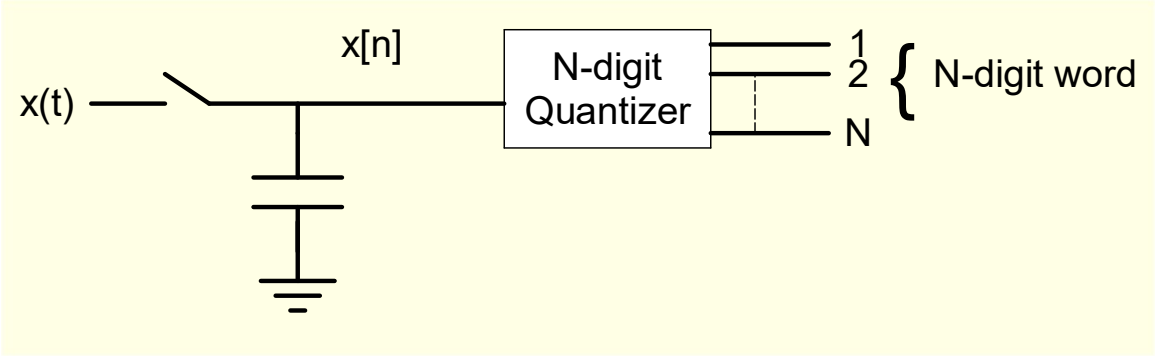


Converting sampled waveform into a sequence of numbers





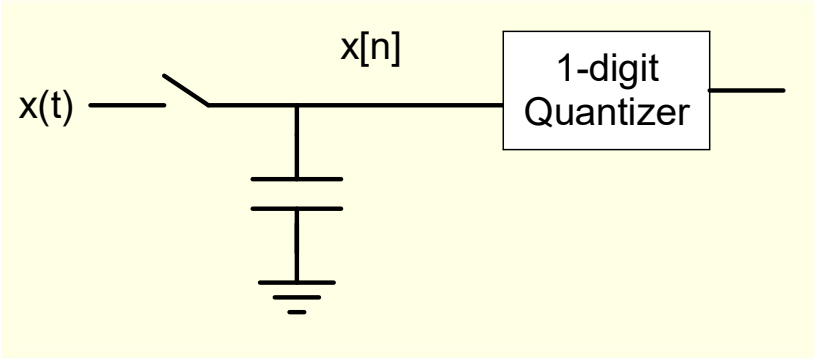
Quantization:

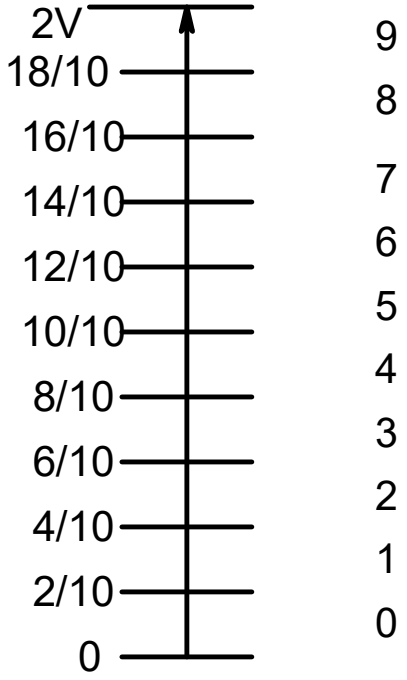


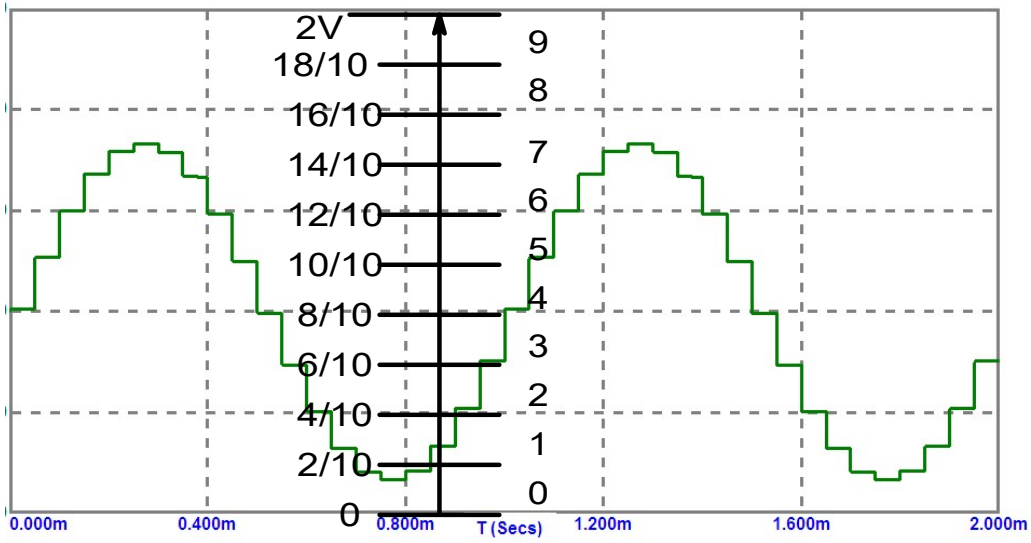
1-digit quantizer: [0,1,3,5,4,3,2,0,1,3,5……..]

2-digit quantizer: [00,12,32,57,42,31,22,00,12,32,57……..]

Quantization:







It is obvious that quantization introduces errors !

Quantization:

$$x(t) = 0.5 + 0.5 \times \sin(\frac{2\pi}{10^{-3}} t)$$

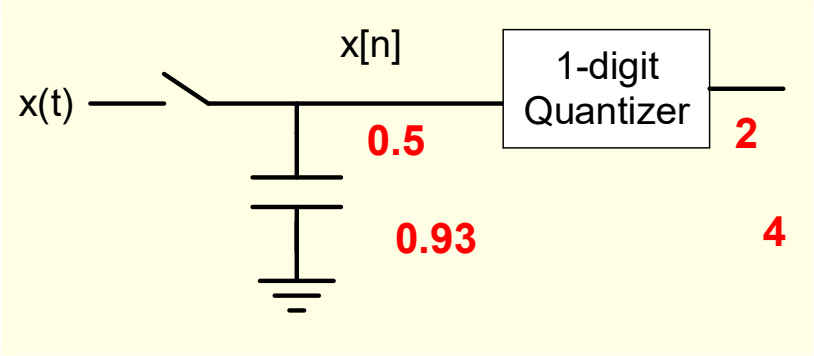
Sample at intervals of T/12

$$x[n\frac{T}{12}] = [0.5, 0.75, 0.93, 1, 0.93, 0.75, 0.5, 0.25, 0.067, 0, 0.067, 0.25, 0.5....]$$

Voltage Range (Volts)	Number
0-0.2	0
0.2-0.4	1
0.4-0.6	2
0.6-0.8	3
0.8-1.0	4
1.0-1.2	5
1.2-1.4	6
1.4-1.6	7
1.6-1.8	8
1.8-2.0	9

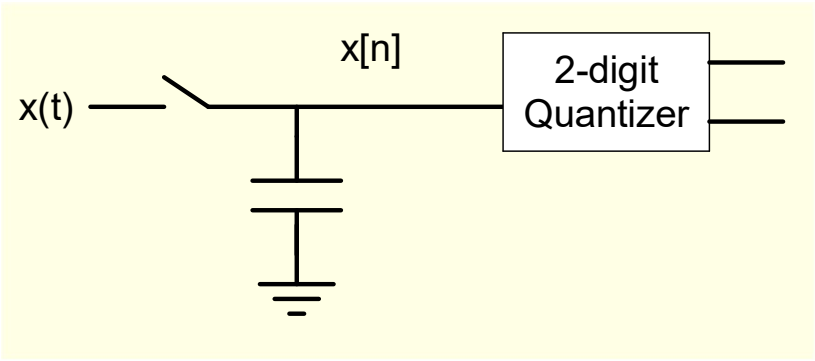
Analog Voltage	Digital voltage or Number
0.5	2
0.75	3
0.93	4
1	5
0.93	4
0.75	3
0.5	2
0.25	1
0.067	0
0	0
0.067	0
0.25	1
0.5	2

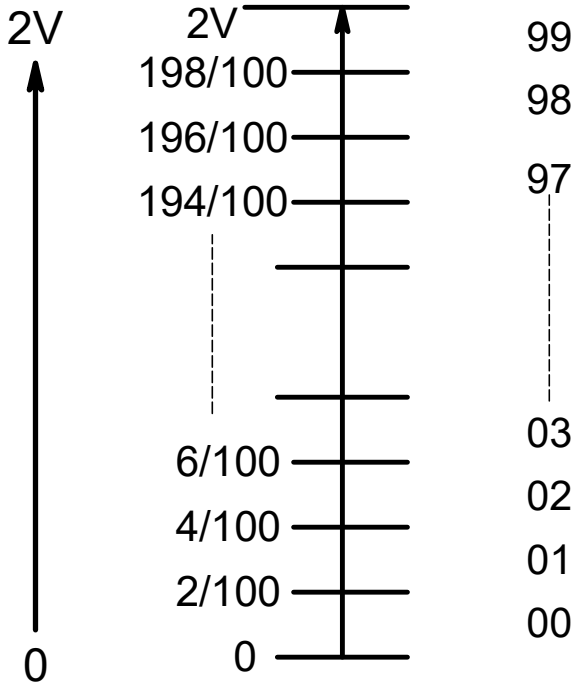
$$x(t) = 0.5 + 0.5 \times \sin(\frac{2\pi}{10^{-3}} t)$$



Analog Voltage	Digital voltage or Number
0.5	2
0.75	3
0.93	4
1	5
0.93	4
0.75	3
0.5	2
0.25	1
0.067	0
0	0
0.067	0
0.25	1
0.5	2

2-digit Quantization:





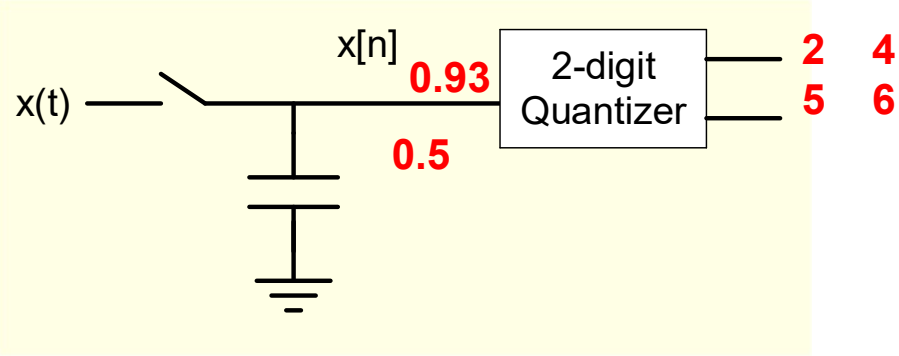
2-digit Quantization: $x(t) = 0.5 + 0.5 \times \sin(\frac{2\pi}{10^{-3}} t)$

Sample at intervals of T/12

Voltage Range (Volts)	Number
0-0.02	00
0.02-0.04	01
0.04-0.06	02
-	-
-	-
-	-
1.92-1.94	96
1.94-1.96	97
1.96-1.98	98
1.98-2.0	99

Analog Voltage	Number
0.5	25
0.75	37
0.93	46
1	50
0.93	46
0.75	37
0.5	25
0.25	12
0.067	03
0	00
0.067	03
0.25	12
0.5	25

$$x(t) = 0.5 + 0.5 \times \sin(\frac{2\pi}{10^{-3}} t)$$

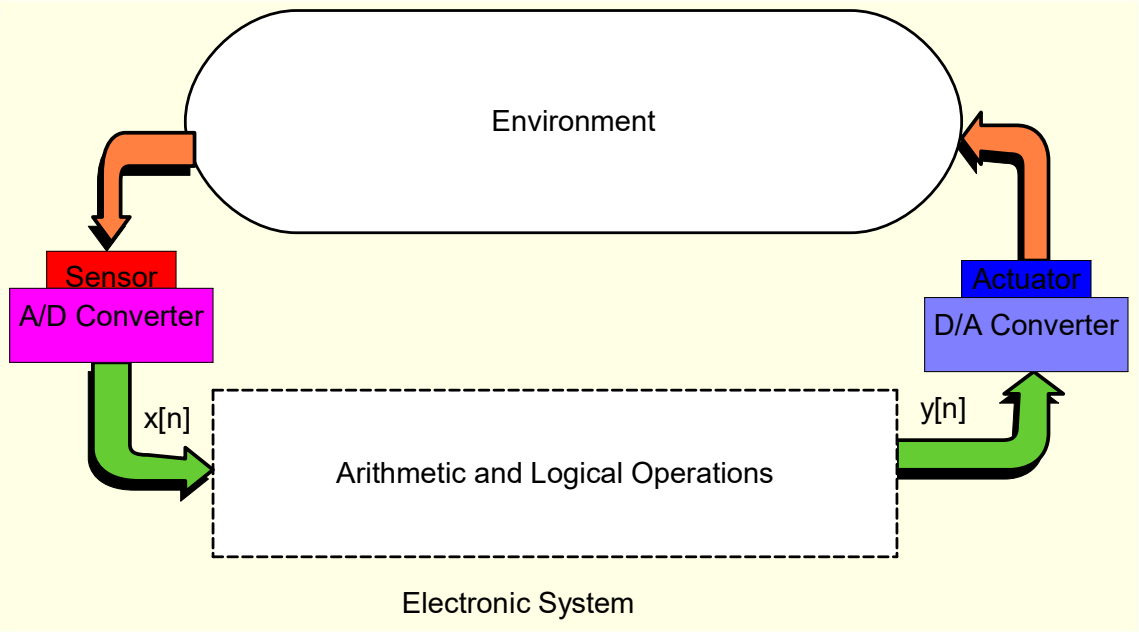


Analog Voltage	Number
0.5	25
0.75	37
0.93	46
1	50
0.93	46
0.75	37
0.5	25
0.25	12
0.067	03
0	00
0.067	03
0.25	12
0.5	25

Converting numbers back to signals

$$x(t) = 0.5 + 0.5 \times \sin(\frac{2\pi}{10^{-3}} t)$$

$$x(n) = [2, 3, 4, 5, 4, 3, 2, 1, 0, 0, 0, 1, 2....]$$



Suppose we do not carry out any processing. Y[n] = x[n]. Are we able to regenerate the original signal?

Digital to Analog Converter

$$x(n) = [2, 3, 4, 5, 4, 3, 2, 1, 0, 0, 0, 1, 2....]$$

Voltage Range (Volts)	Number
0-0.2	0
0.2-0.4	1
0.4-0.6	2
0.6-0.8	3
0.8-1.0	4
1.0-1.2	5
1.2-1.4	6
1.4-1.6	7
1.6-1.8	8
1.8-2.0	9

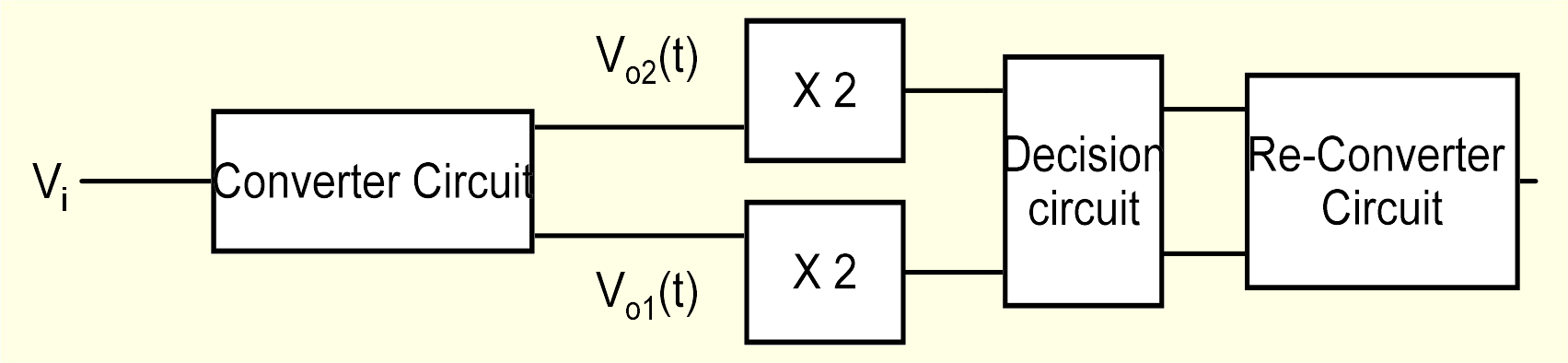
Digital voltage or Number	New Analog Voltage
2	2 x 0.2= 0.4
3	3 x 0.2=0.6
4	0.8
5	1.0
4	0.8
3	0.6
2	0.4
1	0.2
0	0
0	0
0	0
1	0.2
2	0.4

Digital to Analog Converter

Voltage Range (Volts)	Number
0-0.02	00
0.02-0.04	01
0.04-0.06	02
-	-
-	-
-	-
1.92-1.94	96
1.94-1.96	97
1.96-1.98	98
1.98-2.0	99

Analog Voltage	Number	New Analog Voltage
0.5	25	25 x 0.02 = 0.5
0.75	37	37 x 0.02 =0.74
0.93	46	0.92
1	50	1.0
0.93	46	0.92
0.75	37	0.74
0.5	25	0.5
0.25	12	0.24
0.067	03	0.06
0	00	0
0.067	03	0.06
0.25	12	0.24
0.5	25	0.5

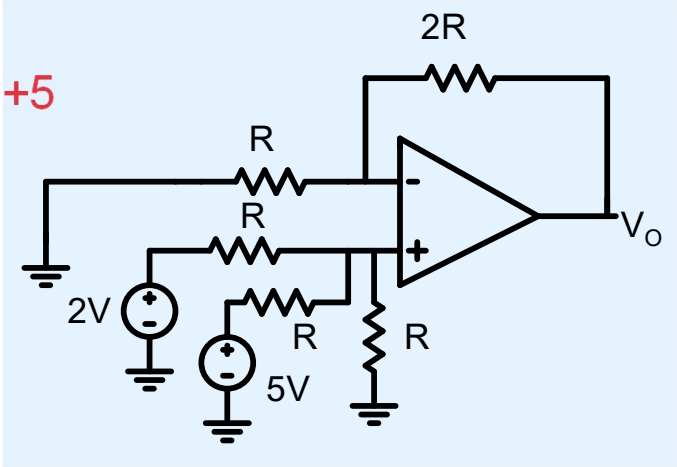
Processing of numbers in decimal system is cumbersome !



Circuits for processing numbers in binary system are much easier to implement

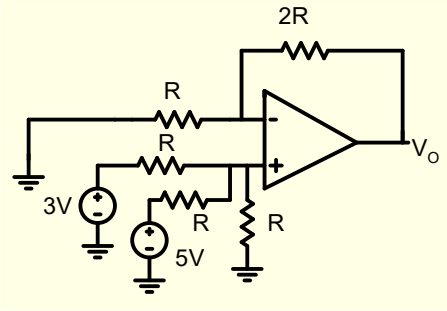
Processing of numbers in decimal system is cumbersome !

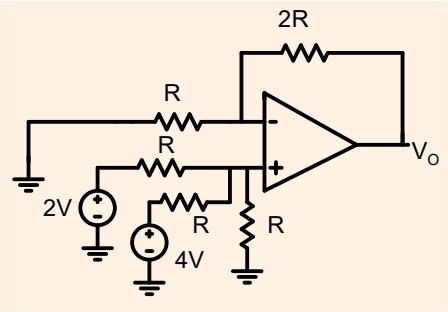
How do we add 2+5



How do we add 23+45?

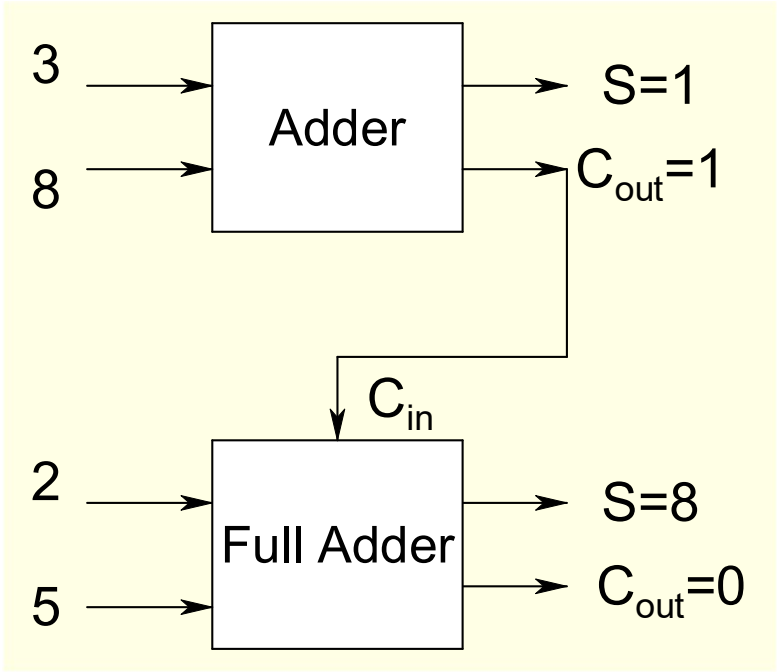
	2	3
	4	5
	6	8





How do we add 23+58 ?

$$\begin{array}{r} {\scriptstyle 1} \\ 2\ 3 \\ 5\ 8 \\ \hline 8\ 1 \end{array}$$

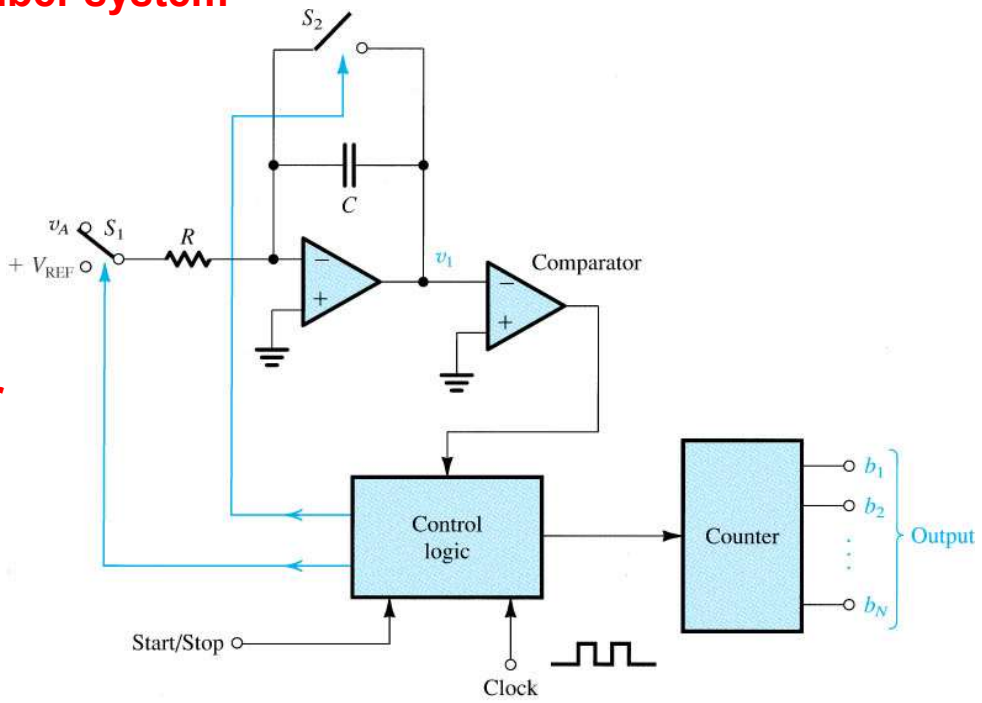


It is not easy to design circuits to carry out this operations using decimal system

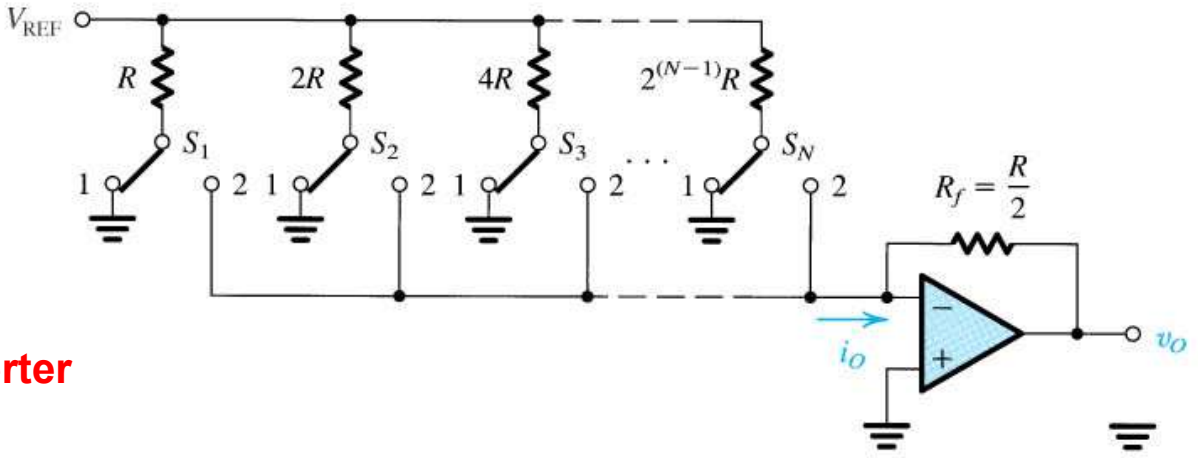
A Binary number system is more convenient !

Converter circuits for binary number system

A/D converter



D/A converter



ESC 201T : Introduction to Electronics

Lecture32: Digital Circuits-2

B. Mazhari
Dept. of EE, IIT Kanpur

Numbers

Every number system is associated with a base or radix

A positional notation is commonly used to express numbers

$$(a_5a_4a_3a_2a_1a_0)_r = a_5r^5 + a_4r^4 + a_3r^3 + a_2r^2 + a_1r^1 + a_0r^0$$

The decimal system has a base of 10 and uses symbols (0,1,2,3,4,5,6,7,8,9) to represent numbers

$$(2009)_{10} = 2\times10^3 + 0\times10^2 + 0\times10^1 + 9\times10^0$$

$$(123.24)_{10} = 1\times10^2 + 2\times10^1 + 3\times10^0 + 2\times10^{-1} + 4\times10^{-2}$$

An octal number system has a base 8 and uses symbols (0,1,2,3,4,5,6,7)

$$(2007)_8 = 2\times8^3 + 0\times8^2 + 0\times8^1 + 7\times8^0$$

What decimal number does it represent?

$$(2007)_8 = 2\times512 + 0\times64 + 0\times8^1 + 7\times8^0 = 1033$$

A hexadecimal system has a base of 16

Number	Symbol
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

$$(2BC9)_{10} = 2\times16^3 + B\times16^2 + C\times16^1 + 9\times16^0$$

How do we convert it into decimal number?

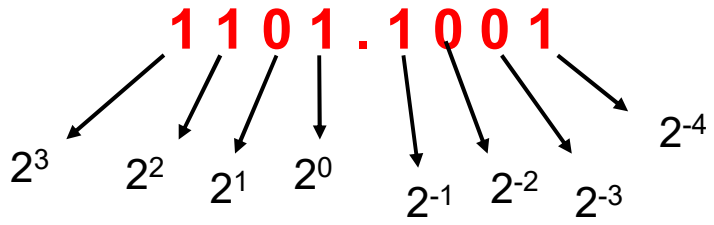
$$(2BC9)_{10} = 2\times4096 + 11\times256 + 12\times16^1 + 9\times16^0 = 11209$$

A Binary system has a base 2 and uses only two symbols 0, 1 to represent all the numbers

$$(1101)_2 = 1\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0$$

Which decimal number does this correspond to ?

$$(1101)_2 = 1\times 8 + 1\times 4 + 0\times 2^1 + 1\times 2^0 = 13$$



2^0	1
2^1	2
2^2	4
2^3	8
2^4	16
2^5	32
2^6	64
2^7	128
2^8	256
2^9	512
2^{10}	1024(K)
2^{20}	1048576(M)

2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
0.5	0.25	0.125	0.0625	0.03125	0.015625

Developing Fluency with Binary Numbers

1 1 0 0 1 = ? 25

1100001 = ? 64+32+1=97

0.101 = ? 0.5+0.125=0.625

11.001 = ? 3+0.125=3.125

Converting decimal to binary number

Convert 45 to binary number

$$(45)_{10} = b_n b_{n-1} \ldots\ldots b_0$$

$$45 = b_n 2^n + b_{n-1} 2^{n-1} \ldots\ldots b_1 2^1 + b_0$$

Divide both sides by 2

$$\frac{45}{2} = 22.5 = b_n 2^{n-1} + b_{n-1} 2^{n-2} \ldots\ldots b_1 2^0 + b_0 \times 0.5$$

$$22 + 0.5 = b_n 2^{n-1} + b_{n-1} 2^{n-2} \ldots\ldots + b_1 2^0 + b_0 \times 0.5$$

$$\Rightarrow \; b_0 = 1$$

$$22+0.5=b_n 2^{n-1}+b_{n-1}2^{n-2}.....+b_1 2^0+b_0\times 0.5 \quad \Rightarrow\ b_0=1$$

$$22=b_n 2^{n-1}+b_{n-1}2^{n-2}.....b_2 2^1+b_1 2^0$$

Divide both sides by 2

$$\frac{22}{2}=11=b_n 2^{n-2}+b_{n-1}2^{n-3}.....b_2 2^0+b_1\times 0.5 \quad \Rightarrow\ b_1=0$$

$$11=b_n 2^{n-2}+b_{n-1}2^{n-3}.....+b_3 2^1+b_2 2^0$$

$$5.5=b_n 2^{n-3}+b_{n-1}2^{n-4}.....+b_3 2^0+0.5b_2 \quad \Rightarrow\ b_2=1$$

$$5=b_n 2^{n-3}+b_{n-1}2^{n-4}.....b_4 2^1+b_3 2^0$$

$$5 = b_n 2^{n-3} + b_{n-1} 2^{n-4} \ldots\ldots b_4 2^1 + b_3 2^0$$

$$2.5 = b_n 2^{n-4} + b_{n-1} 2^{n-5} \ldots\ldots b_4 2^0 + 0.5 b_3 \quad \Rightarrow \; b_3 = 1$$

$$2 = b_n 2^{n-4} + b_{n-1} 2^{n-5} \ldots\ldots b_5 2^1 + b_4 2^0$$

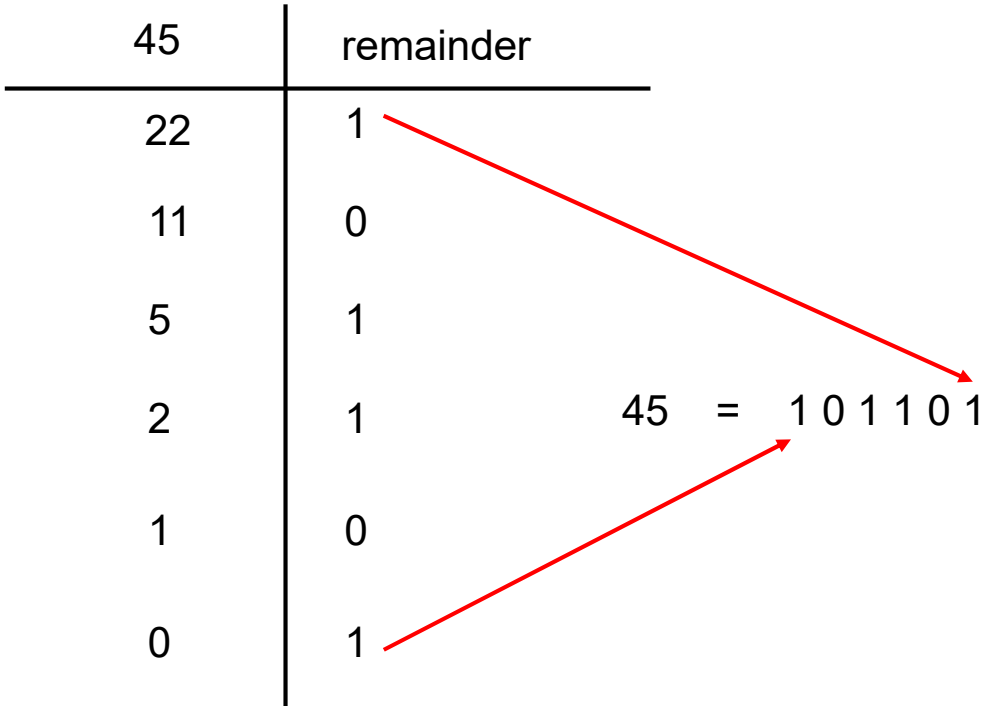
$$1 = b_n 2^{n-5} + b_{n-1} 2^{n-6} \ldots\ldots b_5 2^0 + 0.5 b_4 \quad \Rightarrow \; b_4 = 0$$

$$\Rightarrow \; b_5 = 1$$

$$(45)_{10} = b_5 b_4 b_3 b_2 b_1 b_0 = 101101$$

Converting decimal to binary number

Method of successive division by 2



Convert $(153)_{10}$ to octal number system

$$(153)_{10} = (b_n b_{n-1} \ldots\ldots b_0)_8$$

$$(153)_{10} = b_n 8^n + b_{n-1} 8^{n-1} \ldots\ldots b_1 8^1 + b_0$$

Divide both sides by 8

$$\frac{153}{8} = 19.125 = b_n 8^{n-1} + b_{n-1} 8^{n-2} \ldots\ldots b_1 8^0 + \frac{b_0}{8} \quad \Rightarrow \frac{b_0}{8} = 0.125 \quad \Rightarrow b_0 = 1$$

153	remainder
19	1
2	3
0	2

153 = $(231)_8$

Converting decimal to binary number

Convert $(0.35)_{10}$ to binary number

$$(0.35)_{10} = 0.b_{-1}b_{-2}b_{-3}.....b_{-n}$$

$$0.35 = 0 + b_{-1}2^{-1} + b_{-2}2^{-2} + ...b_{-n}2^{-n}$$

How do we find the b_{-1} b_{-2} …coefficients?

Multiply both sides by 2

$$0.7 = b_{-1} + b_{-2}2^{-1} + ...b_{-n}2^{-n+1} \qquad \Rightarrow b_{-1} = 0$$

$$0.7 = b_{-2}2^{-1} + b_{-3}2^{-2} + ...b_{-n}2^{-n+1}$$

$$0.7 = b_{-2}2^{-1} + b_{-3}2^{-2} + ...b_{-n}2^{-n+1}$$

Multiply both sides by 2

$$1.4 = b_{-2} + b_{-3}2^{-1} + ...b_{-n}2^{-n+2}$$

Note that ½+1/4+1/8+……≤1

$$\Rightarrow b_{-2} = 1$$

$$0.4 = b_{-3}2^{-1} + b_{-4}2^{-2}.....b_{-n}2^{-n+2}$$

$$0.8 = b_{-3} + b_{-4}2^{-1}.....b_{-n}2^{-n+3}$$

$$\Rightarrow b_{-3} = 0$$

Converting decimal to binary number

0.125 = ?

0 .	125	
		x2
0 .	25	
		x2
0 .	5	
		x2
1.	0	

0.125 = $(.001)_2$

0.8125 = ?

0 .	8125	
		x2
1 .	625	
		x2
1 .	25	
		x2
0.	5	
		x2
1.	0	

0.8125 = $(.1101)_2$

Binary numbers

Most significant bit or MSB

1011000111

Least significant bit or LSB

This is a 10 bit number

Binary digit = bit

decimal	2bit	3bit	4bit	5bit
0	00	000	0000	00000
1	01	001	0001	00001
2	10	010	0010	00010
3	11	011	0011	00011
4		100	0100	00100
5		101	0101	00101
6		110	0110	00110
7		111	0111	00111
8			1000	01000
9			1001	01001
10			1010	01010
11			1011	01011
12			1100	01100
13			1101	01101
14			1110	01110
15			1111	01111

N-bit binary number can represent numbers from 0 to 2^N -1

Converting Binary to Hex and Hex to Binary

$$(b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0)_b = (h_1, h_0)_{Hex}$$

$$b_7 2^7 + b_6 2^6 + b_5 2^5 + b_4 2^4 + b_3 2^3 + b_2 2^2 b_1 2^1 + b_0 = h_1 16^1 + h_0$$

$$(b_7 2^3 + b_6 2^2 + b_5 2^1 + b_4)2^4 + (b_3 2^3 + b_2 2^2 b_1 2^1 + b_0) = h_1 16^1 + h_0$$

h_1 h_0

$$(10110011)_b = (1011)(0011) = (B3)_{Hex}$$

$$(110011)_b = (11)(0011) = (33)_{Hex}$$

$$(EC)_{Hex} = (1110)(1100) = (11101100)_b$$

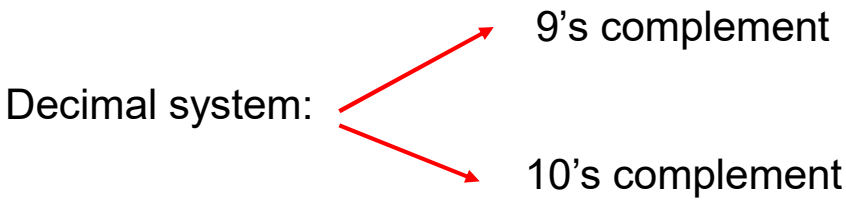
Number	Symbol
0(0000)	0
1(0001)	1
2(0010)	2
3(0011)	3
4(0100)	4
5(0101)	5
6(0110)	6
7(0111)	7
8(1000)	8
9(1001)	9
10(1010)	A
11(1011)	B
12(1100)	C
13(1101)	D
14(1110)	E
15(1111)	F

Binary Addition/Subtraction

$$\begin{array}{r} 0 \\ +\ 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ +\ 0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 0 \\ +\ 1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\ 1 \\ \hline 1\ 0 \end{array} \qquad \begin{array}{r} 1 \\ 1 \\ +\ 1 \\ \hline 1\ 1 \end{array}$$

$$\begin{array}{r} 1\ 0\ 1 \\ +\ 1\ 1\ 0 \\ \hline 1\ 0\ 1\ 1 \end{array} \qquad \begin{array}{r} 1\ 1\ 0\ 1 \\ +\ 1\ 1\ 1\ 0 \\ \hline 1\ 1\ 0\ 1\ 1 \end{array}$$

Complement of a number



9's complement of n-digit number x is $10^n - 1 - x$

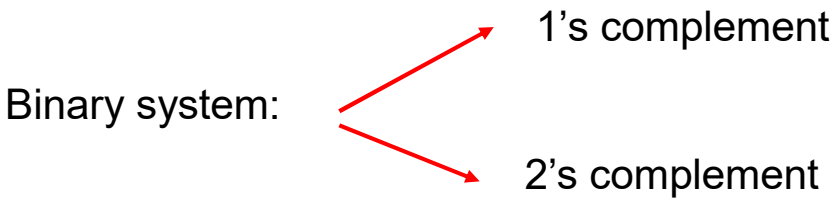
10's complement of n-digit number x is $10^n - x$

9's complement of 85 ? $10^2 - 1 - 85$ $99 - 85 = 14$

$$9\text{'s complement of } 123 = 999 - 123 = 876$$

$$10\text{'s complement of } 123 = 9\text{'s complement of } 123 + 1 = 877$$

Complement of a binary number



1's complement of n-bit number x is $2^n - 1 - x$

2's complement of n-bit number x is $2^n - x$

1's complement of 1011 ? $2^4 - 1 - 1011$ $1111 - 1011 = 0100$

1's complement is simply obtained by flipping a bit (changing 1 to 0 and 0 to 1)

$$1\text{'s complement of } 1001101 = ?$$

$$0110010$$

$$\text{2's complement of } 1010 = \text{1's complement of } 1010+1 = 0110$$

$$\text{2's complement of } 110010 =$$

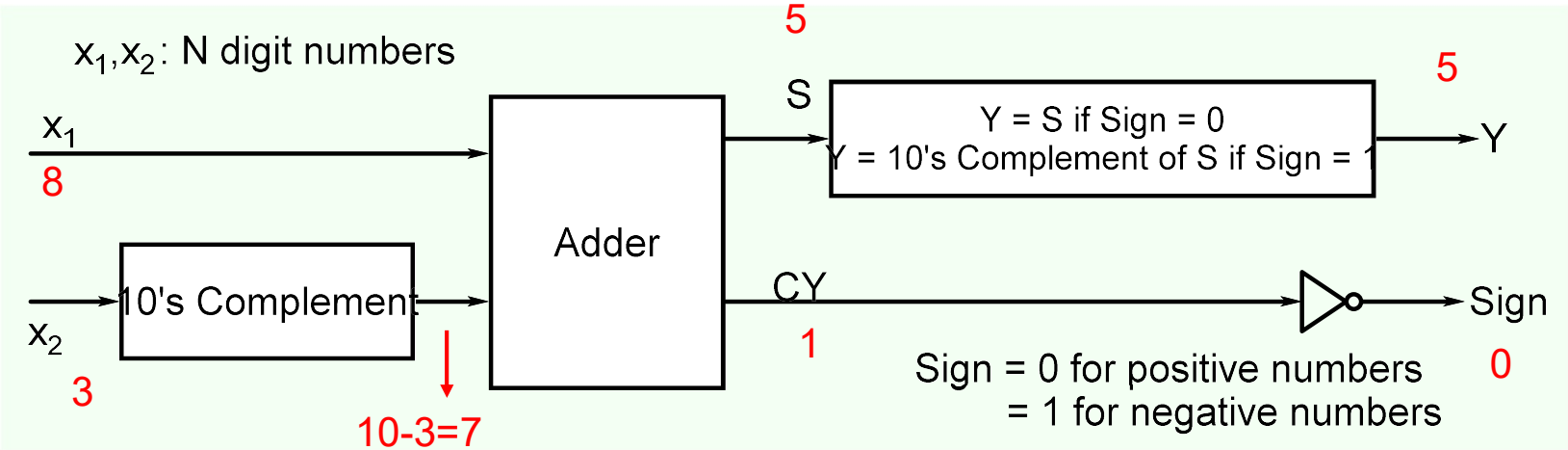
Leave all least significant 0's as they are, leave first 1 unchanged and then flip all subsequent bits

$$001110$$

$$1011 \rightarrow 0101$$

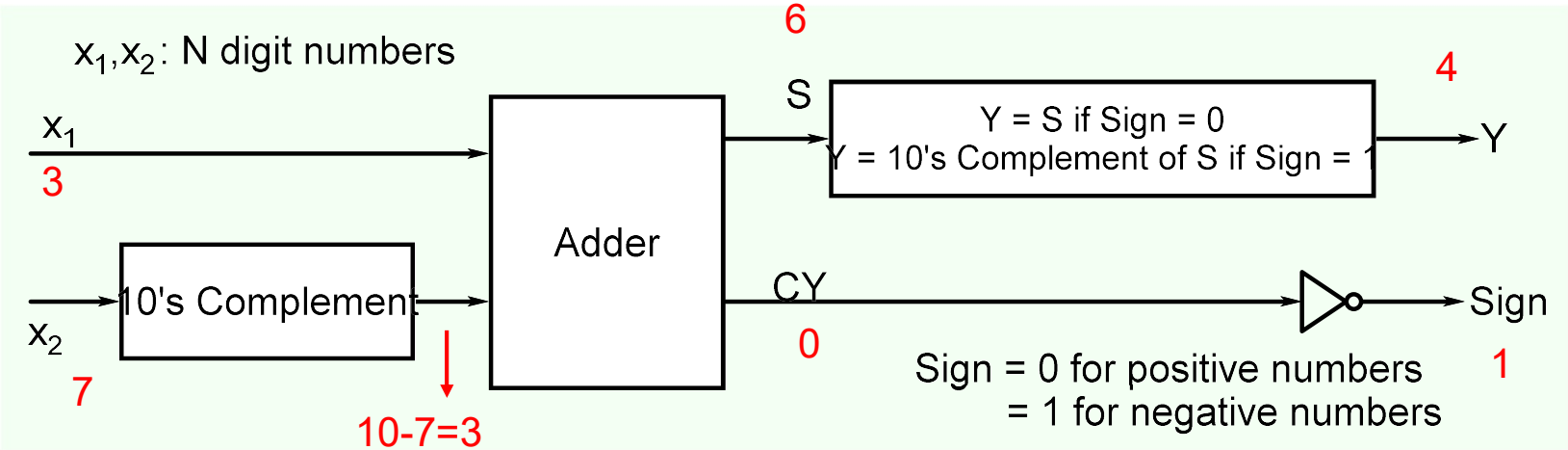
$$101101100 \rightarrow 010010100$$

Subtraction using 10's complement



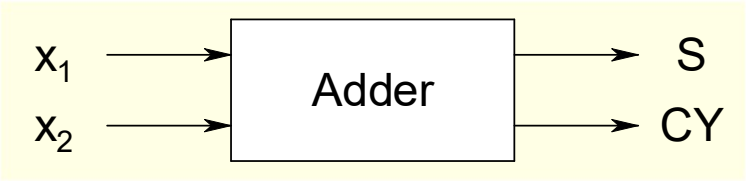
This way of subtraction would make sense only if subtracting a number x_2 from 10^N is much simpler than directly subtracting it directly from x_1

Subtraction using 10's complement

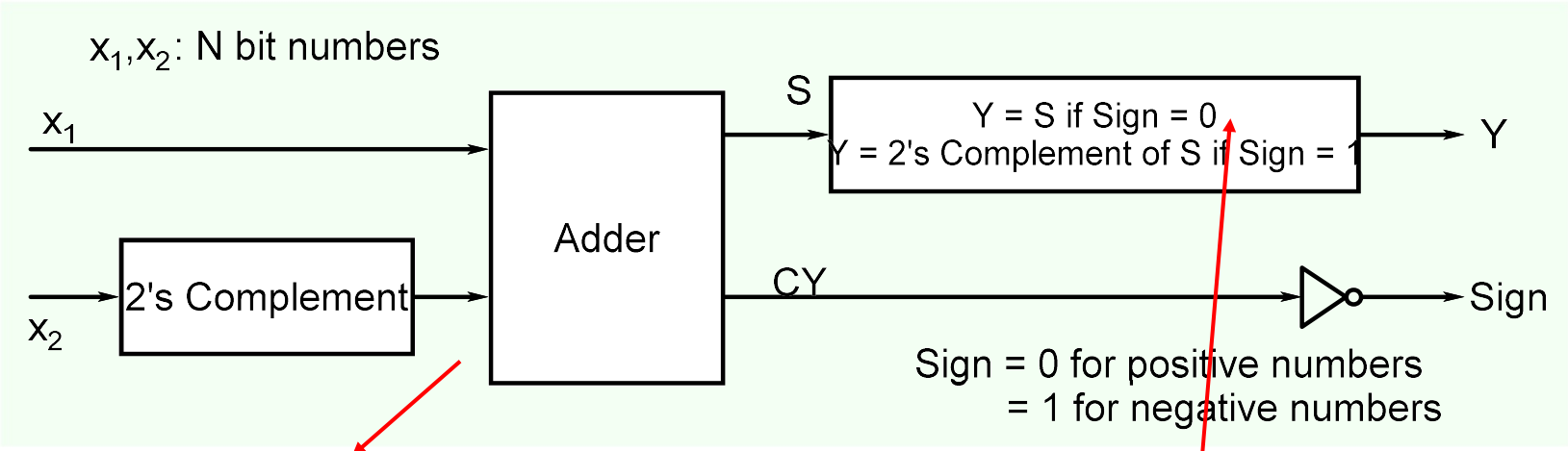


This way of subtraction would make sense only if subtracting a number x_2 from 10^N is much simpler than directly subtracting it directly from x_1

Advantages of using 2's complement

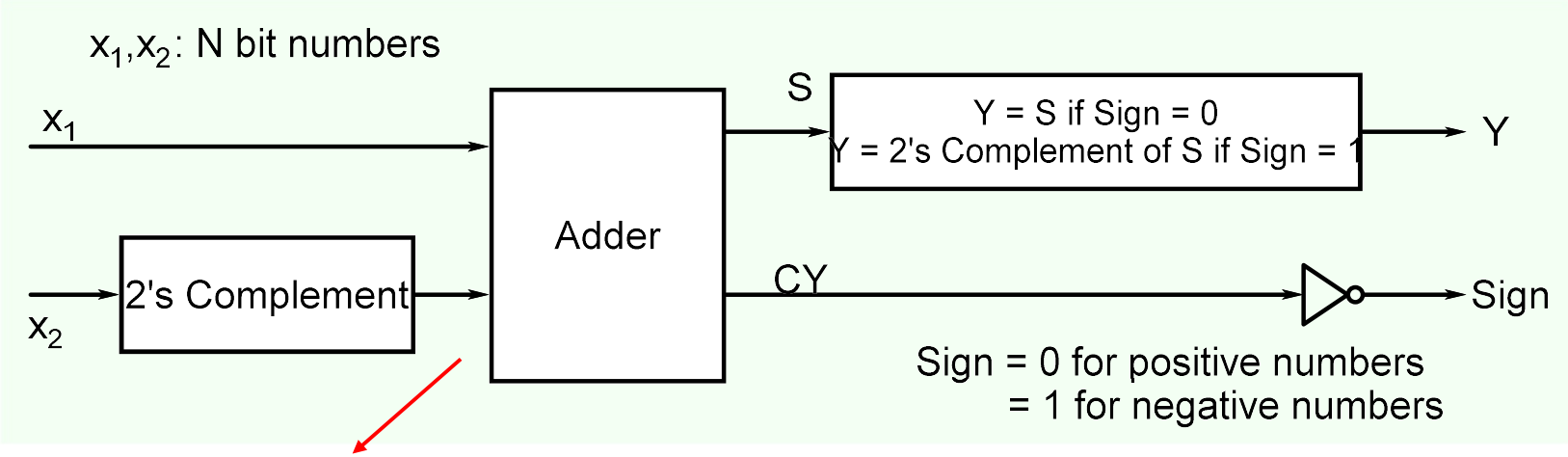


Can we carry out Y = $X_1 - X_2$ using such an adder?



Note that carry will be there only if $x_1 - x_2$ is positive as 2^N is N+1 bits (1 followed by N zeros)

Advantages of using 2's complement



$2^N - x_2$

$$(CY, S) = x_1 + 2^N - x_2$$

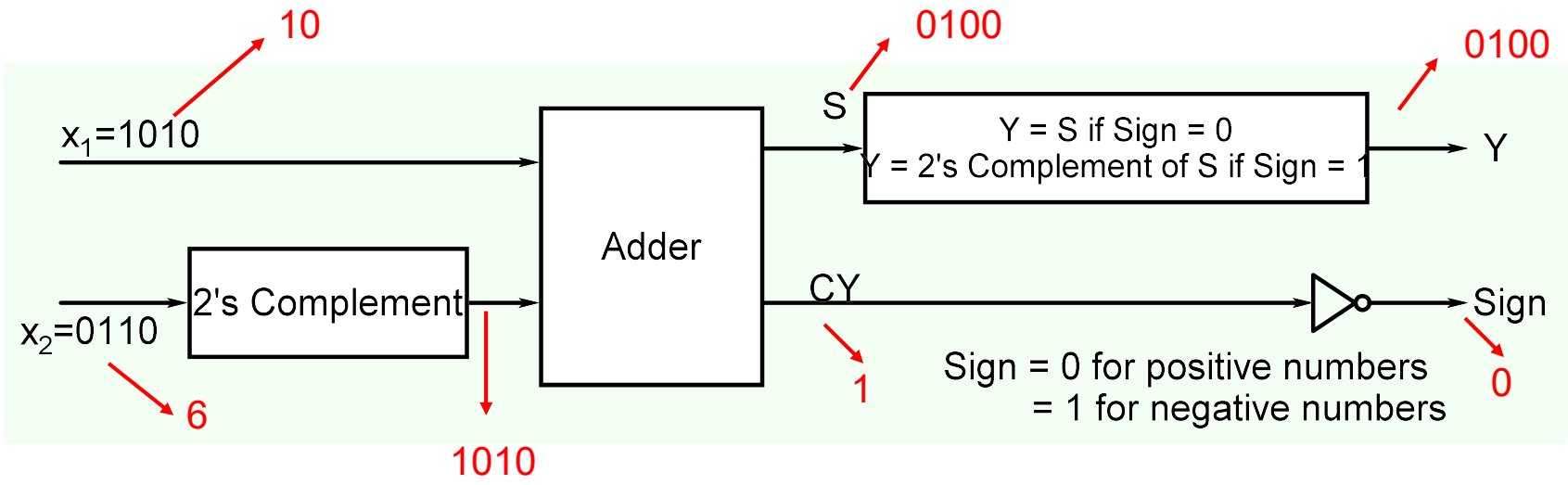
Note that carry will be there only if $x_1 - x_2$ is positive as 2^N is N+1 bits (1 followed by N zeros)

A zero carry implies a negative number whose magnitude ($x_2 - x_1$) can be found as follows:

$$S = x_1 + 2^N - x_2$$

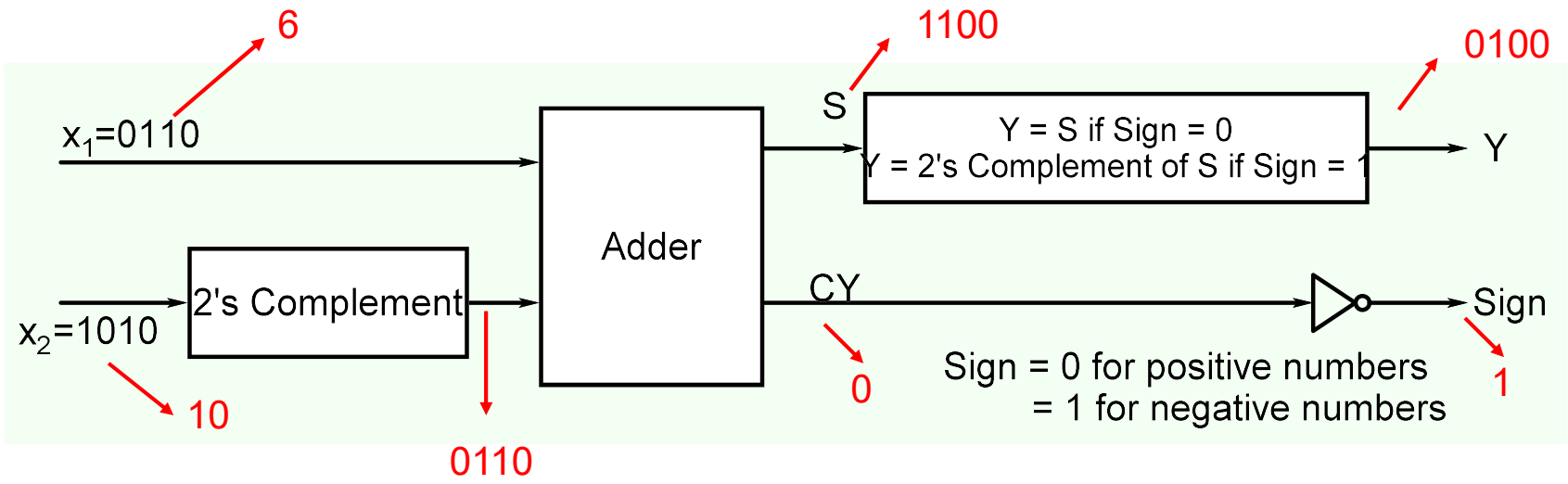
$$2'\text{scomplement of } S = 2^N - (x_1 + 2^N - x_2) = x_2 - x_1$$

Example



$$
\begin{array}{r}
1\,0\,1\,0 \\
+\,1\,0\,1\,0 \\
\hline
1\,0\,1\,0\,0 \\
\hline
\end{array}
$$

Example



$$\begin{array}{r} 0\,1\,1\,0 \\ +\,0\,1\,1\,0 \\ \hline 1\,1\,0\,0 \\ \hline \end{array}$$

It makes sense to use adder as a subtractor as well provided additional circuit required for carrying out 2's complement is simple

Representing positive and negative binary numbers

One extra bit is required to carry sign information. Sign bit = 0 represents positive number and Sign bit = 1 represents negative number

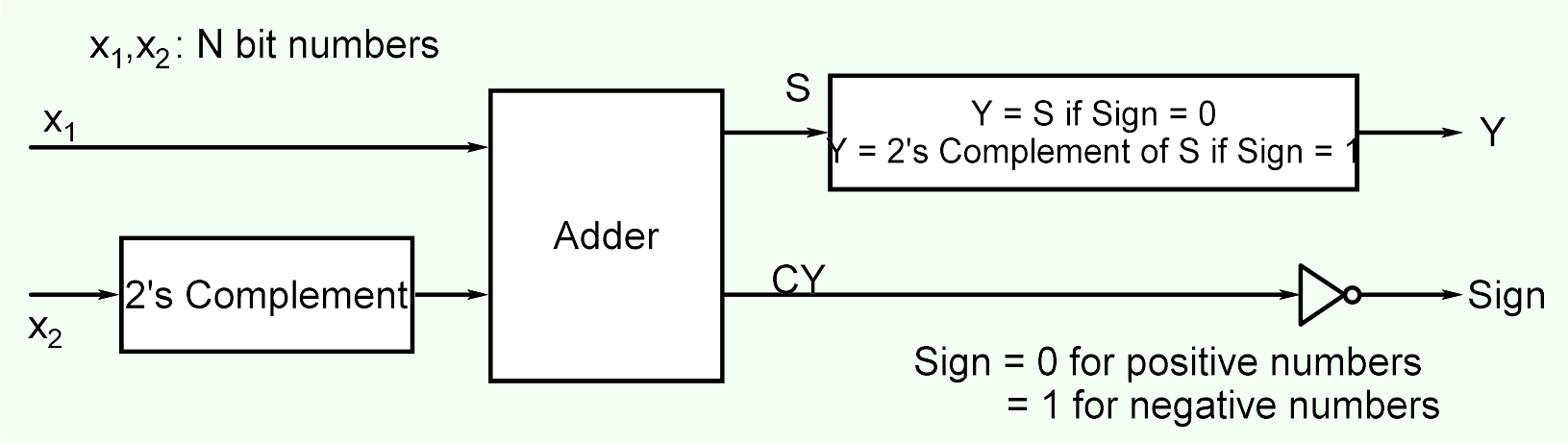
decimal	Signed Magnitude
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
-0	1000
-1	1001
-2	1010
-3	1011
-4	1100
-5	1101
-6	1110
-7	1111

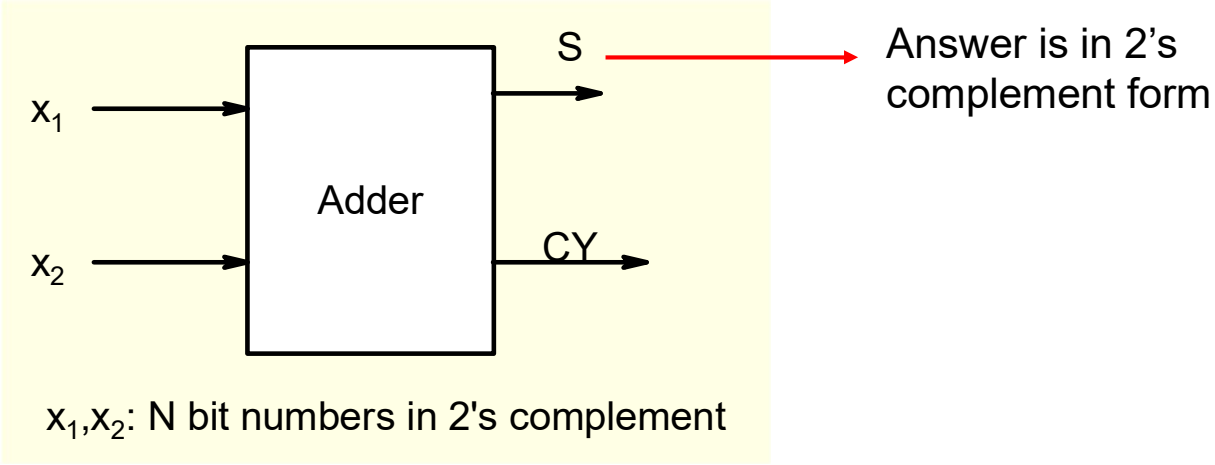
decimal	Signed 1's complement
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
-0	1111
-1	1110
-2	1101
-3	1100
-4	1011
-5	1010
-6	1001
-7	1000

decimal	Signed 2's complement
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001

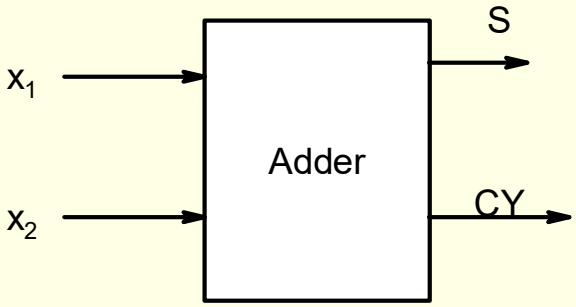
If we represent numbers in 2's complement form carrying out subtraction is same as addition

decimal	Signed 2's complement
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001





Example



x_1,x_2: N bit numbers in 2's complement

+ 5	0 1 0 1
+ 2	+ 0 0 1 0
+7	0 1 1 1

+ 5	0 1 0 1
- 2	+ 1 1 1 0
+3	0 0 1 1

- 5	1 0 1 1
+ 2	+ 0 0 1 0
- 3	1 1 0 1

2's complement is 0011 = 3

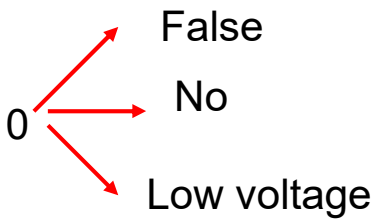
- 5	1 0 1 1
- 2	+ 1 1 1 0
- 7	1 0 0 1

2's complement is 0111 = 7

Boolean Algebra

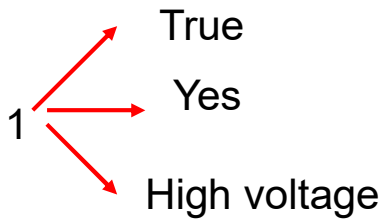
Algebra on Binary numbers

A variable x can take two values {0,1}



Basic operations:

$$\text{AND:} \quad y = x_1 . x_2$$



Y is 1 if and only if both x_1 and x_2 are 1, otherwise zero

Truth Table

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Basic operations:

OR: $y = x_1 + x_2$

Y is 1 if either x_1 and x_2 is 1. Or y= 0 if and only if both variables are zero

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

NOT: $y = \overline{x}$

x	y
0	1
1	0

Boolean Algebra

Basic Postulates

P1:	$x + 0 = x$
P2:	$x + y = y + x$
P3:	$x.(y+z) = x.y+x.z$
P4:	$x + \overline{x} = 1$

P1:	$x \,.\, 1 = x$
P2:	$x \,.\, y = y \,.\, x$
P3:	$x+y.z = (x+y).(x+z)$
P4:	$x \,.\, \overline{x} = 0$

Basic Theorems

T1:	$x + x = x$
T2:	$x + 1 = 1$
T3:	$\overline{(\overline{x})} = x$
T4:	$x + (y+z) = (x+y)+z$
T5	$\overline{(x+y)} = \overline{x} \,.\, \overline{y}$ (DeMorgan's theorem)
T6:	$x+ x.y = x$

T1:	$x \,.\, x = x$
T2:	$x \,.\, 0 = 0$
T4:	$x \,.\, (y.z) = (x.y).z$
T5	$\overline{(x.y)} = \overline{x} + \overline{y}$ (DeMorgan's theorem)
T6:	$x.(x+y) = x$

Proving theorems

P1:	$x + 0 = x$
P2:	$x + y = y + x$
P3:	$x.(y+z) = x.y+x.z$
P4:	$x + \bar{x} = 1$

P1:	$x \,.\, 1 = x$
P2:	$x \,.\, y = y \,.\, x$
P3:	$x+y.z = (x+y).(x+z)$
P4:	$x \,.\, \bar{x} = 0$

Prove T1: $x + x = x$

$x + x = (x+x).\ 1$ (P1)

$= (x+x).\ (x+\bar{x})$ (P4)

$= x + x.\bar{x}$ (P3)

$= x + 0$ (P4)

$= x$ (P1)

Prove T1: $x \,.\, x = x$

$x \,.\, x = x.x + 0$ (P1)

$= x.x + x.\bar{x}$ (P4)

$= x \,.\, (x+\bar{x})$ (P3)

$= x \,.\, 1$ (P4)

$= x$ (P1)

Proving theorems

P1: $x + 0 = x$

P2: $x + y = y + x$

P3: $x.(y+z) = x.y+x.z$

P4: $x + \overline{x} = 1$

P1: $x \,.\, 1 = x$

P2: $x \,.\, y = y \,.\, x$

P3: $x+y.z = (x+y).(x+z)$

P4: $x \,.\, \overline{x} = 0$

Prove : $x + 1 = 1$

$x + 1 = x+(x+ \overline{x})$

$= (x+x)+ \overline{x}$

$= x + \overline{x}$

$= 1$

$x + x \,.y = x$

$= x \,.\, 1 + x.\, y$

$= x.\, (1+ y)$

$= x \,.\, 1$

$= x$

$x + \overline{x} \,.y = x+y$

$= (x + \overline{x}\,).\, (x+ y)$

$= 1.\, (x+ y)$

$= x + y$

DeMorgan's theorem

$$\overline{(x_1 + x_2 + x_3 + ...)} = \overline{x_1} \,.\, \overline{x_2} \,.\, \overline{x_3} \,.$$

$$\overline{(x_1 .\, x_2 .\, x_3 .....)} = (\overline{x_1} + \overline{x_2} + \overline{x_3} + ...)$$

Simplification of Boolean expressions

$$\overline{(x_1 + x_2 + x_3 + ...)} = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot$$

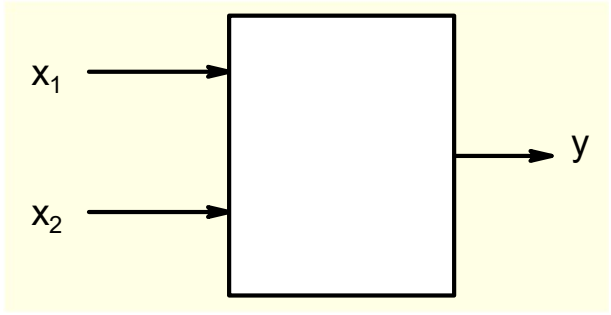
$$\overline{(\overline{x_1} \cdot x_2 + \overline{x_2} \cdot x_3)} = ?$$

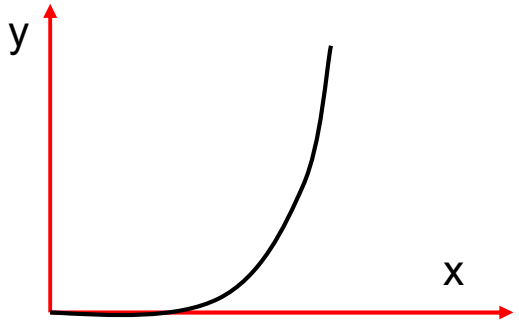
$$\overline{(x_1 \cdot x_2 \cdot x_3 .....)} = (\overline{x_1} + \overline{x_2} + \overline{x_3} + ...)$$

$$= (x_1 + \overline{x_2}) \cdot (x_2 + \overline{x_3})$$

$$= x_1 \cdot x_2 + x_1 \cdot \overline{x_3} + \overline{x_2} \cdot \overline{x_3}$$

Function of Boolean variables





$$y = x^2$$

x_1	x_2	y
0	0	0
0	1	1
1	0	0
1	1	0

Y = 1 when x_1 is 0 and x_2 is 1

$$y = \overline{x_1} \cdot x_2$$

Boolean expression

Obtaining Boolean expressions from truth Table

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	0

$$y = \overline{x_1} \,.\, \overline{x_2}$$

x_1	x_2	y
0	0	0
0	1	0
1	0	1
1	1	0

$$y = x_1 \,.\, \overline{x_2}$$

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	1

$$\overline{x_1} \,.\, \overline{x_2}$$

$$x_1 \,.\, x_2$$

$$y = \overline{x_1} \,.\, \overline{x_2} + x_1 . x_2$$

Obtaining Boolean expressions from truth Table

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

$$y = \overline{x_1} \cdot x_2 + x_1 \cdot \overline{x_2}$$

Instead of writing expressions as sum of terms that make y equal to 1, we can also write expressions using terms that make y equal to 0

x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

$$y = \overline{x_1} \cdot \overline{x_2} + \overline{x_1} \cdot x_2 + x_1 \cdot \overline{x_2}$$

$$y = \overline{x_1} + \overline{x_2}$$

x_1	x_2	y
0	0	1
0	1	0
1	0	1
1	1	1

$$y = x_1 + \overline{x_2}$$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

$$y = x_1 + x_2$$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

$$x_1 + x_2$$

$$y = (x_1 + x_2).(\overline{x_1} + \overline{x_2})$$

$$\overline{x_1} + \overline{x_2}$$

Obtaining Boolean expressions from truth Table

x_1	x_2	x_3	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$$y = \overline{x_1} \,.\, \overline{x_2} \,.\, x_3 + \overline{x_1} \,.\, x_2 \,.\, x_3 + x_1 \,.\, \overline{x_2} \,.\, x_3 + x_1 \,.\, x_2 \,.\, x_3$$

Sum of Products (SOP) form

$$y = (x_1 + x_2 + x_3).(x_1 + \overline{x_2} + x_3).(\overline{x_1} + x_2 + x_3).(\overline{x_1} + \overline{x_2} + x_3)$$

Product of Sum (POS) form

ESC201T : Introduction to Electronics

Lecture 33: Digital Circuits-3

B. Mazhari
Dept. of EE, IIT Kanpur

Obtaining Boolean expressions from truth Table

x_1	x_2	x_3	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$$y = \overline{x_1} \,.\, \overline{x_2} \,.\, x_3 + \overline{x_1} \,.\, x_2 \,.\, x_3 + x_1 \,.\, \overline{x_2} \,.\, x_3 + x_1 \,.\, x_2 \,.\, x_3$$

Sum of Products (SOP) form

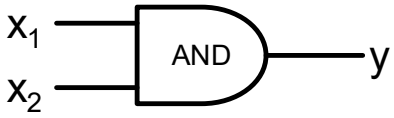
$$y = (x_1 + x_2 + x_3).(x_1 + \overline{x_2} + x_3).(\overline{x_1} + x_2 + x_3).(\overline{x_1} + \overline{x_2} + x_3)$$

Product of Sum (POS) form

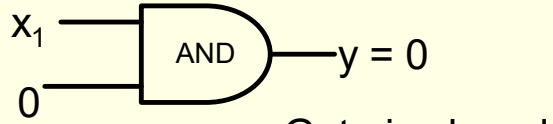
Implementing Boolean expressions

Elementary Gates

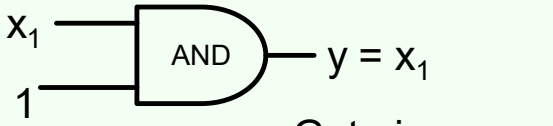
AND: $y = x_1 . x_2$



Why call it a gate?

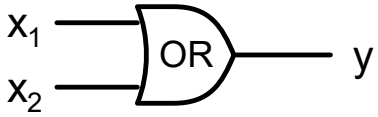


Gate is closed

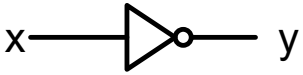


Gate is open

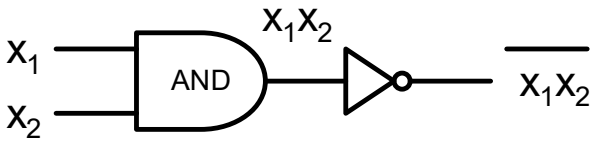
OR: $y = x_1 + x_2$



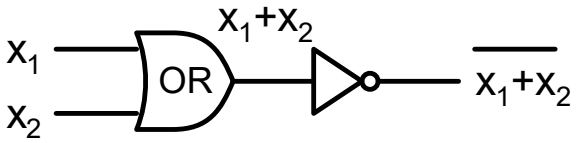
NOT: $y = \overline{x}$



NAND: $y = \overline{x_1 . x_2}$



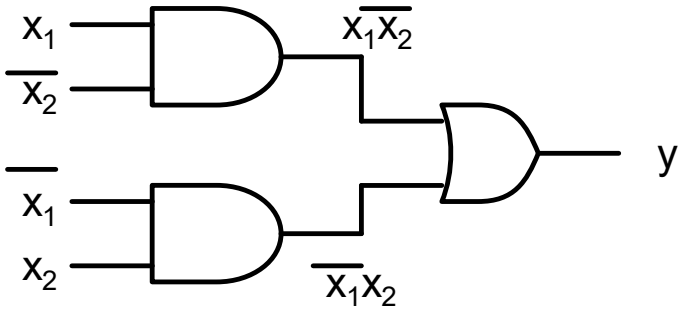
NOR: $y = \overline{x_1 + x_2}$

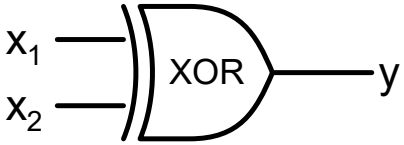


XOR: $y = x_1 \oplus x_2 = x_1.\overline{x_2} + \overline{x_1}.x_2$

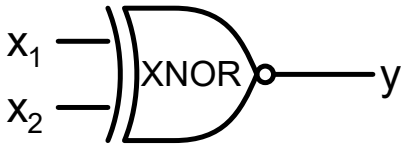
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Y is 1 if only one variable is 1 and the other is zero





XNOR: $y = x_1 \odot x_2 = x_1.x_2 + \overline{x_1}.\overline{x_2}$

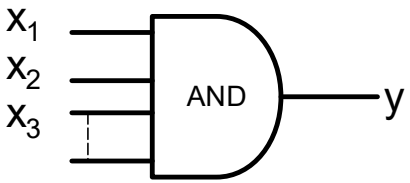


Y is 1 if only both variables are either 0 or 1

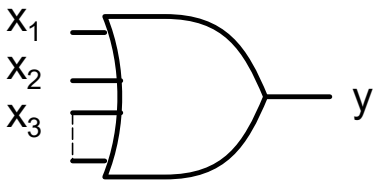
$$y = x_1 \odot x_2 = \overline{x_1 \oplus x_2}$$

Gates with more than 2 inputs

AND: $y = x_1 . x_2 . x_3 \ldots$



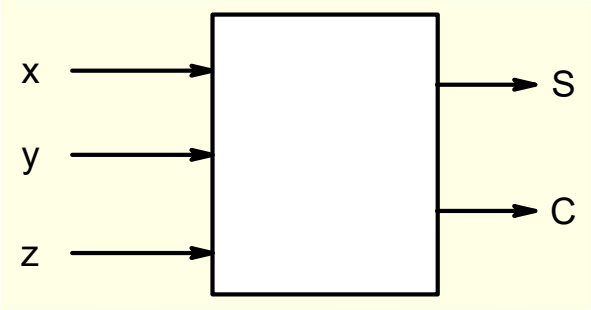
OR: $y = x_1 + x_2 + x_3 + \ldots.$



XOR: $y = x_1 \oplus x_2 \oplus x_3 = x_1 . \overline{x_2}\,\overline{x_3} + \overline{x_1} . x_2 . \overline{x_3} + \overline{x_1} . \overline{x_2} . x_3 + x_1 . x_2 . x_3$

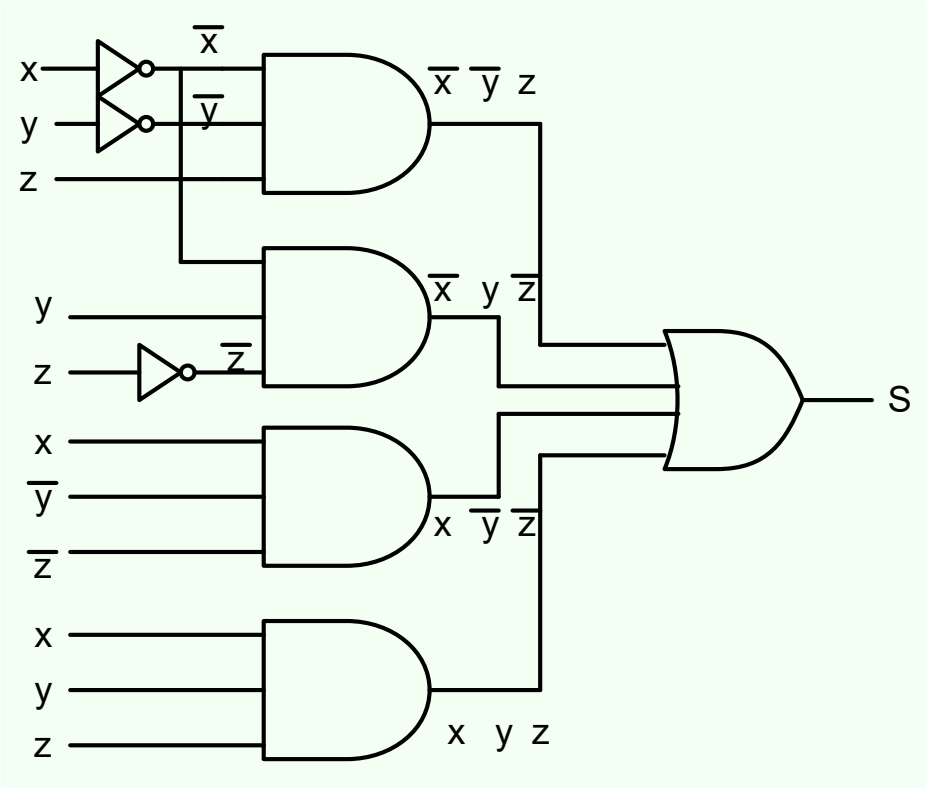
Y = 1 only if odd number of inputs is 1

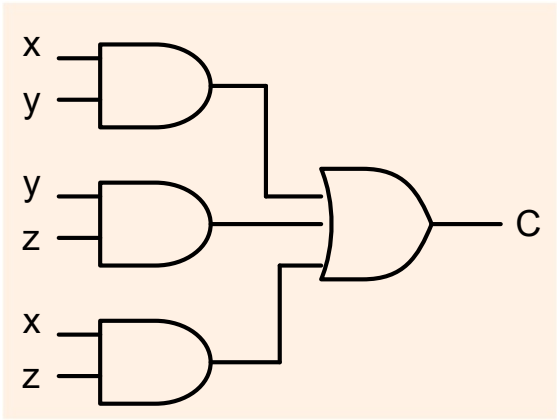
Implementing Boolean expressions using gates



$$S = \bar{x}.\bar{y}.z + \bar{x}.y.\bar{z} + x.\bar{y}.\bar{z} + x.y.z$$

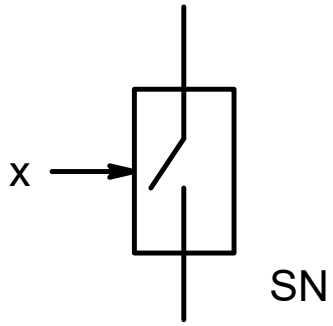
$$C = x.y + x.z + y.z$$





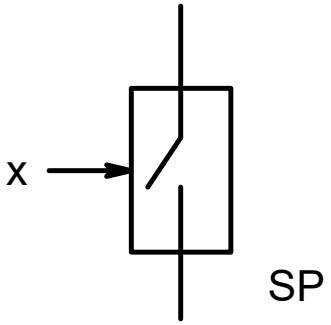
Implementing gates using Switches

Voltage controlled Switch SN:



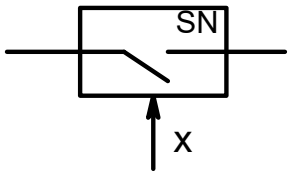
Switch is closed if voltage x is HIGH
Switch is open if voltage x is LOW

Voltage controlled Switch SP:

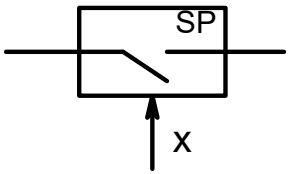


Switch is closed if voltage x is LOW
Switch is open if voltage x is HIGH

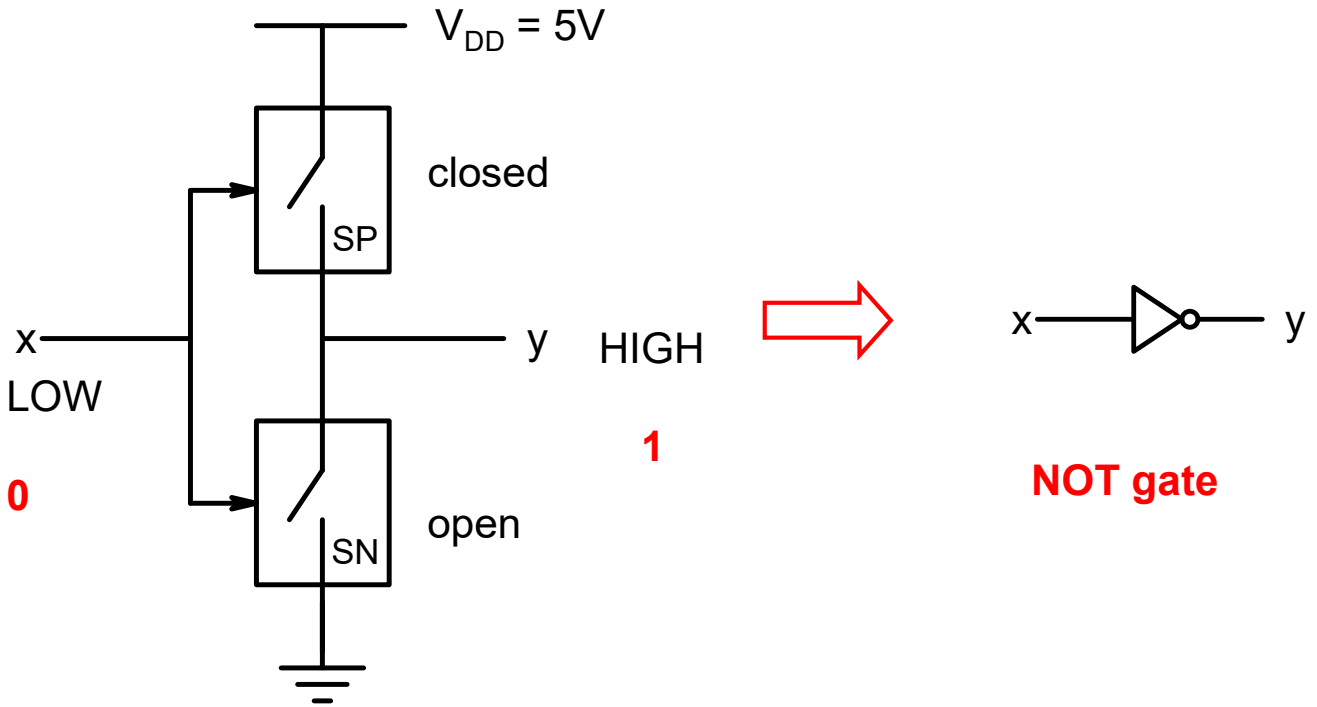
We have seen earlier that transistors act as switches !



Switch is closed if voltage x is HIGH
Switch is open if voltage x is LOW

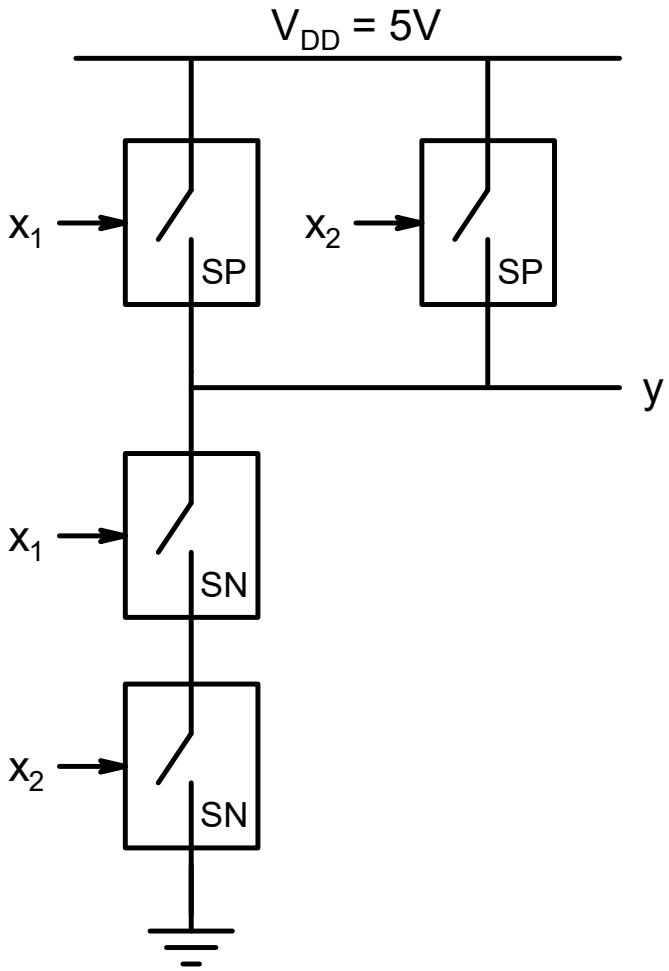


Switch is closed if voltage x is LOW
Switch is open if voltage x is HIGH



NAND Gate

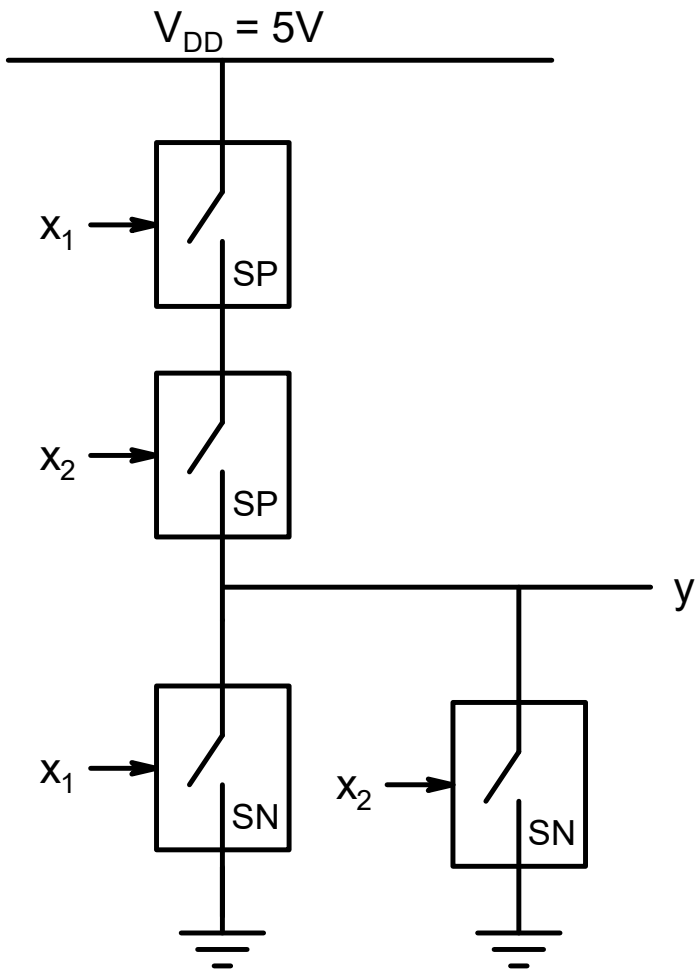
NAND: $y = \overline{x_1 . x_2}$



x_1	x_2	y
LOW	LOW	HIGH
LOW	HIGH	HIGH
HIGH	LOW	HIGH
HIGH	HIGH	LOW

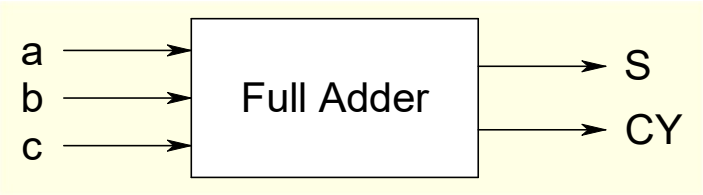
NOR Gate

NOR: $y = \overline{x_1 + x_2}$



x_1	x_2	y
LOW	LOW	HIGH
LOW	HIGH	LOW
HIGH	LOW	LOW
HIGH	HIGH	LOW

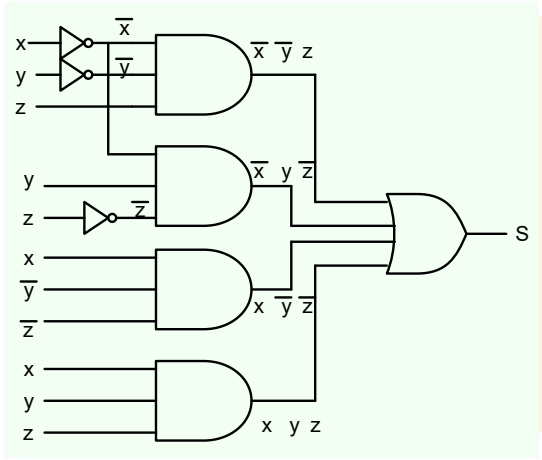
Design Overview

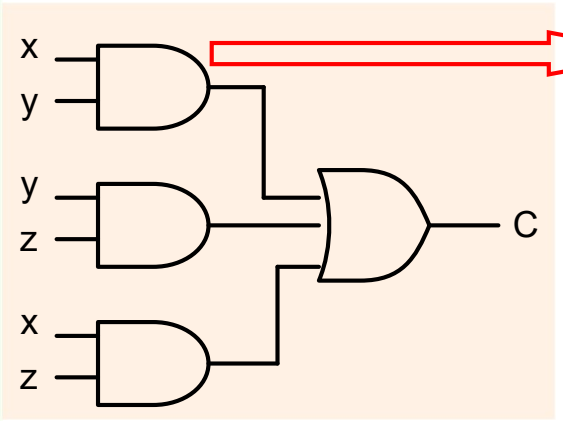


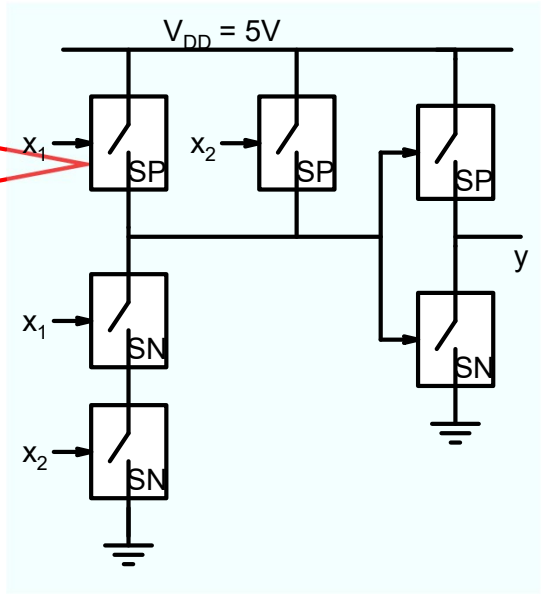
a	b	c	S	CY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = \bar{x}.\bar{y}.z + \bar{x}.y.\bar{z} + x.\bar{y}.\bar{z} + x.y.z$$

$$C = x.y + x.z + y.z$$







Representation of Boolean Expressions

x	y	f_1
0	0	0
0	1	1
1	0	1
1	1	0

x	y	min term
0	0	$\bar{x}.\bar{y}$ m0
0	1	$x.\bar{y}$ m1
1	0	$\bar{x}.y$ m2
1	1	$x.y$ m3

$$f_1 = \bar{x}.\ y + x.\bar{y}$$

$$f_1 = m_1 + m_2$$

$$f_1 = \sum(1,2)$$

$$f_2 = \sum(0,2,3) = ?$$

$$f_2 = \bar{x}.\ \bar{y} + x.\bar{y} + x.y$$

A minterm is a product that contains all the variables used in a function

Three variable functions

x	y	z	min terms	
0	0	0	$\bar{x}.\bar{y}.\bar{z}$	m0
0	0	1	$\bar{x}.\bar{y}.z$	m1
0	1	0	$\bar{x}.y.\bar{z}$	m2
0	1	1	$\bar{x}.y.z$	m3
1	0	0	$x.\bar{y}.\bar{z}$	m4
1	0	1	$x.\bar{y}.z$	m5
1	1	0	$x.y.\bar{z}$	m6
1	1	1	$x.y.z$	m7

$$f_2 = \sum(1,4,7) = ?$$

$$f_2 = \bar{x}.\bar{y}.z + x.\bar{y}.\bar{z} + x.y.z$$

Product of Sum Terms Representation

x	y	f_1
0	0	1
0	1	0
1	0	0
1	1	1

x	y	Max term
0	0	$x + y$ M0
0	1	$x + \bar{y}$ M1
1	0	$\bar{x} + y$ M2
1	1	$\bar{x} + \bar{y}$ M3

$$f_1 = (x + \bar{y}).(\bar{x} + y)$$

$$f_1 = M_1.M_2$$

$$f_1 = \Pi(1,2)$$

$$f_2 = \Pi(0,3) = ?$$

$$f_2 = (x + y).(\bar{x} + \bar{y})$$

x	y	z	Max. terms	
0	0	0	$x + y + z$	M0
0	0	1	$x + y + \bar{z}$	M1
0	1	0	$x + \bar{y} + z$	M2
0	1	1	$x + \bar{y} + \bar{z}$	M3
1	0	0	$\bar{x} + y + z$	M4
1	0	1	$\bar{x} + y + \bar{z}$	M5
1	1	0	$\bar{x} + \bar{y} + z$	M6
1	1	1	$\bar{x} + \bar{y} + \bar{z}$	M7

$$f_1 = \Pi(1,5,7) = ?$$

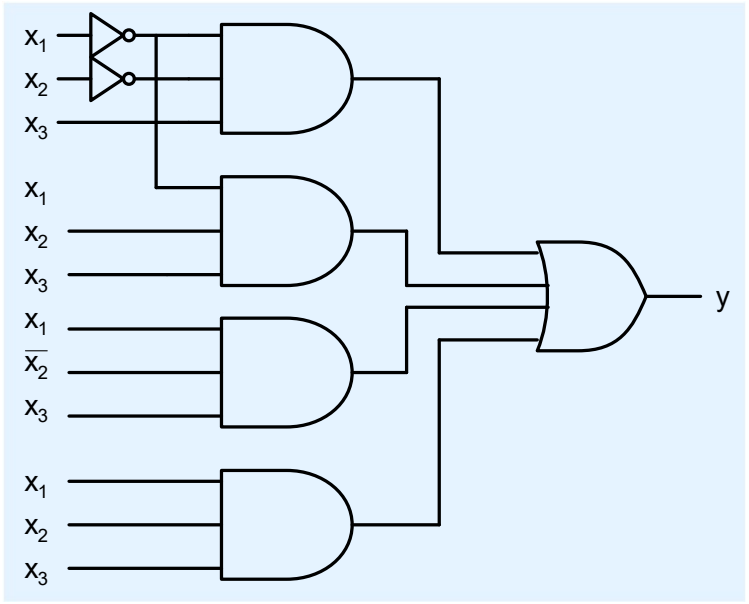
$$f_2 = (x + y + \bar{z}).(\bar{x} + y + \bar{z}).(\bar{x} + \bar{y} + \bar{z})$$

Simplification

x_1	x_2	x_3	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$$y = \sum (1,3,5,7)$$

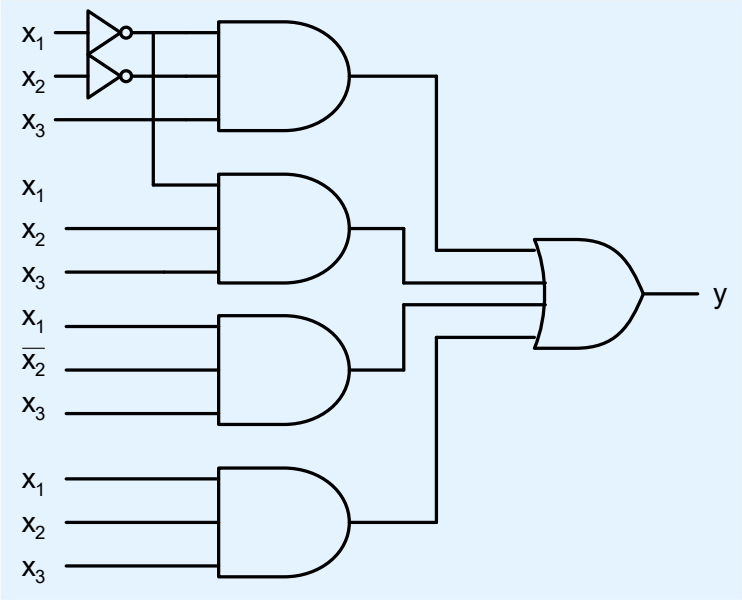
$$y = \overline{x_1} \,.\, \overline{x_2} \,.\, x_3 + \overline{x_1} \,.\, x_2 \,.\, x_3 + x_1 \,.\, \overline{x_2} \,.\, x_3 + x_1 \,.\, x_2 \,.\, x_3$$



Simplification of Boolean expression yields : $y = x_3$!! which does not require any gates at all !

Goal of Simplification

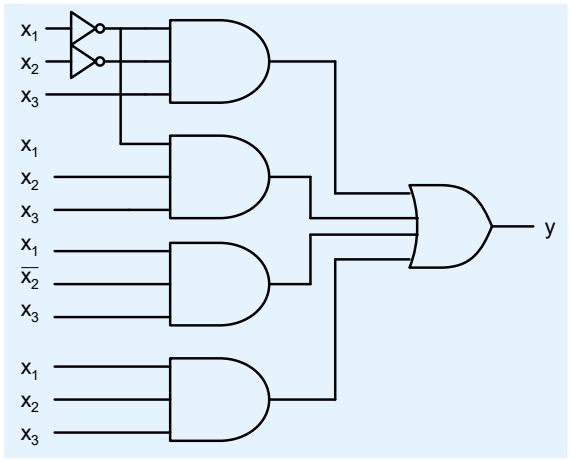
$$y = \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot x_2 \cdot x_3 + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3$$

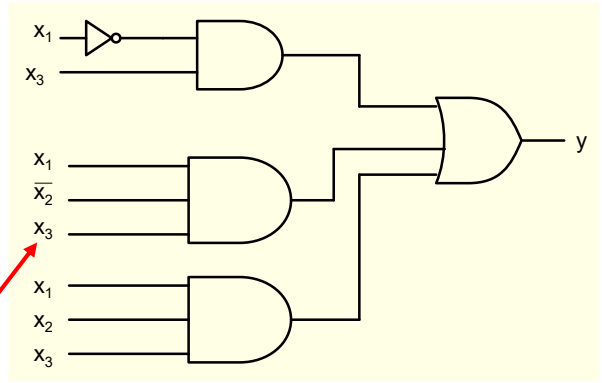


Goal of simplification is to reduce the complexity of gate circuit. This requires that we minimize the number of gates. Since number of gates depends on number of minterms, one of the goals of simplification is to **minimize the number of minterms in SOP expression**

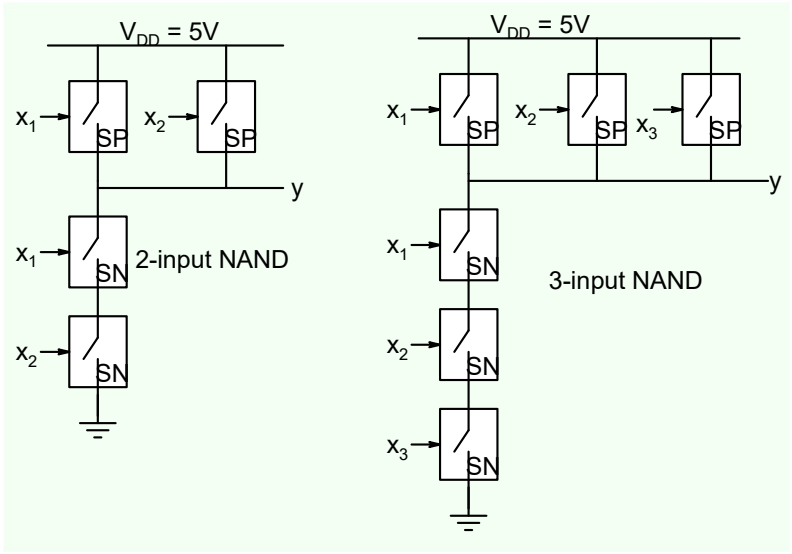
$$y = \overline{x_1}\,.\,\overline{x_2}\,.x_3 + \overline{x_1}.x_2.x_3 + x_1.\overline{x_2}.x_3 + x_1.x_2.x_3$$

$$\Rightarrow \quad y = \overline{x_1}\,.x_3 + x_1.\overline{x_2}.x_3 + x_1.x_2.x_3$$





This circuit is simpler not just because it uses 4 gates instead of 5 but also because circuit-2 uses one 2-input and three 3-input gates as compared to five 3-input gates used in circuit-1



Goal of Simplification

$$y = \overline{x_1}.\overline{x_2}.x_3 + \overline{x_1}.x_2.x_3 + x_1.\overline{x_2}.x_3 + x_1.x_2.x_3 \quad \Rightarrow \quad y = \overline{x_1}.x_3 + x_1.\overline{x_2}.x_3 + x_1.x_2.x_3$$

In the SOP expression:

1. Minimize number of product terms
2. Minimize number of literals in each term

$$\text{Simplification} \Rightarrow \text{Minimization}$$

Minimization

$$y = \overline{x_1} \,.\, \overline{x_2} \,.\, x_3 + \overline{x_1} \,.\, x_2 \,.\, x_3 + x_1 \,.\, \overline{x_2} \,.\, x_3 + x_1 \,.\, x_2 \,.\, x_3$$

$$y = \overline{x_1} \,.\, x_3 \,.\, (\overline{x_2} + x_2) + x_1 \,.\, x_3 \,.\, (\overline{x_2} + x_2)$$

$$y = \overline{x_1} \,.\, x_3 + x_1 \,.\, x_3$$

$$y = (\overline{x_1} + x_1) \,.\, x_3$$

$$y = x_3$$

Principle used: $x + \overline{x} = 1$

$$f = \bar{x}.\bar{y} + \bar{x}.y + x.\bar{y}$$

Apply the Principle: $x + \bar{x} = 1$ to simplify

$$f = \bar{x}.(\bar{y} + y) + x.\bar{y}$$

$$f = \bar{x} + x.\bar{y}$$

How do we simplify further?

$$f = \bar{x}.\bar{y} + \bar{x}.y + x.\bar{y} = \bar{x}.\bar{y} + \bar{x}.\bar{y} + \bar{x}.y + x.\bar{y}$$

Principle used : $x + x = x$

$$f = \bar{x}.\bar{y} + \bar{x}.y + \bar{x}.\bar{y} + x.\bar{y}$$
$$= \bar{x}.(\bar{y} + y) + (\bar{x} + x).\bar{y} = \bar{x} + \bar{y}$$

Simplify $f = \overline{x_1}.x_2.\overline{x_3}.x_4 + \overline{x_1}.x_2.x_3.x_4 + x_1.x_2.\overline{x_3}.x_4 + x_1.x_2.x_3.x_4 + x_1.\overline{x_2}.\overline{x_3}.x_4 + x_1.\overline{x_2}.x_3.x_4$

Principle: $x + \overline{x} = 1$ and $x + x = x$

Need a systematic and simpler method for applying these two principles

Karnaugh Map (K map) is a popular technique for carrying out simplification

It represents the information in problem in such a way that the two principles become easy to apply

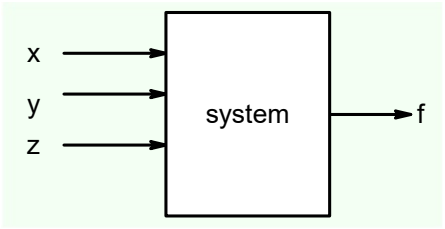
ESC201T : Introduction to Electronics

Lecture 34: Minimization (Kmap)

B. Mazhari
Dept. of EE, IIT Kanpur

Design Flow

System Description



Truth Table

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

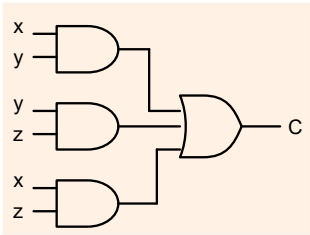
Boolean Expression

$$f = \bar{x}.\bar{y}.z + \bar{x}.y.z + x.\bar{y}.z + x.y.z$$

Minimized Boolean Expression

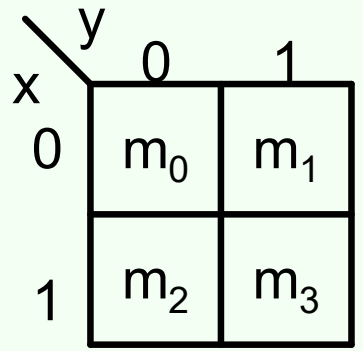
$$\Rightarrow f = \bar{x}.\bar{z} + x.z$$

Gate Netlist

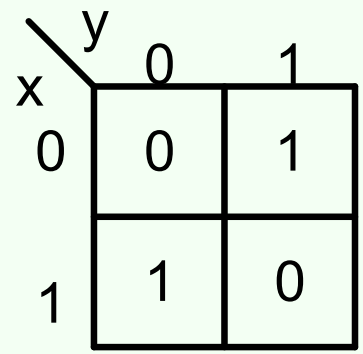


K-map representation of truth table

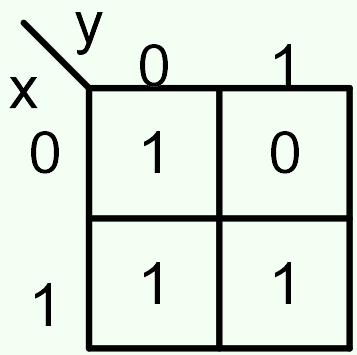
x	y	min term
0	0	$\bar{x}.\bar{y}$ m0
0	1	$\bar{x}.y$ m1
1	0	$x.\bar{y}$ m2
1	1	$x.y$ m3



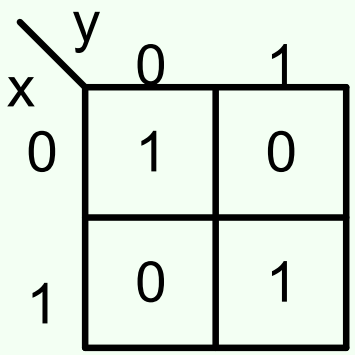
x	y	f_1
0	0	0
0	1	1
1	0	1
1	1	0



$$f_2 = \sum(0,2,3)$$



x \ y	0	1
0	1	0
1	1	1

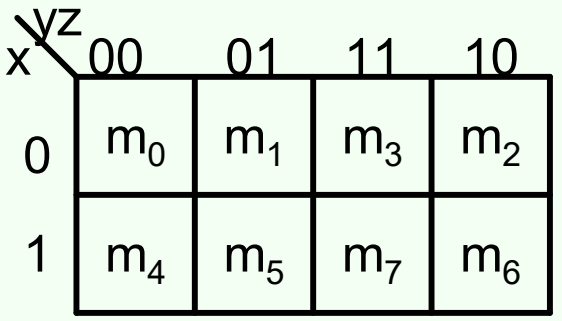


x \ y	0	1
0	1	0
1	0	1

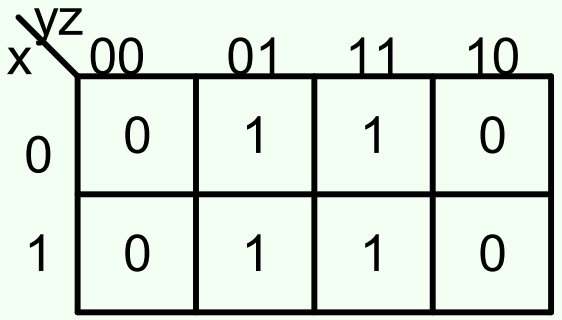
$$f = \overline{x}.\overline{y} + x.y$$

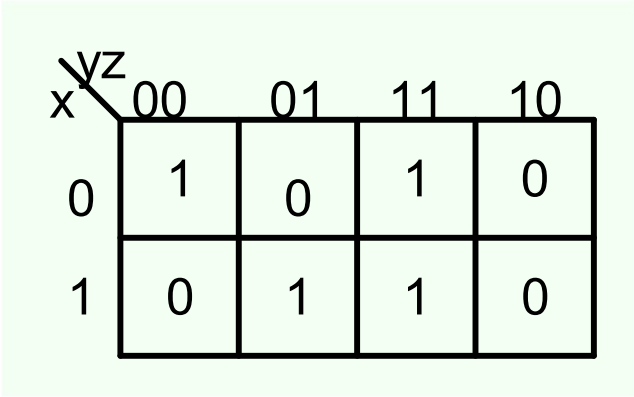
3-variable K-map representation

x	y	z	min terms	
0	0	0	$\bar{x} . \bar{y} . \bar{z}$	m0
0	0	1	$\bar{x} . \bar{y} . z$	m1
0	1	0	$\bar{x} . y . \bar{z}$	m2
0	1	1	$\bar{x} . y . z$	m3
1	0	0	$x . \bar{y} . \bar{z}$	m4
1	0	1	$x . \bar{y} . z$	m5
1	1	0	$x . y . \bar{z}$	m6
1	1	1	$x . y . z$	m7



x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

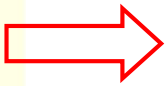




$$f = \overline{x}.\overline{y}.\overline{z} + \overline{x}.y.z + x.\overline{y}.z + x.y.z$$

4-variable K-map representation

w	x	y	z	min terms
0	0	0	0	m_0
0	0	0	1	m_1
0	0	1	0	m_2
0	0	1	1	m_3
⋮	⋮	⋮	⋮	⋮
1	1	1	0	m_{14}
1	1	1	1	m_{15}



wx \ yz	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

wx \ yz	00	01	11	10
00	1	0	1	0
01	0	1	1	0
11	1	0	0	1
10	1	0	0	0

$$f = \overline{w}.\overline{x}.\overline{y}.\overline{z} + \overline{w}.\overline{x}.y.z + \overline{w}.x.\overline{y}.z + \overline{w}.x.y.z + w.x.\overline{y}.\overline{z} + w.x.y.\overline{z} + w.\overline{x}.\overline{y}.\overline{z}$$

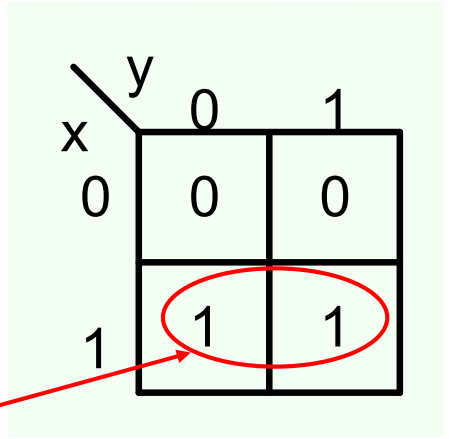
Minimization using Kmap

$$f_2 = \sum(2,3)$$

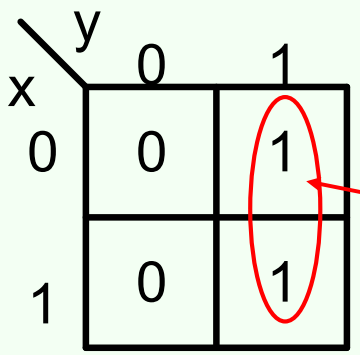
$$f = x.\overline{y} + x.y$$

$$f = x.(\overline{y} + y)$$

$$f = x$$



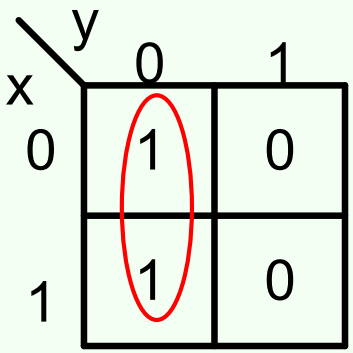
Combine terms which differ in only one bit position. As a result, whatever is common remains.



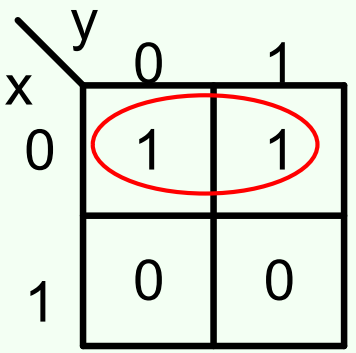
$$f = \bar{x}.y + x.y$$

$$f = (\bar{x}. + x).y$$

$$\Rightarrow f = y$$



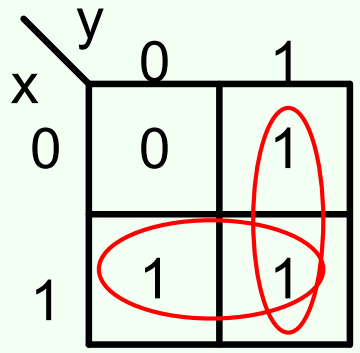
$$\Rightarrow f = \bar{y}$$



$$\Rightarrow f = \bar{x}$$

Principle: $x + \bar{x} = 1$ and $x + x = x$

$$f_2 = \sum(0,2,3)$$



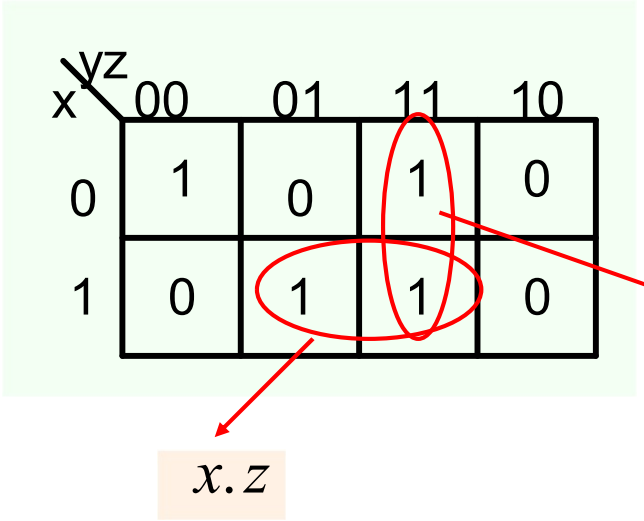
$$f = x.\bar{y} + x.y + \bar{x}.y$$

$$f = x.(\bar{y} + y) + \bar{x}.y$$
$$= x + \bar{x}.y$$

$$f = x + \bar{x}.y + x.y$$
$$= x + (\bar{x} + x).y$$
$$= x + y$$

The idea is to cover all the 1's with as few and as simple terms as possible

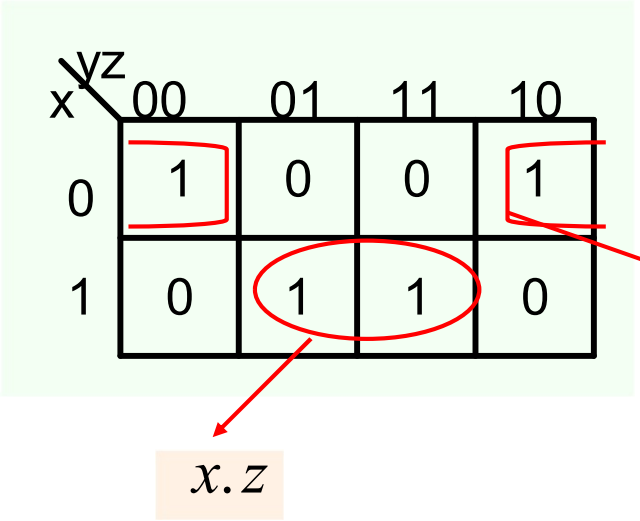
3-variable minimization



$$f = \bar{x}.\bar{y}.\bar{z} + x.\bar{y}.\bar{z} + x.y.\bar{z} + x.y.z$$

$$f = \bar{x}.\bar{y}.\bar{z} + y.z + x.z$$

3-variable minimization



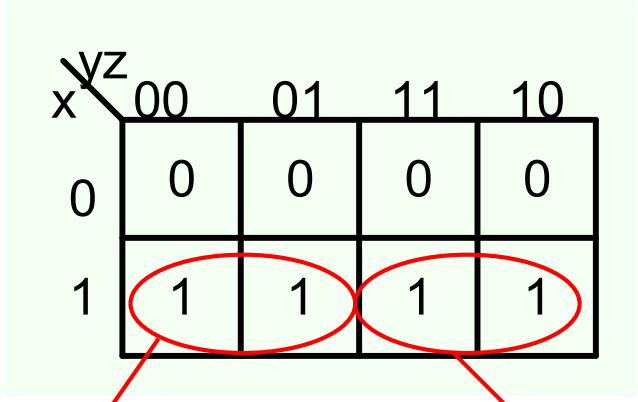
$$f = \bar{x}.\bar{y}.\bar{z} + \bar{x}.y.\bar{z} + x.\bar{y}.z + x.y.z$$

$\bar{x}.\bar{z}$

$x.z$

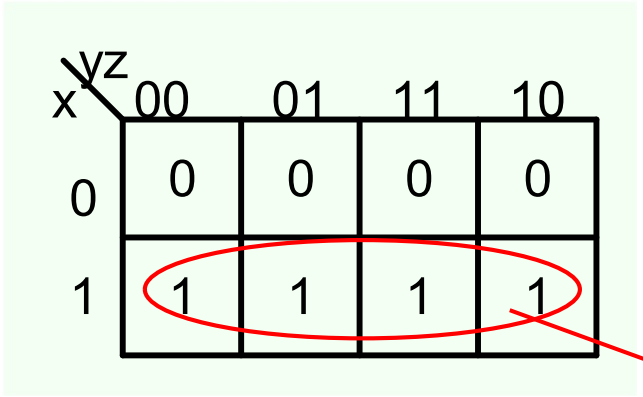
$$f = \bar{x}.\bar{z} + x.z$$

3-variable minimization

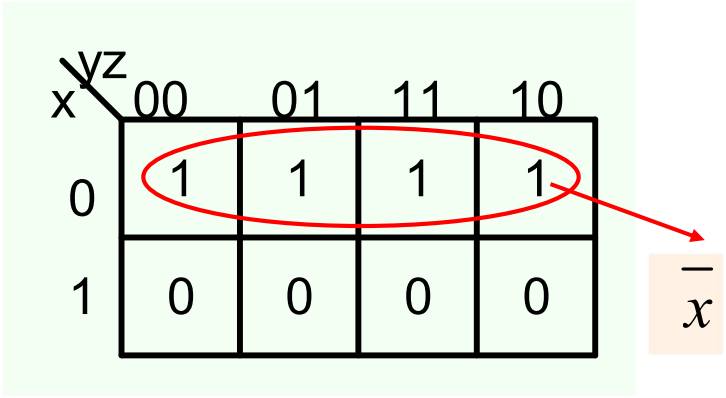


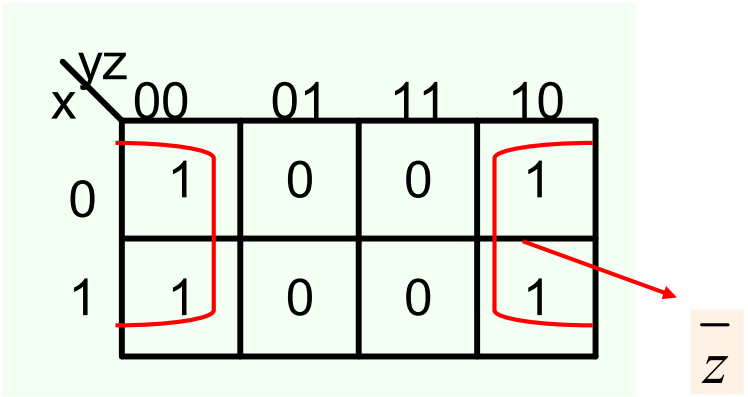
$$f = \overline{x}.\overline{y}.\overline{z} + x.\overline{y}.\overline{z} + x.y.\overline{z} + x.y.z$$

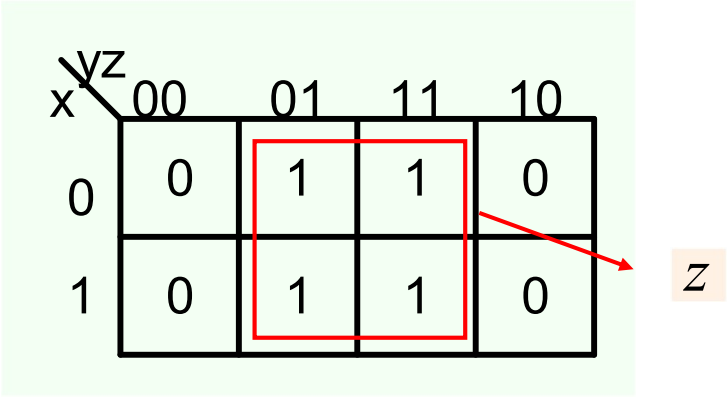
$$f = x.\overline{y} + x.y$$

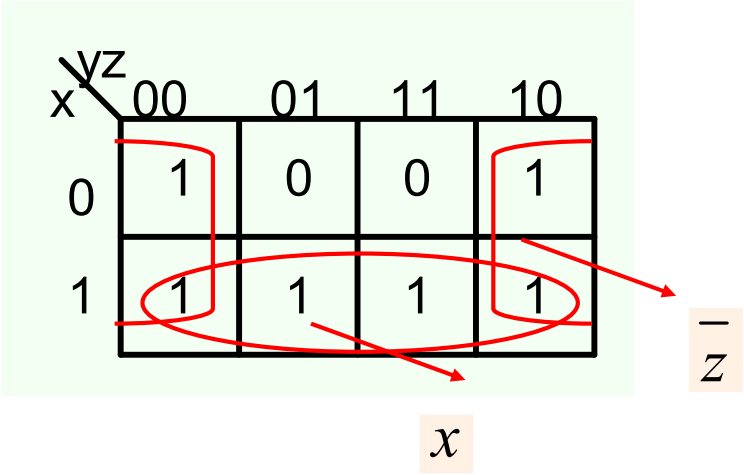


$$f = x.(\overline{y} + y) = x$$


yz
x
00
01
11
10
0
1
1
1
1
1
0
0
0
0
$\bar{x}$

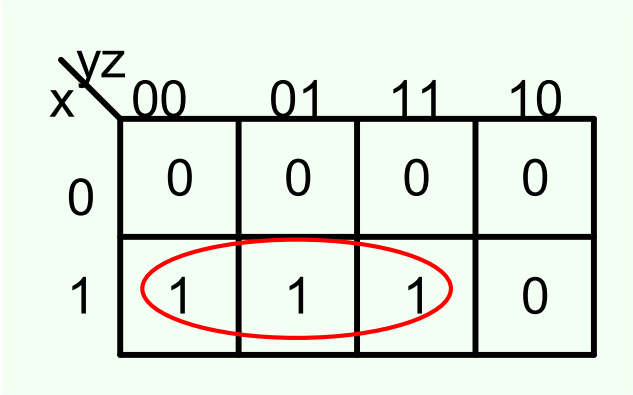

yz
x
00
01
11
10
0
1
1
0
0
1
1
0
0
1
$\bar{z}$


yz
x
00
01
11
10
0
1
0
1
1
0
0
1
1
0
z


yz
x
00
01
11
10
0
1
1
0
0
1
1
1
1
1
1
$\bar{z}$
x

$$f = x + \bar{z}$$

Can we do this ?

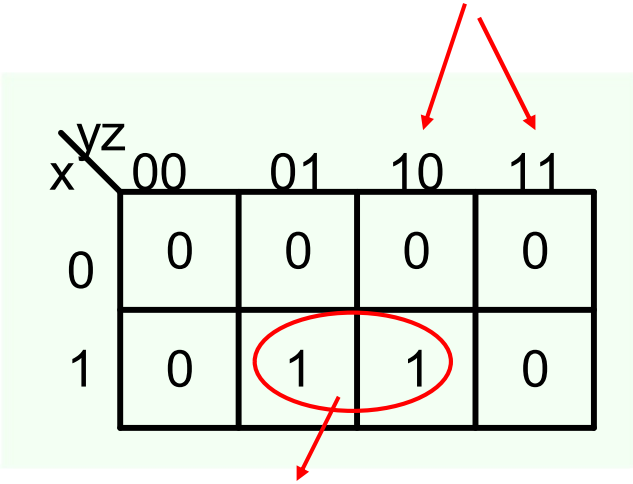


Note that each encirclement should represent a single product term. In this case it does not.

$$f = x.\bar{y}.\bar{z} + x.\bar{y}.z + x.y.z$$
$$= x.\bar{y} + x.z$$

We do not get a single product term. In general we cannot make groups of 3 terms.

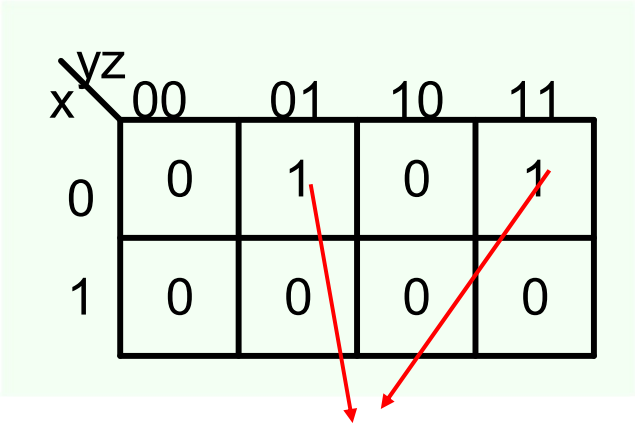
Can we use kmap with the following ordering of variables?



Can we combine these two terms into a single term ?

$$f = x.\bar{y}.z + x.y.\bar{z}$$
$$= x.(\bar{y}.z + y.\bar{z})$$

Note that no simplification is possible. Kmap requires information to be represented

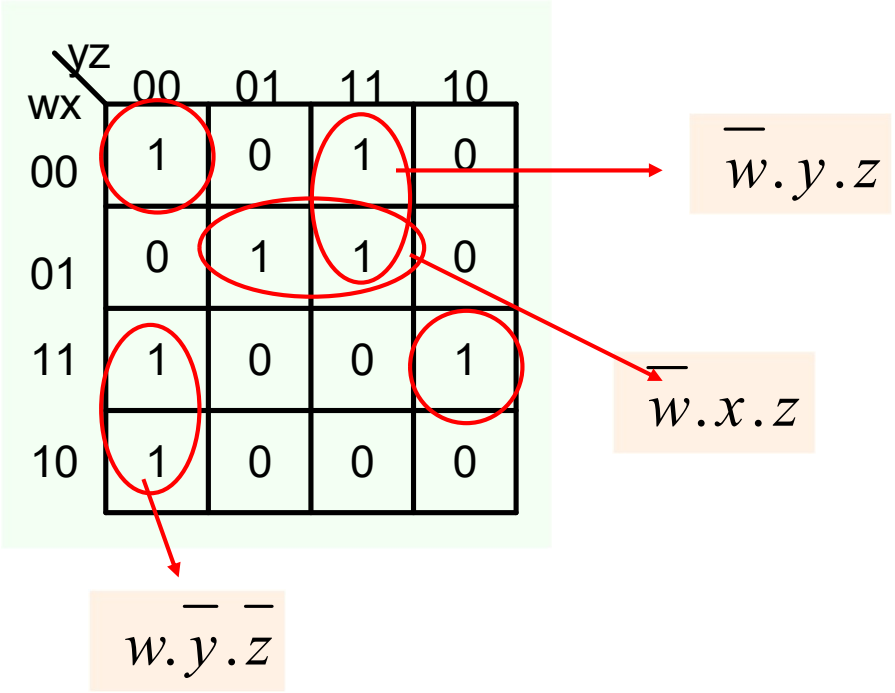


These two terms can be combined into a single term but it is not easy to show that on the diagram.

$$f = \bar{x}.\bar{y}.z + \bar{x}.y.z$$
$$= \bar{x}.(\bar{y} + y).z = \bar{x}.z$$

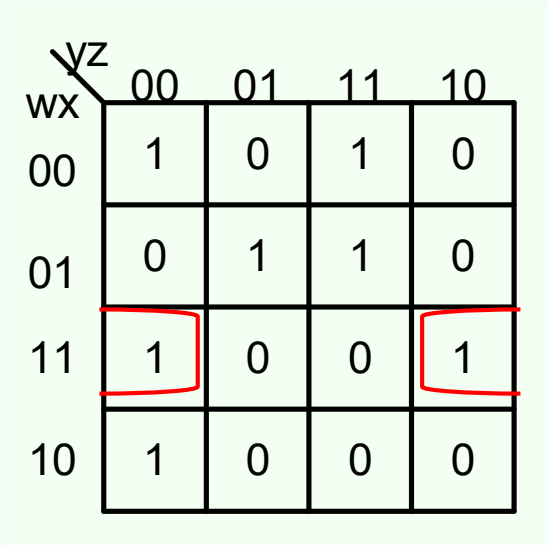
Kmap requires information to be represented in such a way that it is easy to apply the principle $x + \bar{x} = 1$

4-variable minimization

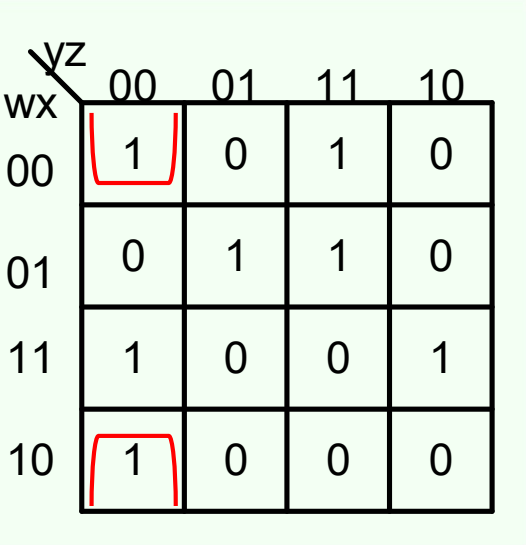


$$f = \overline{w}.y.z + \overline{w}.x.z + w.\overline{y}.\overline{z} + \overline{w}.\overline{x}.\overline{y}.z + w.x.y.\overline{z}$$

But is this the simplest expression ?

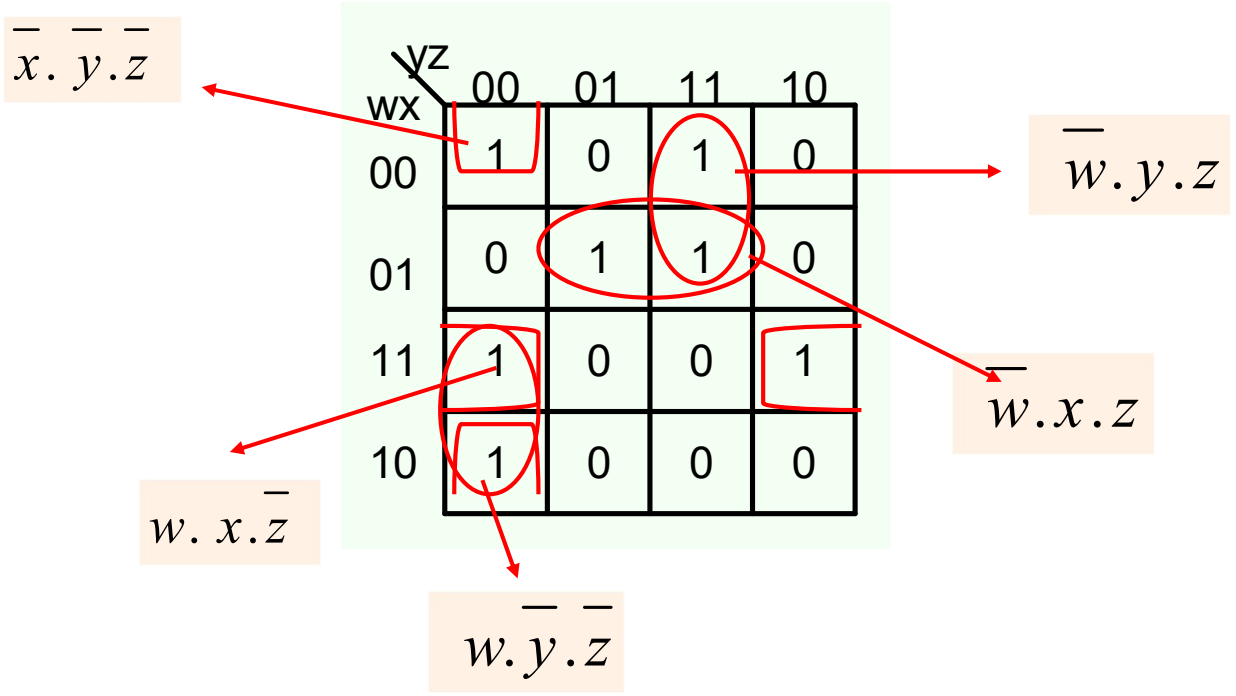


$$w.\ x.\overline{y}.\overline{z} + w.\ x.y.\overline{z} = w.\ x.\overline{z}$$



$$w.\ \overline{x}.\overline{y}.\overline{z} + w.\ x.y.z = \overline{x}.\ \overline{y}.\overline{z}$$

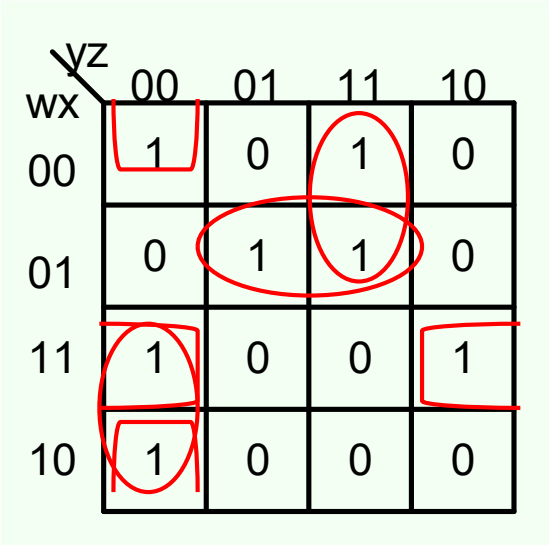
4-variable minimization



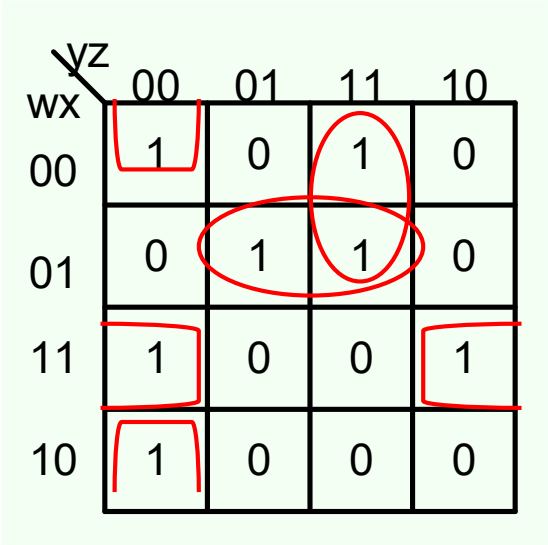
$$f = \overline{w}.y.z + \overline{w}.x.z + w.\overline{y}.\overline{z} + w.x.\overline{z} + \overline{x}.\overline{y}.\overline{z}$$

Is this the best that we can do ?

Cover the 1's with minimum number of terms

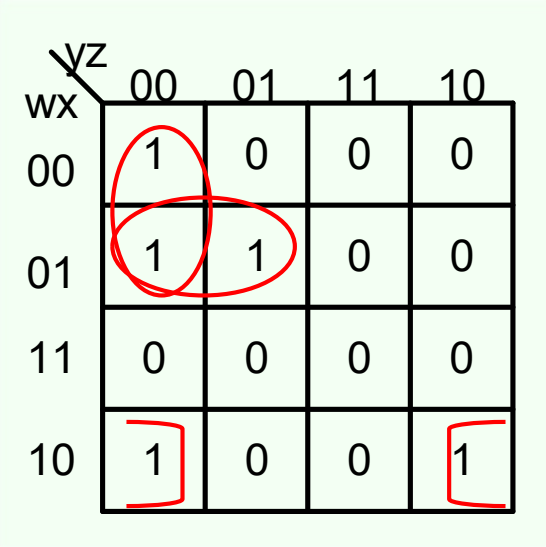


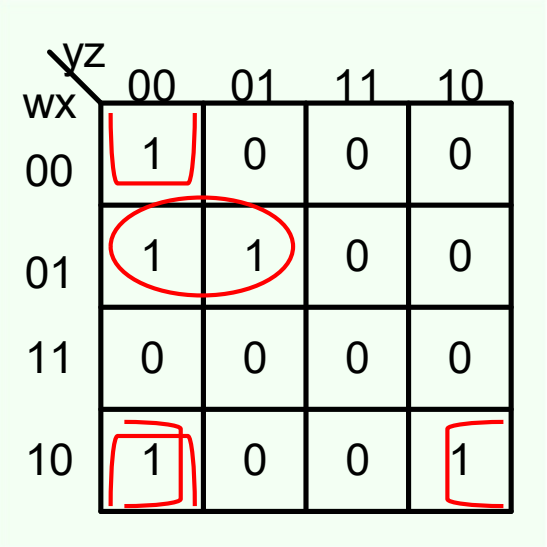
$$f = \overline{w}.y.z + \overline{w}.x.z + w.\overline{y}.\overline{z} + w.x.\overline{z} + \overline{x}.y.z$$



$$f = \overline{w}.y.z + \overline{w}.x.z + w.x.\overline{z} + \overline{x}.y.z$$

4-variable minimization

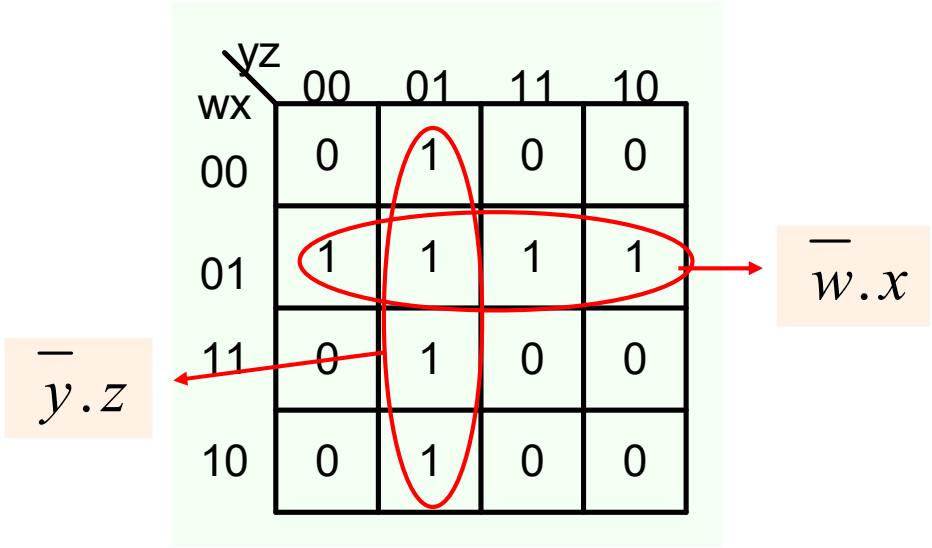


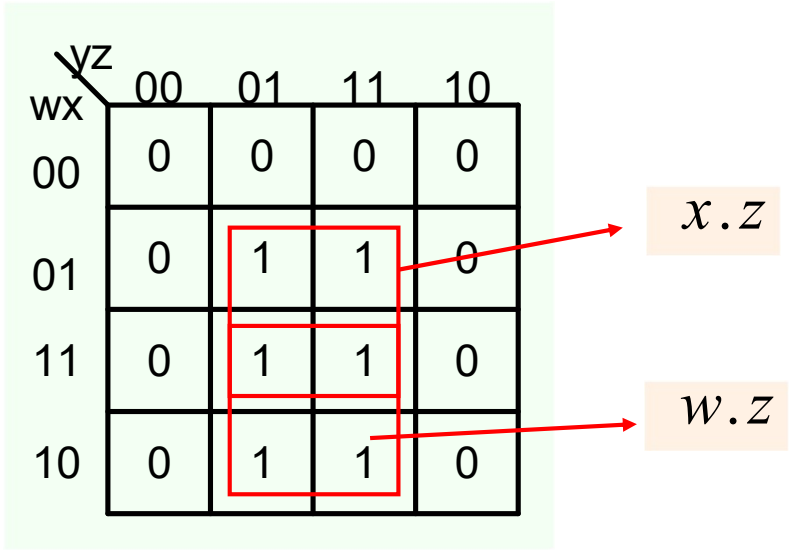


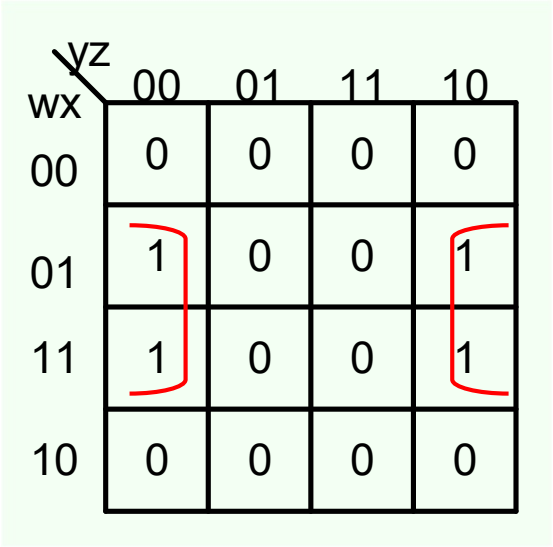
$$f = \overline{w}.x.\overline{y} + w.\overline{x}.\overline{z} + \overline{w}.\overline{y}.\overline{z}$$

$$f = \overline{w}.x.\overline{y} + w.\overline{x}.\overline{z} + \overline{x}.\overline{y}.\overline{z}$$

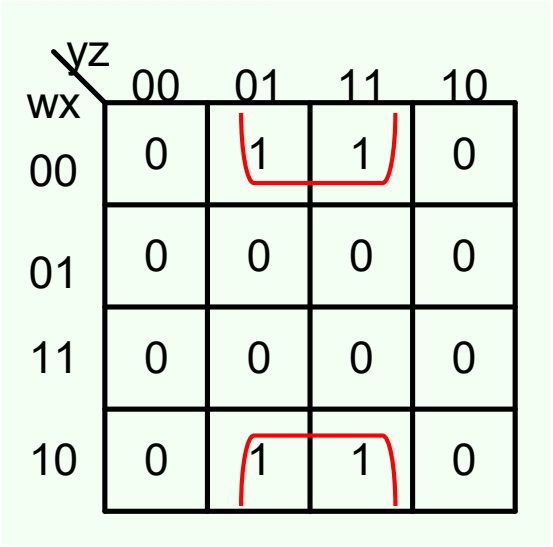
Groups of 4


yz
wx
00
01
11
10
0 1 0 0
1 1 1 1
0 1 0 0
0 1 0 0
$\overline{w}.x$
$\overline{y}.z$

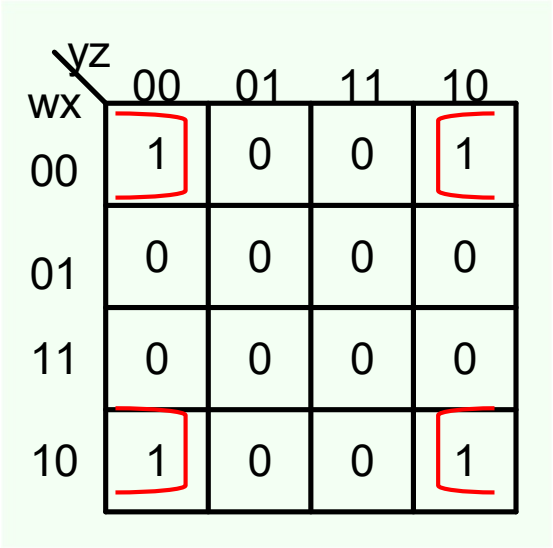

yz
wx
00
01
11
10
0 0 0 0
0 1 1 0
0 1 1 0
0 1 1 0
$x.z$
$w.z$



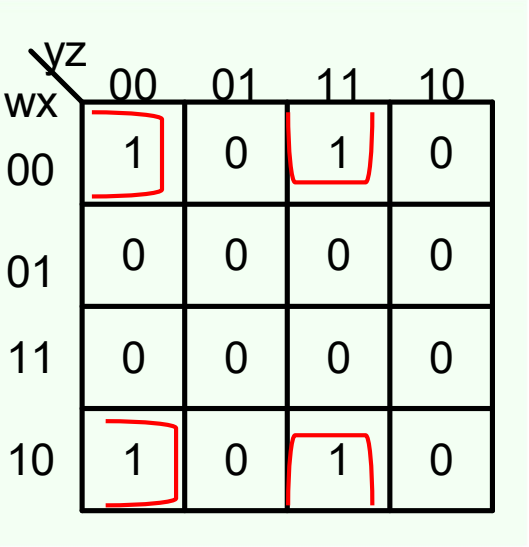
$x . \overline{z}$



$\overline{x} . z$

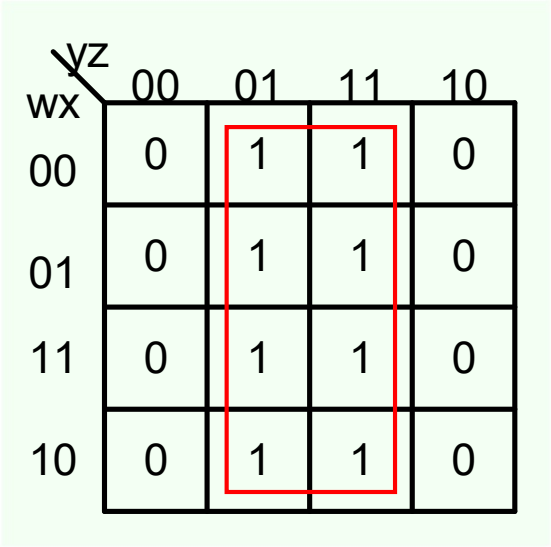


$\overline{x} . \overline{z}$

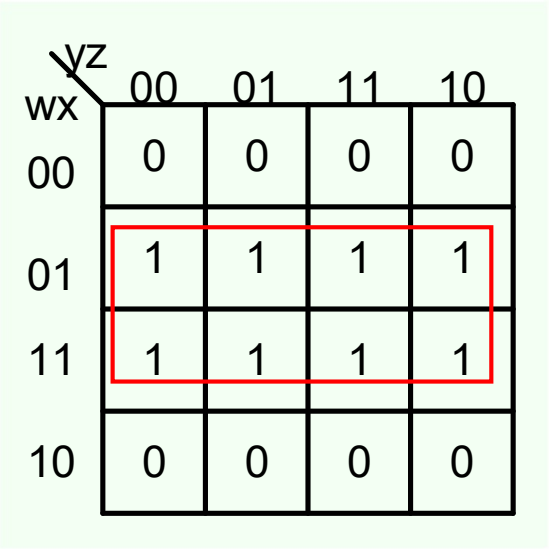


??

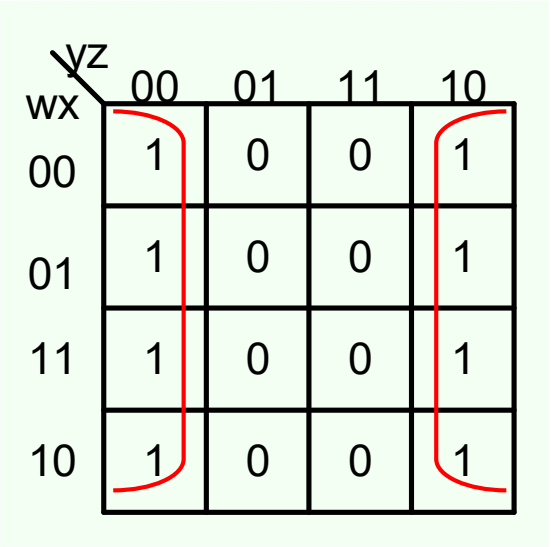
Groups of 8



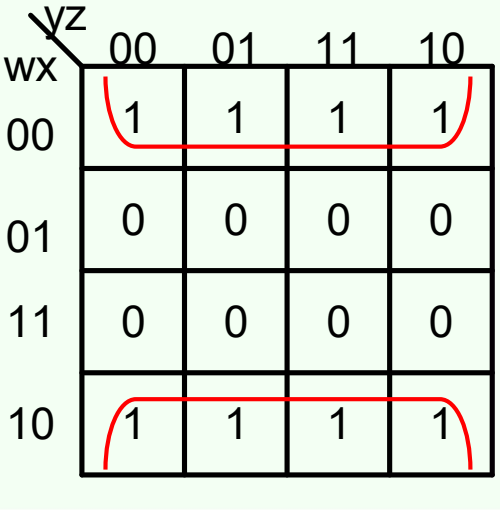
z



x

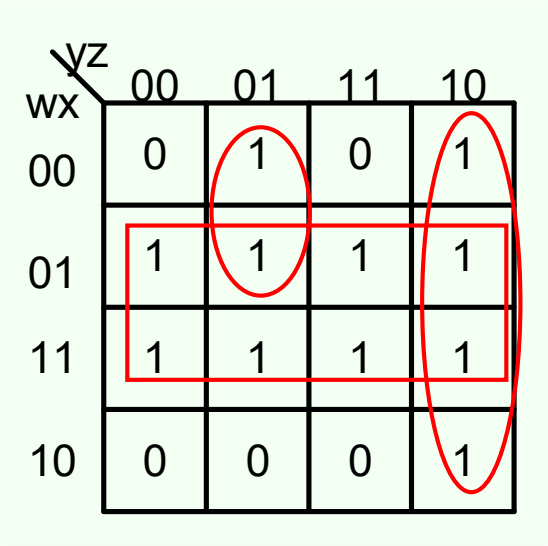


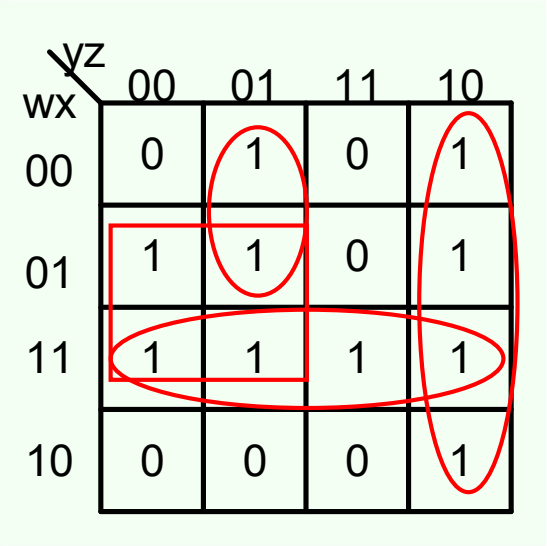
$\overline{z}$



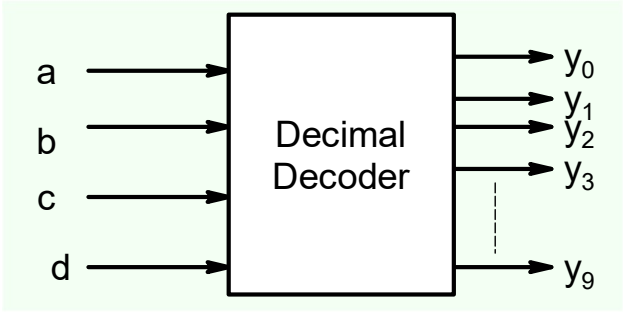
$\overline{x}$

Examples


yz
wx
00
01
11
10
0 1 0 1
1 1 1 1
1 1 1 1
0 0 0 1


yz
wx
00
01
11
10
0 1 0 1
1 1 0 1
1 1 1 1
0 0 0 1

Don’t care terms



Y_3

ab \ cd	00	01	11	10
00	0	0	1	0
01	0	0	0	0
11	x	x	x	x
10	0	0	x	x

$$y_3 = \overline{a}.\overline{b}.c.d$$

a	b	c	d	$y_0y_1y_2y_3y_4y_5y_6y_7y_8y_9$
0	0	0	0	1 0 0 0 0 0 0 0 0 0 0
0	0	0	1	0 1 0 0 0 0 0 0 0 0 0
0	0	1	0	0 0 1 0 0 0 0 0 0 0
0	0	1	1	0 0 0 1 0 0 0 0 0 0
0	1	0	0	0 0 0 0 1 0 0 0 0 0
0	1	0	1	0 0 0 0 0 1 0 0 0 0
0	1	1	0	0 0 0 0 0 0 1 0 0 0
0	1	1	1	0 0 0 0 0 0 0 1 0 0
1	0	0	0	0 0 0 0 0 0 0 0 1 0
1	0	0	1	0 0 0 0 0 0 0 0 0 1
1	0	1	0	x x x x x x x x x x
1	0	1	1	x x x x x x x x x x
1	1	0	0	x x x x x x x x x x
1	1	0	1	x x x x x x x x x x
1	1	1	0	x x x x x x x x x x
1	1	1	1	x x x x x x x x x x

Don’t care terms can be chosen as 0 or 1. Depending on the problem, we can choose the don’t care term as 1 and use it to obtain a simpler Boolean expression

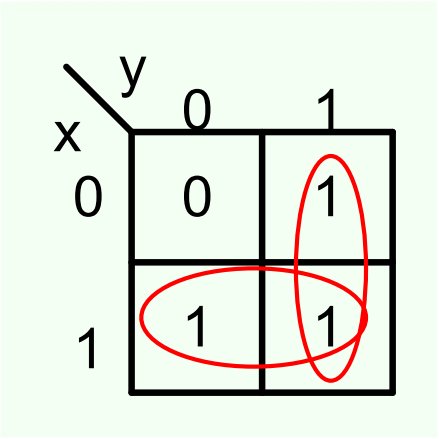
Y_3

ab \ cd	00	01	11	10
00	0	0	1	0
01	0	0	0	0
11	x	x	x	x
10	0	0	x	x

$$y_3 = \bar{b}.c.d$$

Don’t care terms should only be included in encirclements if it helps in obtaining a larger grouping or smaller number of groups.

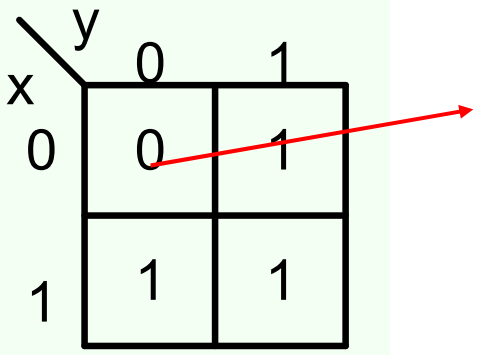
Minimization of Product of Sum Terms using Kmap



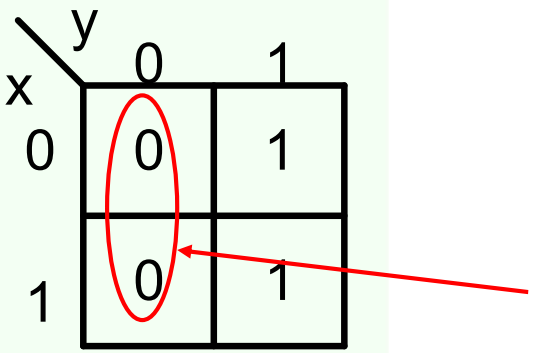
$$f = x + \bar{x}.y + x.y$$

$$= x + (\bar{x} + x).y$$

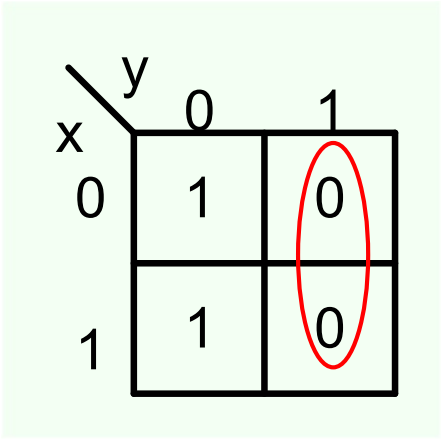
$$= x + y$$



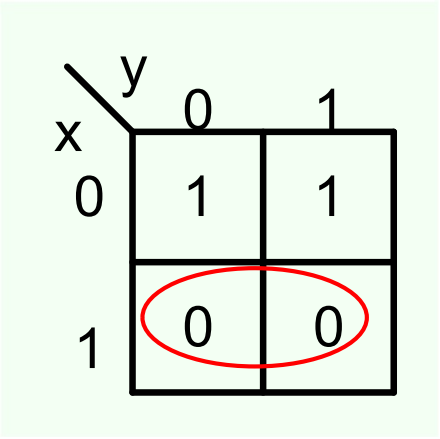
$f = x + y$



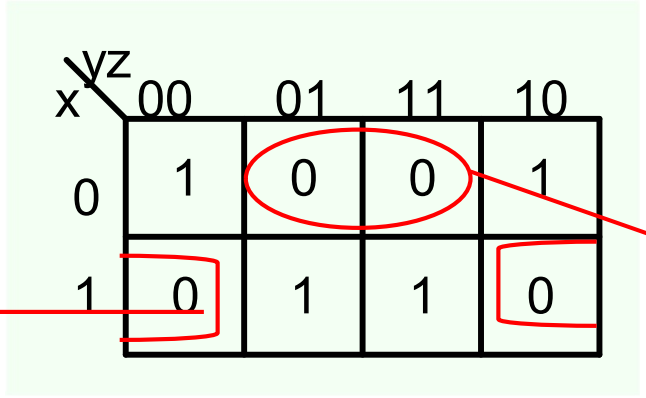
$f = y$



$\Rightarrow f = \overline{y}$



$\Rightarrow f = \overline{x}$

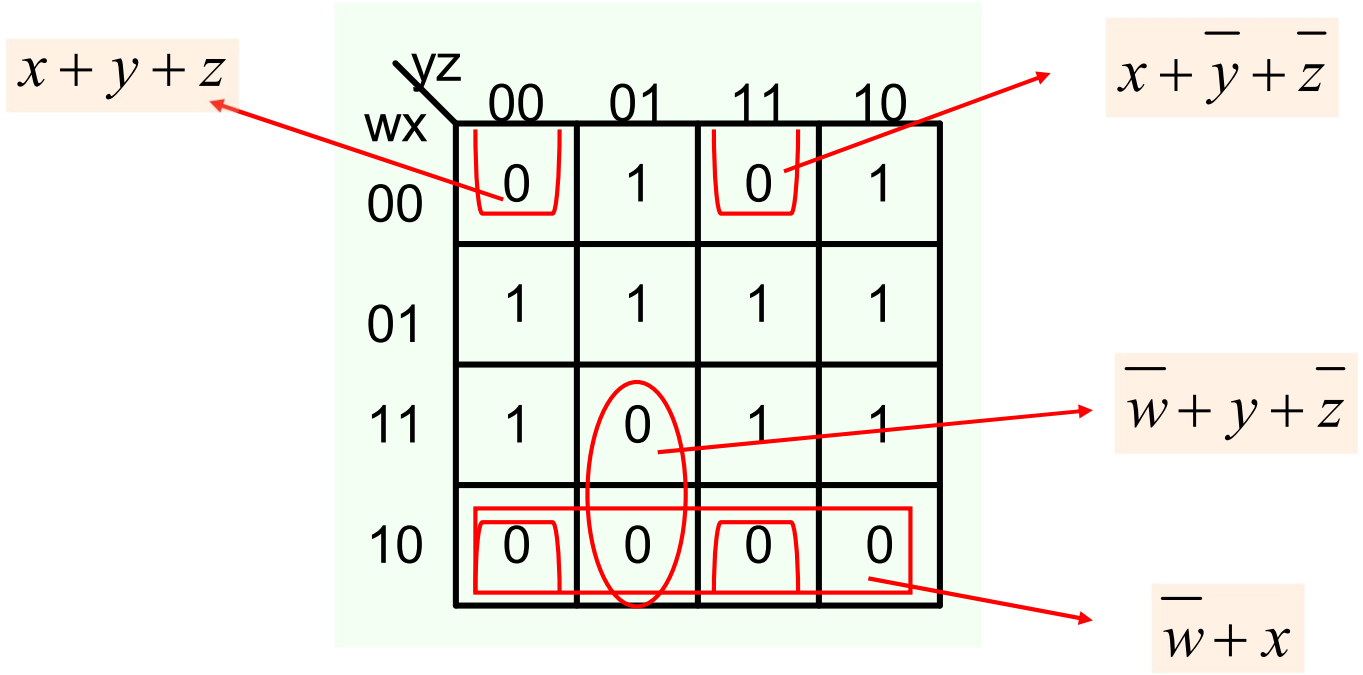


$\overline{x}.+z$

$x+\overline{z}$

$f = (\overline{x}\ .+z).(x+\overline{z})$

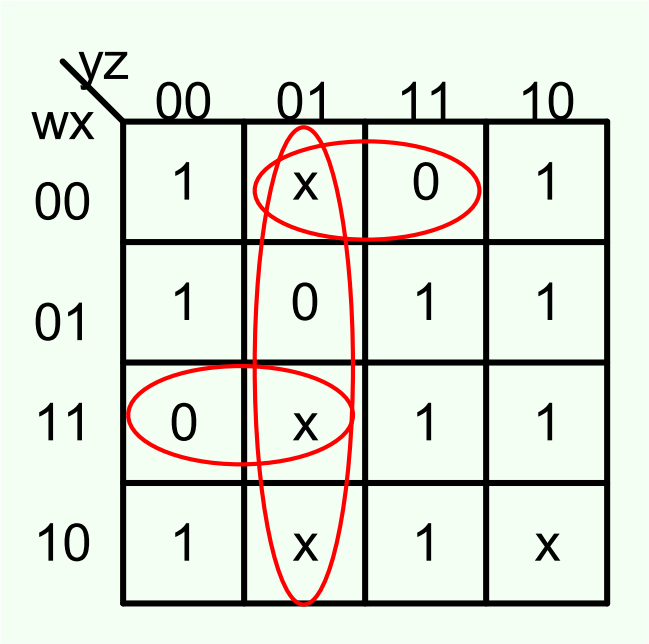
$\Rightarrow f = \overline{x}\ .\overline{z} + x.z$



$$f = (x + y + z).(x + \overline{y} + \overline{z}).(\overline{w} + y + \overline{z}).(\overline{w} + x)$$

Example

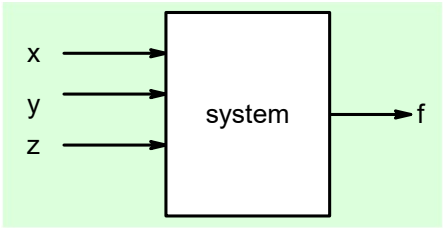
Obtain the minimized PoS by suitably using don't care terms



$$f = (x + w + \bar{z}).(\bar{x} + \bar{w} + y).(y + \bar{z})$$

Design Flow

System Description



Truth Table

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

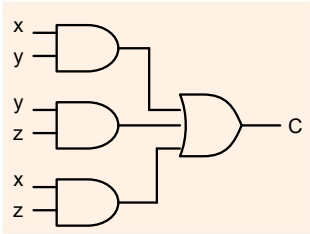
Boolean Expression

$$f = \bar{x}.\bar{y}.z + \bar{x}.y.z + x.\bar{y}.z + x.y.z$$

Minimized Boolean Expression

$$\Rightarrow f = \bar{x}.\bar{z} + x.z$$

Gate Netlist



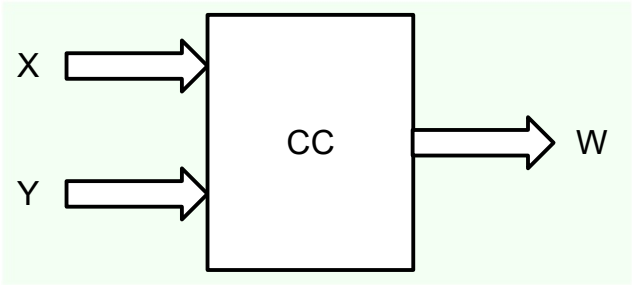
ESC201T : Introduction to Electronics

Lecture 35: Combination circuit design-2

B. Mazhari
Dept. of EE, IIT Kanpur

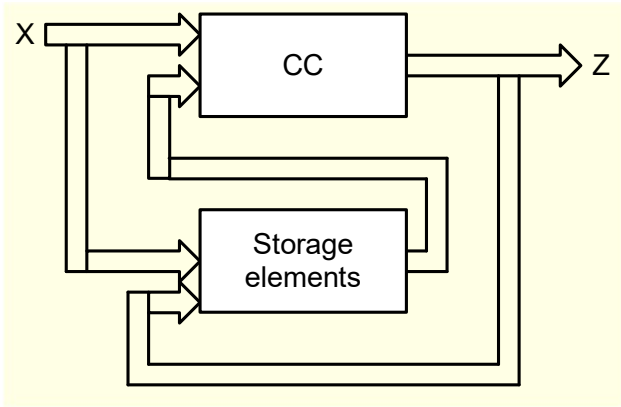
Digital Circuits

Combinational Circuits



Output is determined by current values of inputs only.

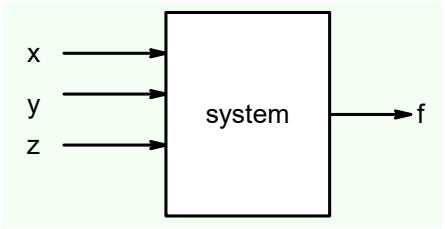
Sequential Circuits



Output is determined in general by current values of inputs and past values of inputs/outputs as well.

Design Flow

System Description



Truth Table

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

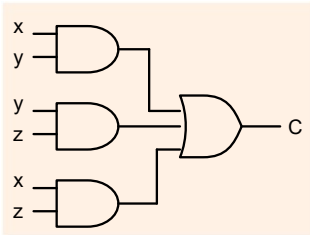
Boolean Expression

$$f = \bar{x}.\bar{y}.z + \bar{x}.y.z + x.\bar{y}.z + x.y.z$$

Minimized Boolean Expression

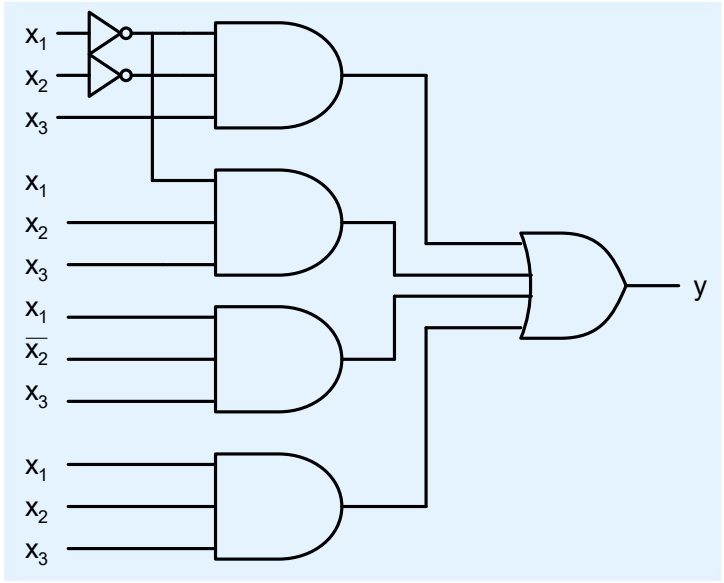
$$\Rightarrow f = \bar{x}.\bar{z} + x.z$$

Gate Netlist



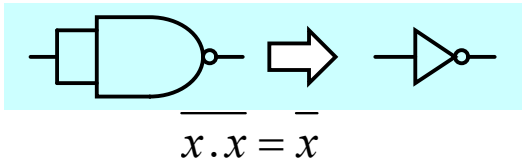
Mapping of Boolean expression to a Network of gates available in the library

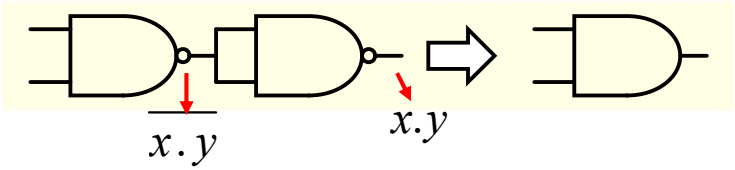
$$y = \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot x_2 \cdot x_3 + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3$$

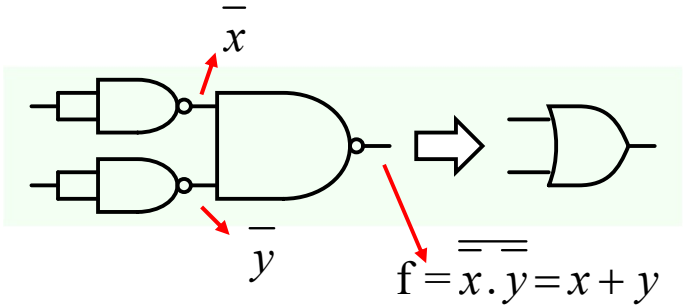


Library of available Gates	Cost
Inverter	1
Two input NAND	2
Three input NAND	3
AND-OR-Invert $Y = \overline{AB + C}$	3

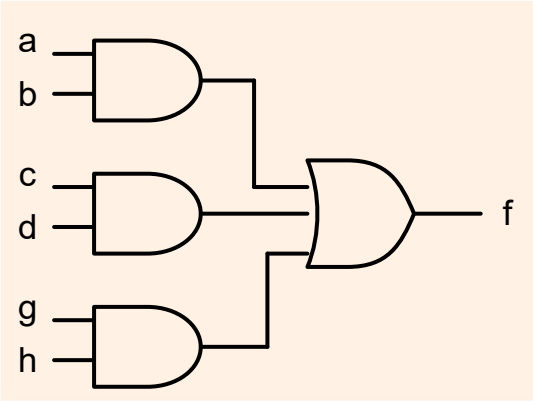
Implementation using only NAND gates

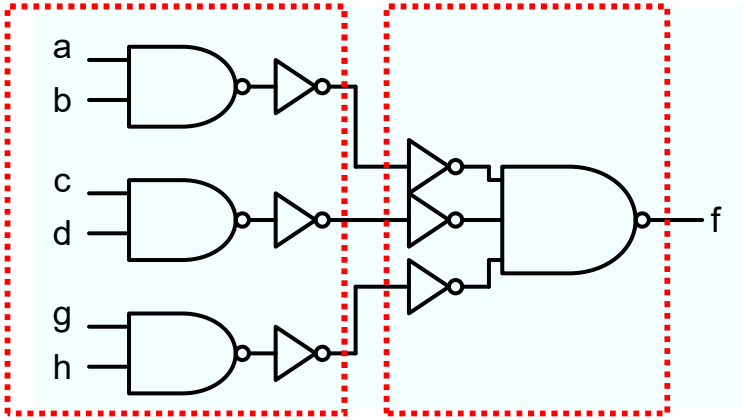


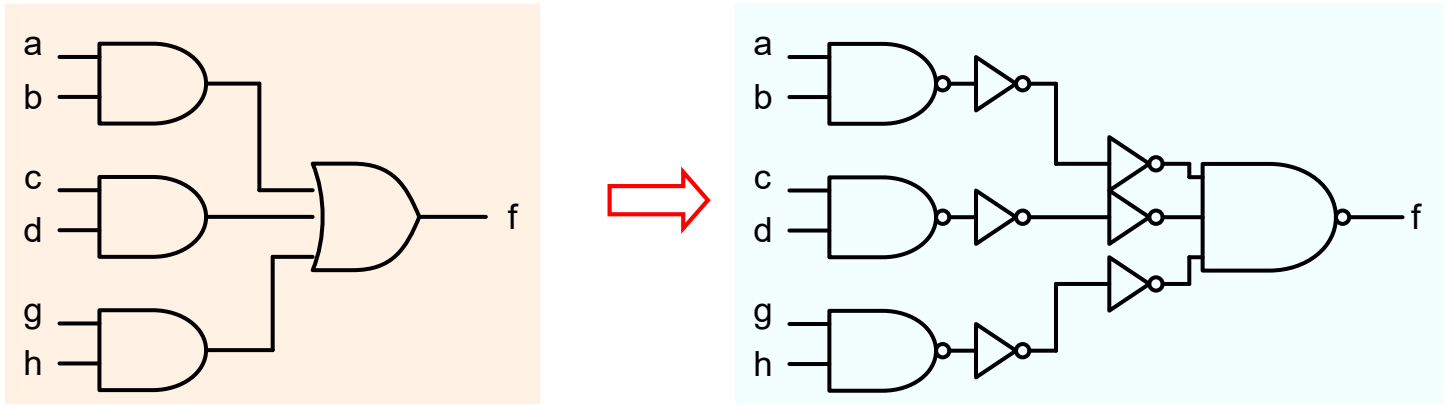


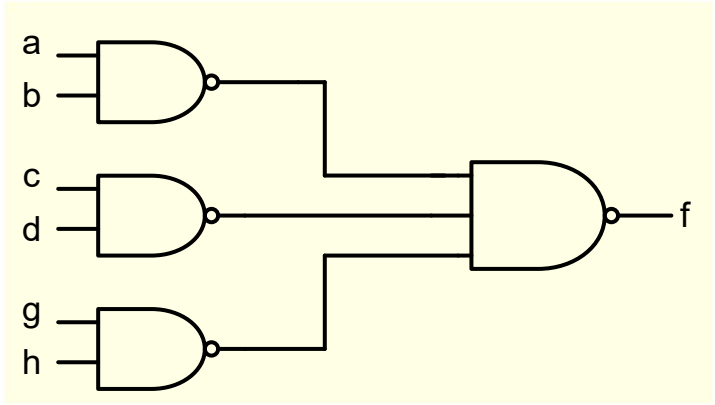


A SoP expression is easily implemented with NAND gates. $f = a.b + c.d + g.h$





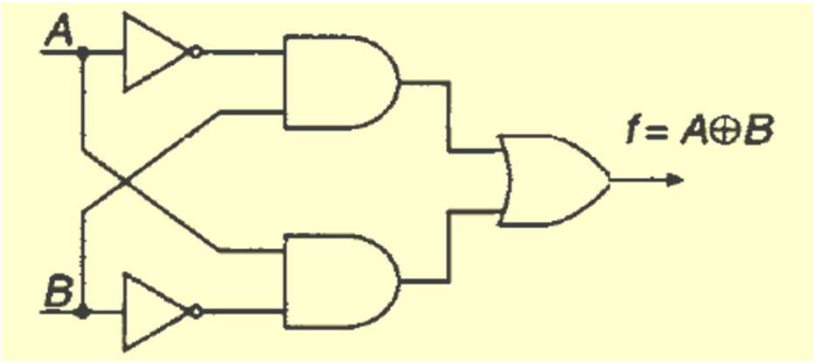


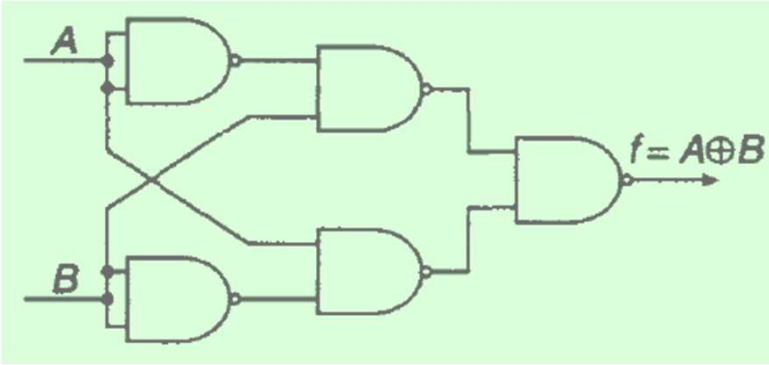


There is a one-to-one mapping between AND-OR network and NAND network

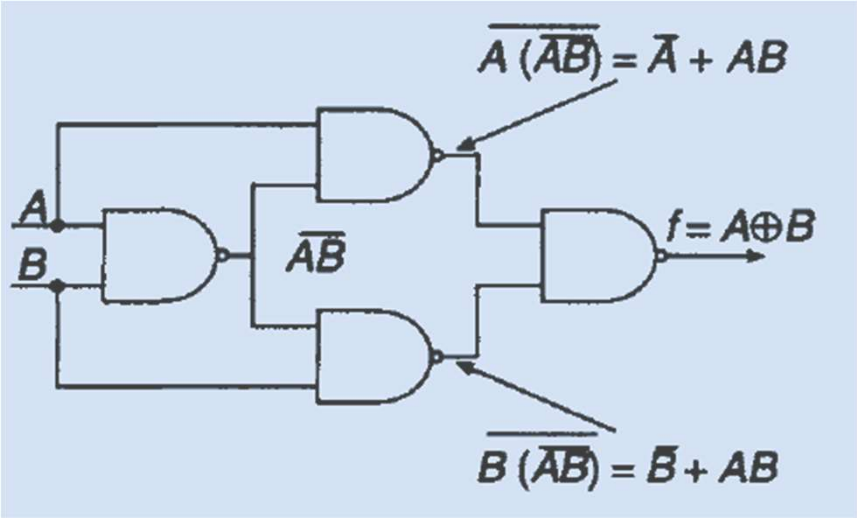
Often there is lot of further optimization that can be done

Consider implementation of XOR gate $f = \overline{A}.B + A.\overline{B}$

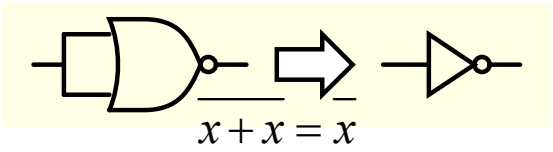


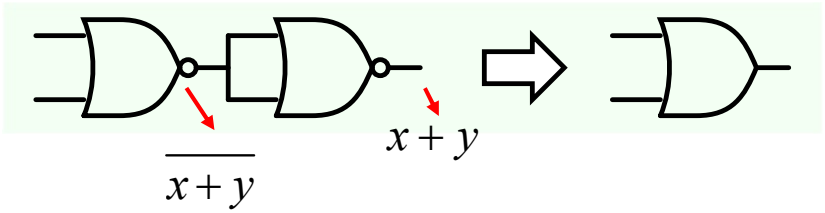


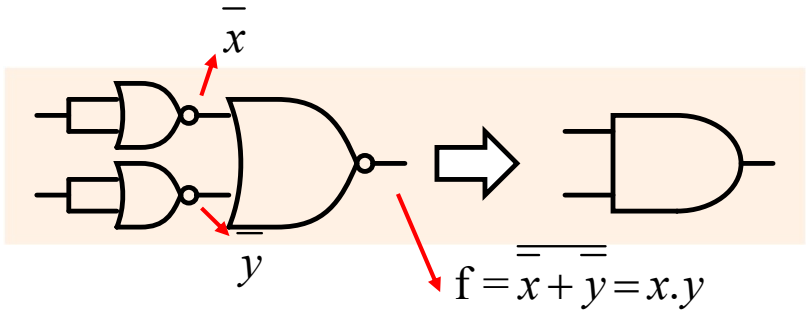
$$f = \overline{A}.B + B.\overline{B} + A.\overline{B} + A.\overline{A}$$
$$= B(\overline{A} + \overline{B}) + A(\overline{A} + \overline{B})$$



Implementation using only NOR gates



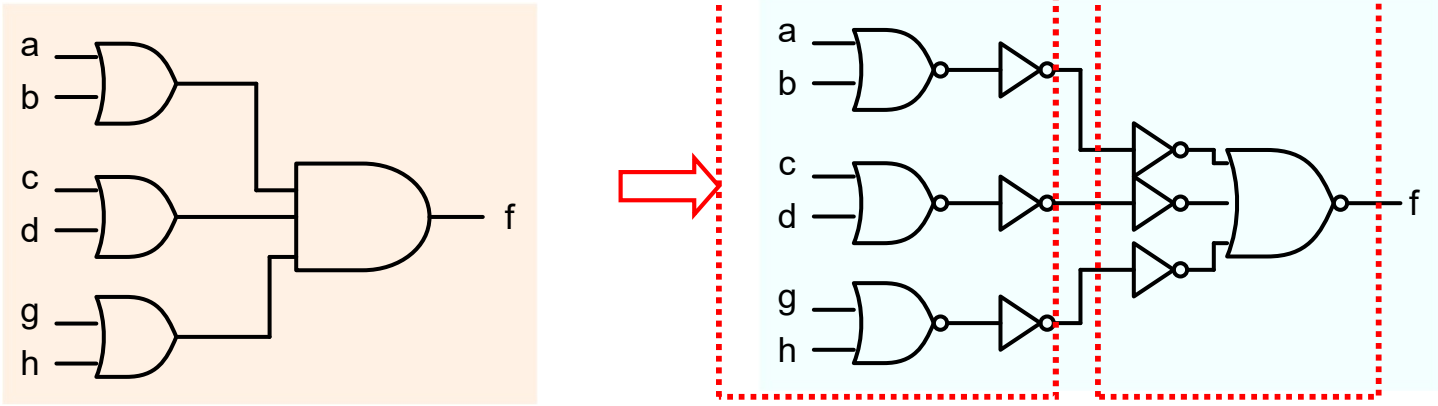


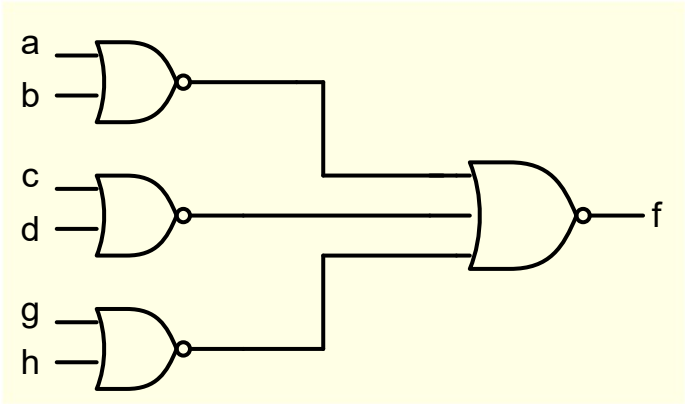


To implement using NOR gates, it is easiest to start with minimized Boolean expression in POS form

$$f = (a+b).(c+d).(g+h)$$

$$f = (a + b).(c + d).(g + h)$$





There is a one-to-one mapping between OR-AND network and NOR network

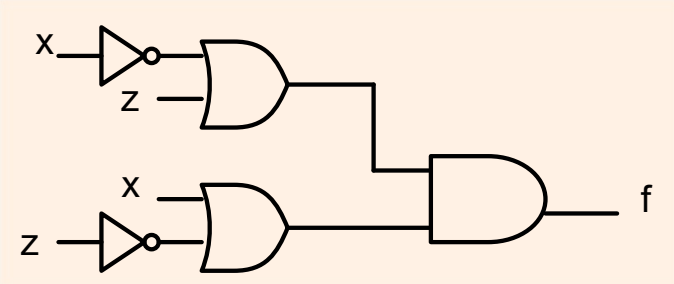
To implement SoP expression using NOR gates, determine first the corresponding PoS expression and then follow the procedure outlined earlier

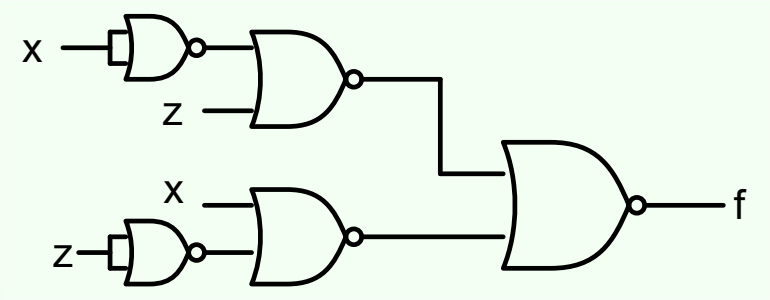
Implement $f(x,y,z) = \bar{x}\,.\,\bar{z} + x\,.\,z$ using NOR gates

$\bar{x}.\bar{z}.(y+\bar{y}) + x.z.(y+\bar{y})$

x \ yz	00	01	11	10
0	1	0	0	1
1	0	1	1	0

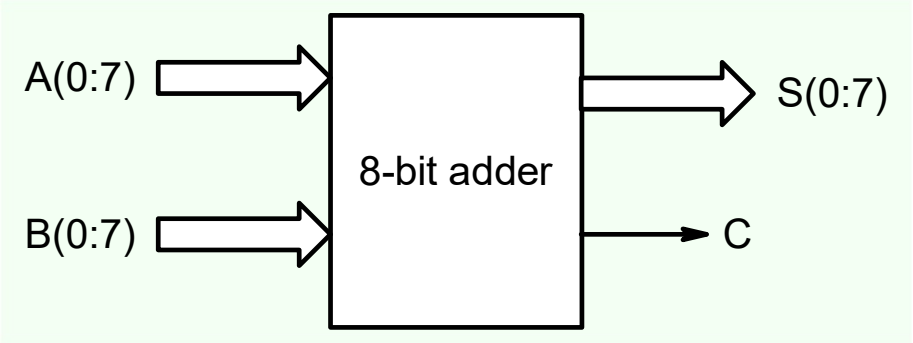
$\Rightarrow f = (\bar{x}\,.+z).(x+\bar{z})$





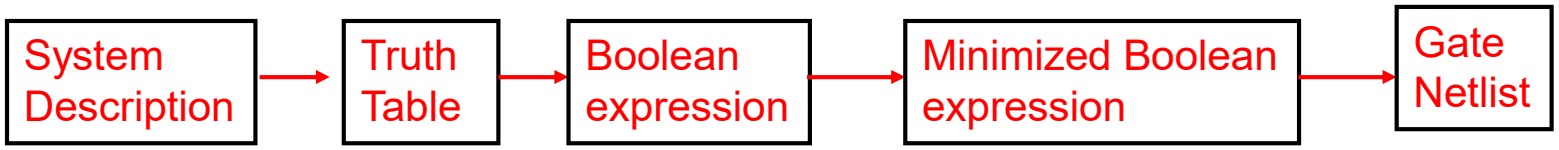
Similarly PoS expression can be implemented as NAND network by first converting it to SoP expression and then following the procedure outlined earlier

Design of Complex Combinational circuits



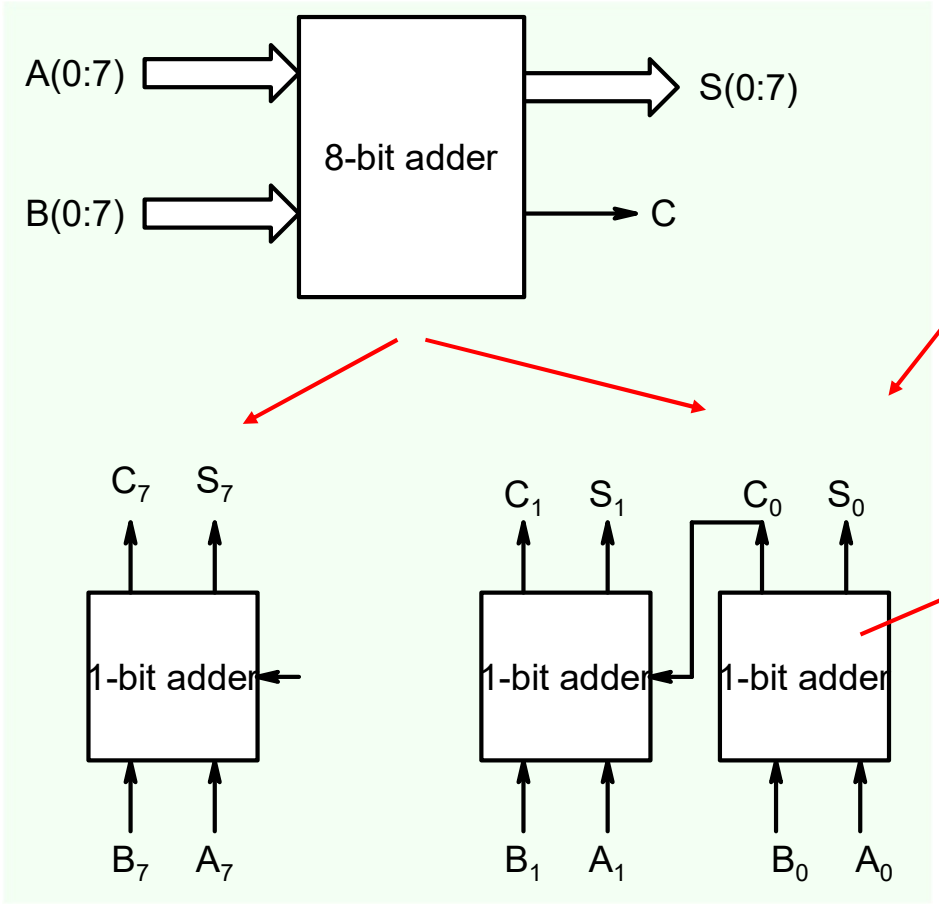
$A_7A_6...A_0$	$B_7B_6...B_0$	$S_7S_6...S_0$	C
00...0	00...0	00...0	0
00...1	00...0	00...1	0
00...0	00...1	00...1	0
⋮	⋮	⋮	⋮

Truth table has 2^{16} entries



This design approach becomes difficult to use

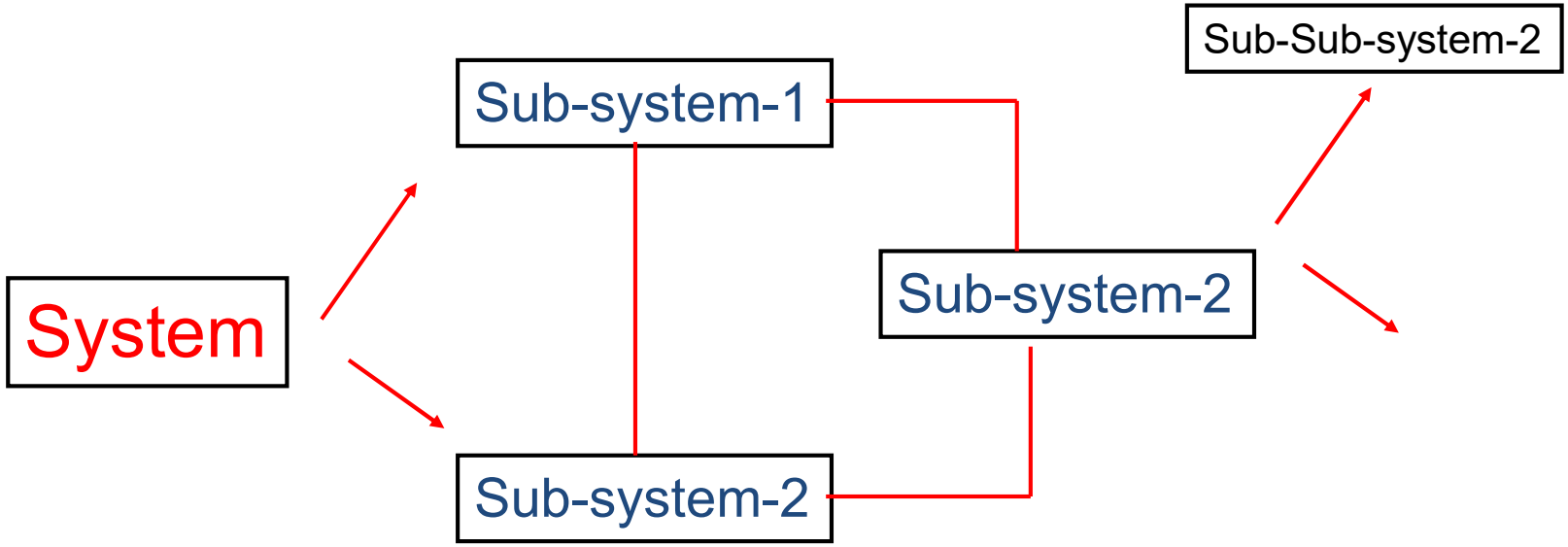
Design system as a network of sub-systems that are of manageable size and can be implemented using the earlier approach of truth table, minimization etc.




```
  1 1 0 1
+ 1 1 1 0
-------
1 1 0 1 1
```

a	b	c	S	CY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

General Approach

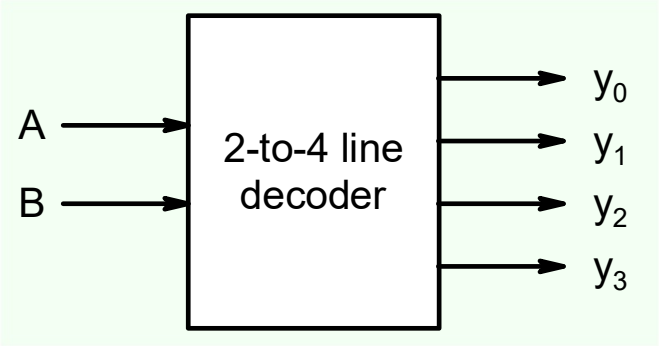


There are certain sub-systems or blocks that are used quite often such as :

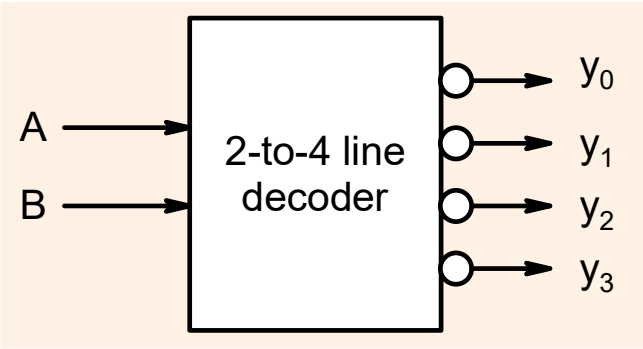
1. **decoders, encoders**
2. **Multiplexers**
3. **Adder/Subtractors, Multipliers**
4. **Comparators**
5. **Parity Generators**
6. **……..**

Decoders

Maps a smaller number of inputs to a larger set of outputs in general



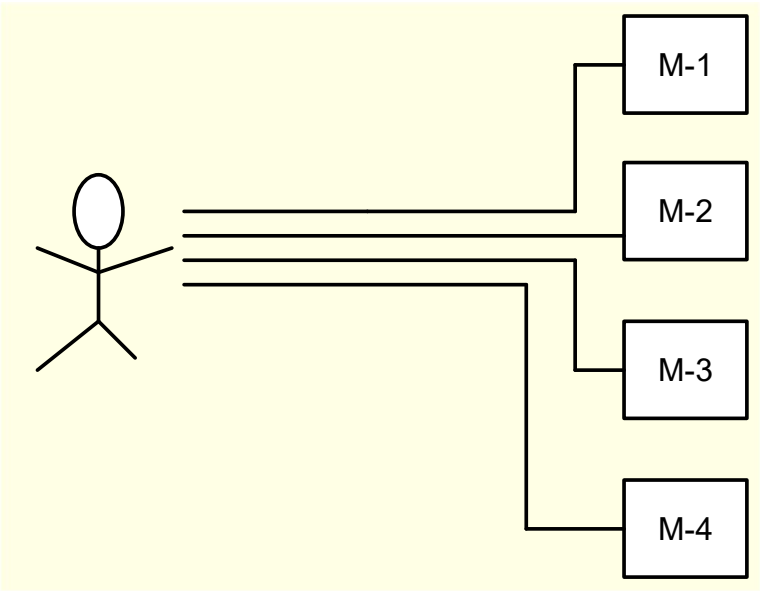
B	A	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

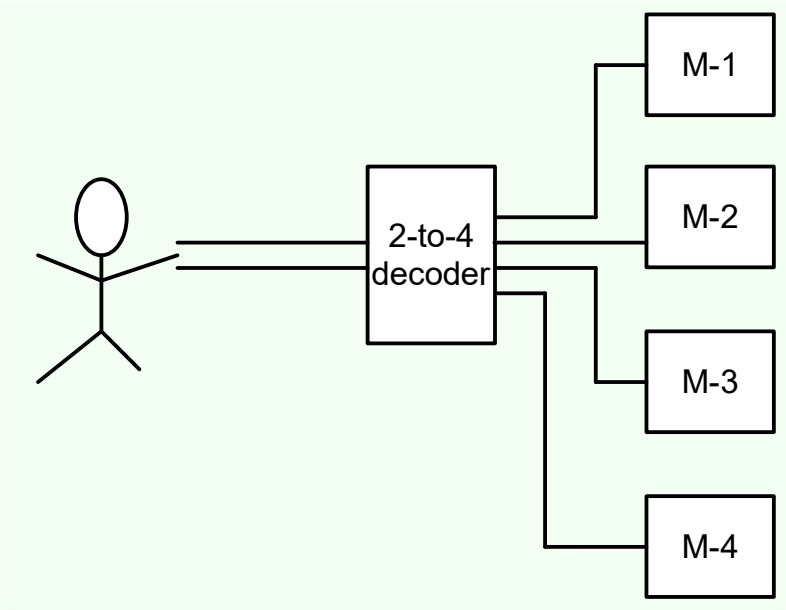


Active Low

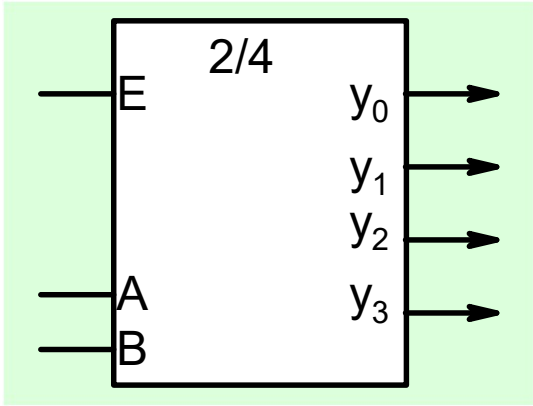
b	a	Y_0	Y_1	Y_2	Y_3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Example

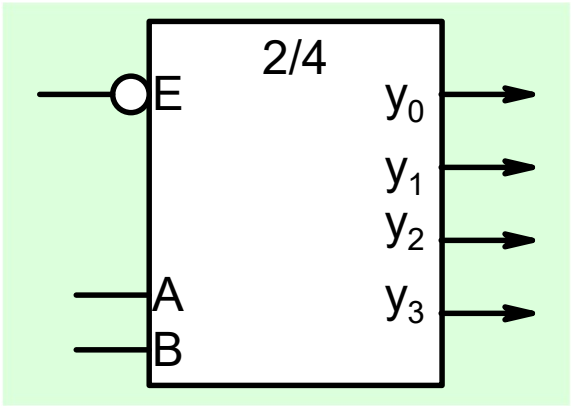




Decoder with Enable Input

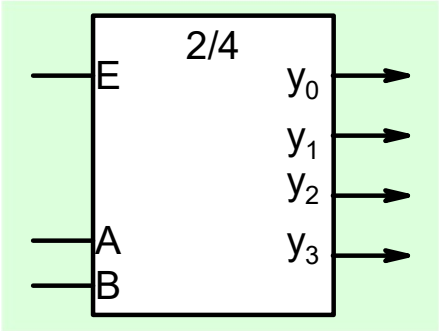


E	B	A	Y_0	Y_1	Y_2	Y_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1



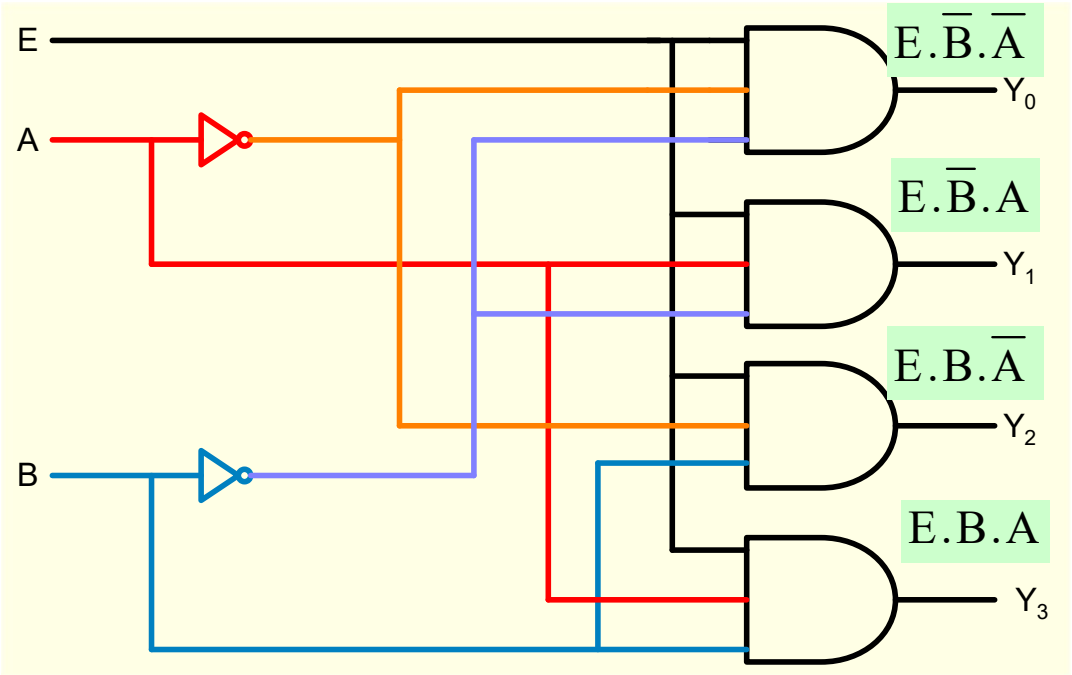
E	B	A	Y_0	Y_1	Y_2	Y_3
1	x	x	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1

Decoder: gate Implementation



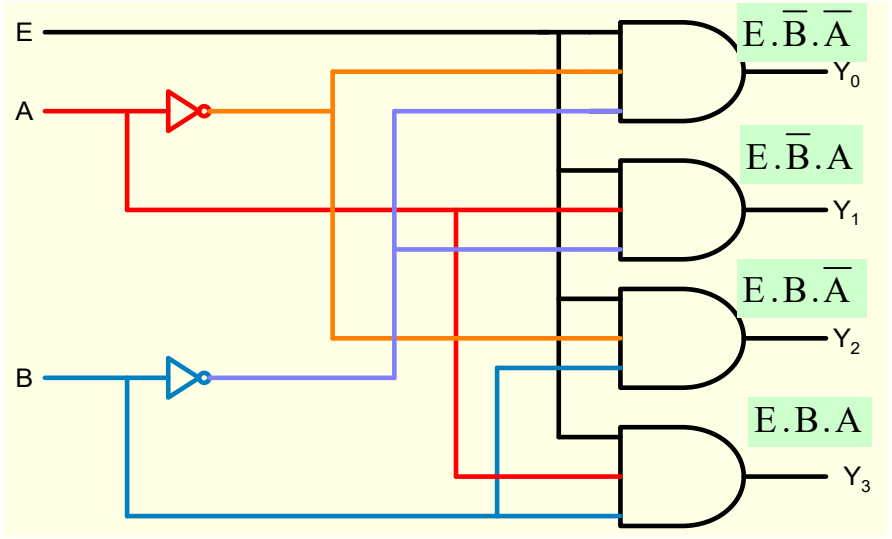
E	B	A	Y_0	Y_1	Y_2	Y_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

$$Y_0 = E.\overline{B}.\overline{A}\ ; Y_1 = E.\overline{B}.A ; Y_2 = E.B.\overline{A}\ ; Y_3 = E.B.A$$



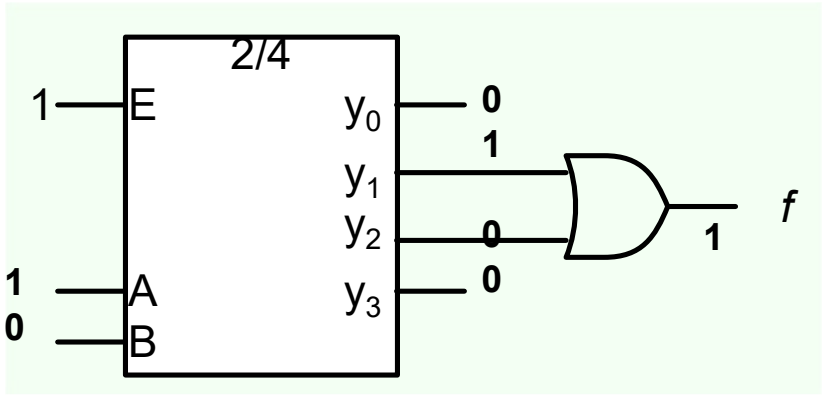
A n to 2^n decoder is a minterm generator

x	y	min term
0	0	$\overline{x}.\overline{y}$ m0
0	1	$\overline{x}.y$ m1
1	0	$x.\overline{y}$ m2
1	1	$x.y$ m3



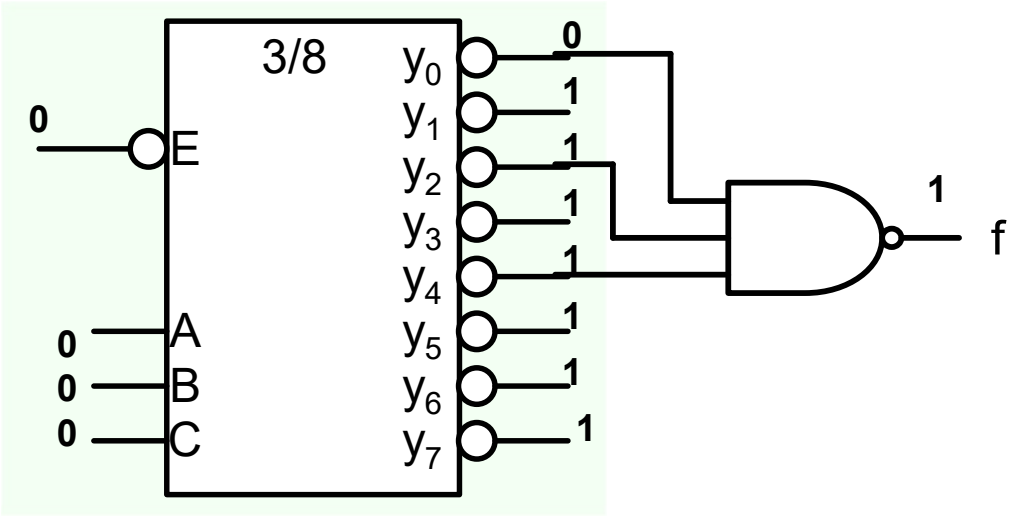
It can be used to implement any combinational circuit

B	A	f_1
0	0	0
0	1	1
1	0	1
1	1	0



Implementation of a 3-variable function with a 3-to-8 decoder

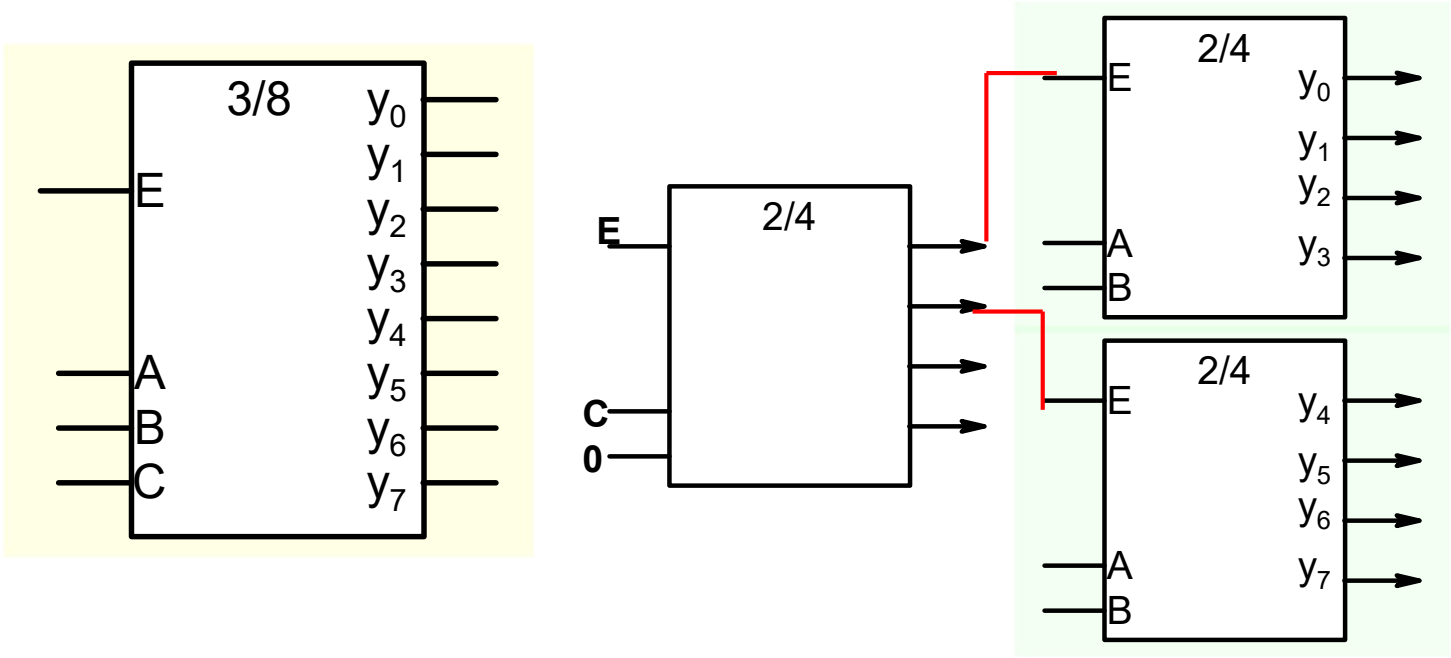
C	B	A	*f*
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0



Although it is easy to implement any combinational circuit with this method , it is often very inefficient in terms of gate utilization. Note that this method does not require any minimization.

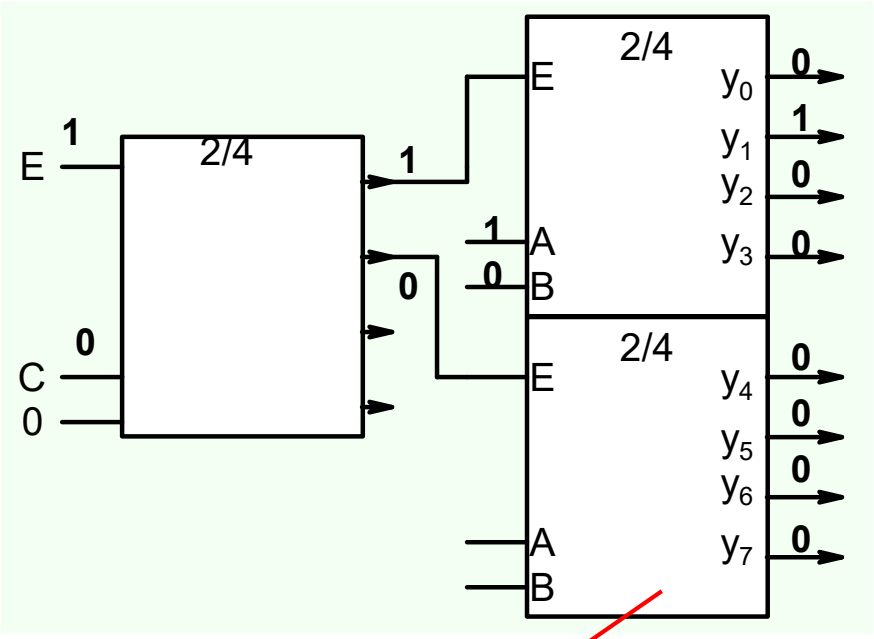
Implementing larger decoders using simpler ones.

3/8 decoder using 2/4 decoders



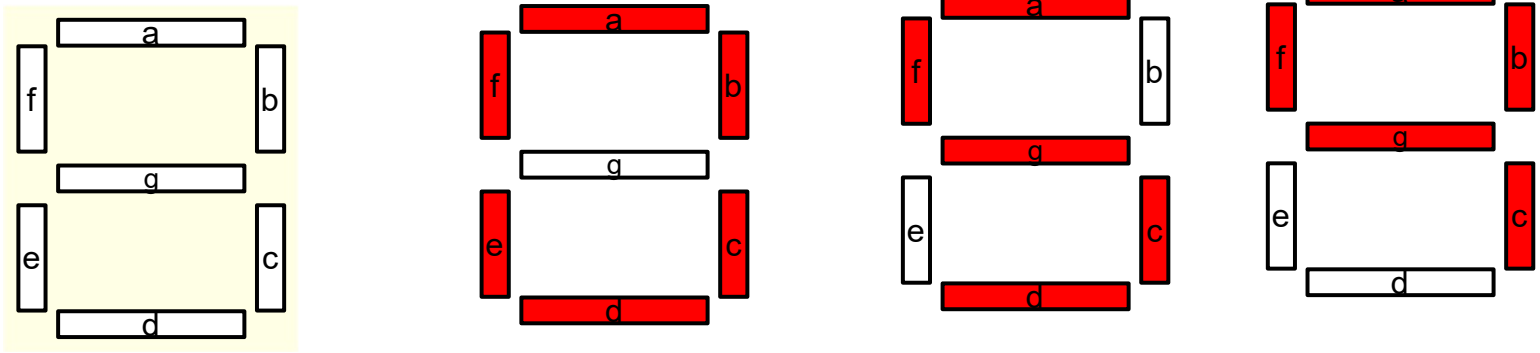
How many 2/4 decoders are required to implement a 4/16 decoder ?

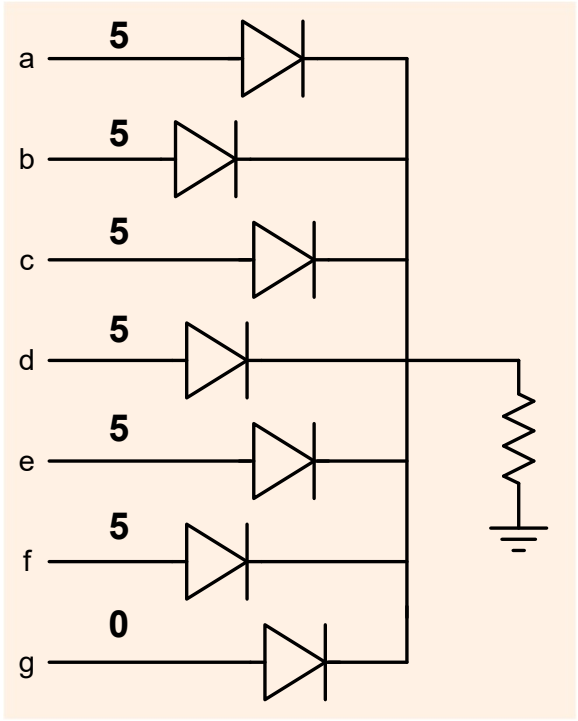
E	C	B	A	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

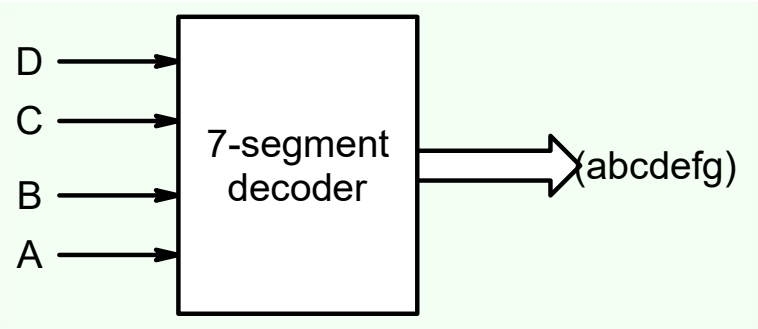


E	B	A	Y_0	Y_1	Y_2	Y_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

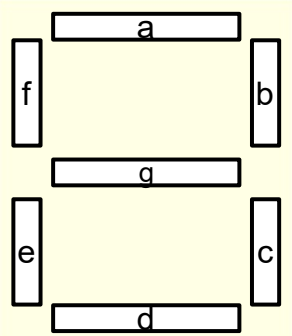
Seven segment decoder

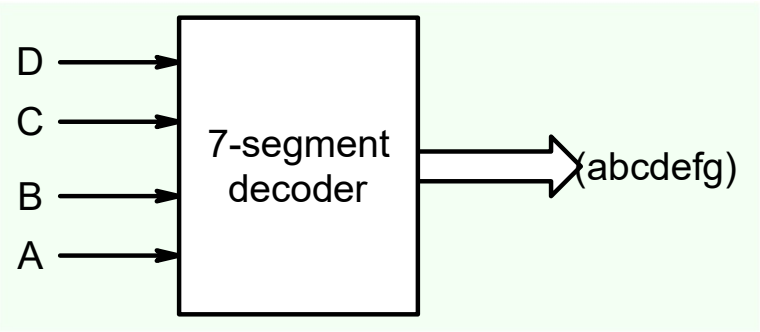






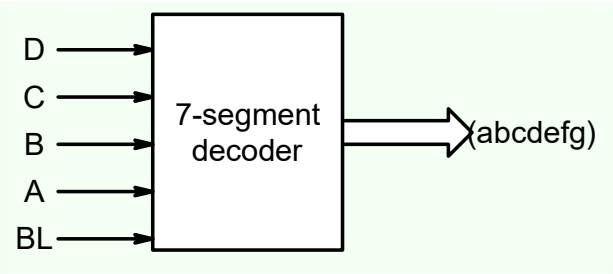
Seven segment decoder

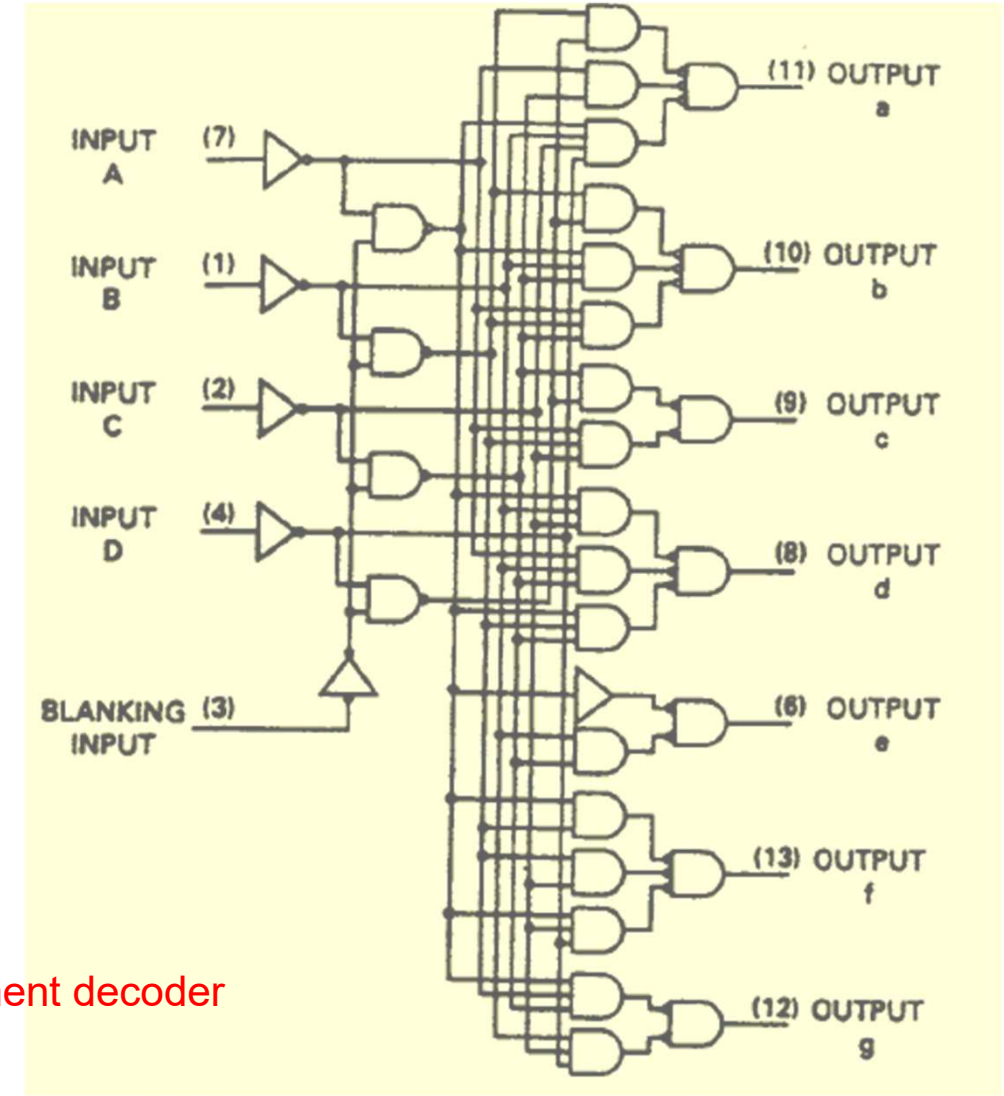




Dec or Function	Input				BI	Output						
	D	C	B	A		a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	1	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	1	0	0	1
4	0	1	0	0	1	0	1	1	0	0	1	1
5	0	1	0	1	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	0	1	1
10	1	0	1	0	1	0	0	0	1	1	0	1
11	1	0	1	1	1	0	0	1	1	0	0	1
12	1	1	0	0	1	0	1	0	0	0	1	1
13	1	1	0	1	1	1	0	0	1	0	1	1
14	1	1	1	0	1	0	0	0	1	1	1	1
15	1	1	1	1	1	0	0	0	0	0	0	0
BI	×	×	×	×	0	0	0	0	0	0	0	0

DC \ BA	00	01	11	10
00	1	0	1	1
01	0	1	1	0
11	0	1	0	0
10	1	1	0	0

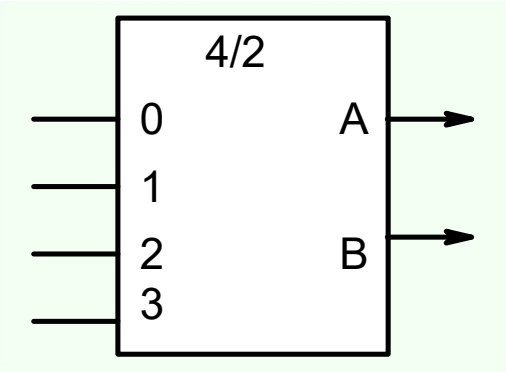




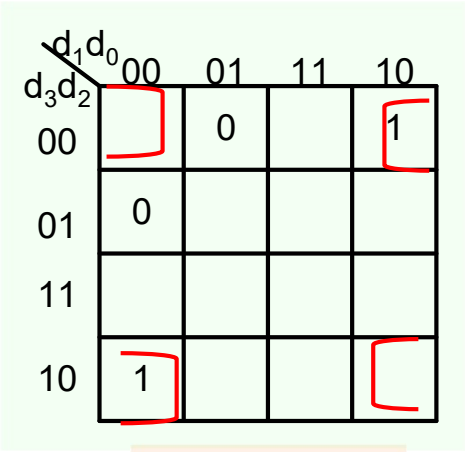
7449 BCD to seven segment decoder

Encoders

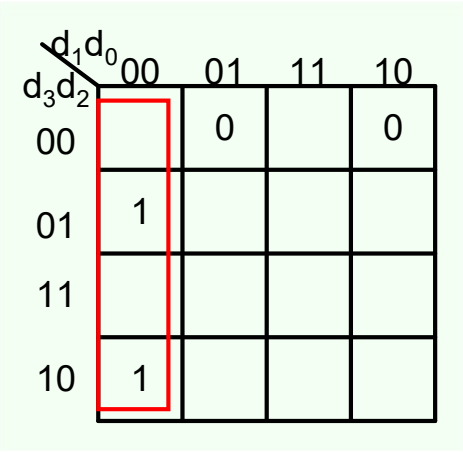
An encoder performs the inverse operation of a decoder.



d_3	d_2	d_1	d_0	B	A
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

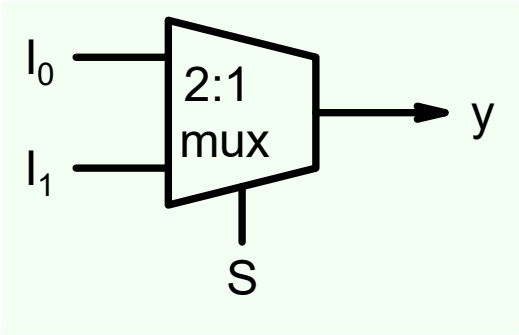


$$A = \overline{d_2}\ \overline{d_0}$$

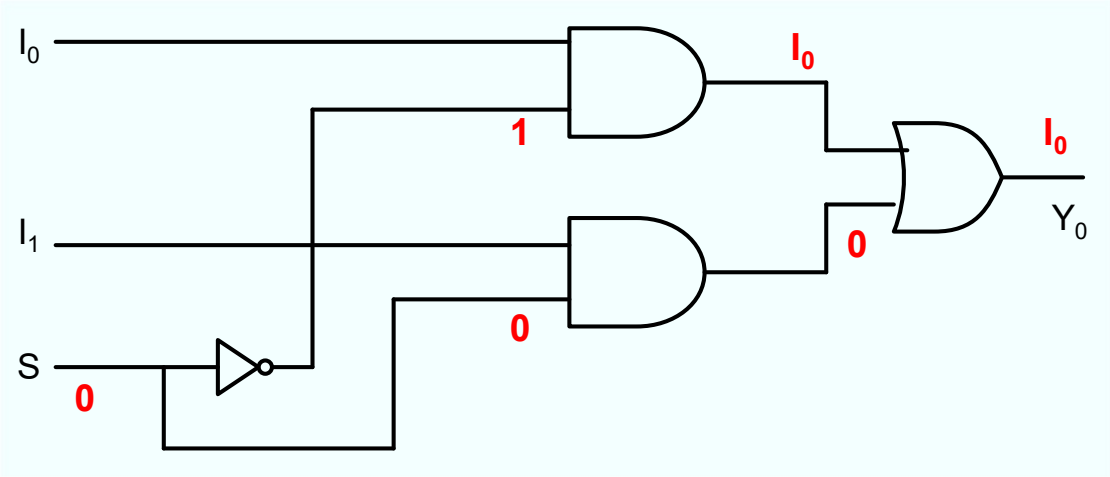


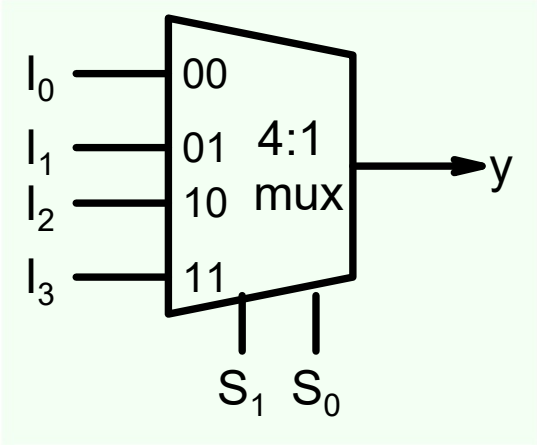
$$B = \overline{d_1}\ \overline{d_0}$$

Multiplexers

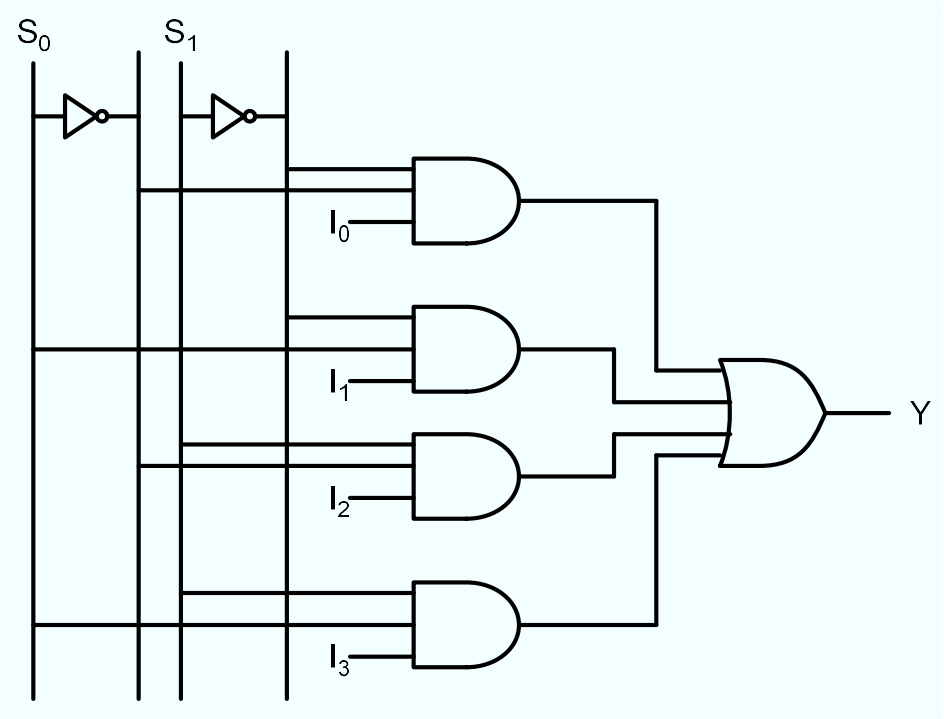


S	*y*
0	I_0
1	I_1

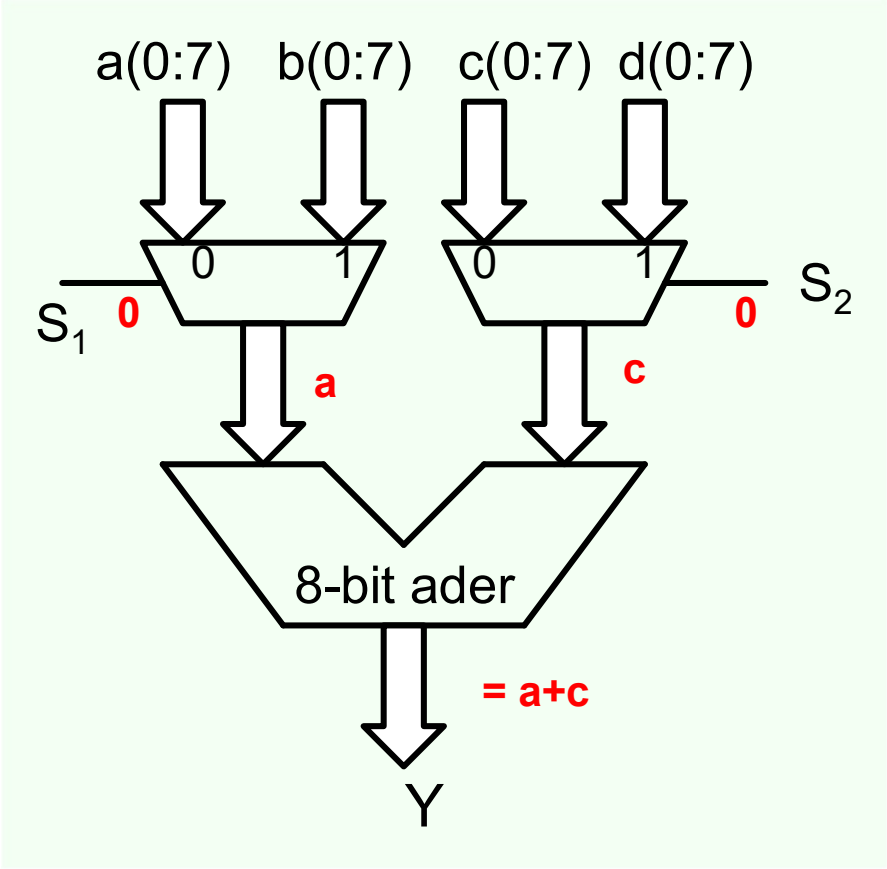




S_1	S_0	y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



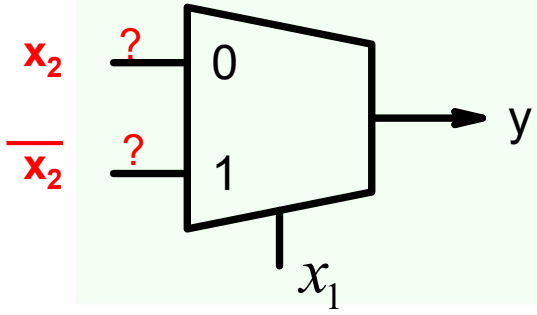
Mux is often used when resources have to be shared



S_1	S_0	y =
0	0	a+c
0	1	a+d
1	0	b+c
1	1	b+d

Implementing Boolean expressions using Multiplexers

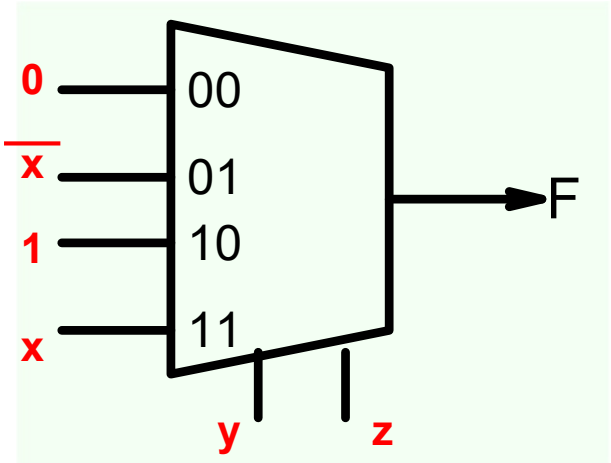
$$y = x_1\overline{x_2} + \overline{x_1}x_2$$



x_1	x_2	y	
0	0	0	
0	1	1	$y = x_2$ when $x_1 = 0$
1	0	1	
1	1	0	$y = \overline{x_2}$ when $x_1 = 1$

$$F(x, y, z) = \sum(1, 2, 6, 7)$$

A 3 variable function can be implemented with a 4:1 mux with 2 select lines



x	y	z	F
0	0	0	0
1	0	0	0
0	0	1	1
1	0	1	0
0	1	0	1
1	1	0	1
0	1	1	0
1	1	1	1

F = 0 when yz = 00

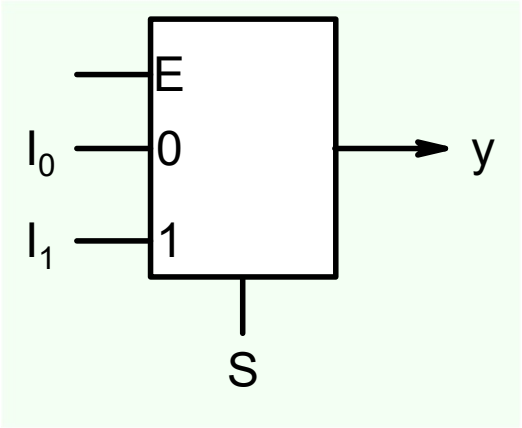
F = $\overline{x}$ when yz =01

F = 1 when yz = 10

F = x when yz = 11

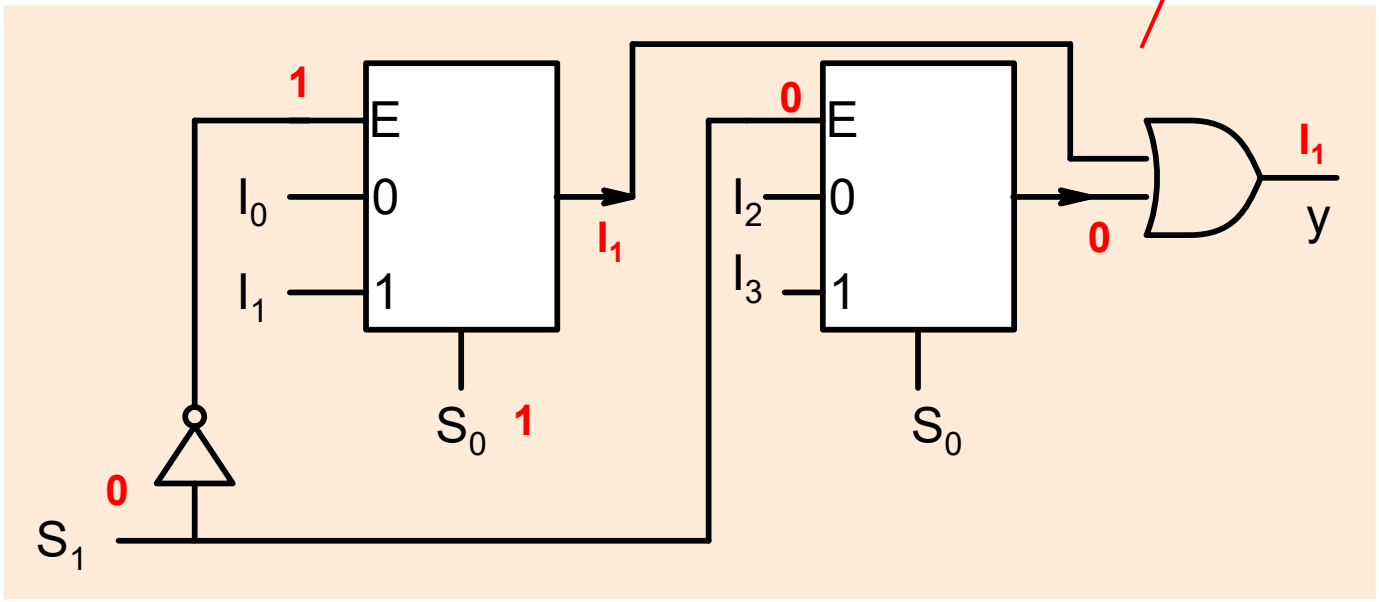
Mux is more efficient way of implementing combinational circuits as compared to decoders.

Mux. expansion

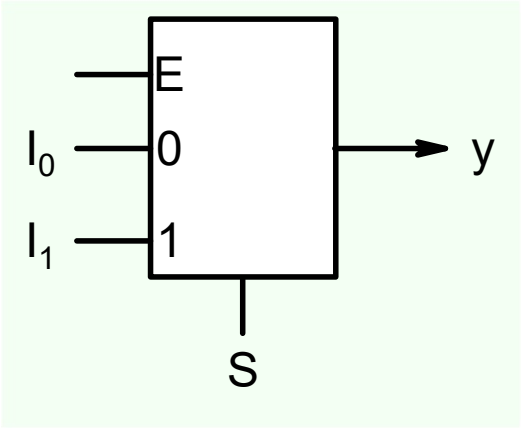


E	S	y
0	x	0
1	0	I_0
1	1	I_1

S_1	S_0	y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

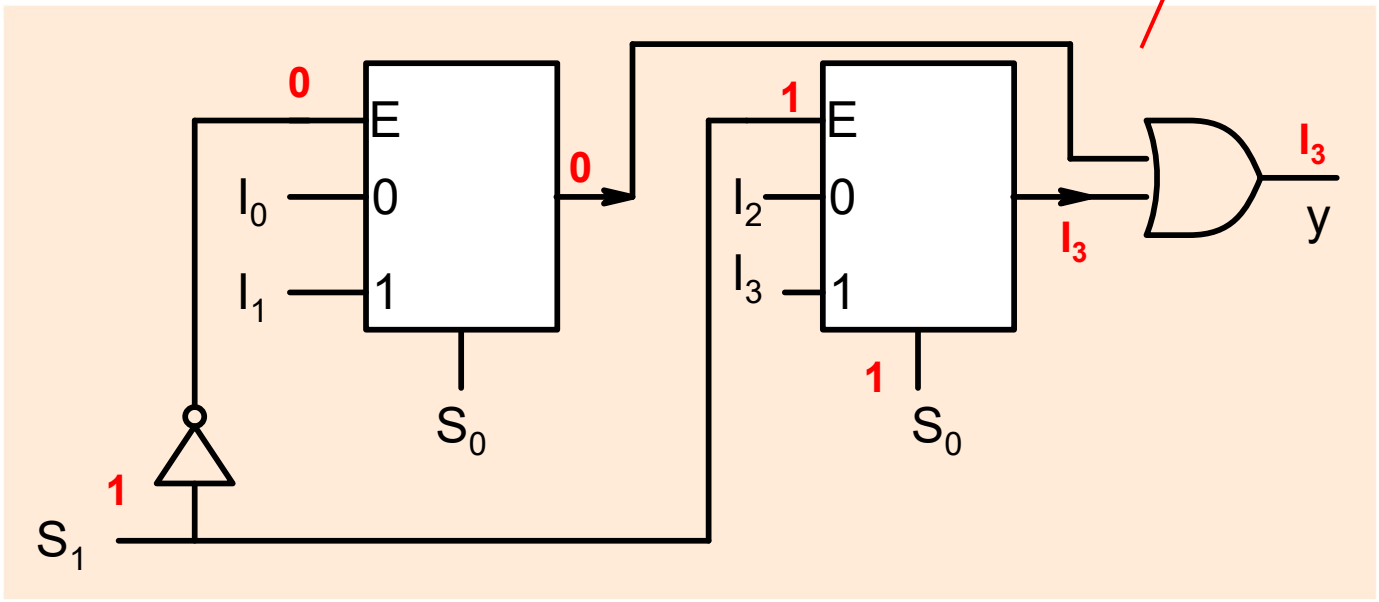


Mux. expansion

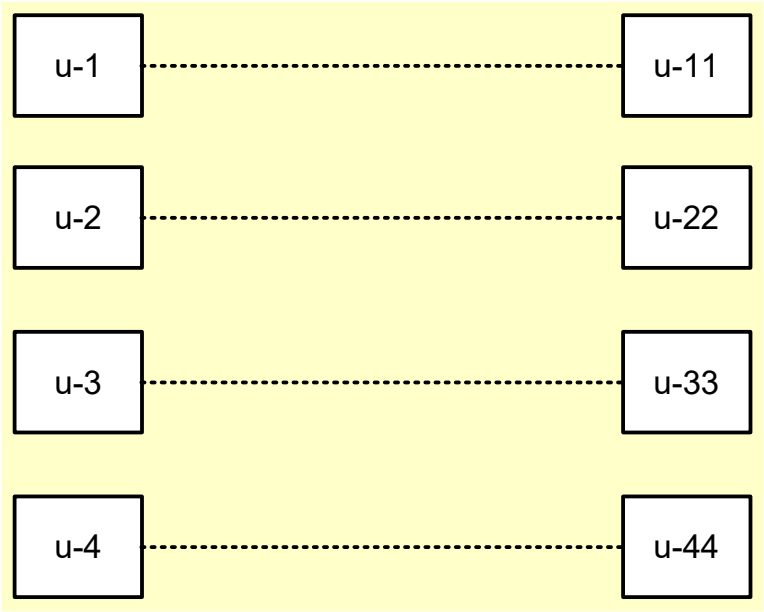


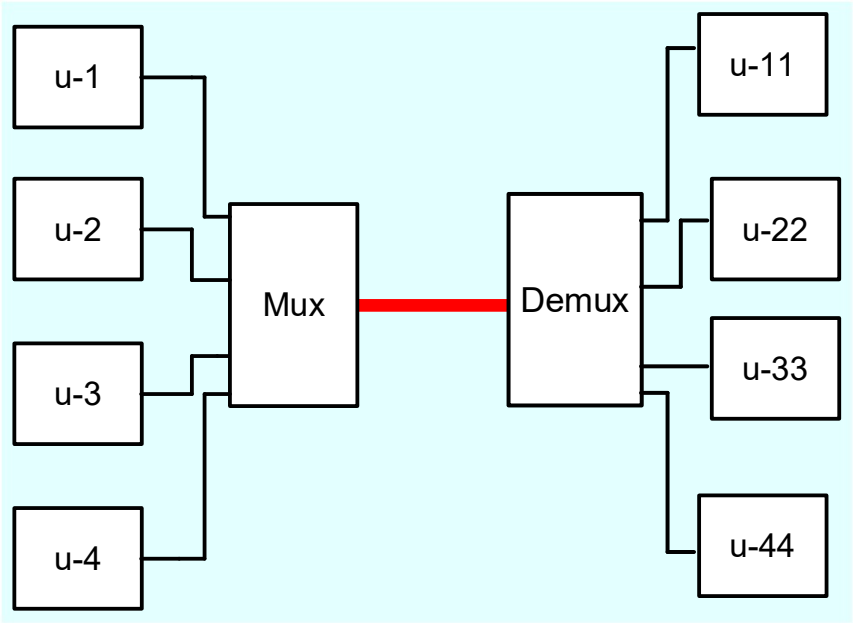
E	S	y
0	x	0
1	0	I_0
1	1	I_1

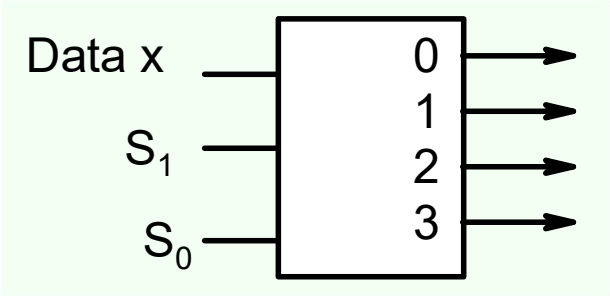
S_1	S_0	y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



DeMultiplexer

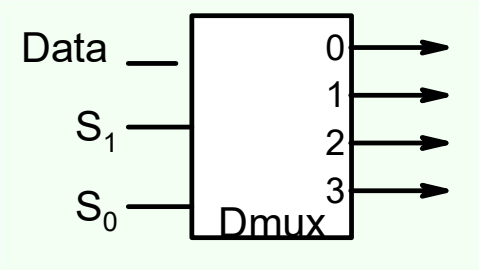


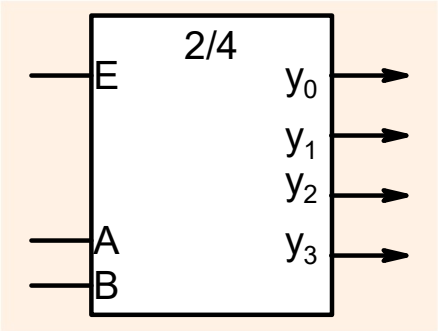




S_1	S_0	y_0	y_1	y_2	y_3
0	0	x	0	0	0
0	1	0	x	0	0
1	0	0	0	x	0
1	1	0	0	0	x

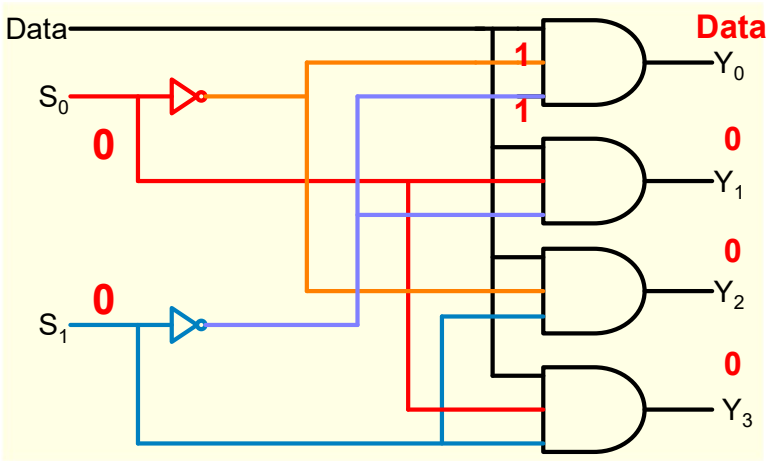
Demultiplexer is very much like a decoder

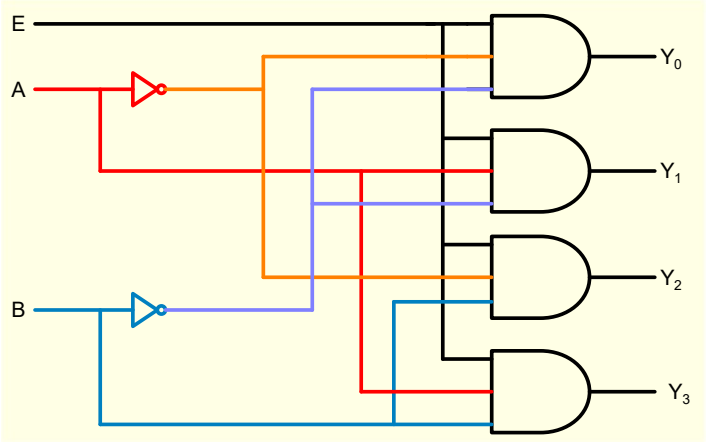




S_1	S_0	y_0	y_1	y_2	y_3
0	0	x	0	0	0
0	1	0	x	0	0
1	0	0	0	x	0
1	1	0	0	0	x

E	B	A	Y_0	Y_1	Y_2	Y_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1



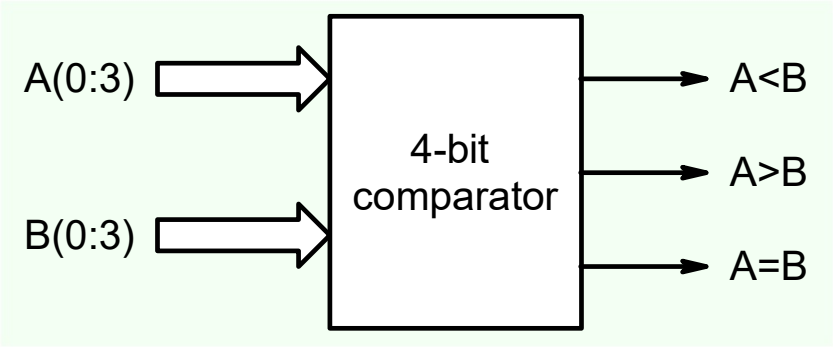


ESC201T : Introduction to Electronics

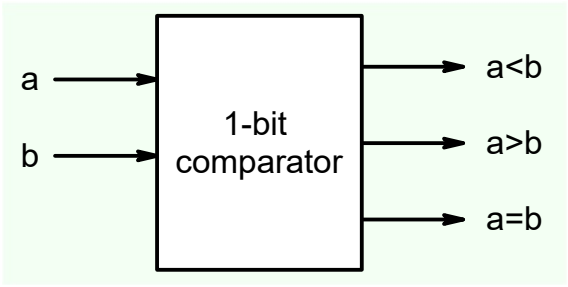
Lecture 36: Combination circuit-3

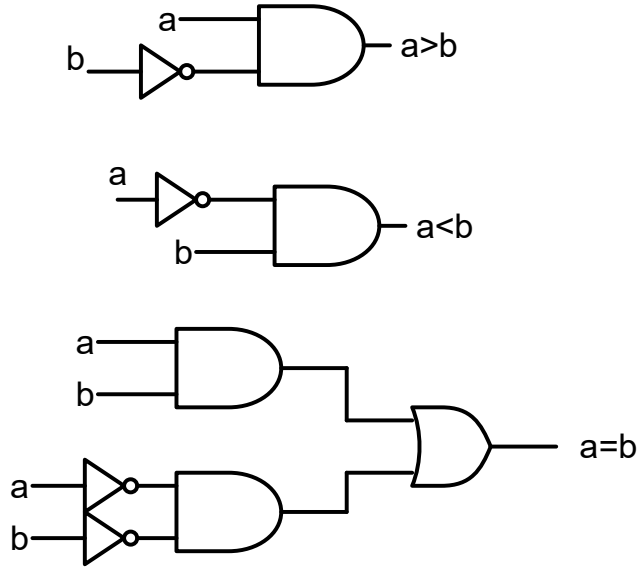
B. Mazhari
Dept. of EE, IIT Kanpur

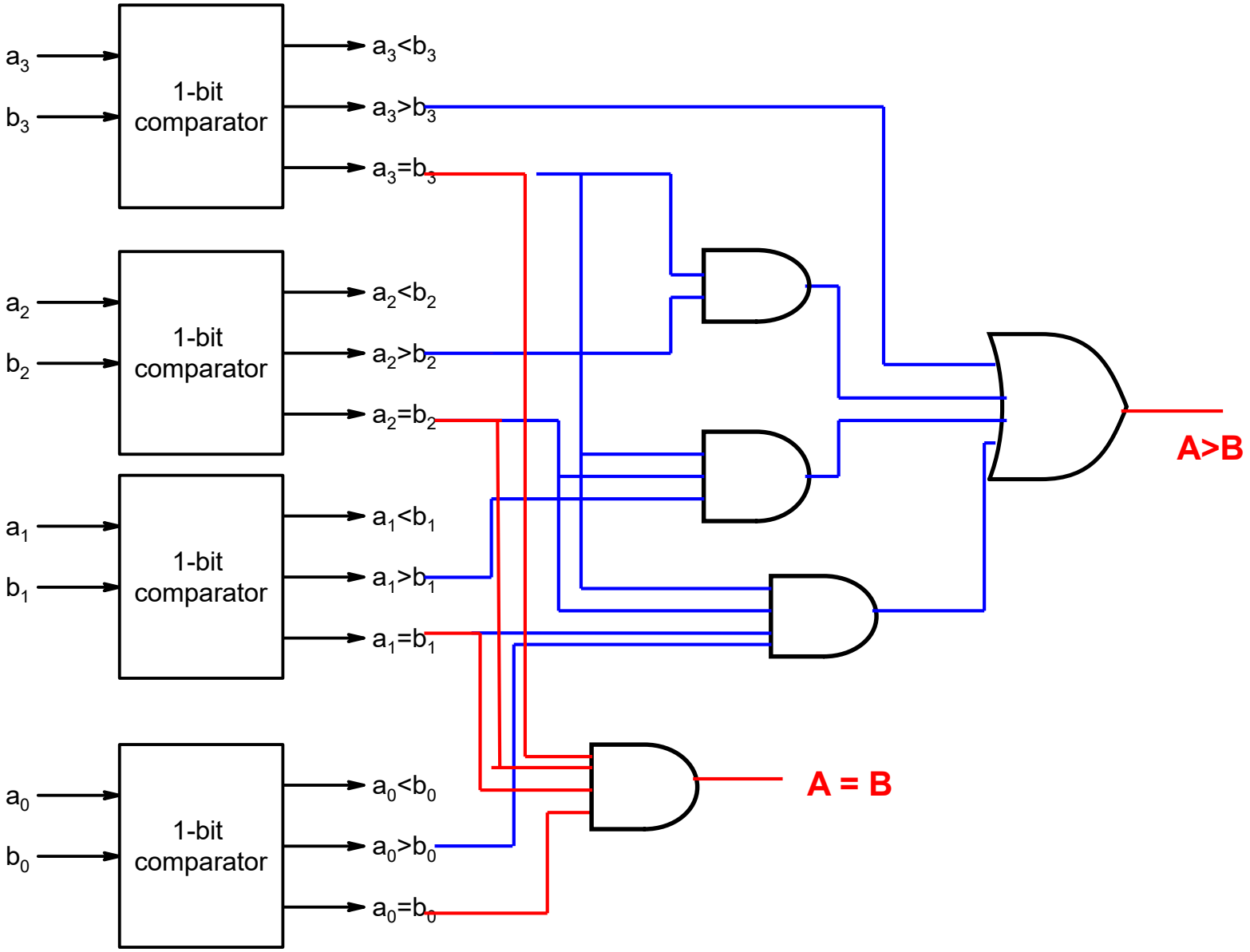
Comparator



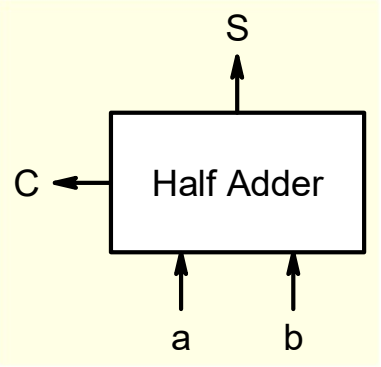
$A_3A_2A_1A_0$	$B_3B_2B_1B_0$	A<B	A>B	A=B
0000	0000	0	0	1
0000	0001	1	0	0
0001	0000	0	1	0
⋮	⋮	⋮	⋮	⋮



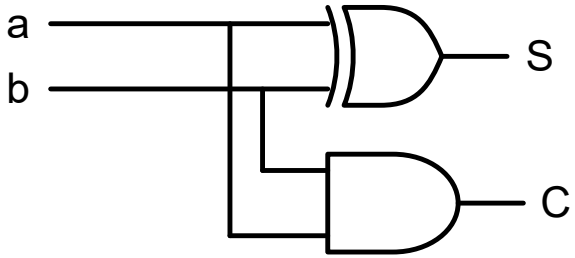



a3
b3
1-bit comparator
a3<b3
a3>b3
a3=b3
a2
b2
1-bit comparator
a2<b2
a2>b2
a2=b2
a1
b1
1-bit comparator
a1<b1
a1>b1
a1=b1
a0
b0
1-bit comparator
a0<b0
a0>b0
a0=b0
A>B
A = B

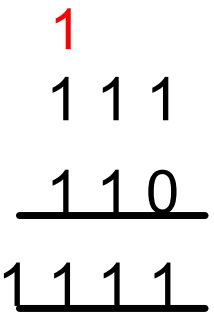
Adder/Subtractor

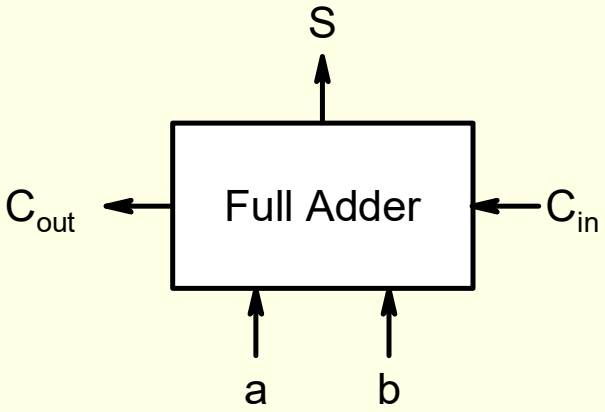


a	b	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



$$S = \overline{a}.b + a.\overline{b}; C = a.b$$





a	b	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

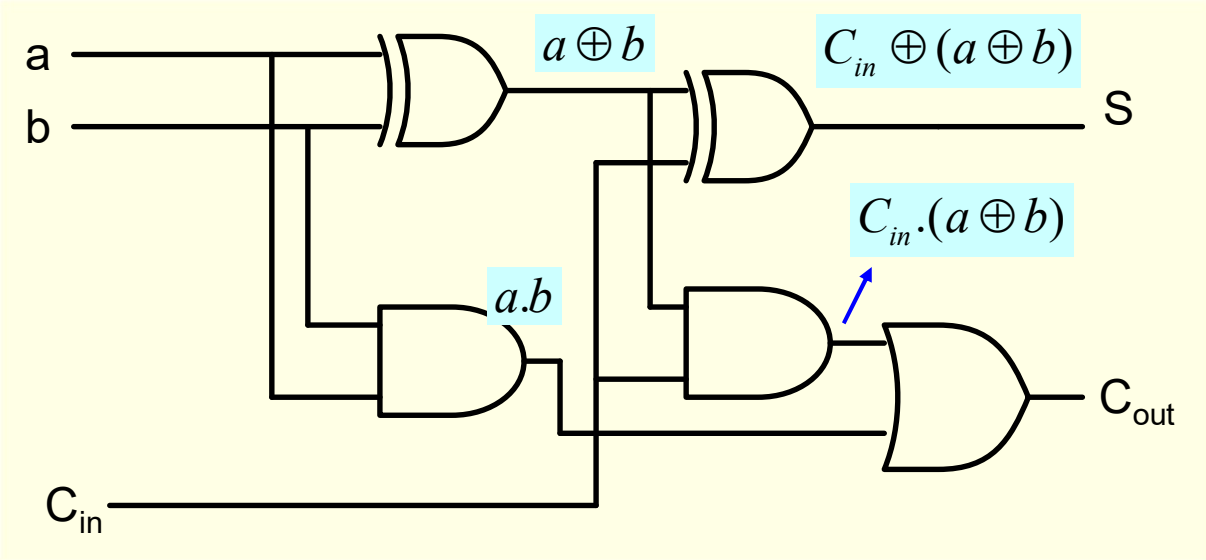
$$S = \overline{a}.\overline{b}.c_{in} + \overline{a}.b.\overline{c_{in}} + a.\overline{b}.\overline{c_{in}} + a.b.c_{in};\ C_{out} = a.b + a.c_{in} + b.c_{in}$$

$$S = \bar{a}.\bar{b}.c_{in} + \bar{a}.b.\overline{c_{in}} + a.\bar{b}.\overline{c_{in}} + a.b.c_{in}$$

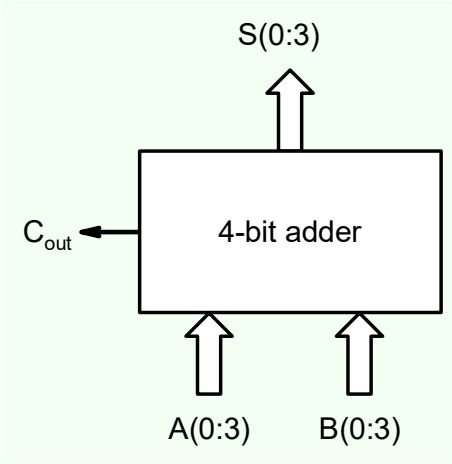
$$S = C_{in} \oplus (a \oplus b)$$

$$C_{out} = a.b + a.C_{in} + b.C_{in}$$

$$C_{out} = C_{in}(a.\bar{b} + \bar{a}.b) + a.b = C_{in}.(a \oplus b) + a.b$$



4-bit Adder



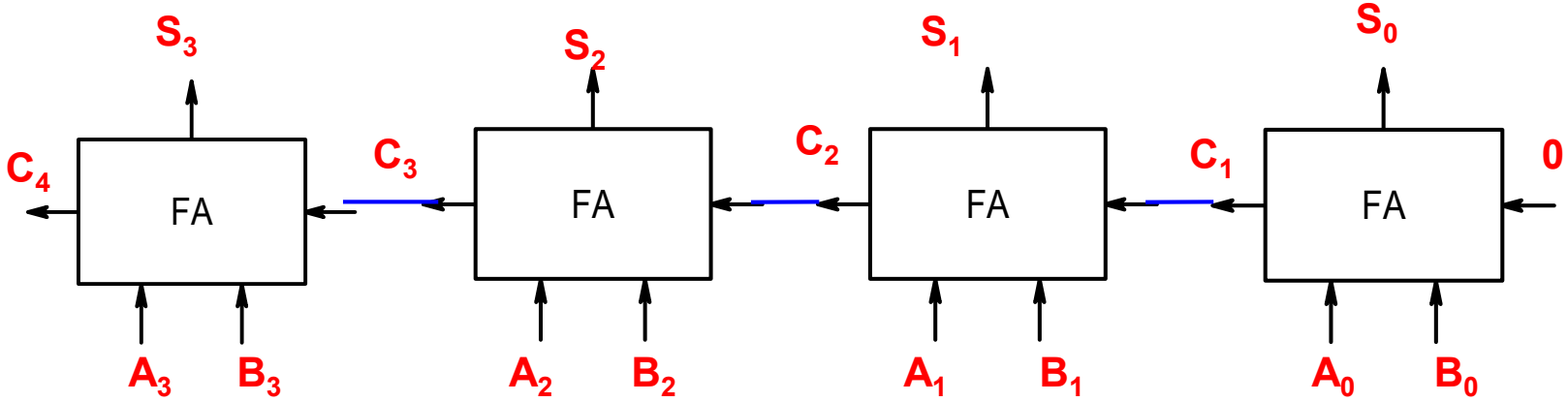
$A_3A_2A_1A_0$	$B_3B_2B_1B_0$	$S_3S_2S_1S_0$	C_{out}
0000	0000	0000	1
0000	0001	0001	0
0001	0000	0001	0
⋮	⋮	⋮	⋮

$C_3\,C_2\,C_1$

$A_3\;A_2\;A_1A_0$

$B_3\;B_2\;B_1B_0$

$C_4\;\;S_3\;S_2\;S_1S_0$



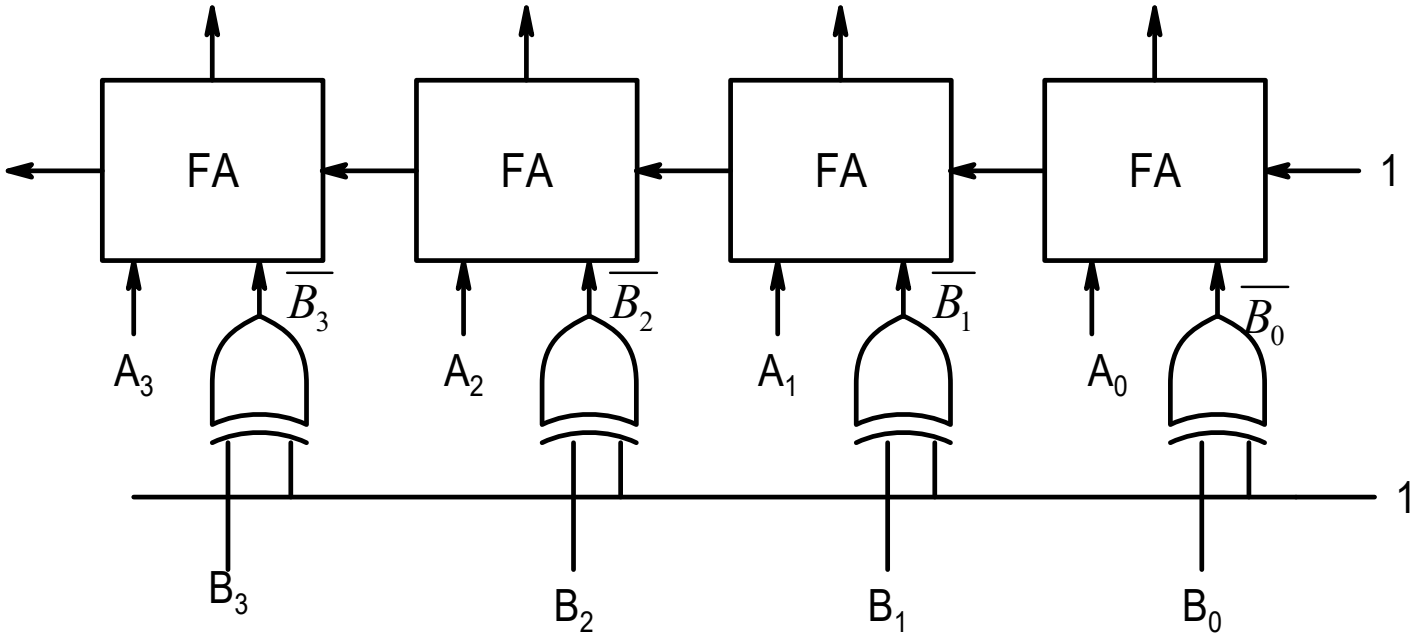
Ripple Carry Adder (20 gate circuit)

Subtraction

A - B = A + 2's complement of B

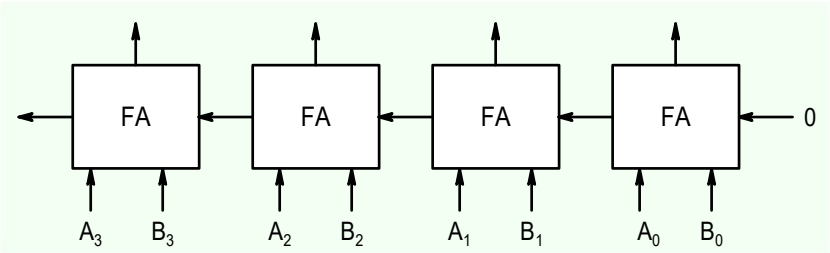
A - B = A + 1's complement of B+1

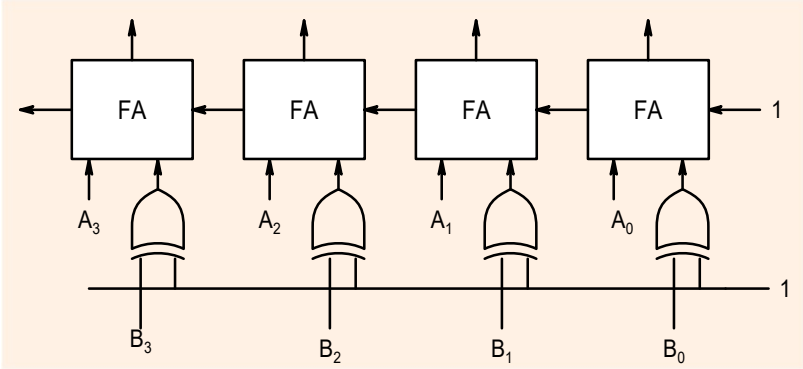
$$A - B = A + \overline{B} + 1$$

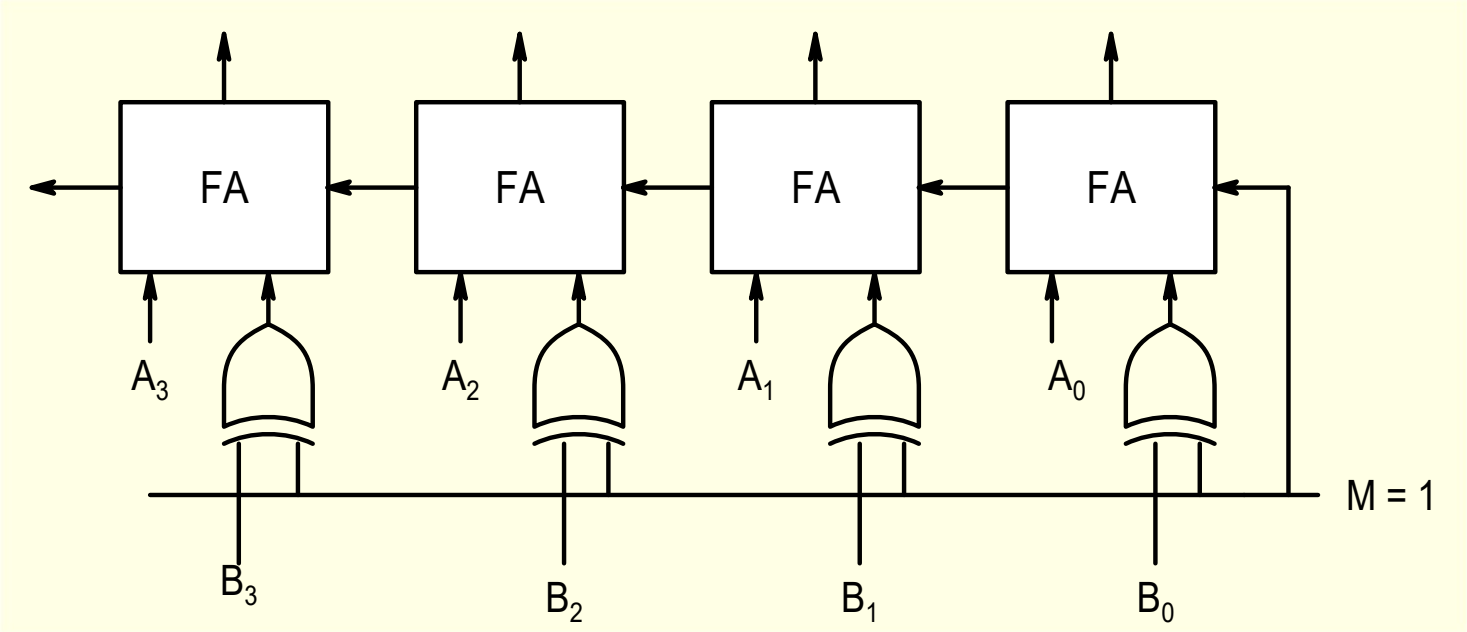


One needs add a circuit for predicting errors resulting from overflow

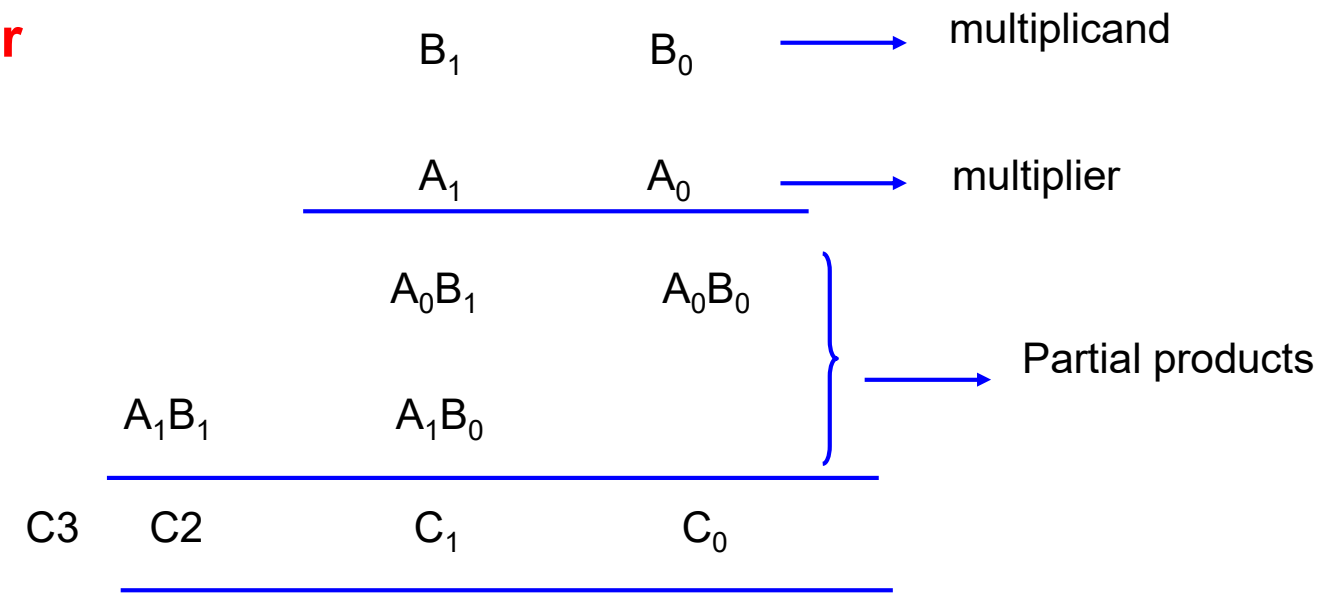
Adder/Subtractor

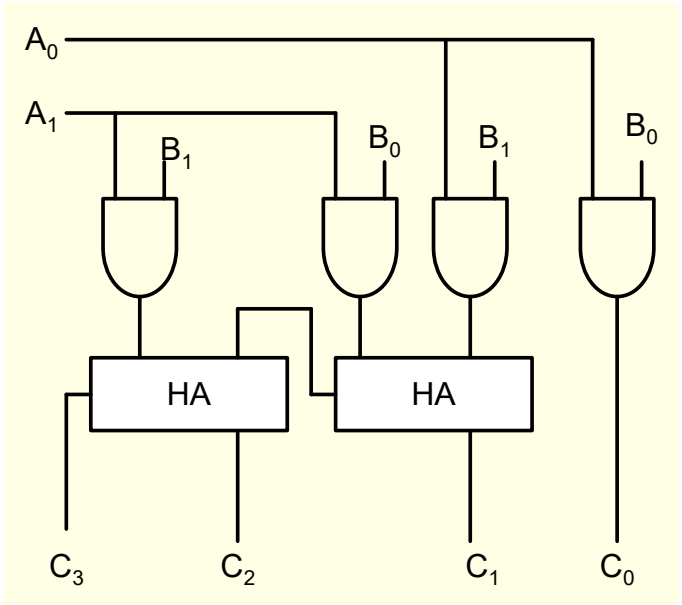






Multiplier


B1 B0
multiplicand
A1 A0
multiplier
A0B1 A0B0
A1B1 A1B0
Partial products
C3 C2 C1 C0


A0
A1
B1
B0
B1
B0
HA
HA
C3
C2
C1
C0

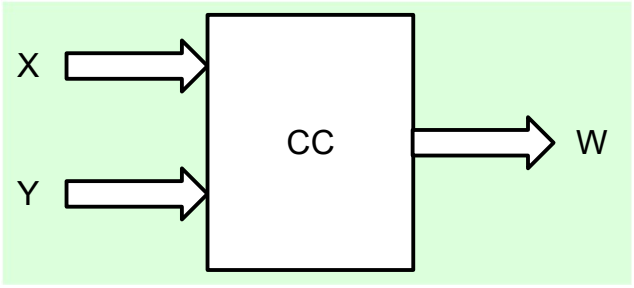
ESC201T : Introduction to Electronics

Lecture 37: Sequential circuit design-1

B. Mazhari
Dept. of EE, IIT Kanpur

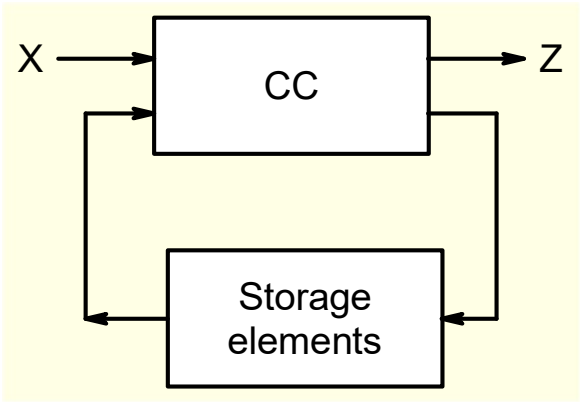
Digital Circuits

Combinational Circuits



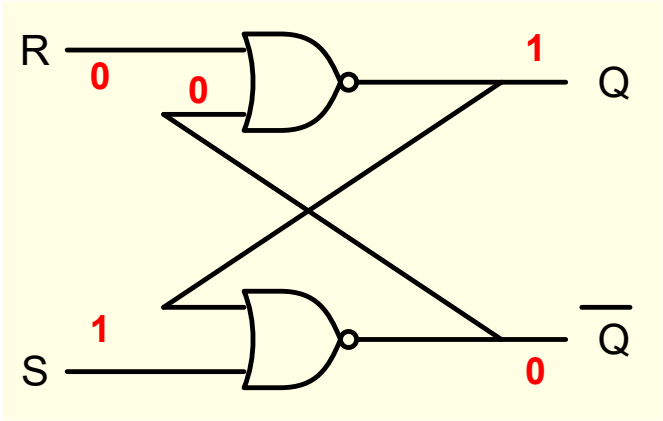
Output is determined by current values of inputs only.

Sequential Circuits



Output is determined in general by current values of inputs and past values of inputs/outputs as well.

NOR SR Latch

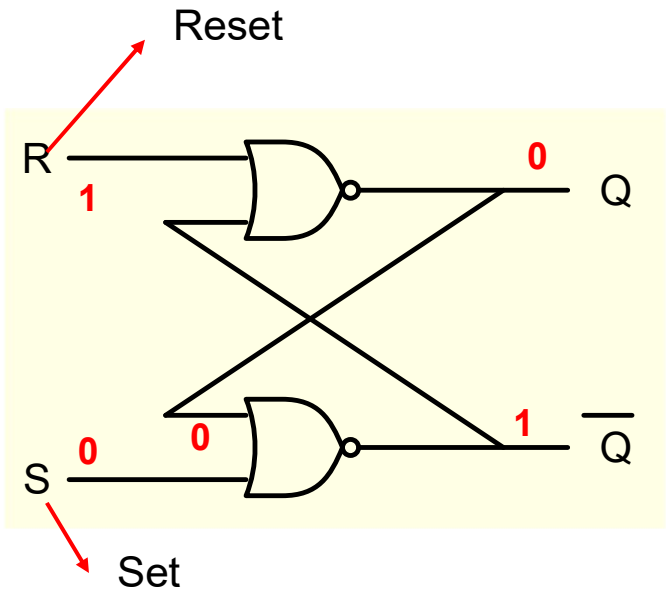


$Q = 1; \overline{Q} = 0$ Set State

$Q = 0; \overline{Q} = 1$ Reset State

S	R	Q	$\overline{Q}$	State
1	0	1	0	SET

NOR SR Latch

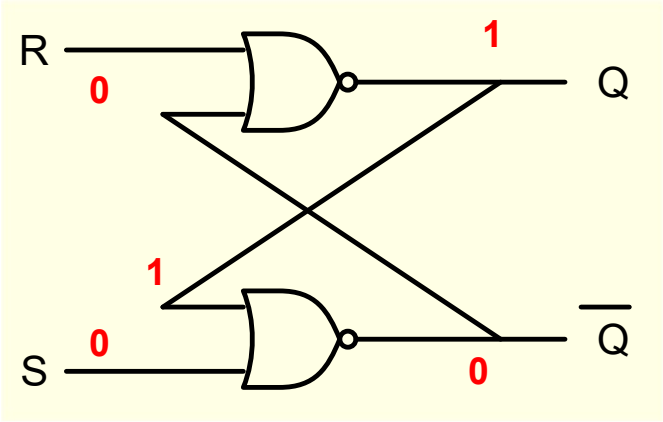


$Q = 1; \overline{Q} = 0$ Set State

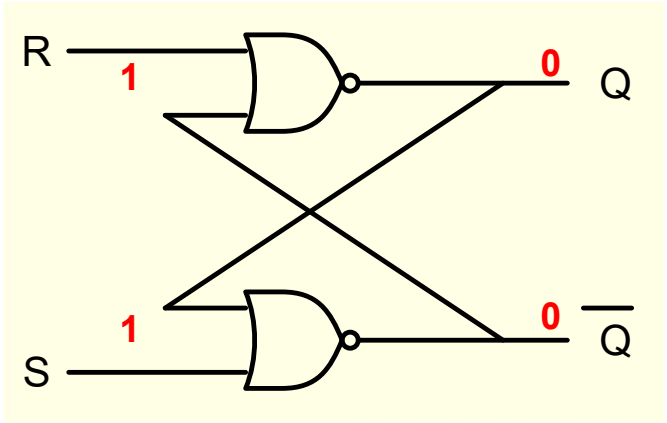
$Q = 0; \overline{Q} = 1$ Reset State

S	R	Q	$\overline{Q}$	State
1	0	1	0	SET
0	1	0	1	RESET

HOLD State

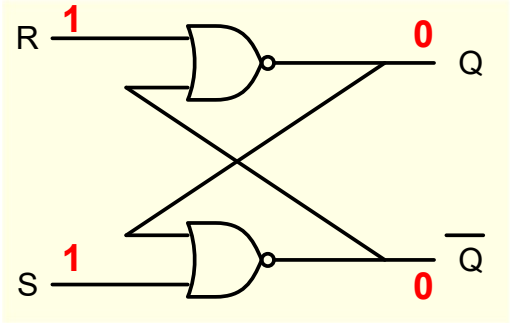


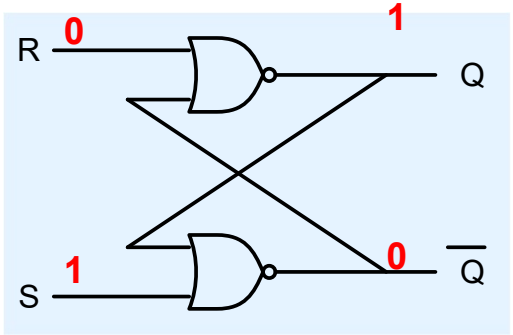
S	R	Q	$\overline{Q}$	State
1	0	1	0	SET
0	1	0	1	RESET
0	0	Q	$\overline{Q}$	HOLD
1	1	0	0	INVALID

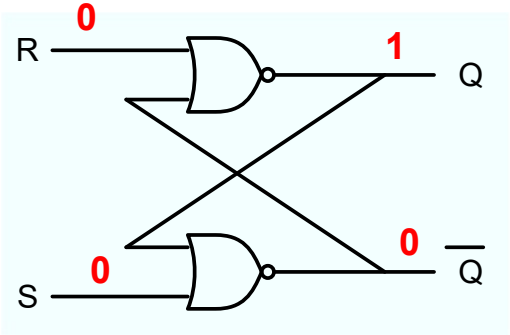


Both the outputs are well defined and 0. the first problem is that we do not get complementary output.

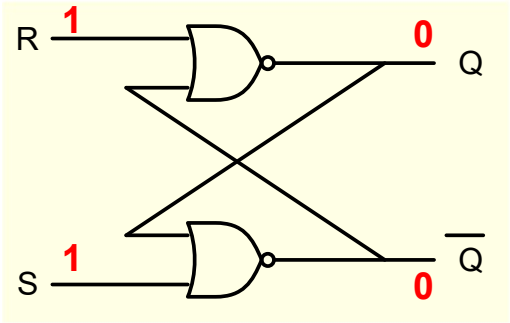
A more serious problem occurs when we switch the latch to the hold state by changing RS from 11 → 00 . Suppose the inputs do not change simultaneously and we get the situation 11 → 01* → 00

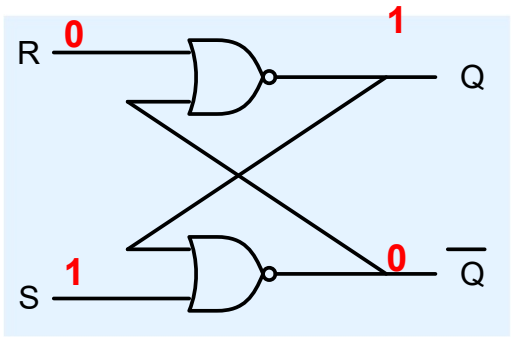


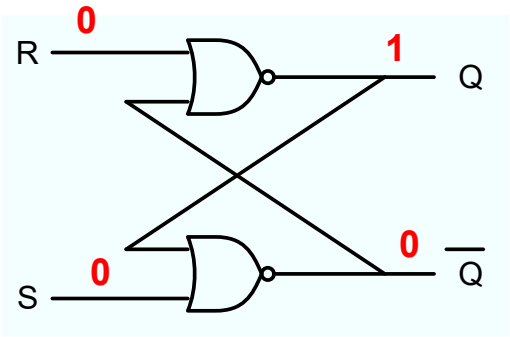




Q = 1

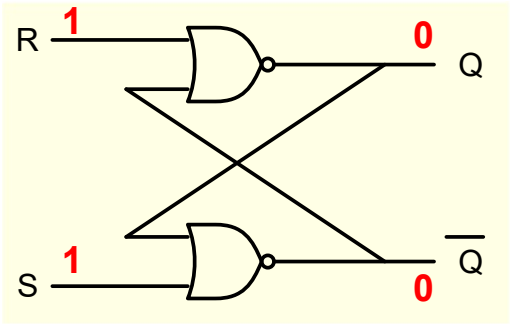


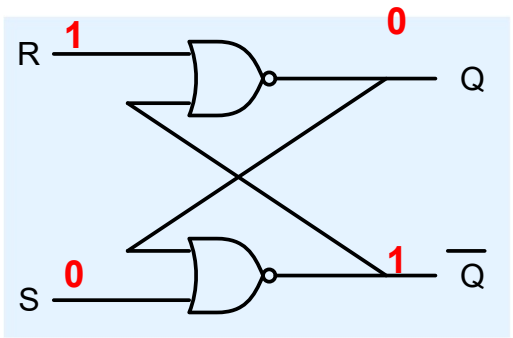


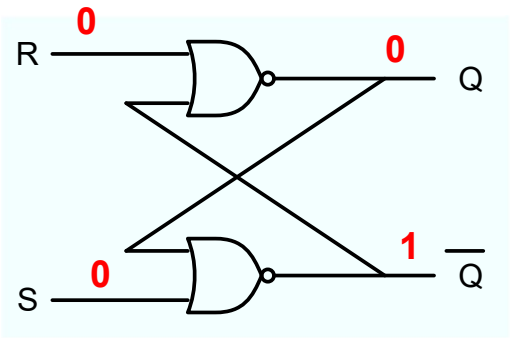


Q = 1

Suppose the inputs change as RS = 11 → 10* → 00



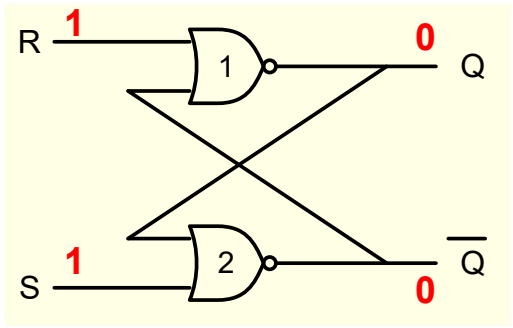


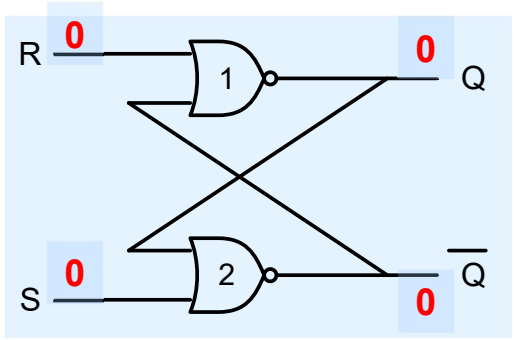


Q = 0

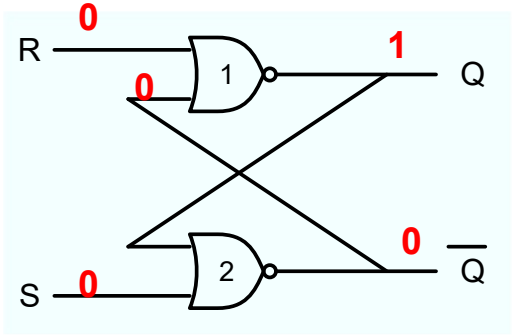
So although output is well defined when we apply RS = 11, it becomes unpredictable once we switch the latch to hold state by applying RS = 00. That is why RS = 11 is not used as an input combination.

The error can occur also due to unequal gate delays.



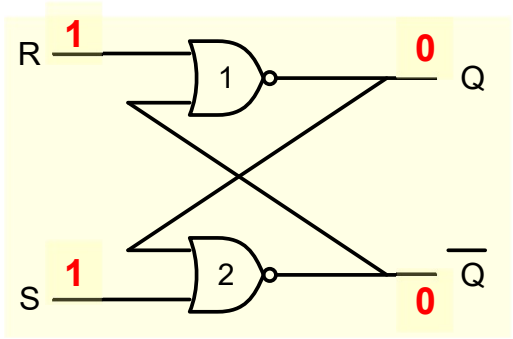


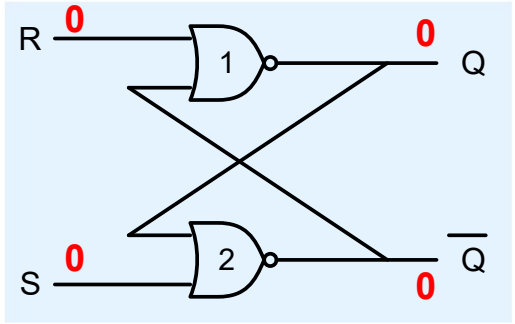
Suppose gate-1 is faster

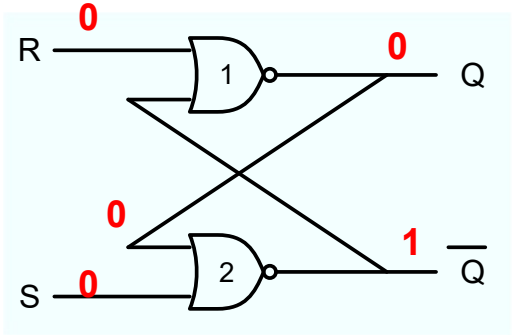


Q = 1

On the other hand suppose that gate-2 is faster.



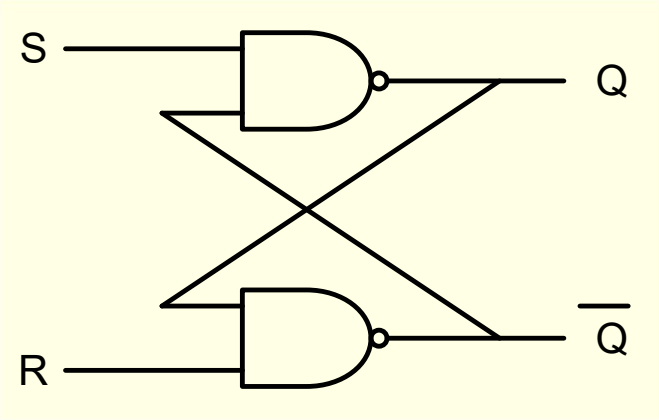




Again the output is unpredictable in general

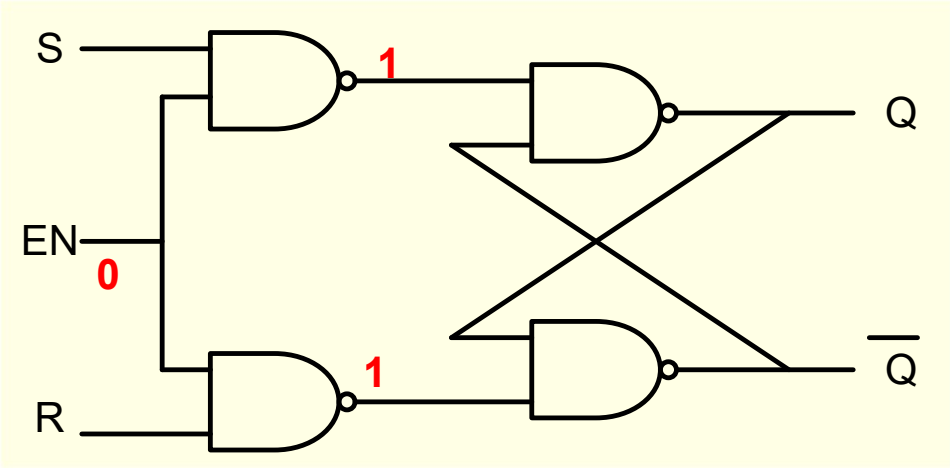
Q = 0

NAND Latch

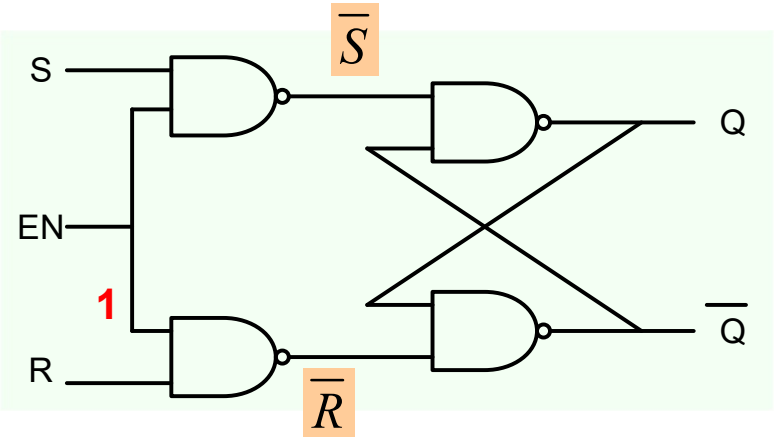


S	R	Q	$\overline{Q}$	State
0	1	1	0	SET
1	0	0	1	RESET
1	1	Q	$\overline{Q}$	HOLD
0	0	1	1	INVALID

RS NAND Latch with Enable

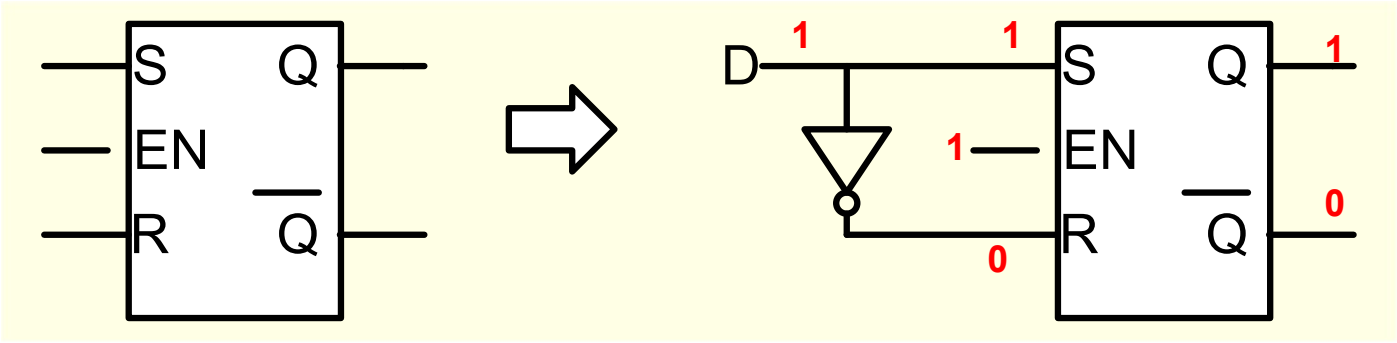


Hold State

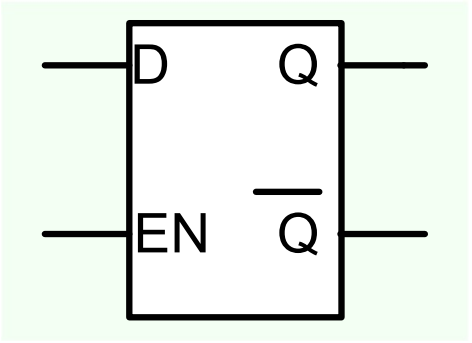


Enable	S R	Q $\overline{Q}$	State
0	x x	Q $\overline{Q}$	Hold
1	1 0	1 0	Set
1	0 1	0 1	Reset
1	0 0	Q $\overline{Q}$	Hold
1	1 1	0 0	Invalid

D latch

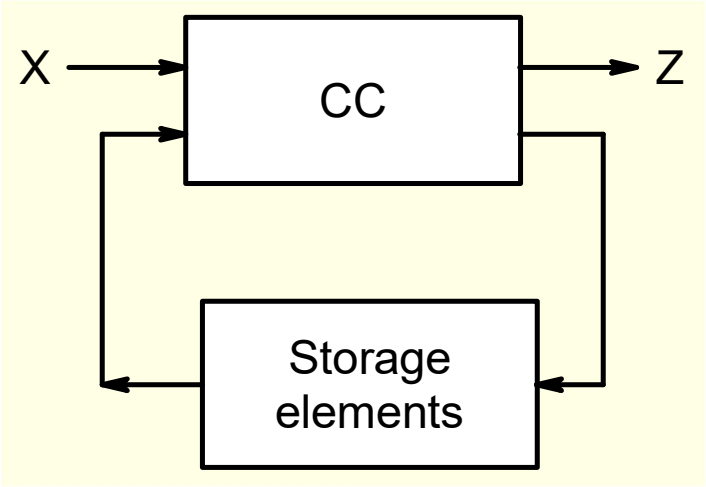


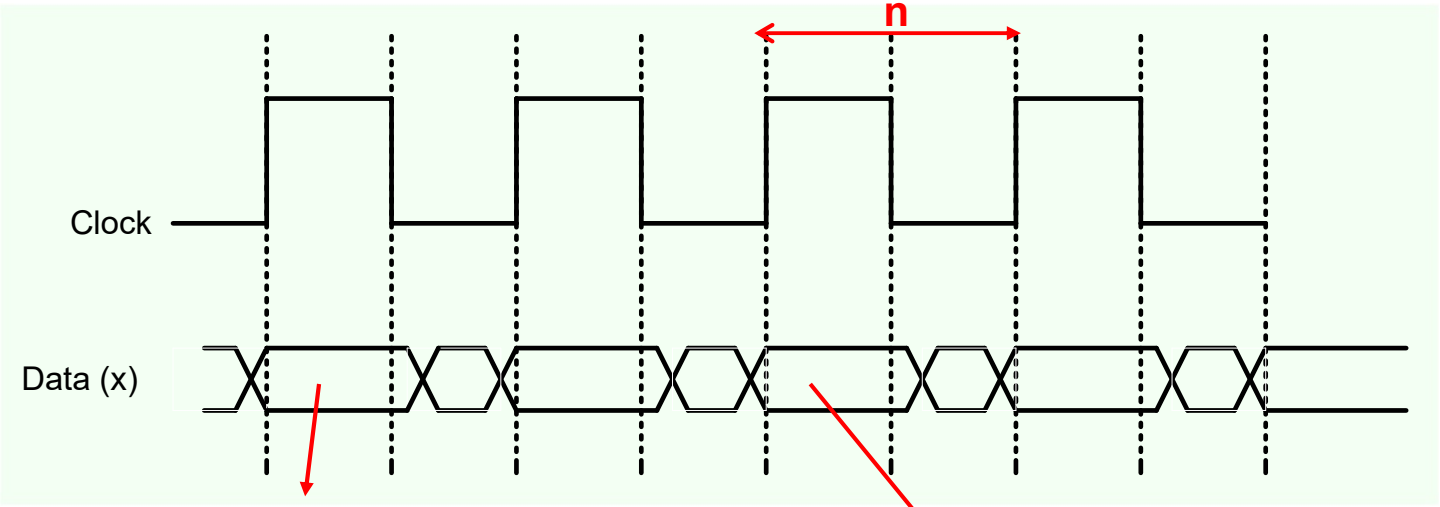
Enable	S	R	Q	$\overline{Q}$	State
0	x	x	Q	$\overline{Q}$	Hold
1	1	0	1	0	Set
1	0	1	0	1	Reset
1	0	0	Q	$\overline{Q}$	Hold
1	1	1	0	0	Invalid



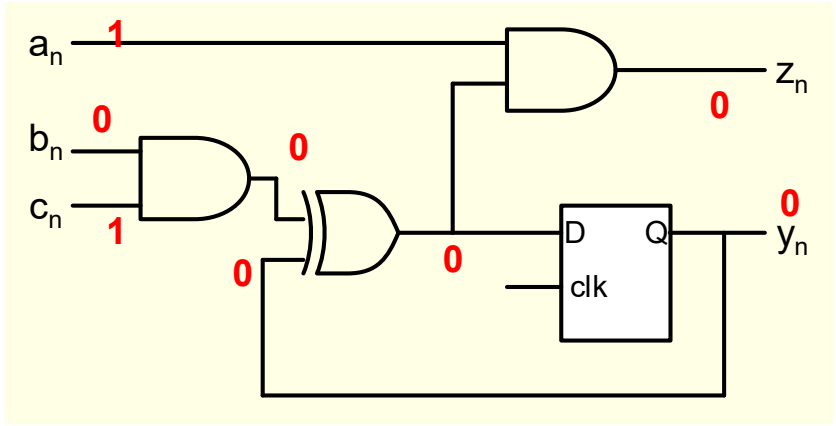
If EN = 1 then Q = D otherwise the latch is in Hold state

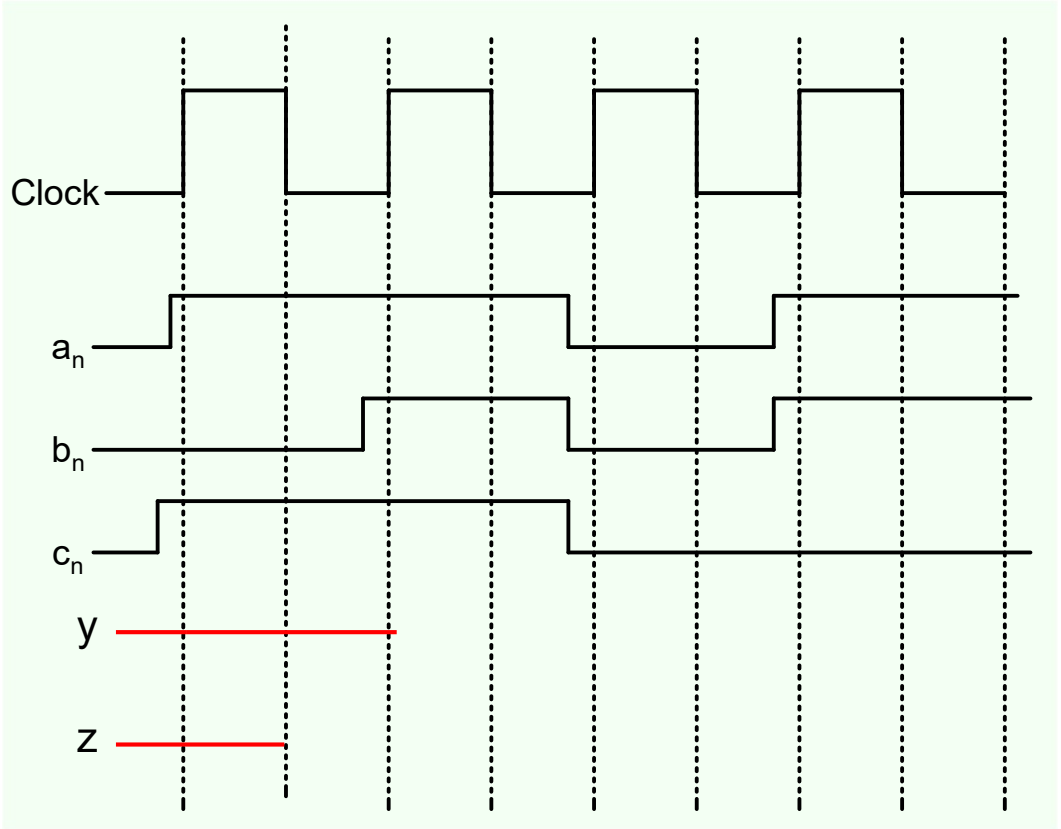
Synchronous Sequential Circuits



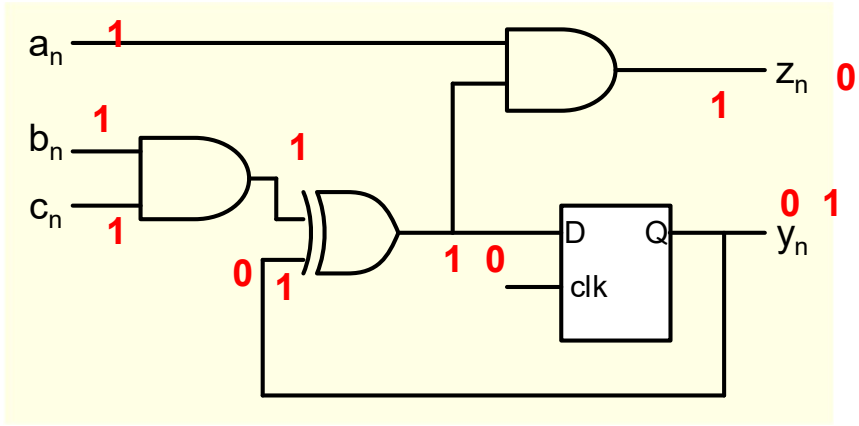


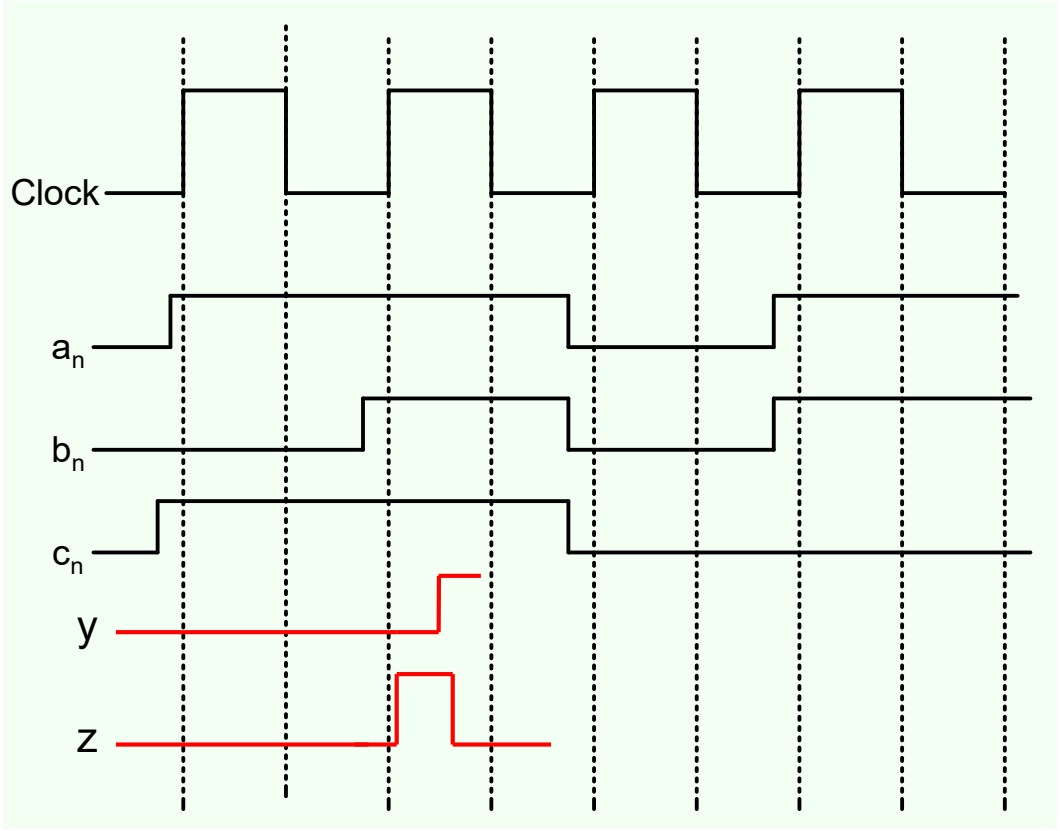
Example



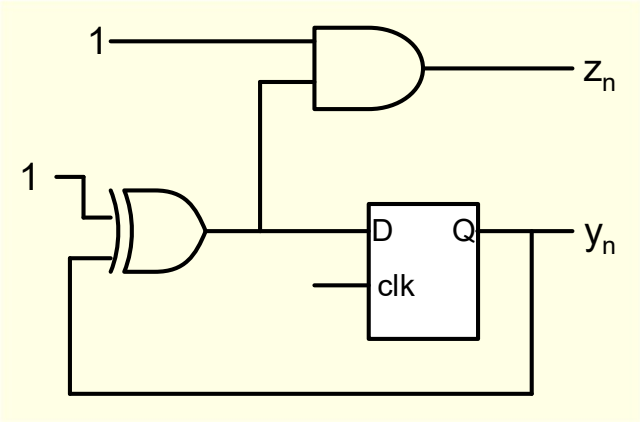


Example

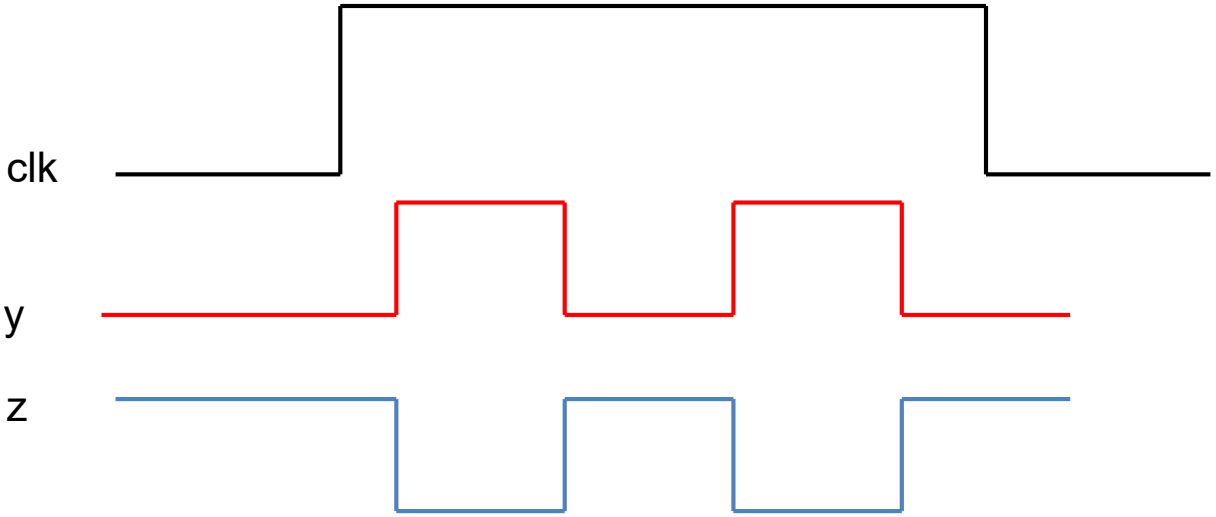




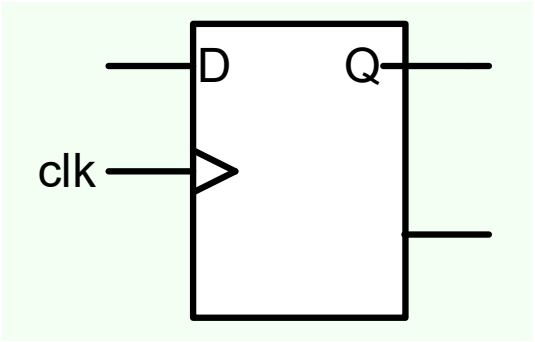
Problem with Latch

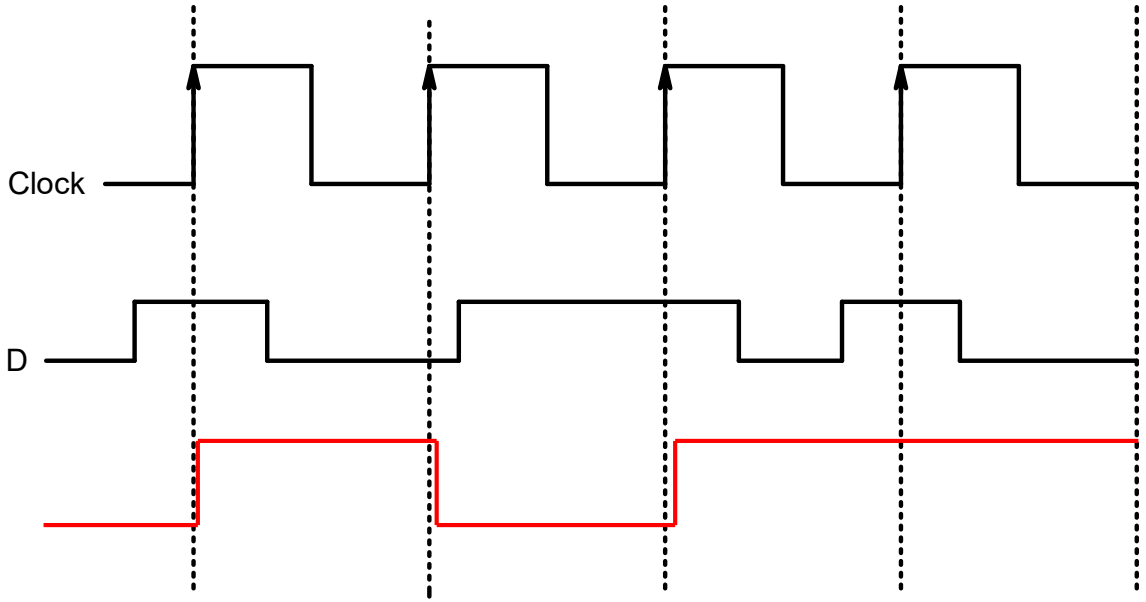


Circuits are designed with the idea there would be single change in output or memory state in single clock cycle.



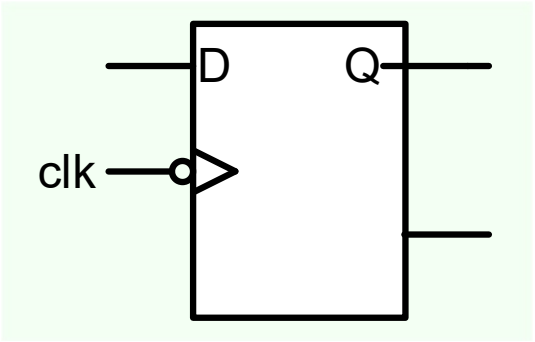
Edge Triggered Latch or Flip-flop

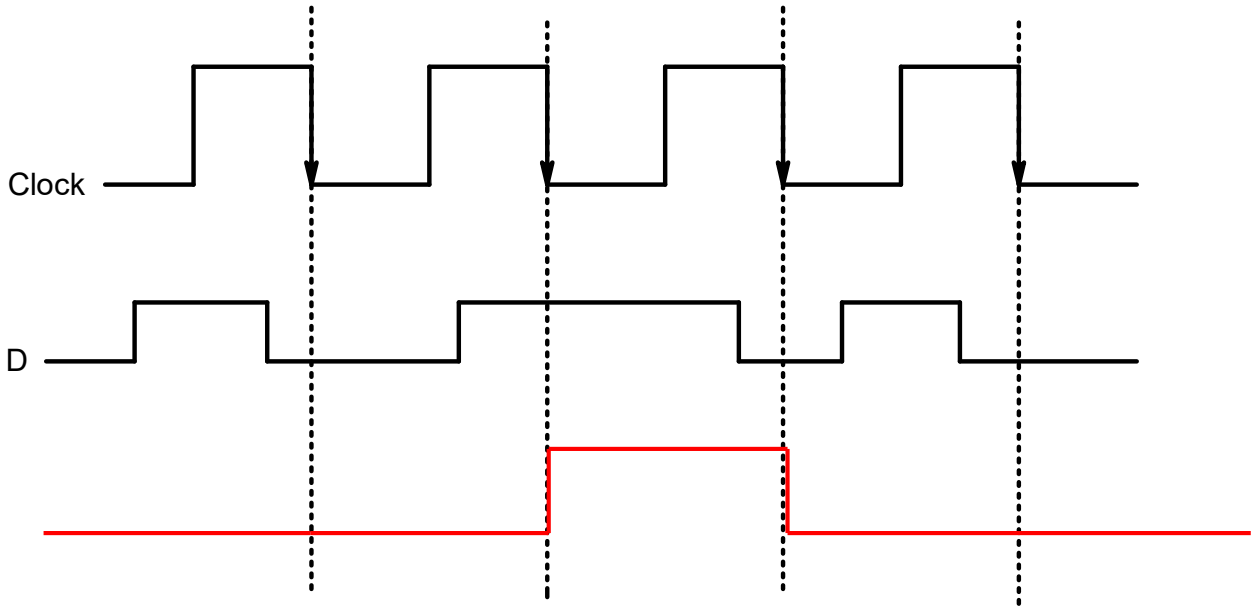




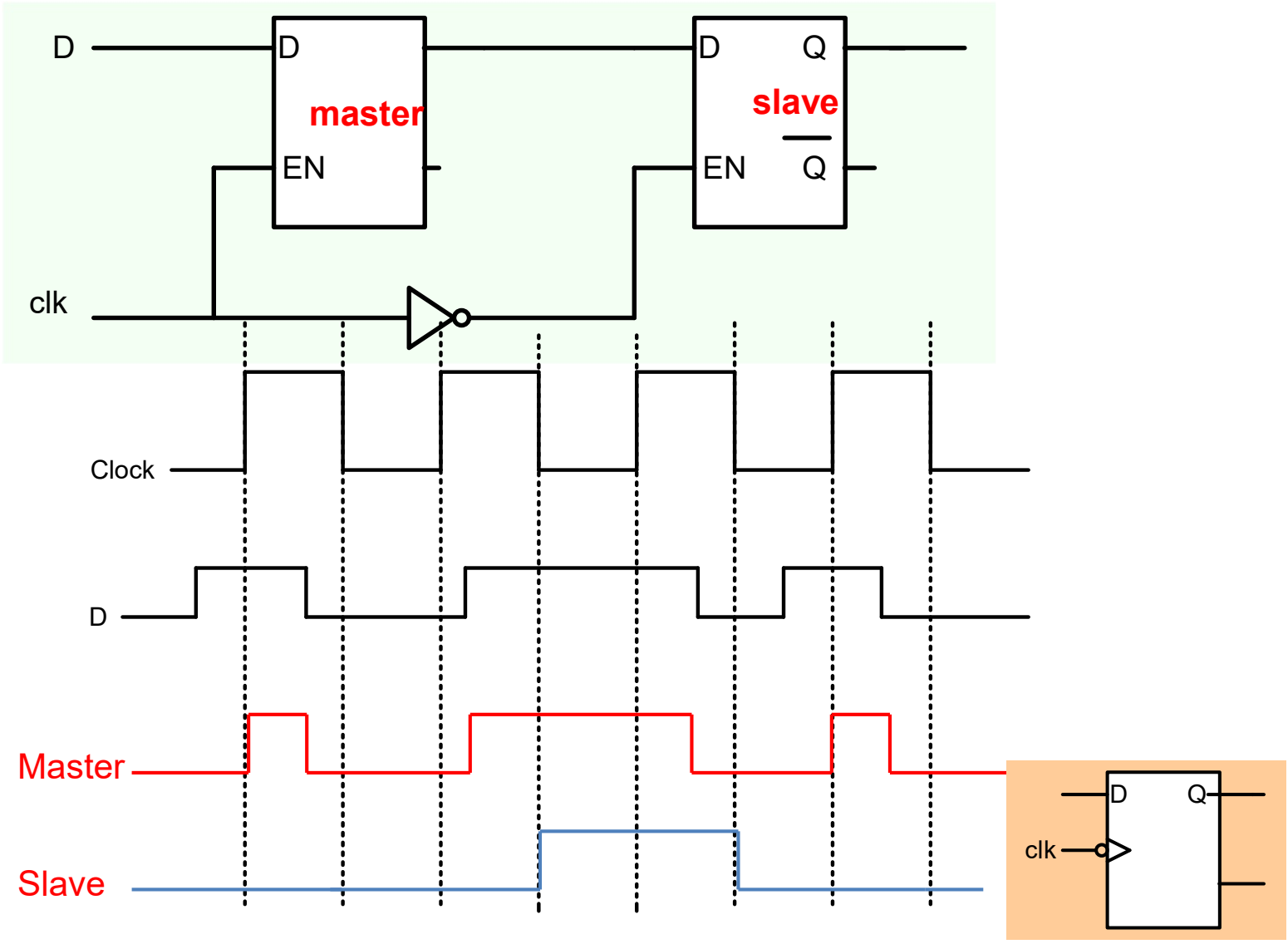
Positive edge triggered flipflop

Negative Edge Triggered Latch or Flip-flop

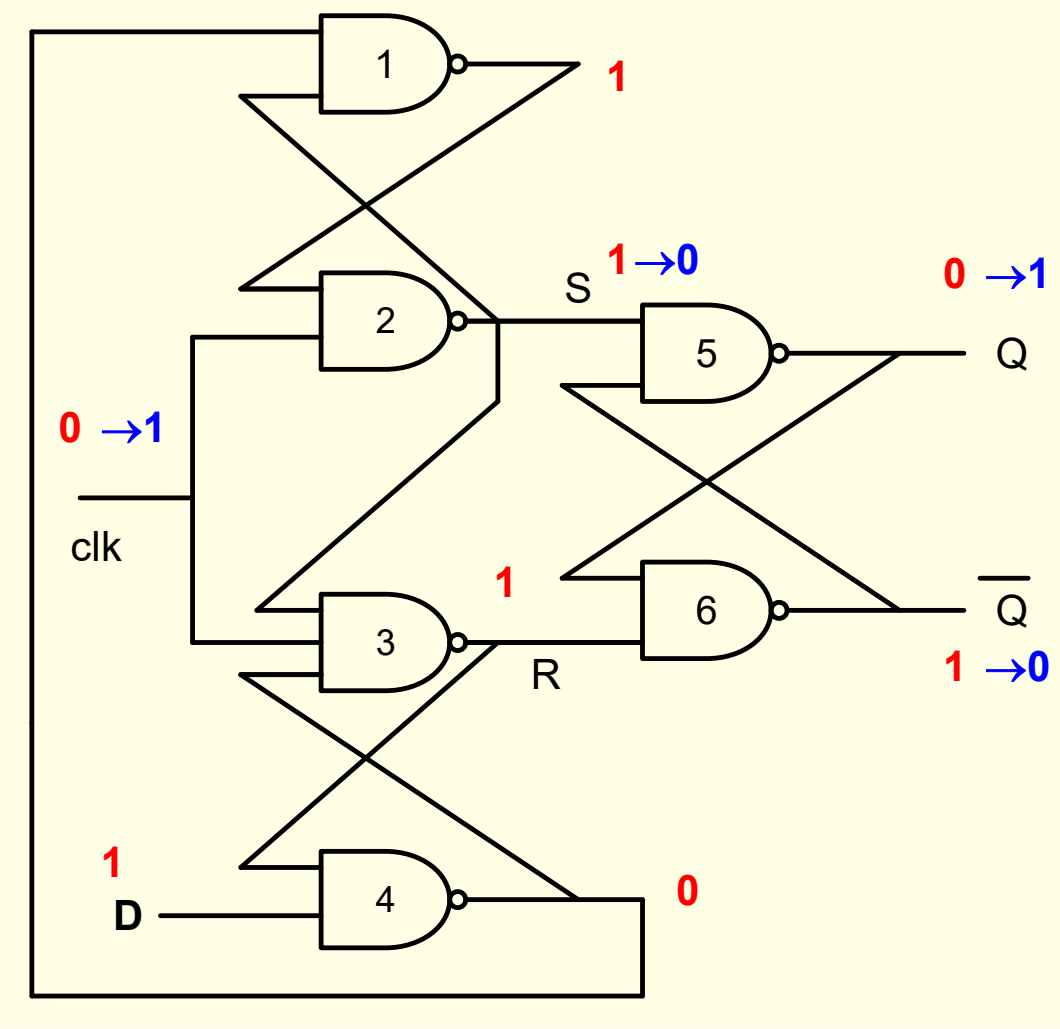




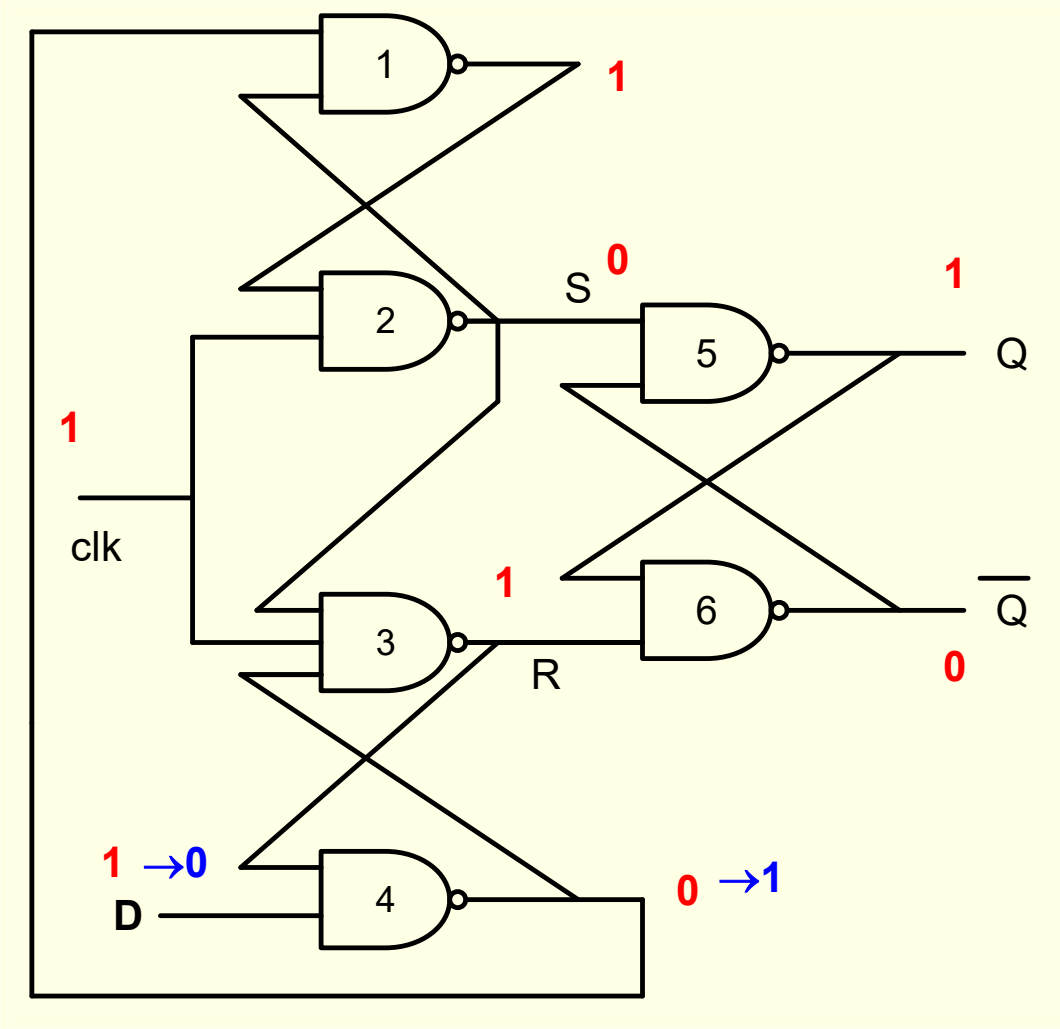
Master-Slave D Flip-flop



Positive edge triggered Flip-flop

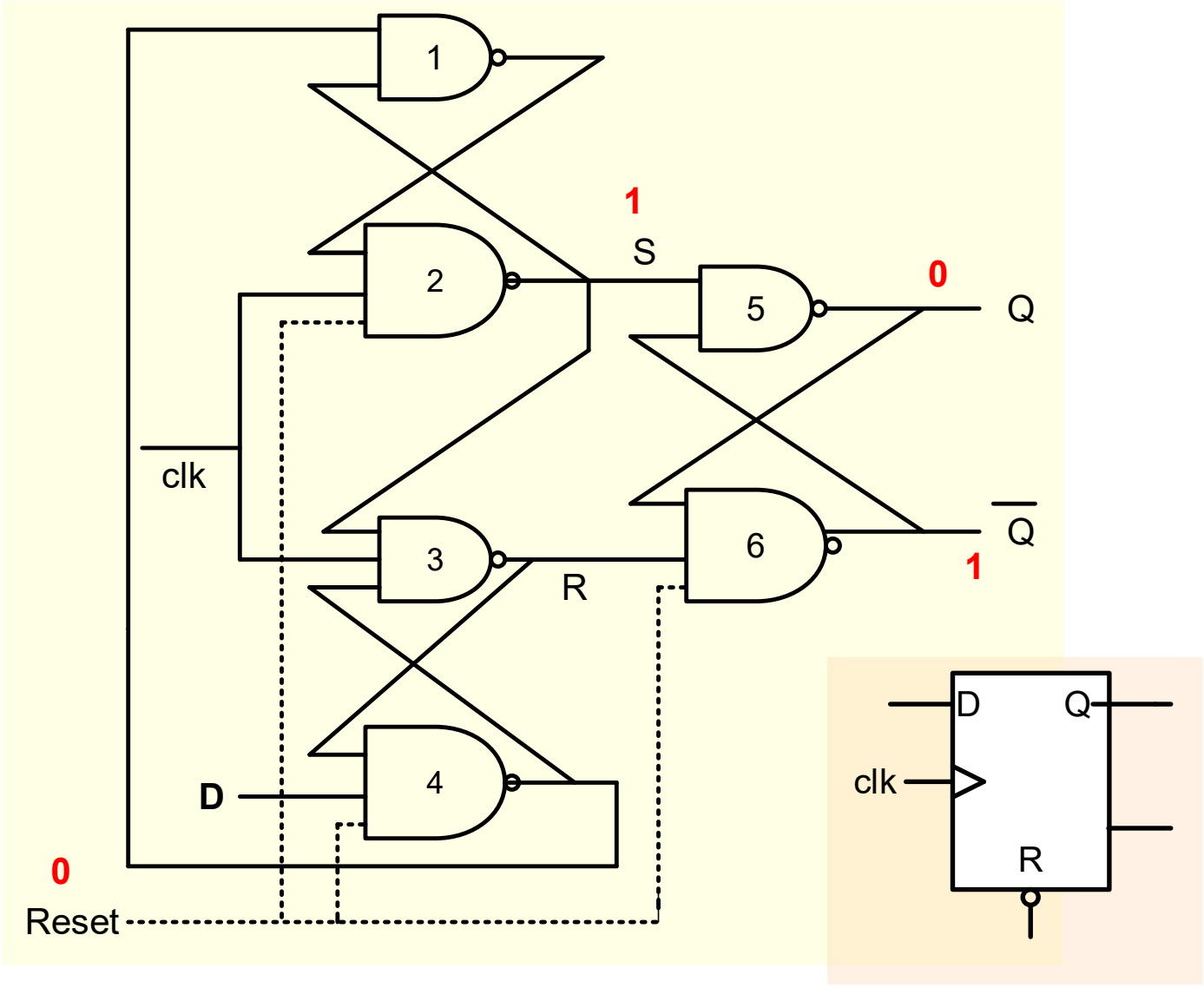


Positive edge triggered Flip-flop



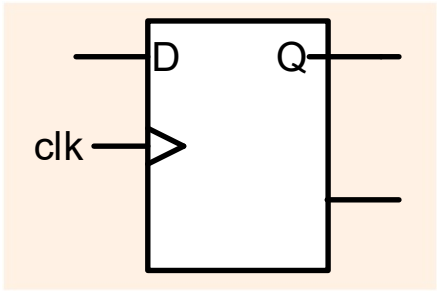
A change in input has no effect if it occurs after the clock edge

Positive edge Triggered Flip-flop with Asynchronous Reset



Characteristic table

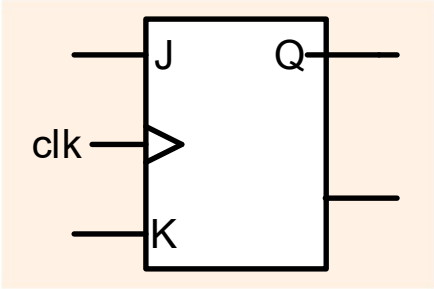
Given a input and the present state of the flip-flop, what is the next state of the flip-flop



Inputs (D)	Q(t+1)
0	0
1	1

$$Q(t+1) = D$$

JK Flip-flop

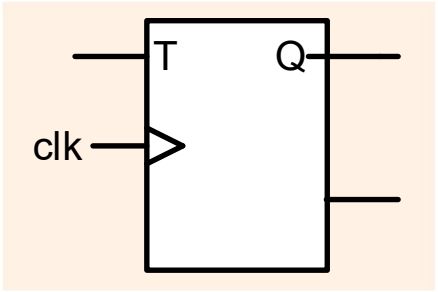


Inputs J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\overline{Q(t)}$

$$Q(t+1) = \overline{Q(t)}.J + Q(t).\overline{K}$$

→**Characteristic equation**

Toggle or T Flip-flop



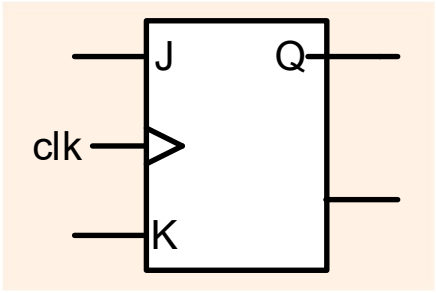
Inputs (T)	Q(t+1)
0	Q(t)
1	$\overline{Q(t)}$

$$Q(t+1) = \overline{Q(t)}.T + Q(t).\overline{T}$$

Excitation Table What inputs are required to effect a particular state change

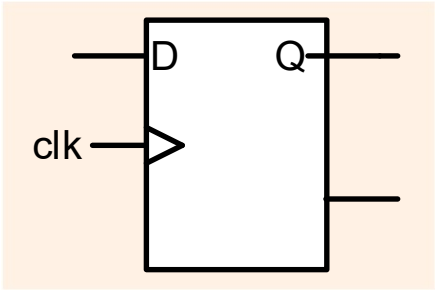
		Inputs
Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Excitation Table



J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\overline{Q(t)}$

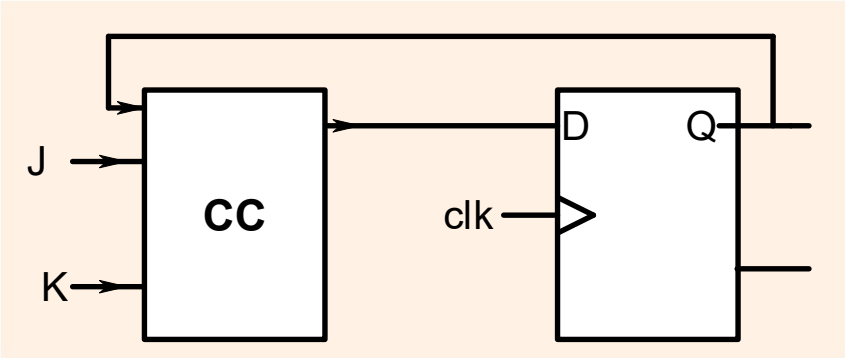
Q(t)	Q(t+1)	Inputs J	Inputs K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0



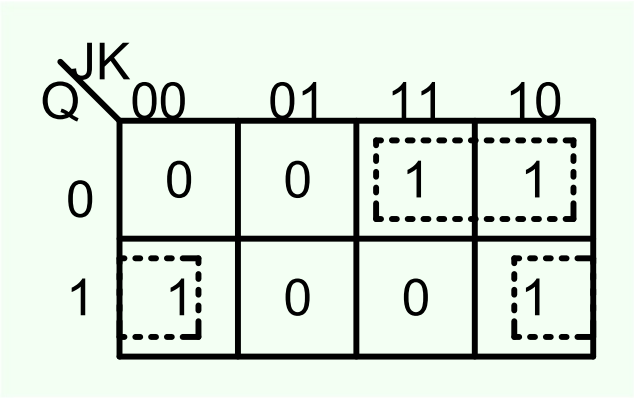
D	Q(t+1)
0	0
1	1

Q(t)	Q(t+1)	Inputs D
0	0	0
0	1	1
1	0	0
1	1	1

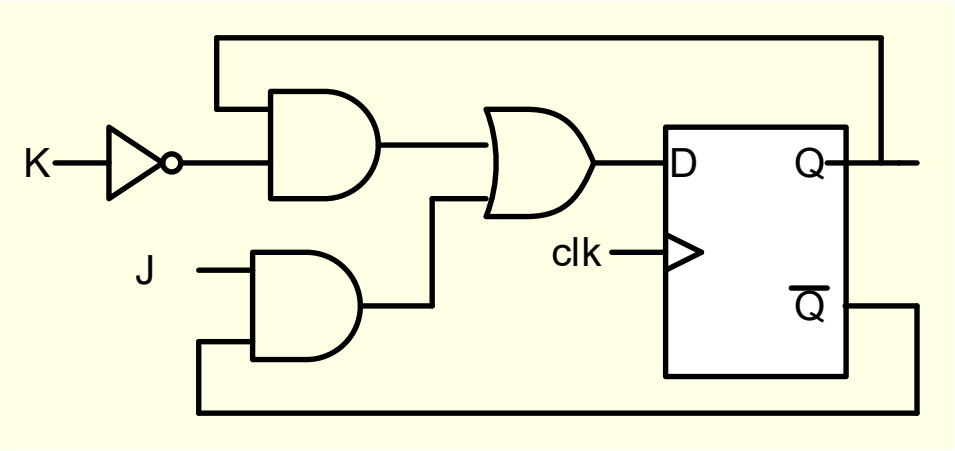
Convert a D FF to JK FF



J	K	Q(t+1)	D
0	0	Q(t)	**Q(t)**
0	1	0	**0**
1	0	1	**1**
1	1	$\overline{Q(t)}$	$\overline{\mathbf{Q(t)}}$



$$D = \overline{Q}.J + Q.\overline{K}$$



ESC201T : Introduction to Electronics

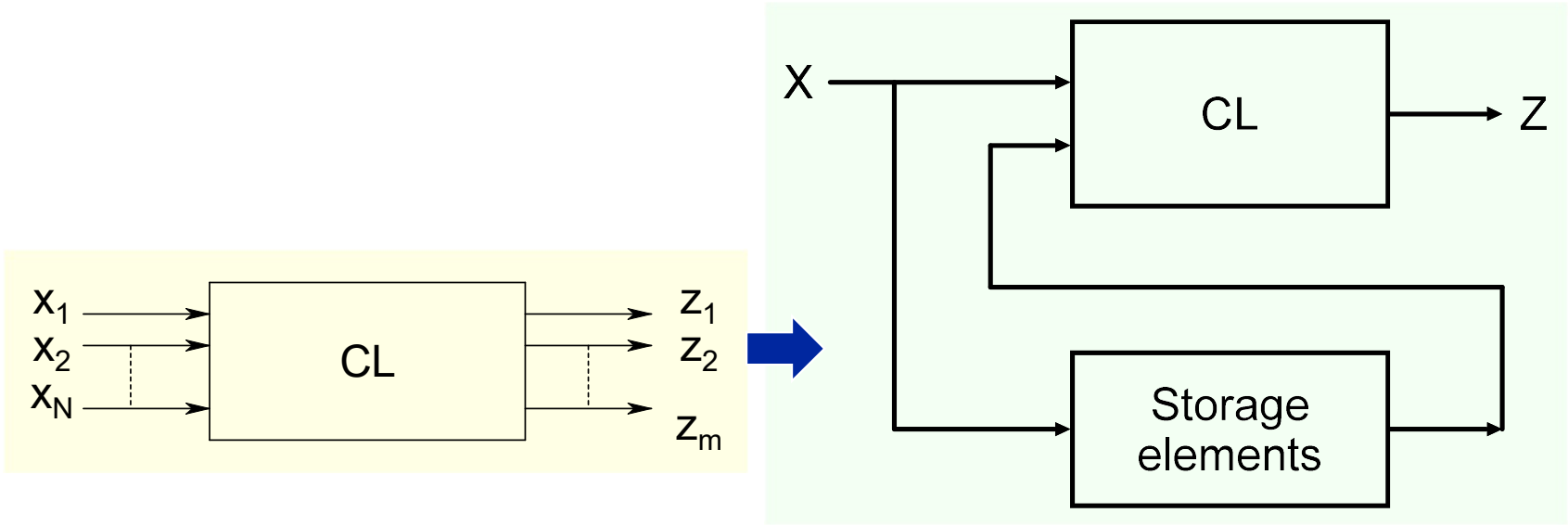
Lecture 38: Sequential circuit design-2

B. Mazhari
Dept. of EE, IIT Kanpur

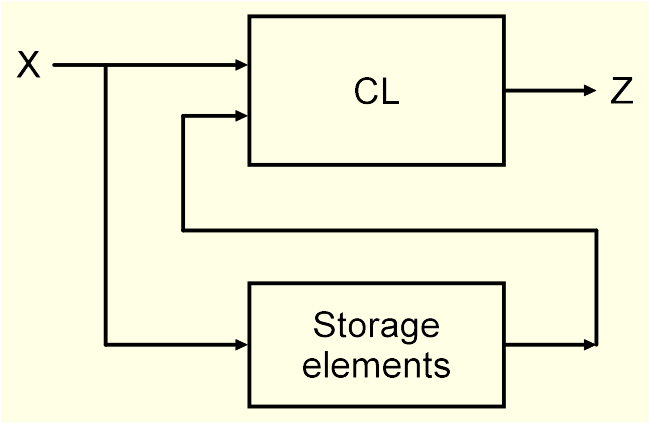
Limitations of Combinational logic:

decisions can only be based on the present value of inputs

- A more general purpose decision making machine should be able to make decisions based on past values of inputs as well.



Limitations:



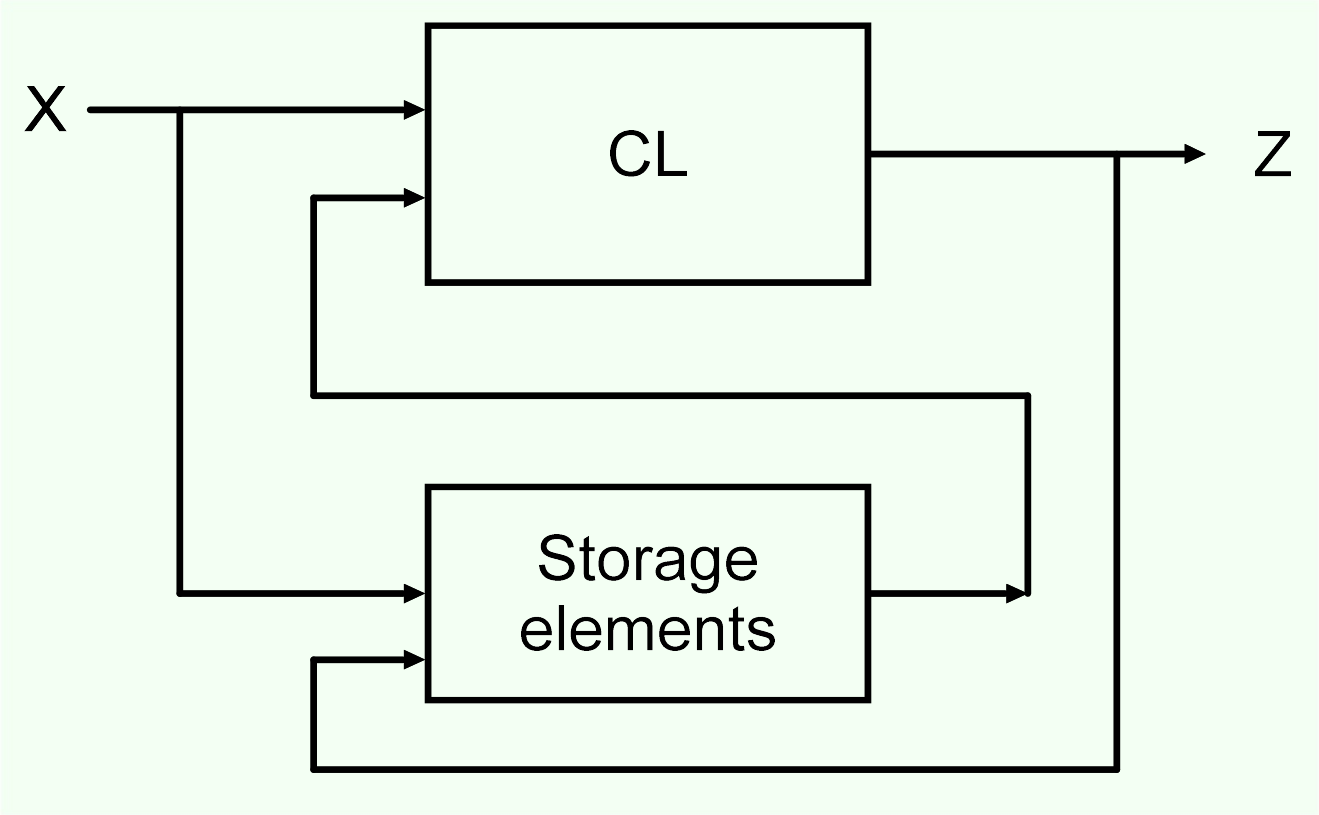
$z[n] = x[n] .OR. z[n-1]$

$z[n] = x[n] .OR. x[n-1] .OR. x[n-2]\ldots\ldots.x[0]$

Requires infinite memory !

$\Rightarrow$Make provision for storage of past values of Outputs as well

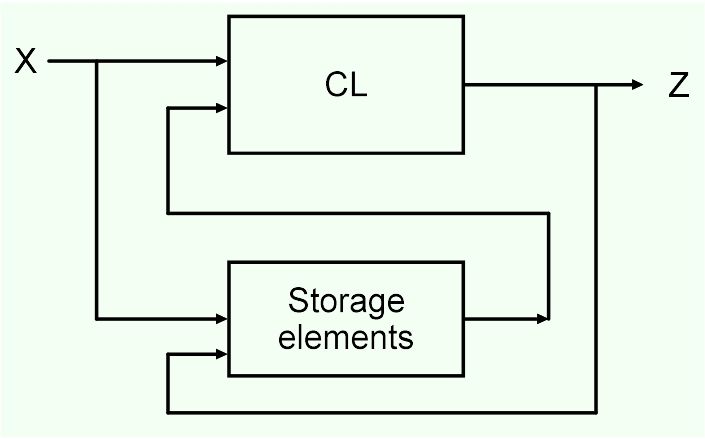
Improved System:



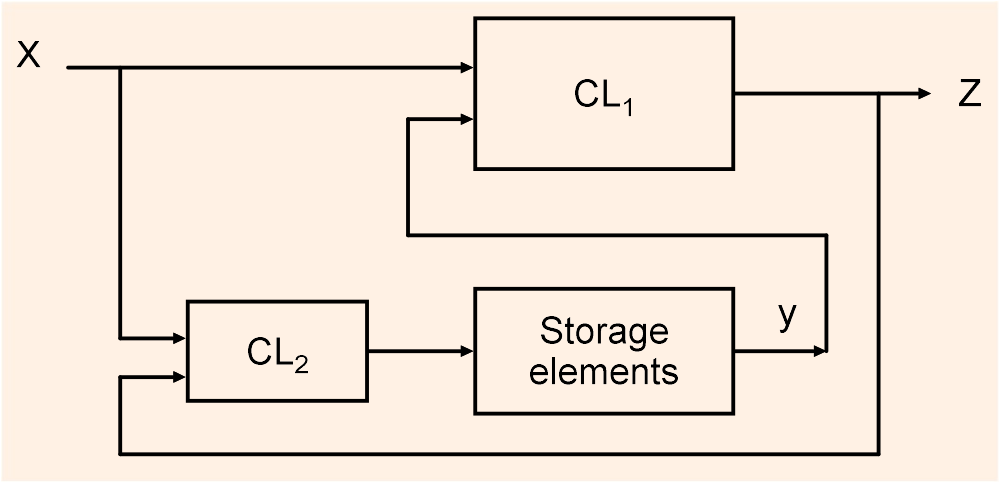
Limitations:

z[n] = x[n] .OR. (x[n-1] .AND. z[n-1])

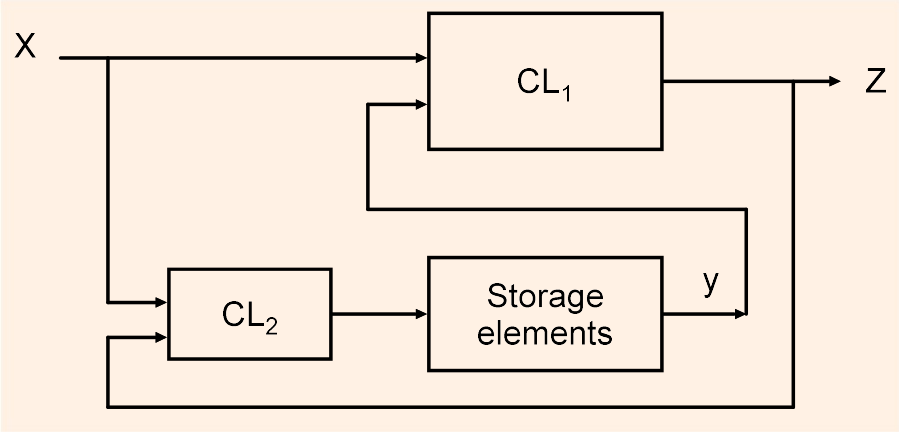
Requires two storage elements



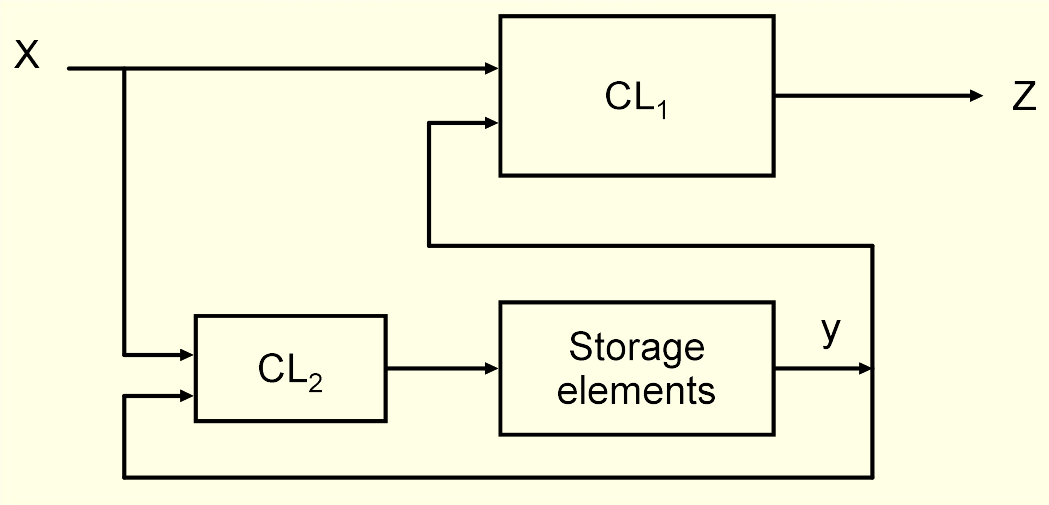
However, by defining a new variable
y[n] = x[n-1].AND. z[n-1] , we can use only one storage element.

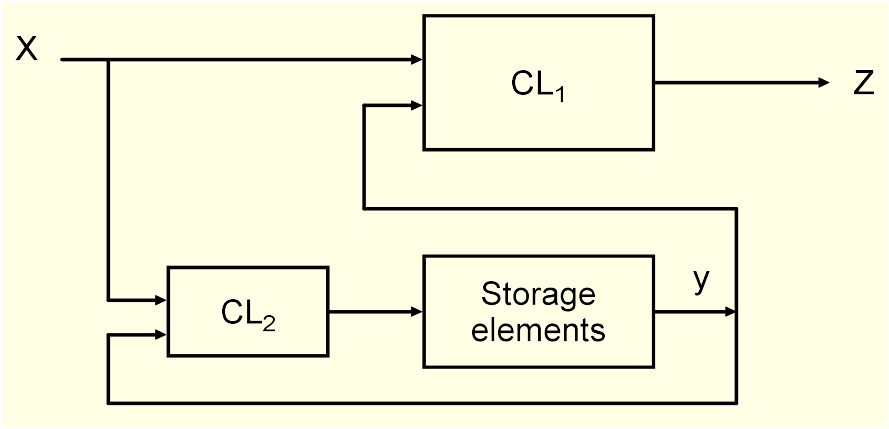


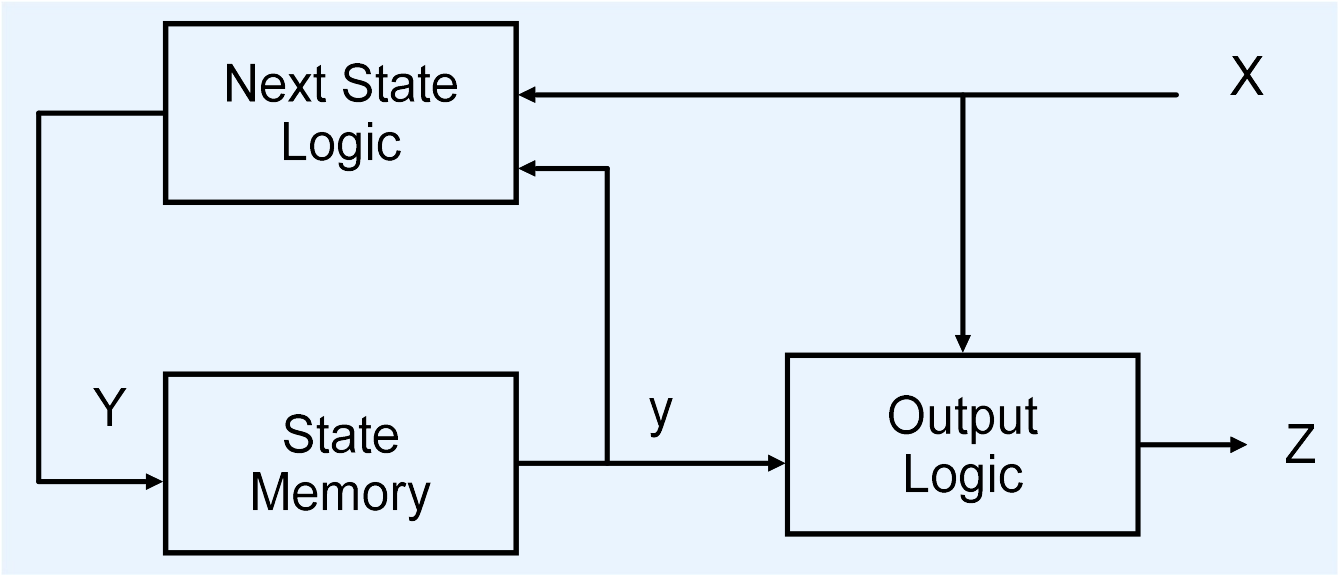
Improved Decision Making Machine:



Since Output Z is a function of Y and X

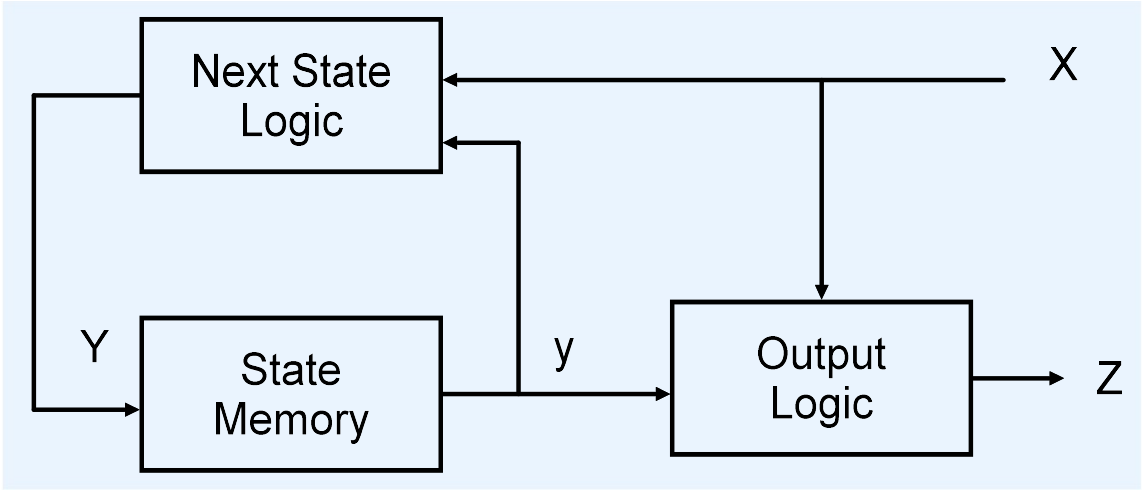


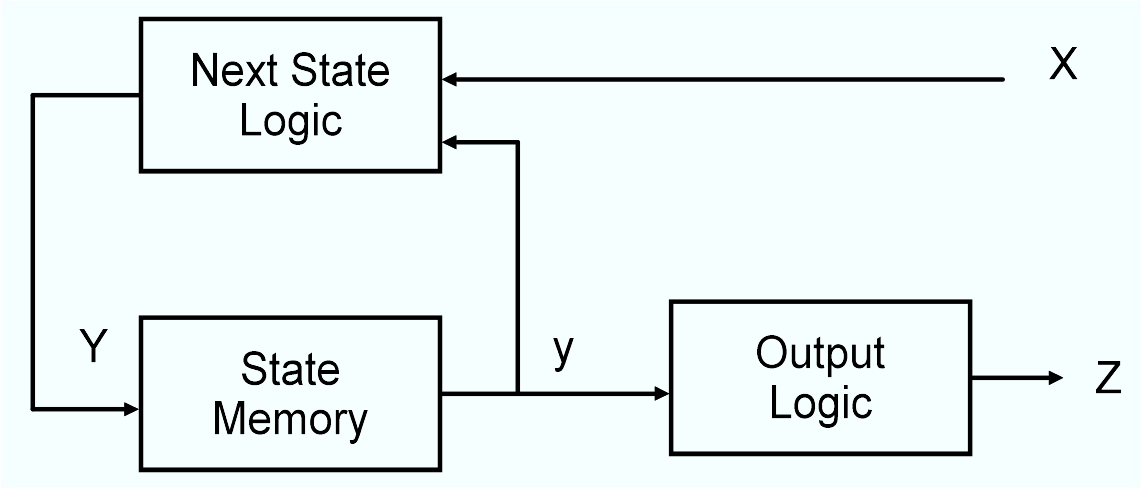




Mealy Sequential Machine

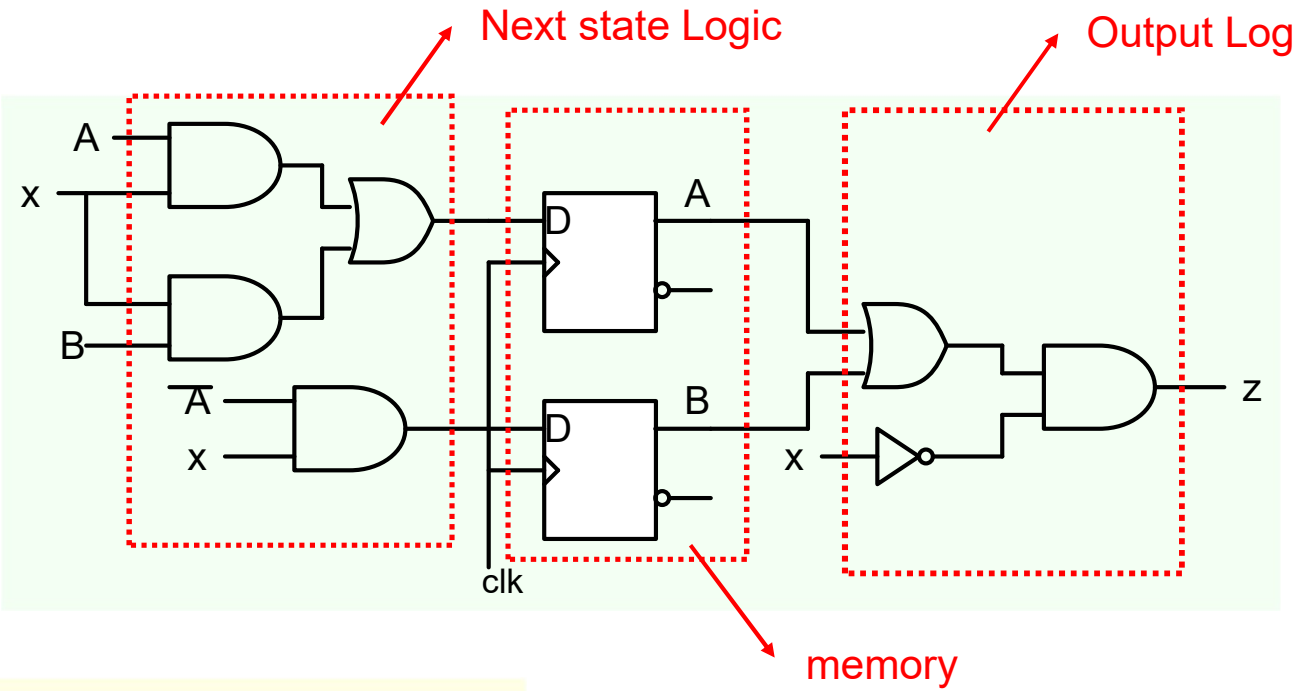
y: Present State
Y : Next State

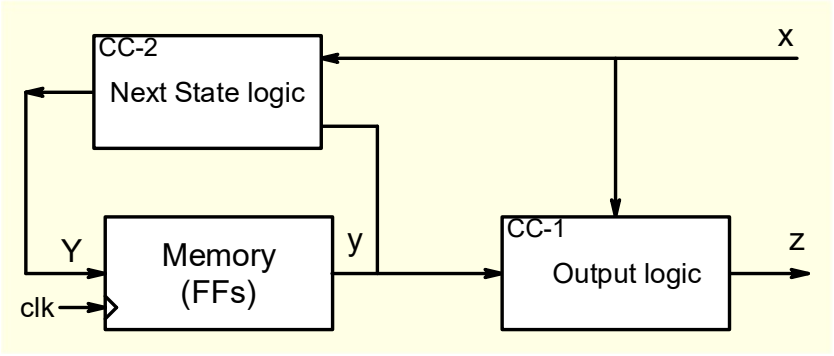




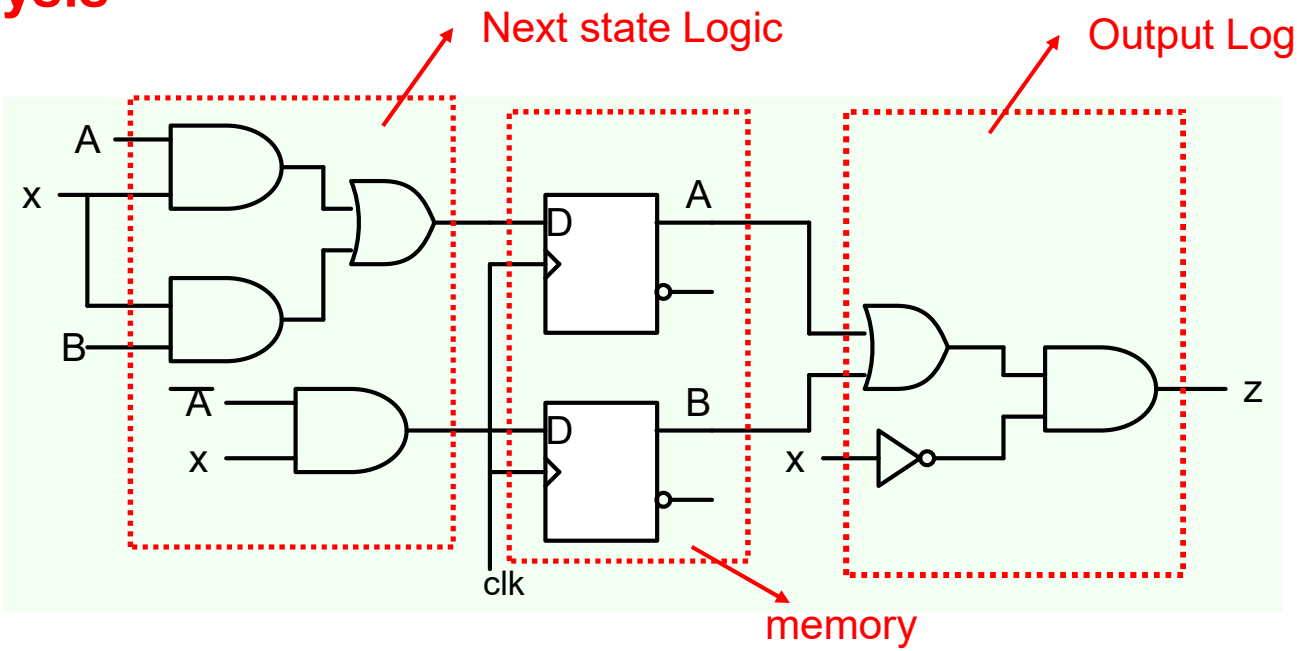
Moore Sequential Machine

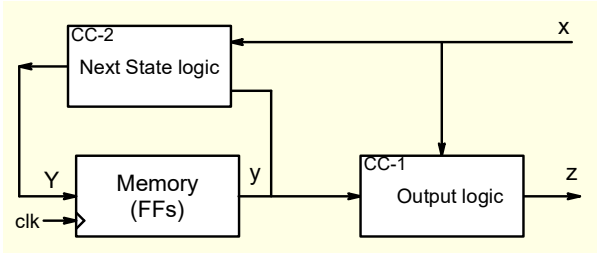
Example of a synchronous sequential circuit





Analysis





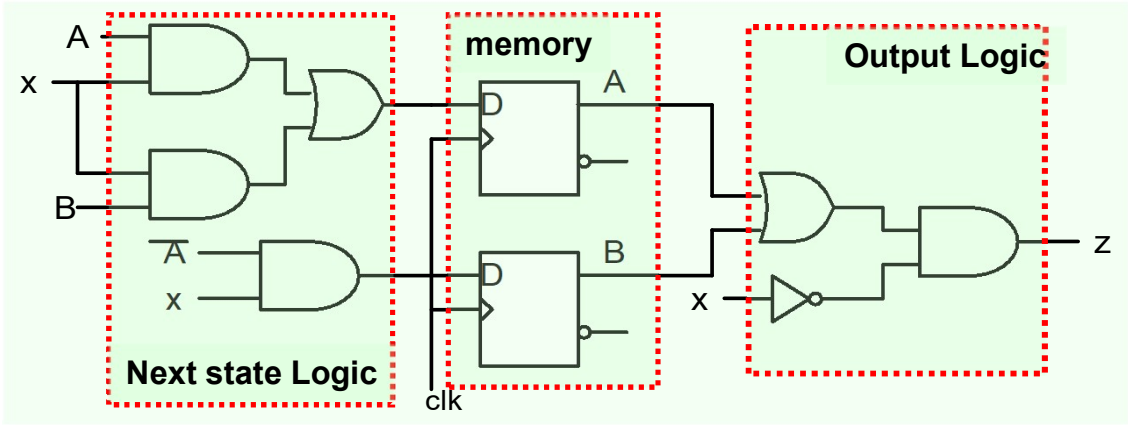
The dependence of output z on input x depends on the state of the memory (A,B)

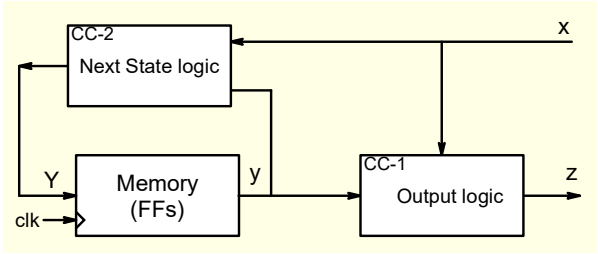
The memory has 2 FFs and each FF can be in state 0 or 1. Thus there are four possible states: AB: 00,01,10,11.

To describe the behavior of a sequential circuit, we need to show

1. how the system goes from one memory state to the next as the input changes
2. How the output responds to input in each state

Analysis of Sequential Circuits



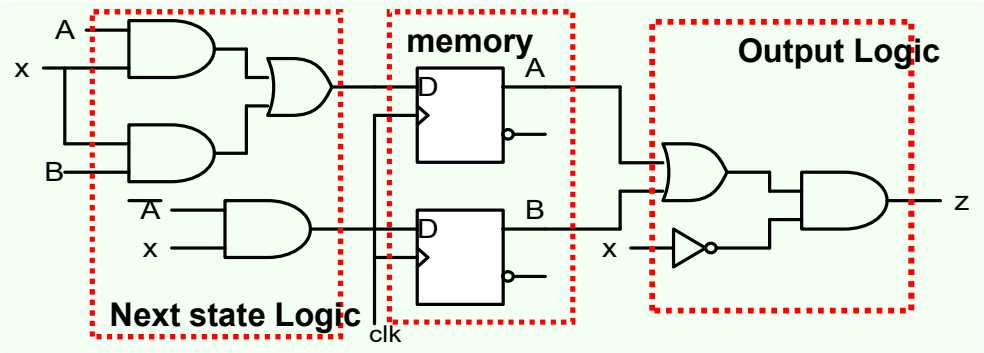


$$D_A = A.x + B.x \quad ; D_B = \overline{A}.x \; ; z = (A + B).\overline{x}$$

$$A(t+1) = A(t).x + B(t).x$$
$$B(t+1) = \overline{A(t)}.x$$
$$z = (A + B).\overline{x}$$

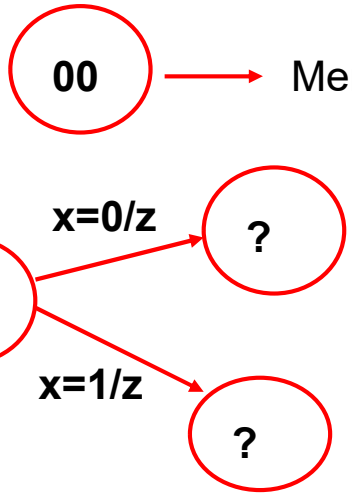
State Transition Table

Present State A	Present State B	Input x	Next State A	Next State B	Output z
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



State Transition Table

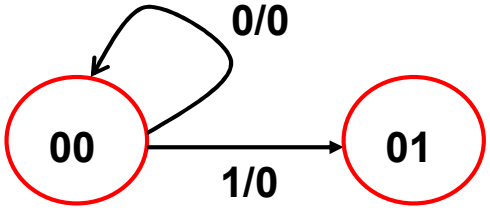
Present State A	Present State B	Input x	Next State A	Next State B	Output z
0	0	0	**0**	**0**	**0**
0	0	1	**0**	**1**	**0**
0	1	0	**0**	**0**	**1**
0	1	1	**1**	**1**	**0**
1	0	0	**0**	**0**	**1**
1	0	1	**1**	**0**	**0**
1	1	0	**0**	**0**	**1**
1	1	1	**1**	**0**	**0**



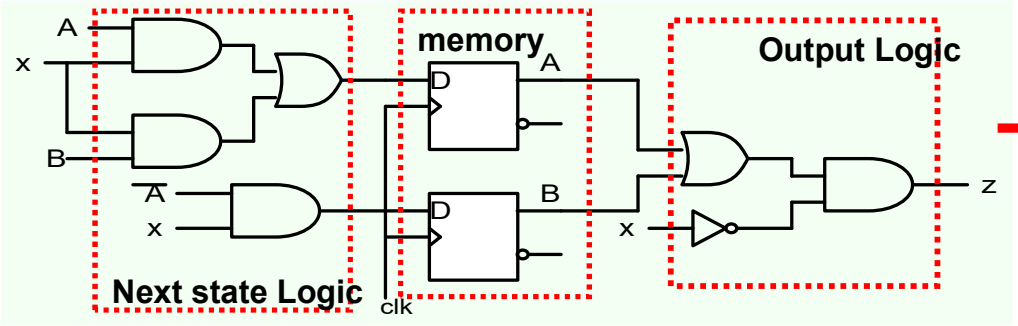
00 → Memory state in which FF A& B have output values 00

If x = 0 then z = 0, When the clock edge comes the system would stay in 00 state.

If x = 1 then z = 0. When the clock edge comes the system would go to 01 state.



Analysis of Sequential Circuits



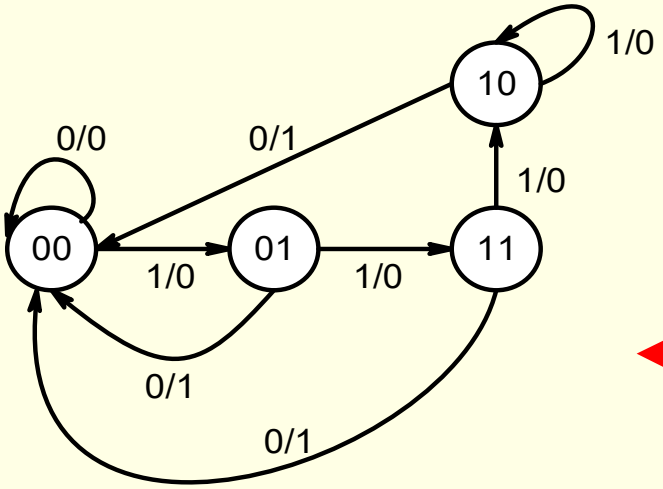
$$A(t+1) = A(t).x + B(t).x$$

$$B(t+1) = \overline{A(t)}.x$$

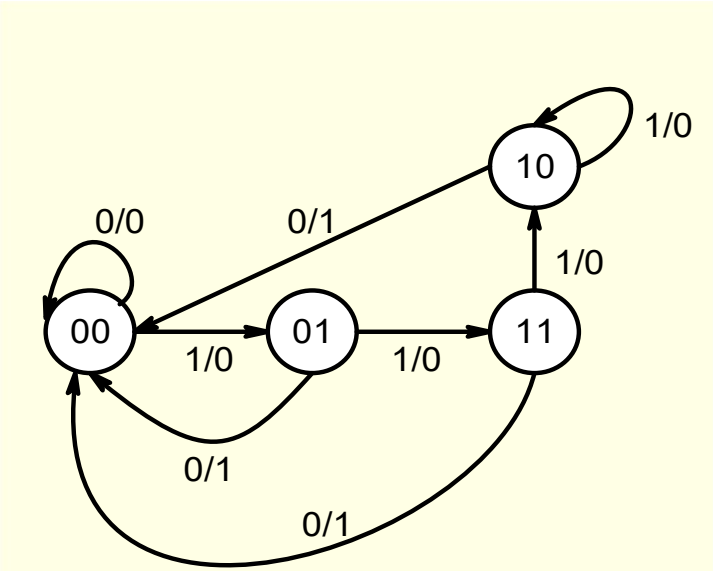
$$z = (A + B).\overline{x}$$

State Transition Table

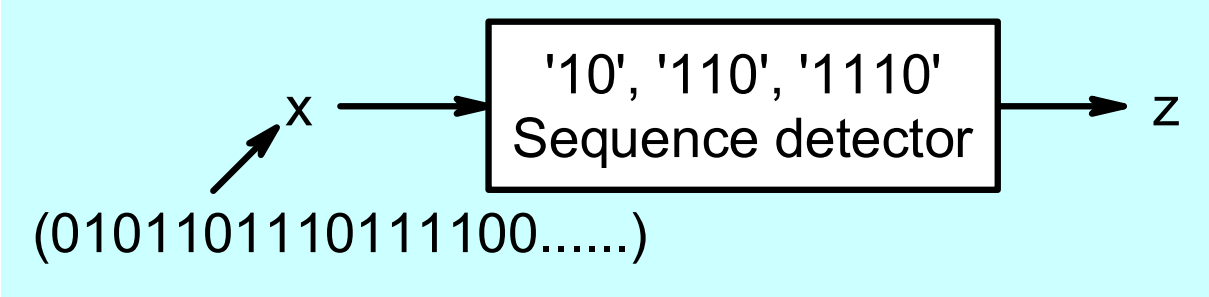
Present State A	Present State B	Input x	Next State A	Next State B	Output z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0



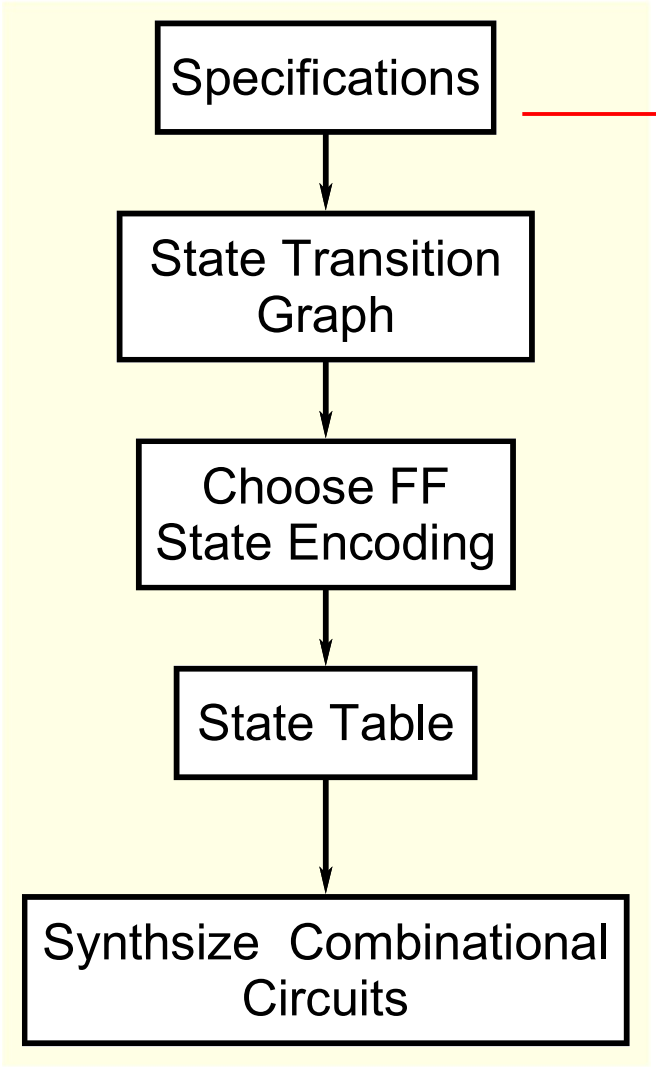
State transition Graph

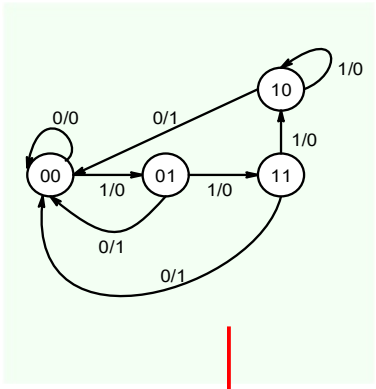


State transition Graph



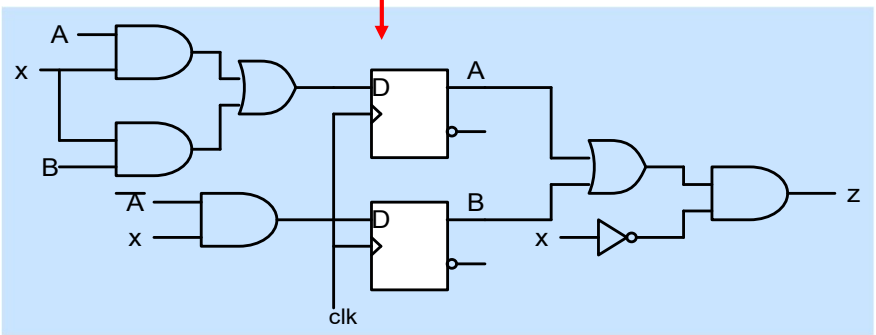
Design of Sequential Circuits



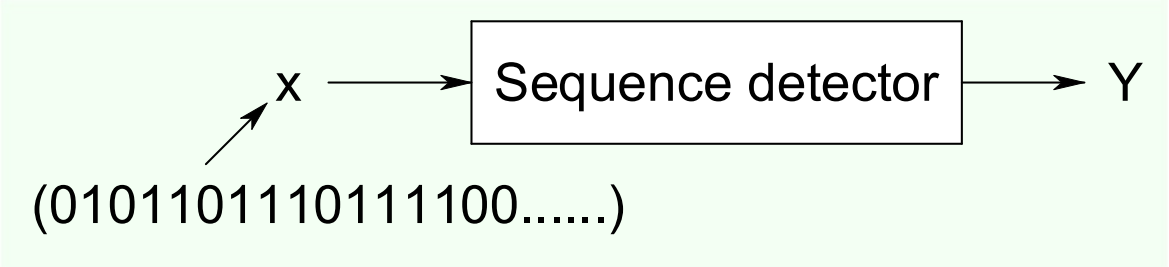


State Transition Table

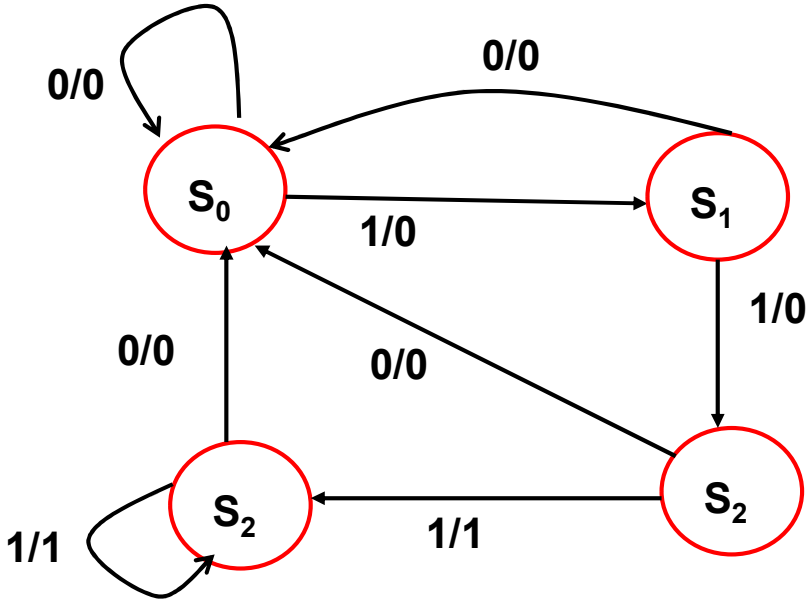
Present State A	Present State B	Input x	Next State A	Next State B	Output z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0



System specification to State Transition Graph

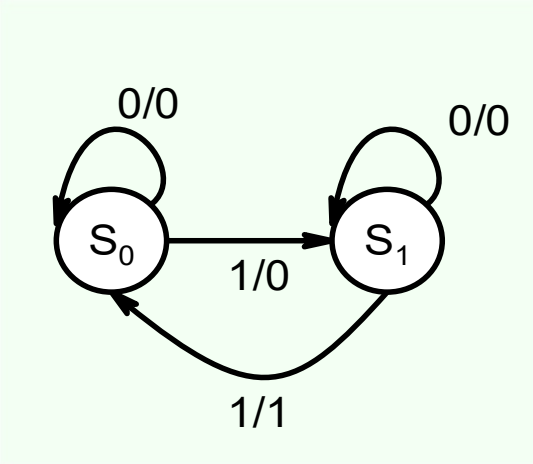


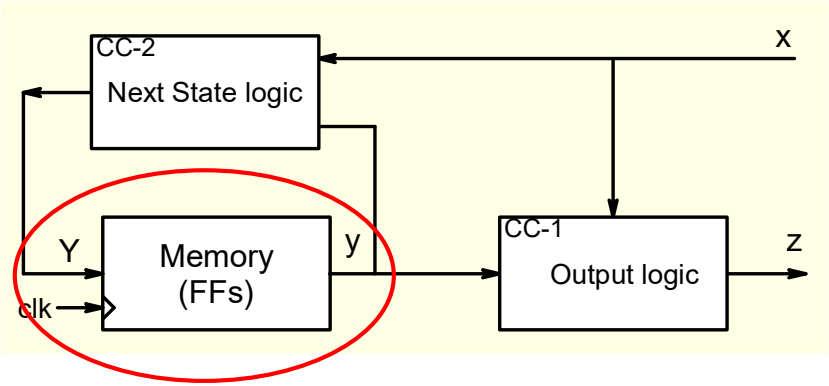
Detect 3 or more consecutive 1's in the input stream



Conversion of State transition graph to a circuit

Example-1



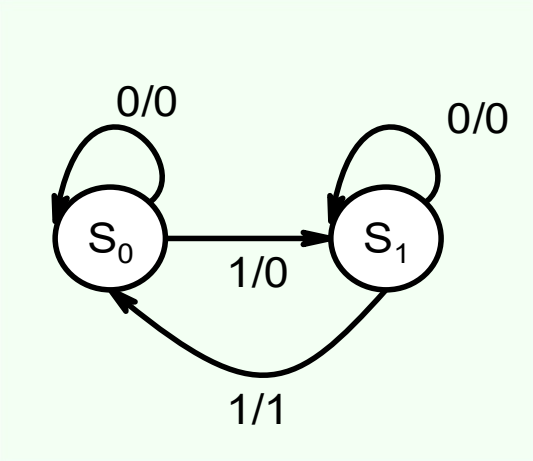


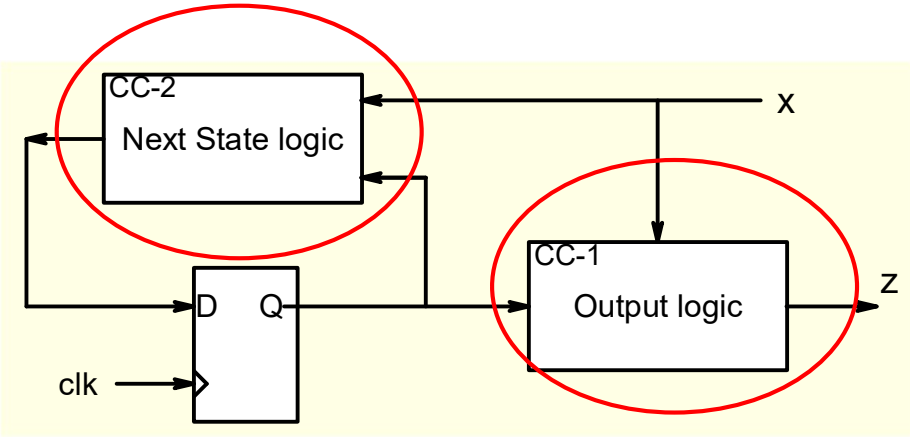
3 blocks need to be designed

1. How many FFs do we need? **N FFS can represent 2^N states so Minimum is 1**

2. Which FF do we choose? **Say D FF**

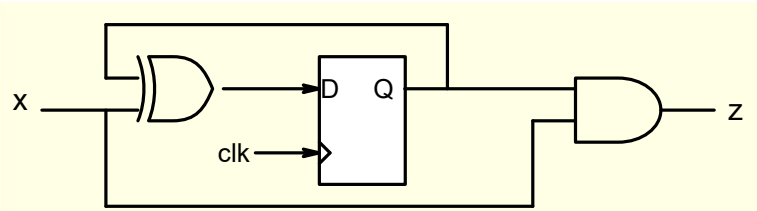
3. How are the states encoded? **Say FF output Q=0 represents S_0 and Q=1 represents S_1 state**





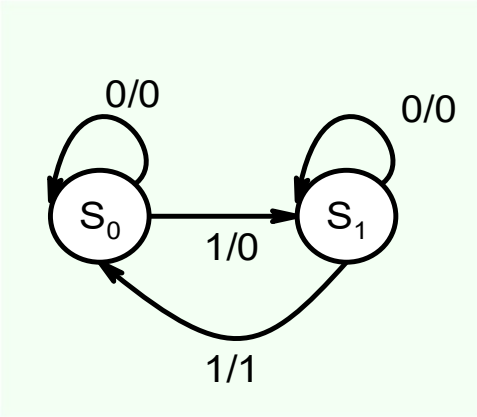
State Transition Table

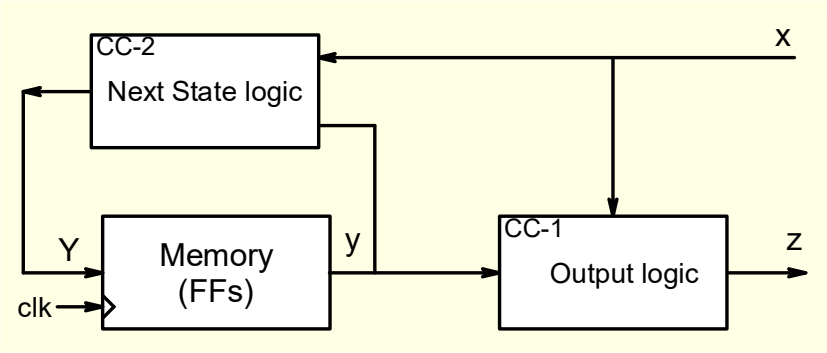
Present State Q(t)	Input x	Next State Q(t+1)	D	Output z
0	0	0	**0**	0
0	1	1	**1**	0
1	0	1	**1**	0
1	1	0	**0**	1



$$D = \overline{Q}.x + Q.\overline{x} \quad ; \quad z = Q.x$$

Example-2

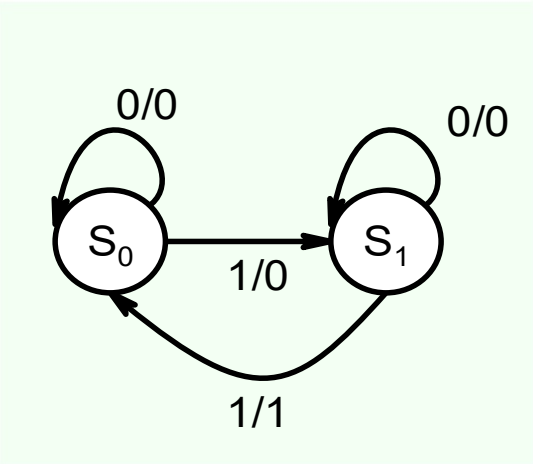


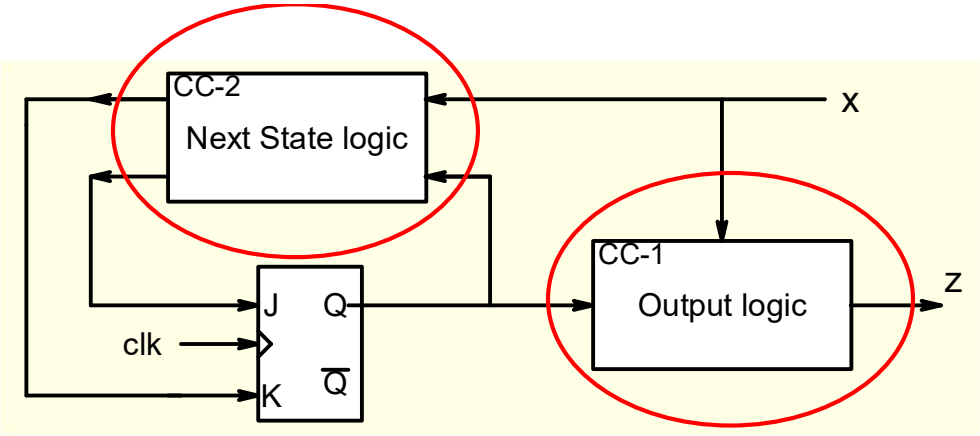


1. How many FFs do we need? **1**

2. Which FF do we choose? **Say JK FF**

3. How are the states encoded? **Say FF output Q=0 represents S_0 and Q=1 represents S_1 state**



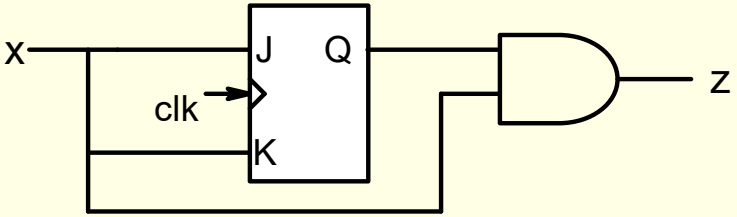


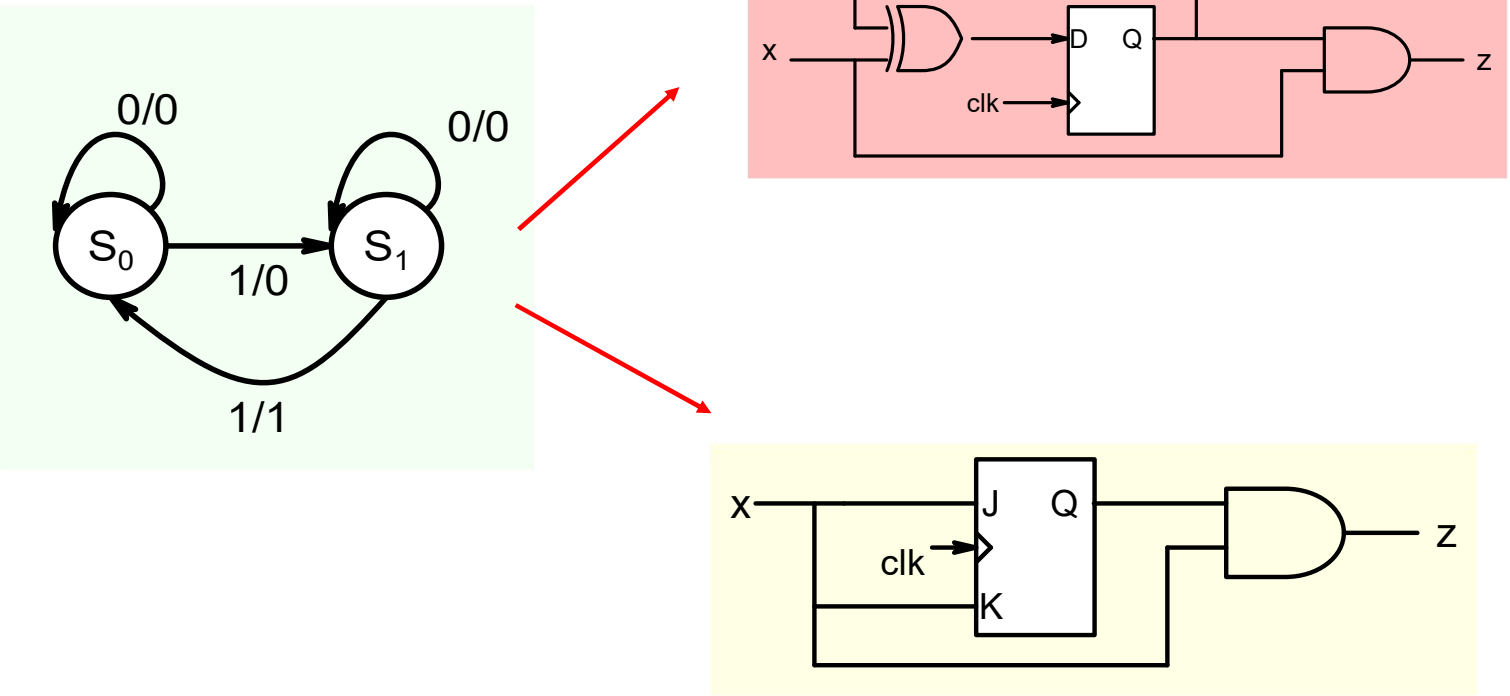
State Transition Table

Present State Q(t)	Input x	Next State Q(t+1)	J K	Output z
0	0	0	**0 X**	0
0	1	1	**1 X**	0
1	0	1	**X 0**	0
1	1	0	**X 1**	1

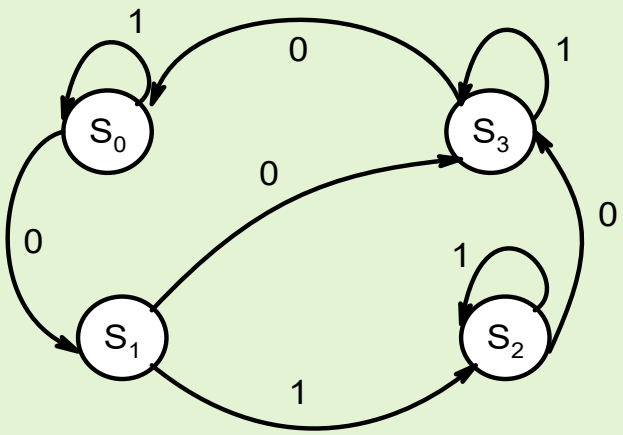
Q(t)	Q(t+1)	J K
0	0	0 X
0	1	1 X
1	0	X 1
1	1	X 0

$$J = x \;; K = x; \;\; z = Q.x$$

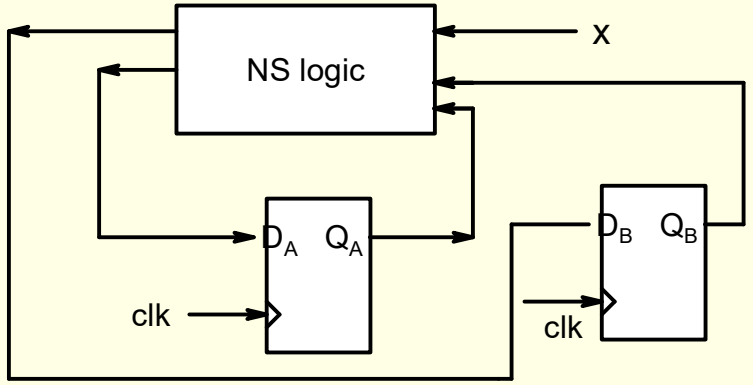



0/0
0/0
S_0
S_1
1/0
1/1
x
D
Q
clk
z
x
J
Q
clk
K
z

Example-3



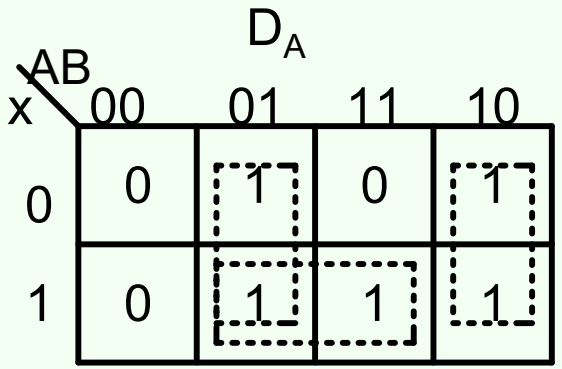
For 4 states a minimum of two FFs will be required. Let us choose 2 D FFs A &B



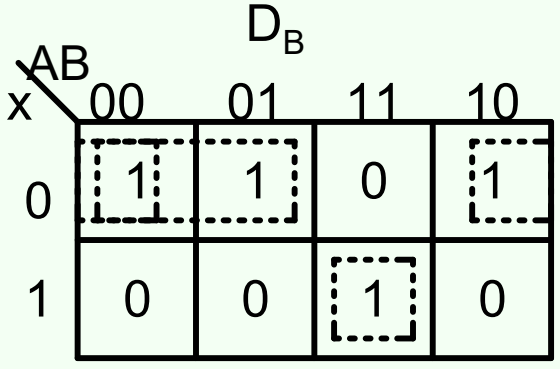
State	FF O/P A	B
S_0	0	0
S_1	0	1
S_2	1	0
S_3	1	1

Present State A	B	Input x	Next State A	B	D_A	D_B
0	0	0	0	1	0	1
0	0	1	0	0	0	0
0	1	0	1	1	1	1
0	1	1	1	0	1	0
1	0	0	1	1	1	1
1	0	1	1	0	1	0
1	1	0	0	0	0	0
1	1	1	1	1	1	1

Present State		Input	Next State			
A	B	x	A	B	D_A	D_B
0	0	0	0	1	0	1
0	0	1	0	0	0	0
0	1	0	1	1	1	1
0	1	1	1	0	1	0
1	0	0	1	1	1	1
1	0	1	1	0	1	0
1	1	0	0	0	0	0
1	1	1	1	1	1	1



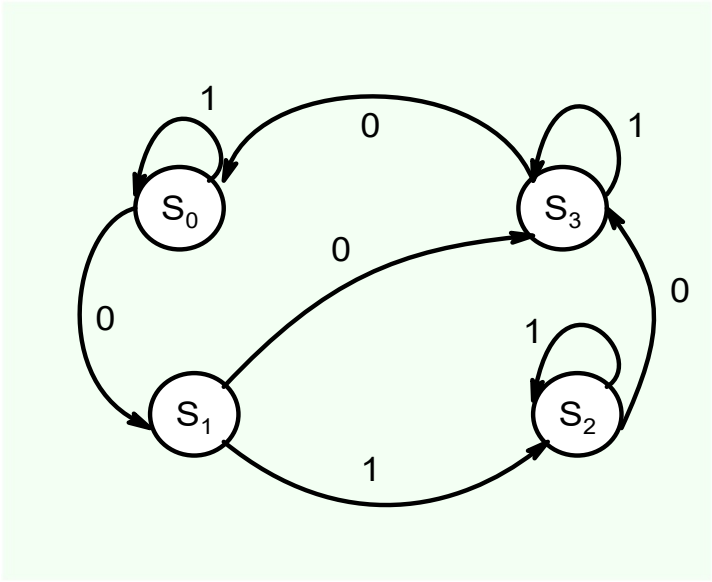
$$D_A = \overline{A}B + xB + A\overline{B}$$
$$= A \oplus B + x.B$$

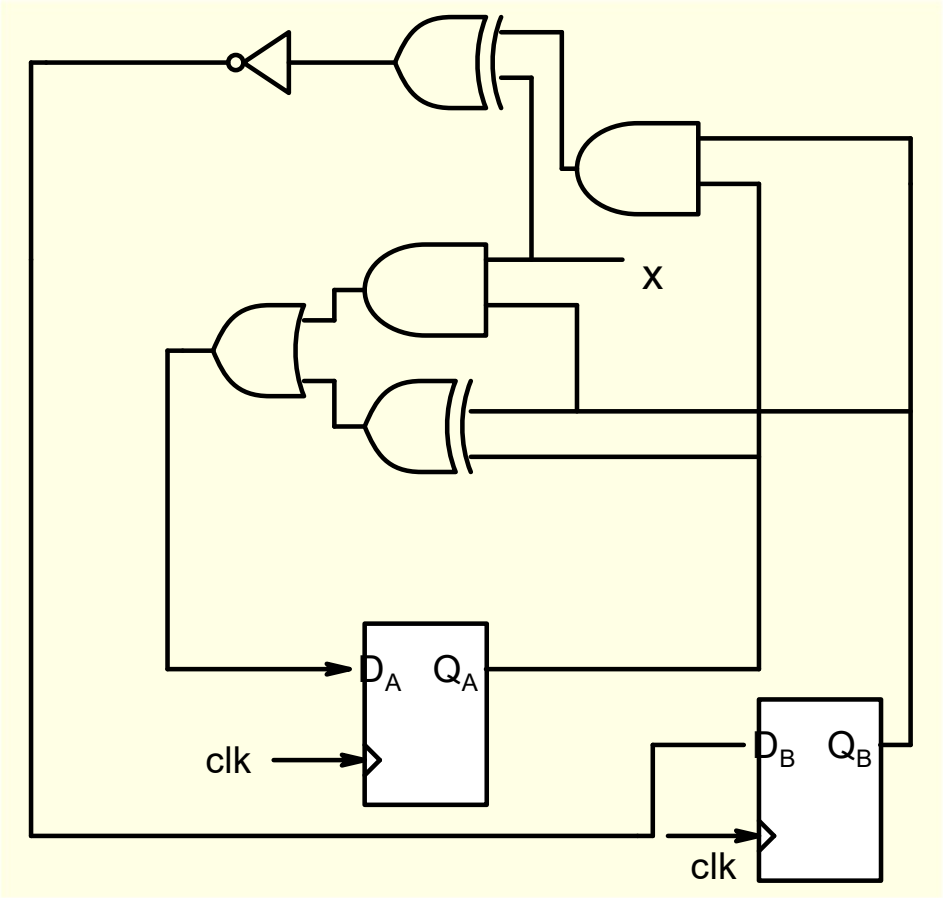


$$D_B = \overline{x}.\overline{A} + \overline{x}.\overline{B} + x.A.B$$
$$= \overline{x}.(\overline{A} + \overline{B}) + x.A.B$$
$$= \overline{x}.\overline{AB} + x.AB = \overline{x \oplus AB}$$

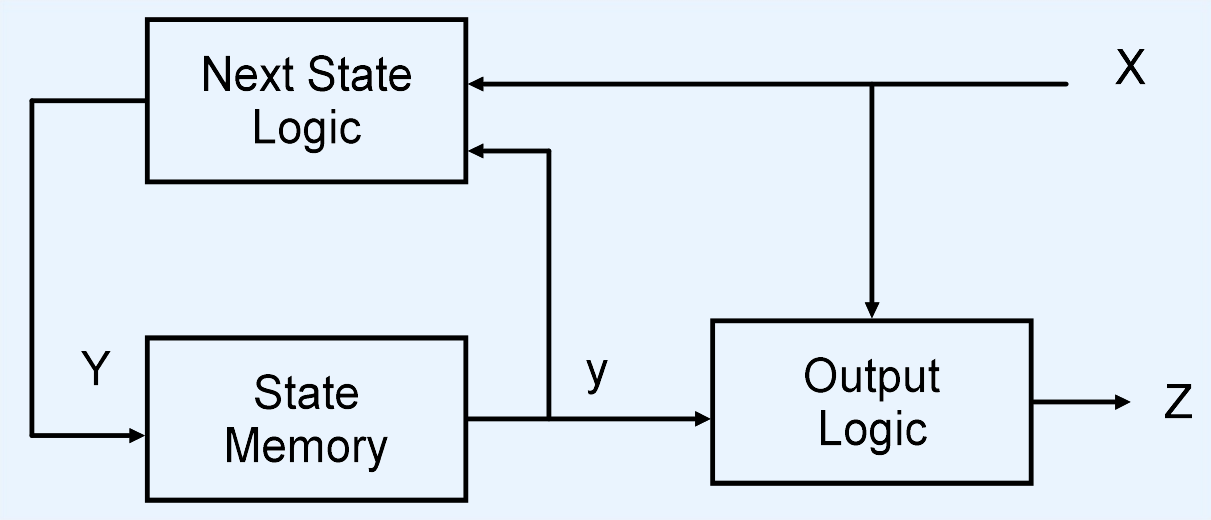
$$D_A = A \oplus B + x.B$$

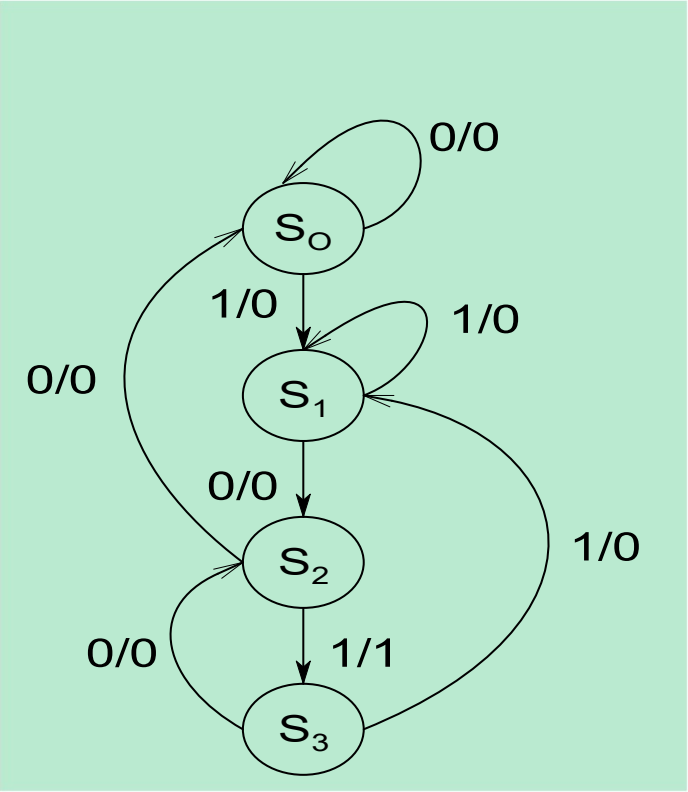
$$D_B = \overline{x \oplus AB}$$





Sequential Circuits: Summary

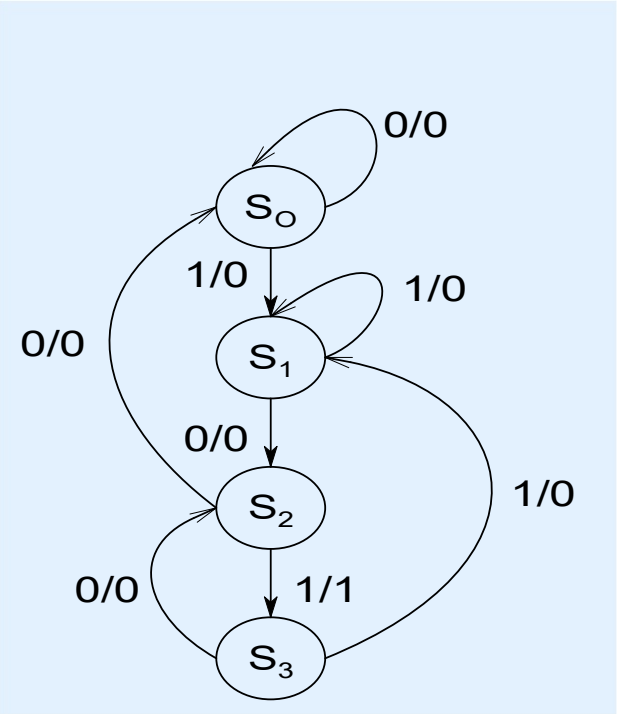


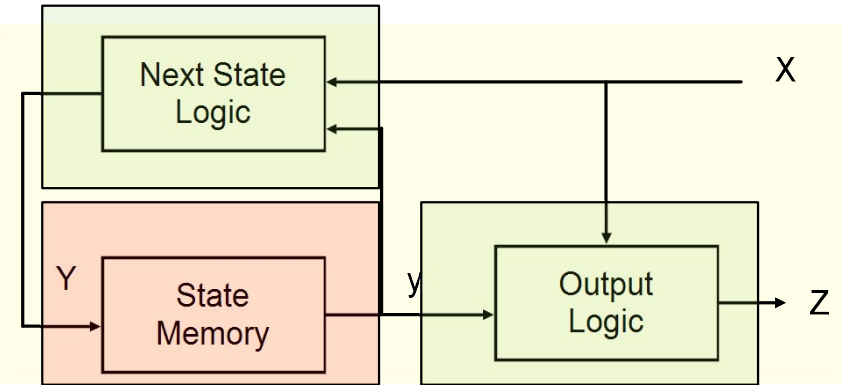


- we need to know how the system goes from one state to another in response to the inputs and how the outputs respond to these changes

Synthesis involves the following tasks:

- Number of storage elements
- State Encoding
- Choosing a flipflop type to implement states
- Synthesize next state logic
- Synthesize output logic



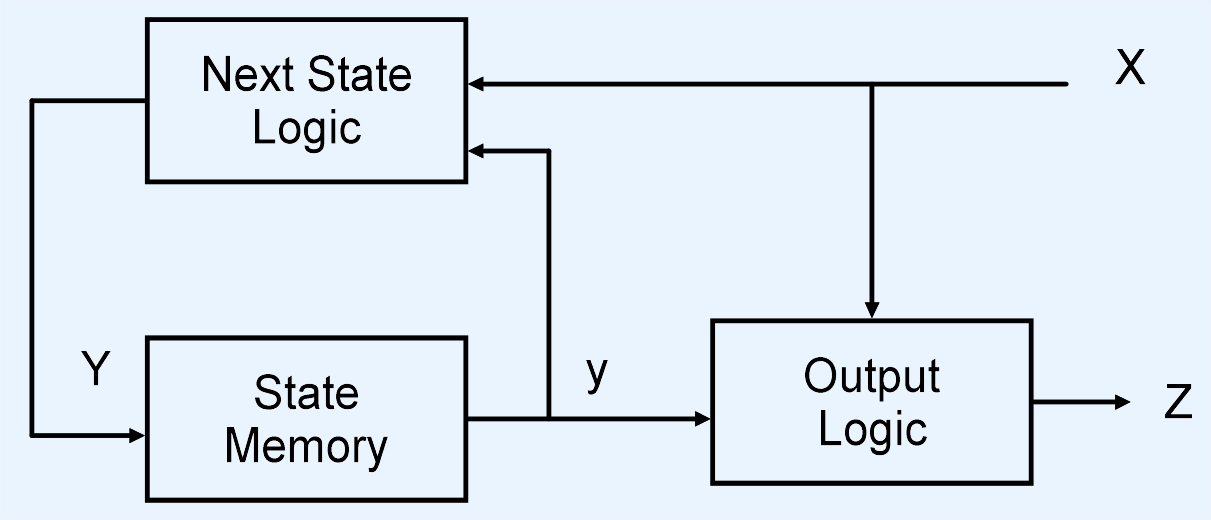


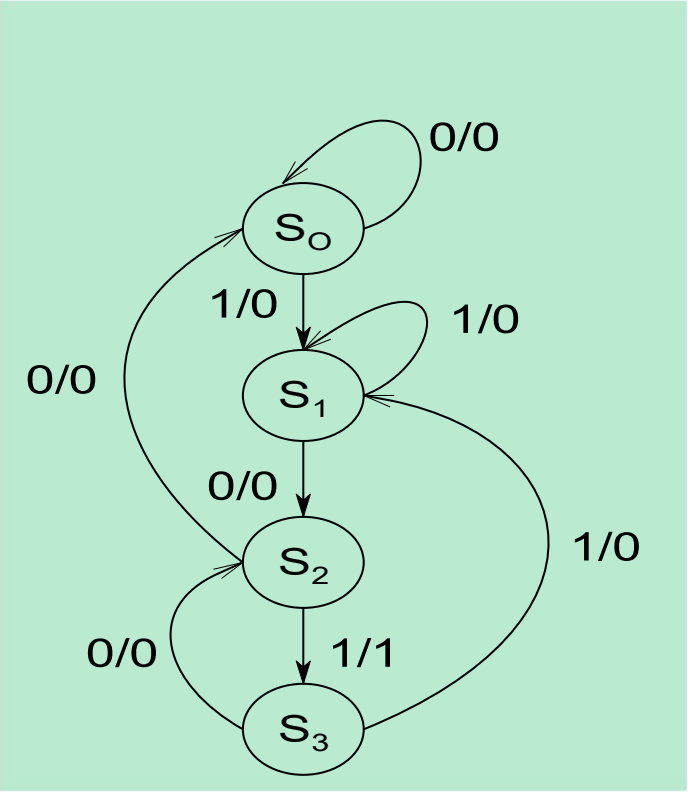
ESC201T : Introduction to Electronics

Lecture 39: Sequential circuit design-3

B. Mazhari
Dept. of EE, IIT Kanpur

Sequential Circuits: Summary

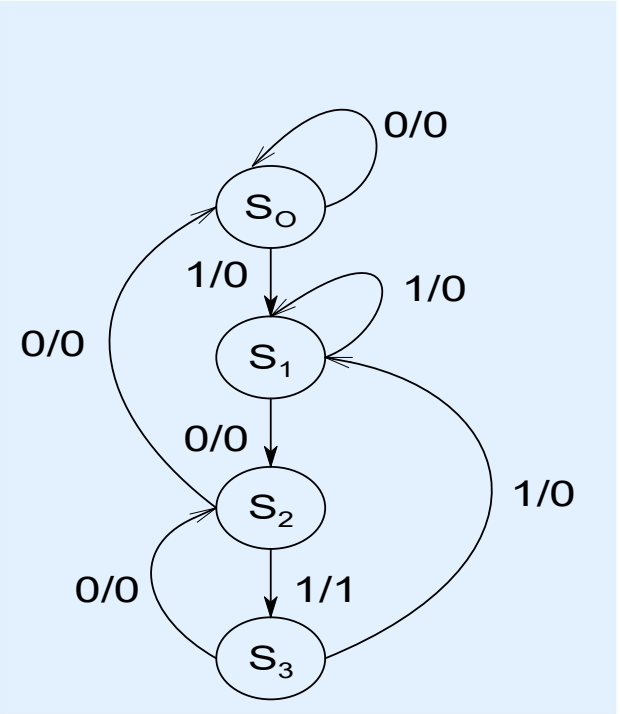


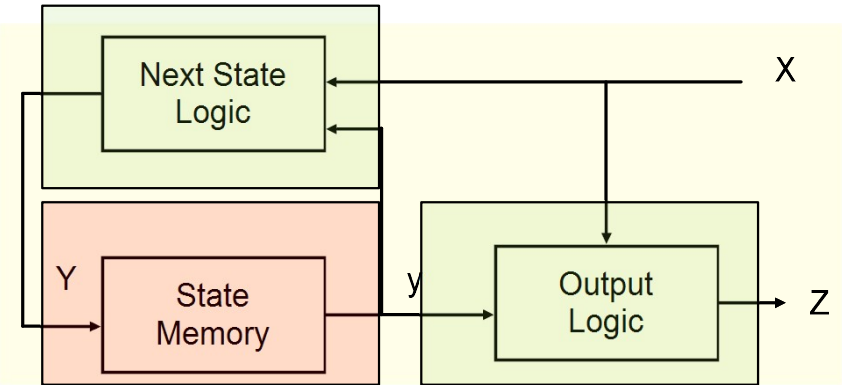


- we need to know how the system goes from one state to another in response to the inputs and how the outputs respond to these changes

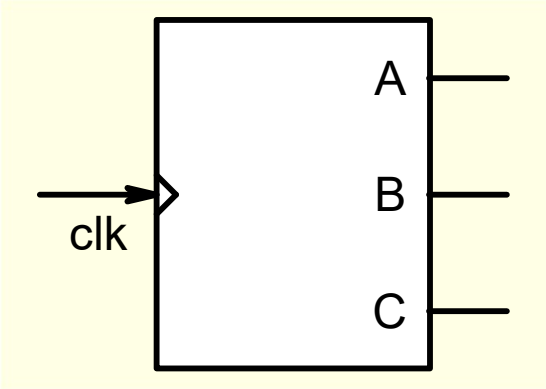
Synthesis involves the following tasks:

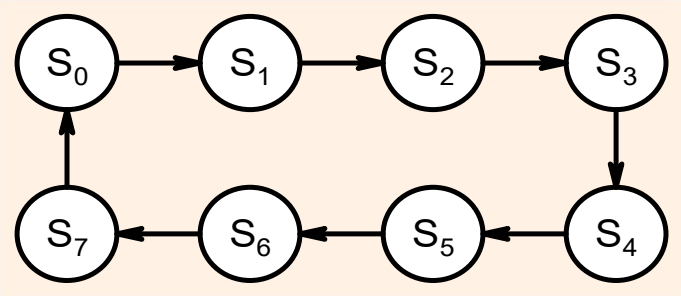
- Number of storage elements
- State Encoding
- Choosing a flipflop type to implement states
- Synthesize next state logic
- Synthesize output logic





Counters

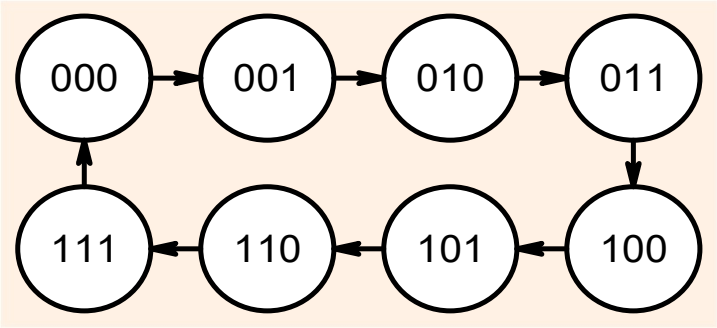




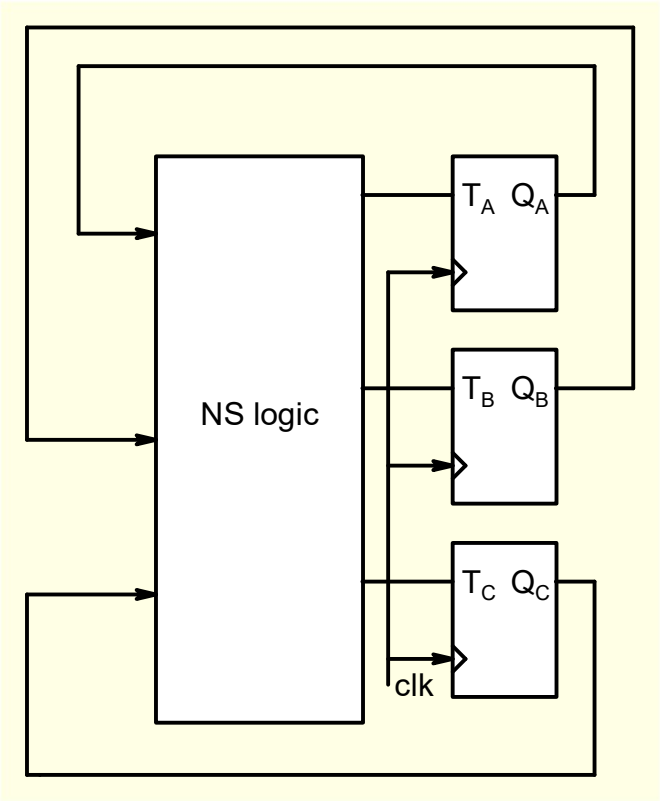
A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

In state S_0, the output ABC is 000, in S_1 001 and so on

There are 8 states so 3 FFs are at least required. Let us choose T FF.

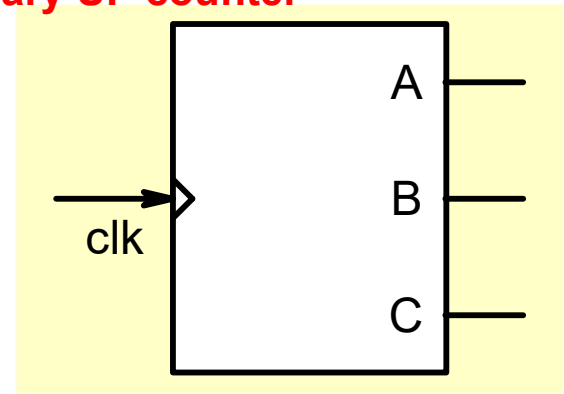


PS A B C	NS A B C	T_A T_B T_C
0 0 0	0 0 1	0 0 1
0 0 1	0 1 0	0 1 1
0 1 0	0 1 1	0 0 1
0 1 1	1 0 0	1 1 1
1 0 0	1 0 1	0 0 1
1 0 1	1 1 0	0 1 1
1 1 0	1 1 1	0 0 1
1 1 1	0 0 0	1 1 1

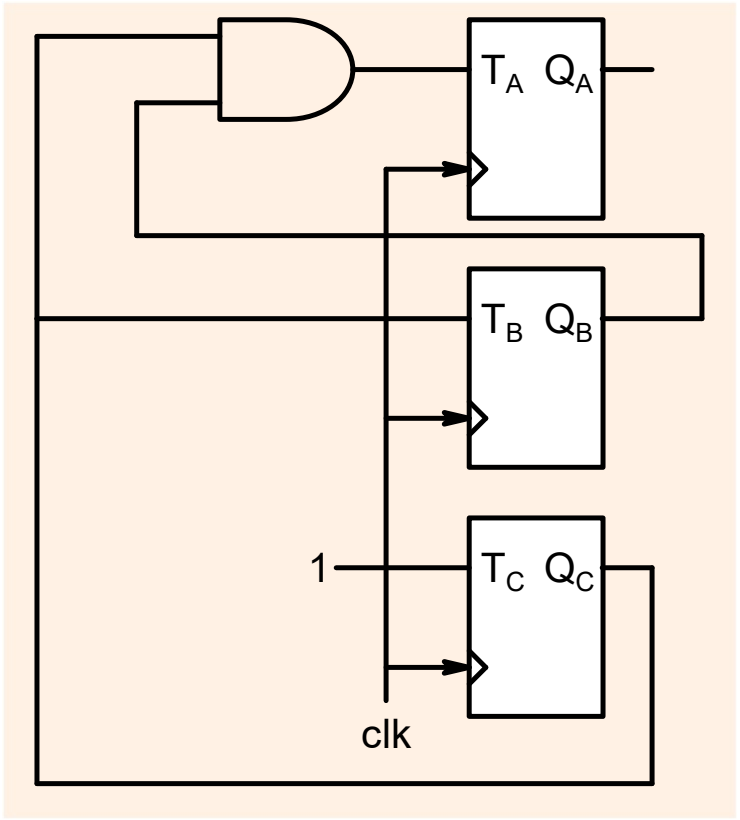


$$T_A = B.C \; ; \; T_B = C \; ; \; T_C = 1$$

Binary UP counter

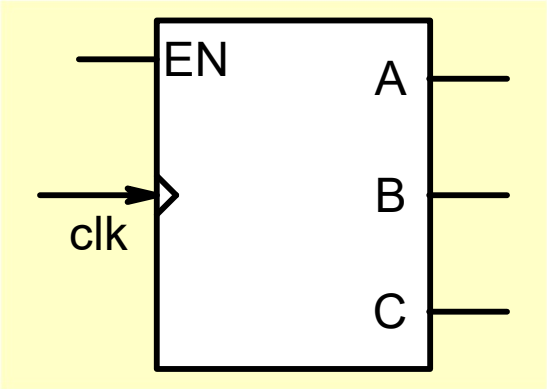


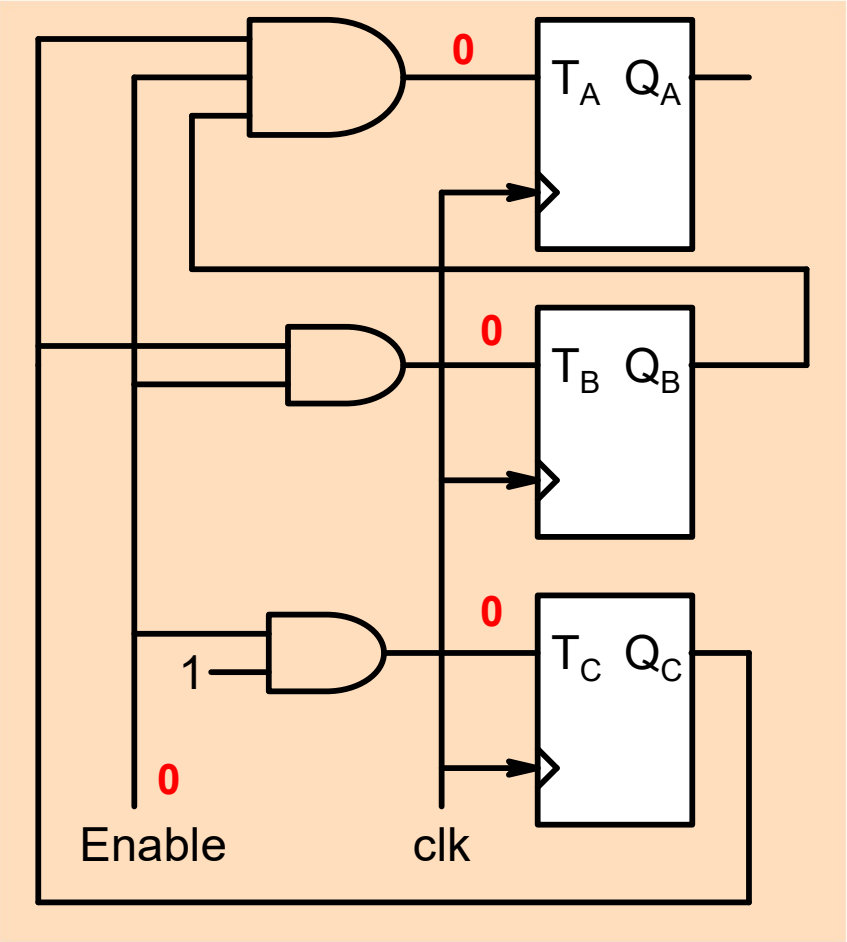
A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



$$T_A = B.C \; ; \; T_B = C \; ; \; T_C = 1$$

Counter with Enable

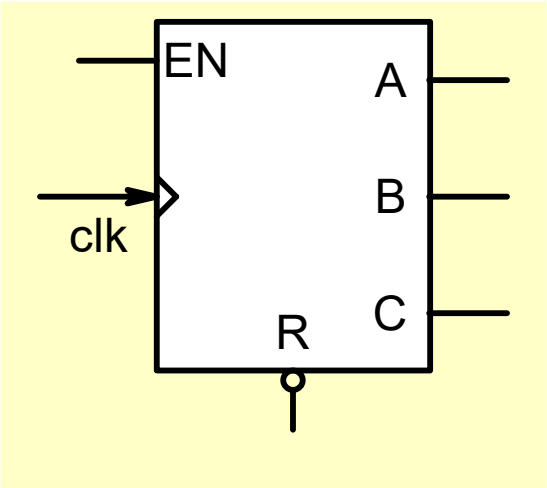


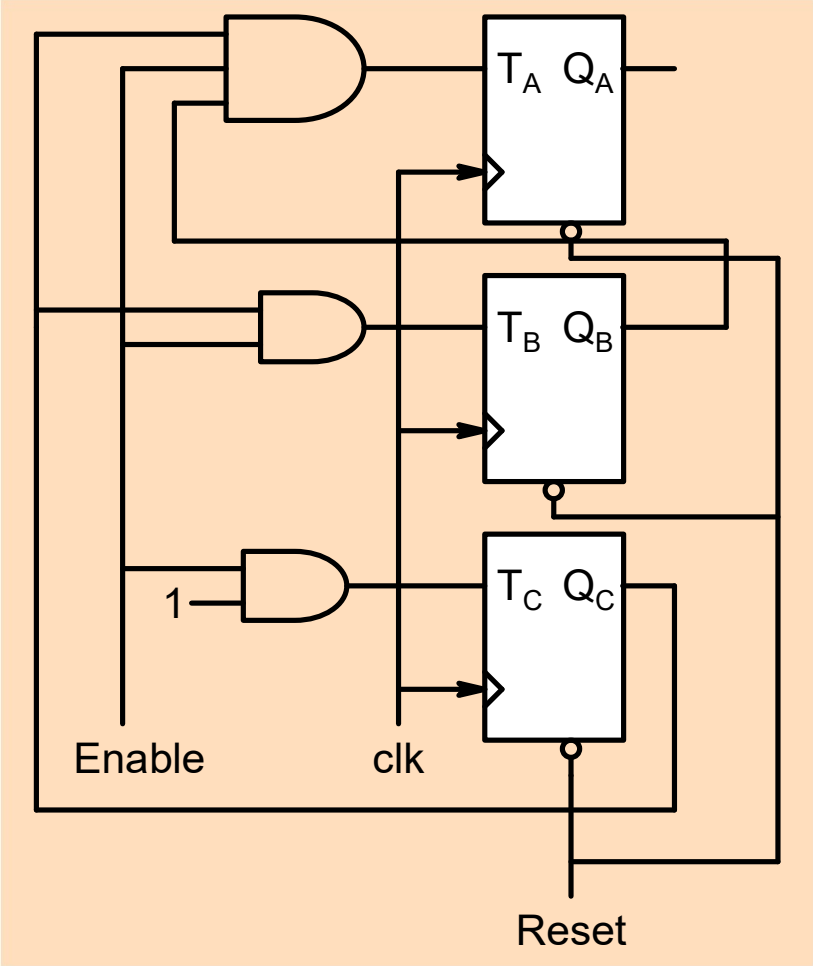


Counter is in Hold state.

When Enable = 1, the counter begins the count.

Counter with Asynchronous Reset



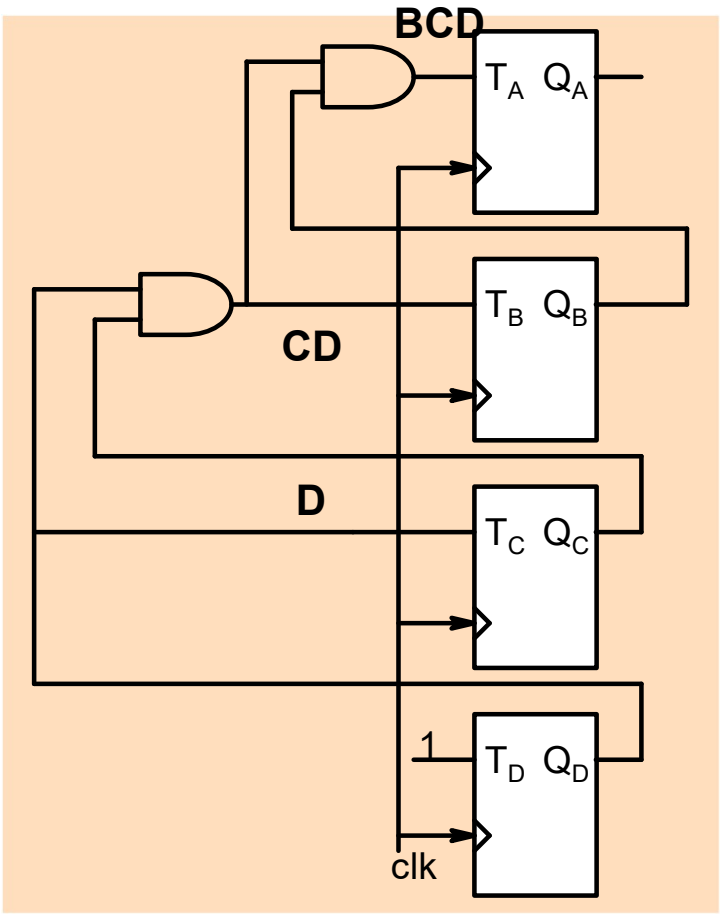


When Enable = 1, the counter begins the count.

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
0	0	0	0

-D toggles every clock cycle
-B toggles only when D is 1
-C toggles only when both C and D are 1
-A toggles only when B C D are 1

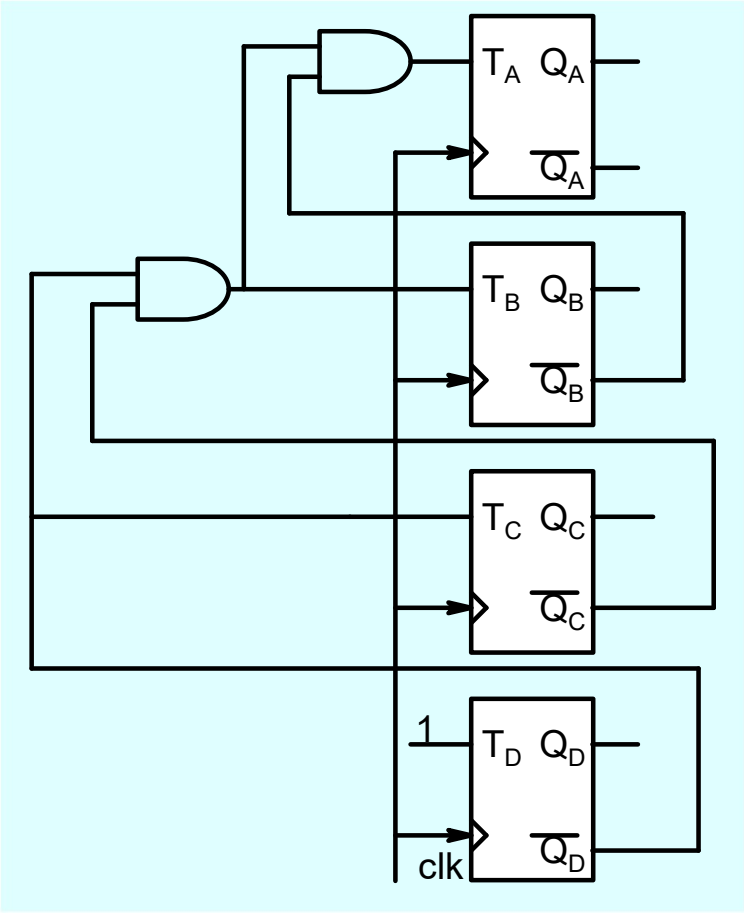
T FF toggles when T=1



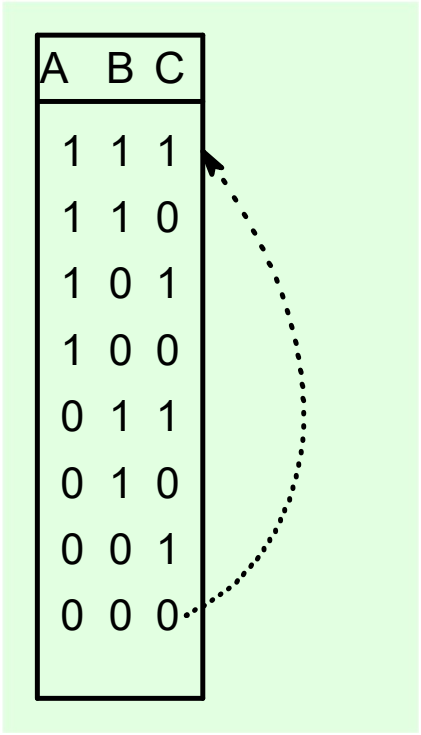
4-bit Down Counter

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

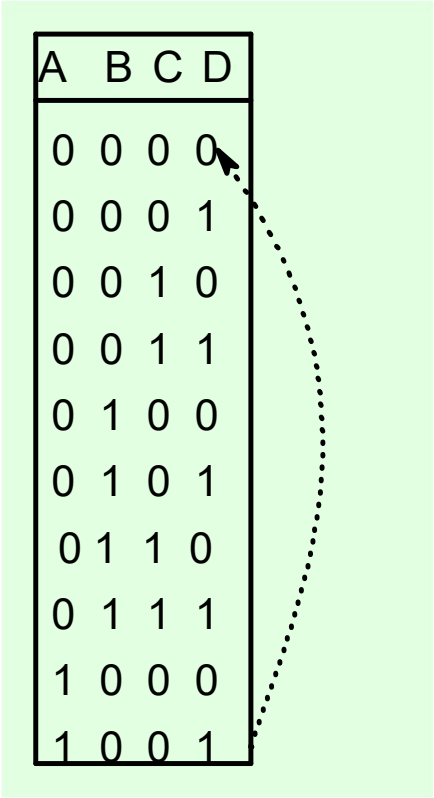
-D toggles every clock cycle
-C toggles only when D is 0
-B toggles only when both C and D are 0
-A toggles only when D C B are 0



Counters

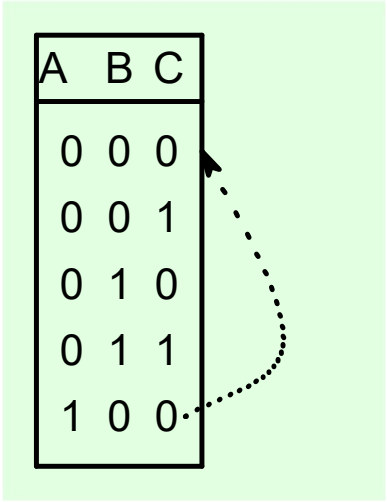


Binary down counter



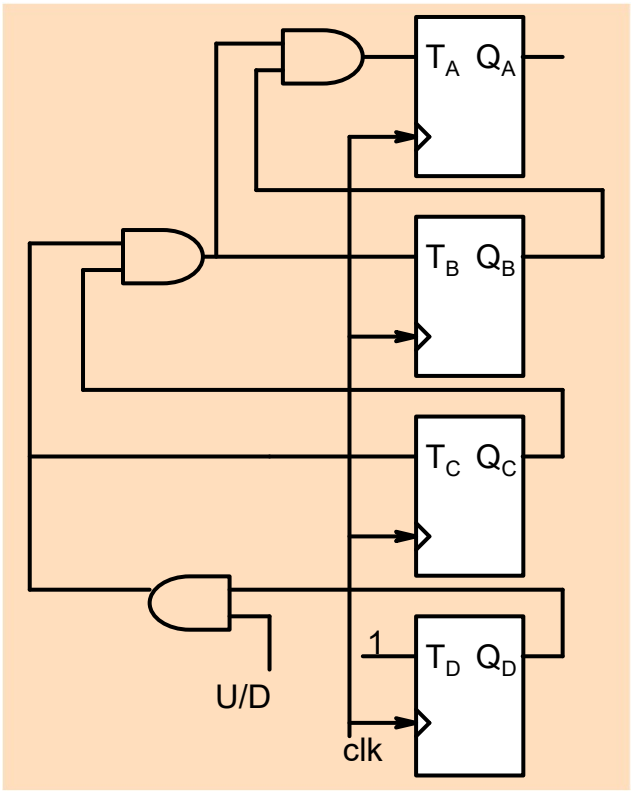
Decade counter

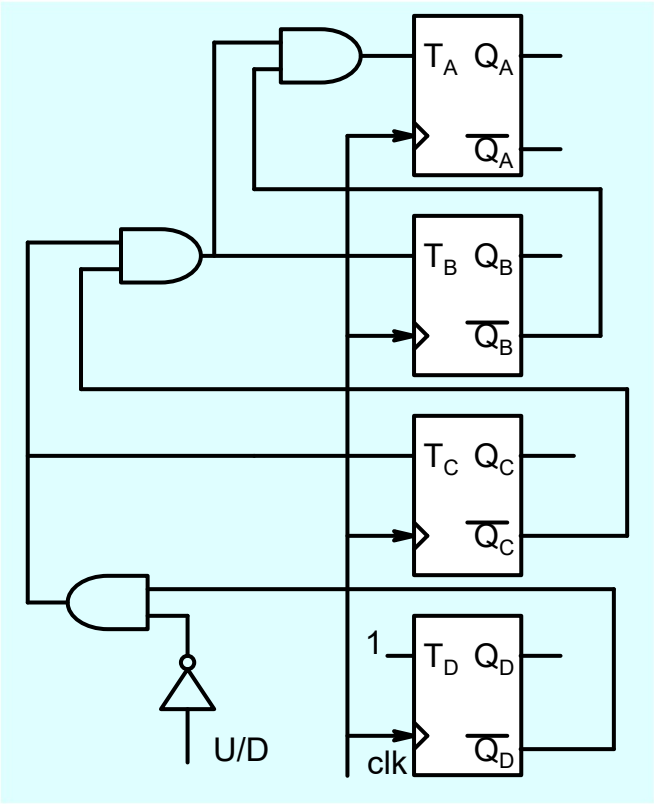
Modulo-10 Counter



Modulo-5 Counter

4-bit Up-Down Counter





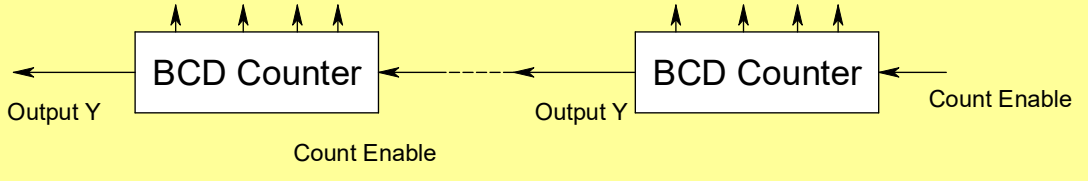
Merging of the two structures gives an Up/down counter

BCD Counter

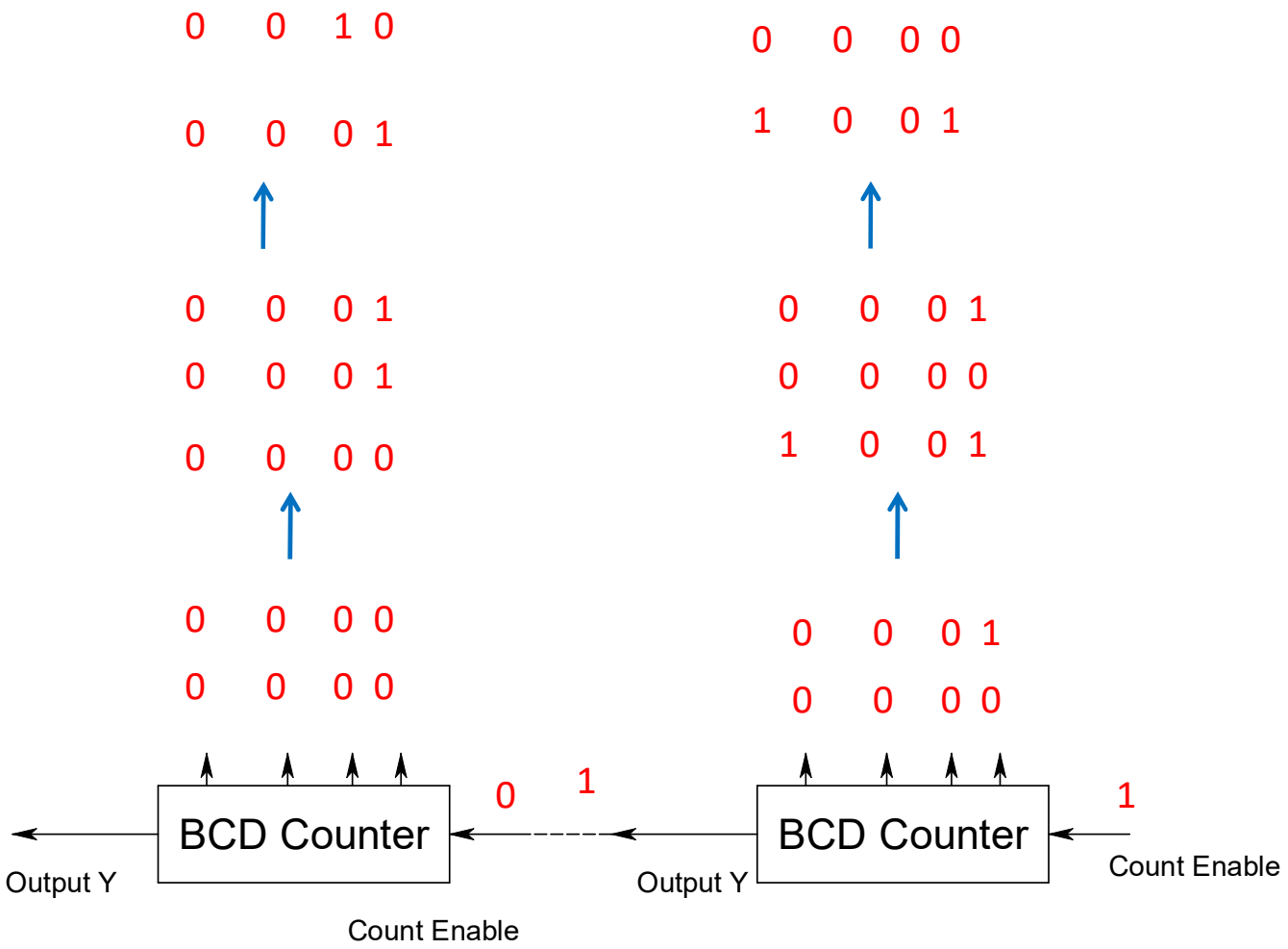
Binary Coded Decimal (BCD): each decimal digit is coded as a 4-bit binary number

25 = 0010 0101

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



BCD counter from 0 to 99



Counter with Unused States

PS			NS			J_A	K_A	J_B	K_B	J_C	K_C
A	B	C	A	B	C						
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	0	0	1	X	X	1	0	X
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X

There are two unused states 011 and 111. one approach to handle this situation is that, while evaluating expressions for J K , we use don't care conditions corresponding to these unused states

Counter with Unused States

PS			NS			J_A	K_A	J_B	K_B	J_C	K_C
A	B	C	A	B	C						
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	0	0	1	X	X	1	0	X
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X

A \ BC	00	01	11	10
0	0	0	X	1
1	X	X	X	X

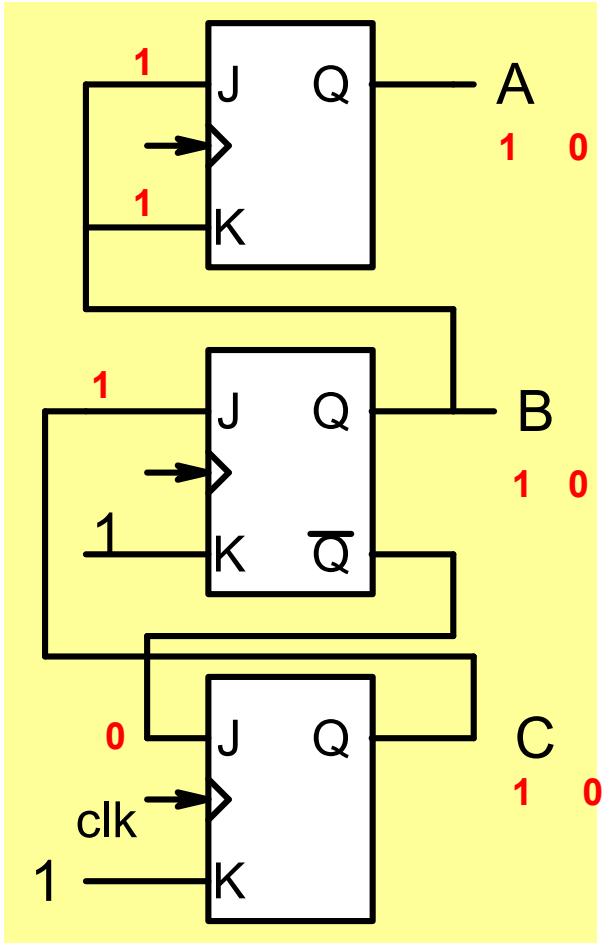
$J_A = B$

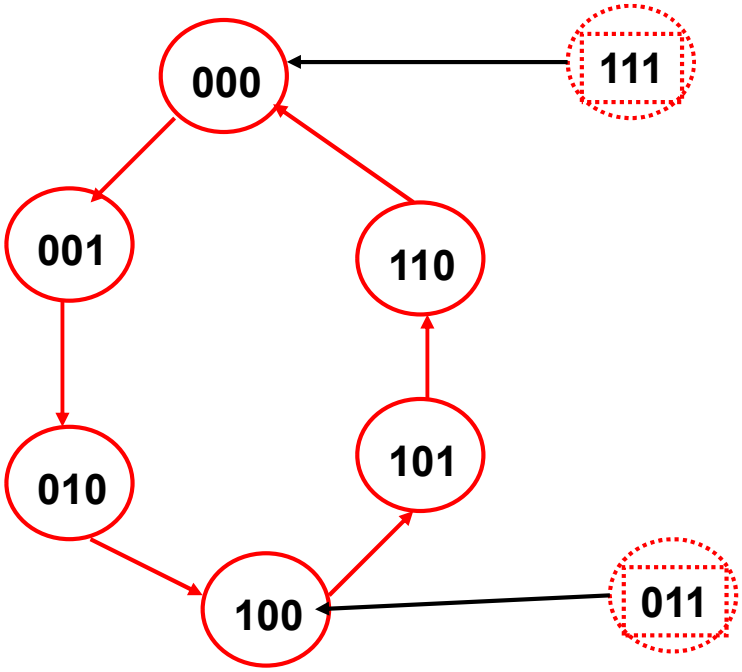
Counter with Unused States

PS	NS						
A B C	A B C	J_A	K_A	J_B	K_B	J_C	K_C
0 0 0	0 0 1	0	X	0	X	1	X
0 0 1	0 1 0	0	X	1	X	X	1
0 1 0	1 0 0	1	X	X	1	0	X
1 0 0	1 0 1	X	0	0	X	1	X
1 0 1	1 1 0	X	0	1	X	X	1
1 1 0	0 0 0	X	1	X	1	0	X

$$J_A = B \qquad K_A = B$$
$$J_B = C \qquad K_B = 1$$
$$J_C = \overline{B} \qquad K_C = 1$$

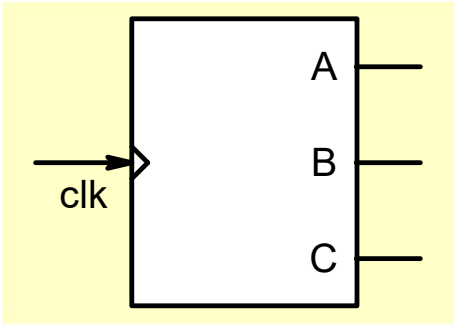
After synthesizing the circuit, one needs to check that if by chance the counter goes into one of the unused states, after one or more clock cycles, it enters a used state and then remains among the used states



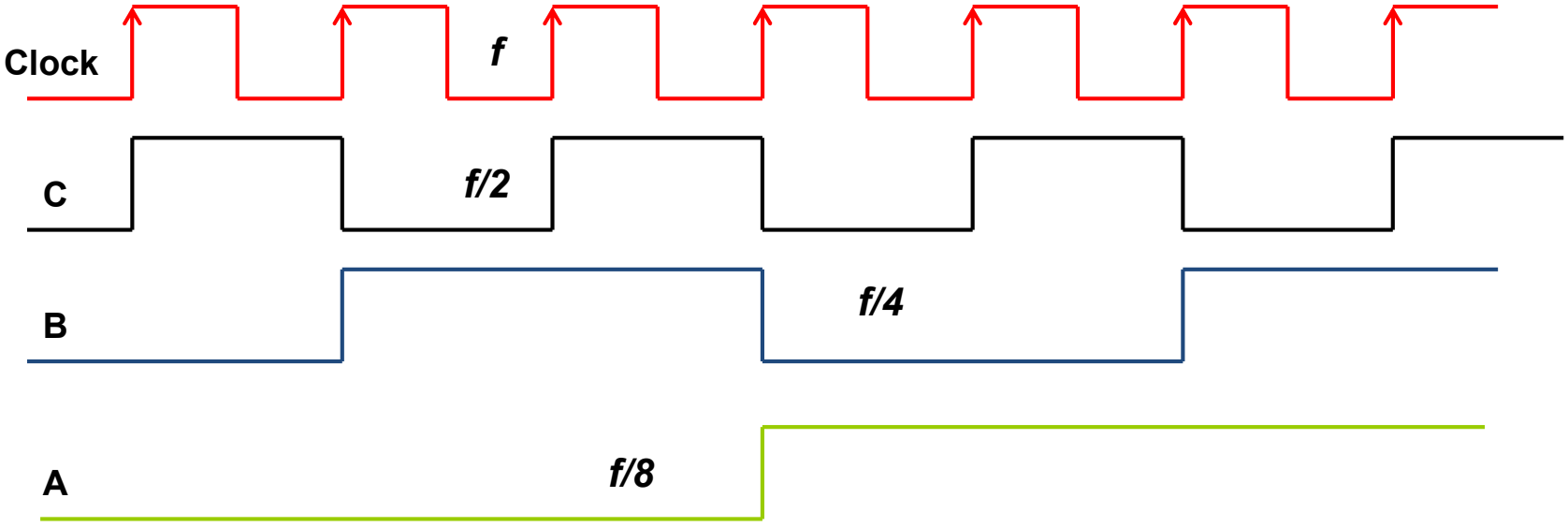


We can see that if by chance the counter goes into unused states 111 or 011, then after a clock cycle it enters one of the used states.

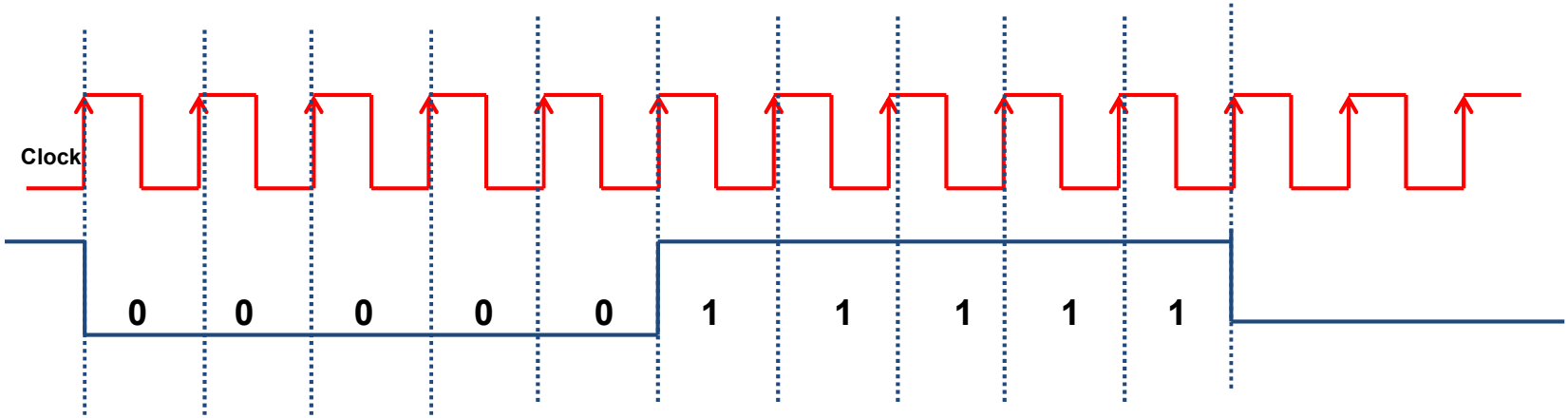
Counter as frequency divider



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



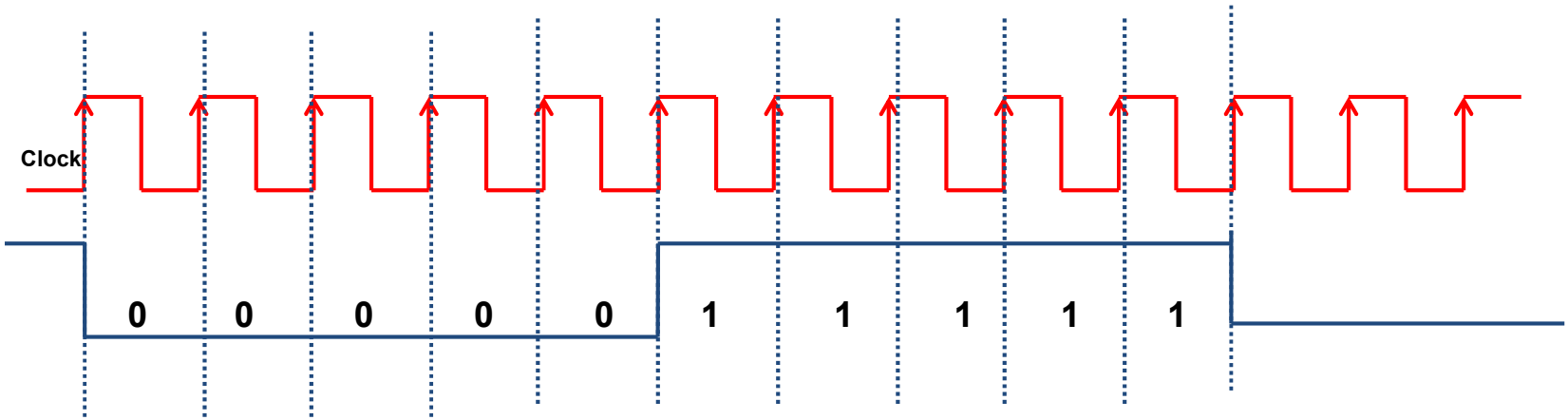
Example From a frequency of 10KHz, generate the following signal of frequency 1KHz



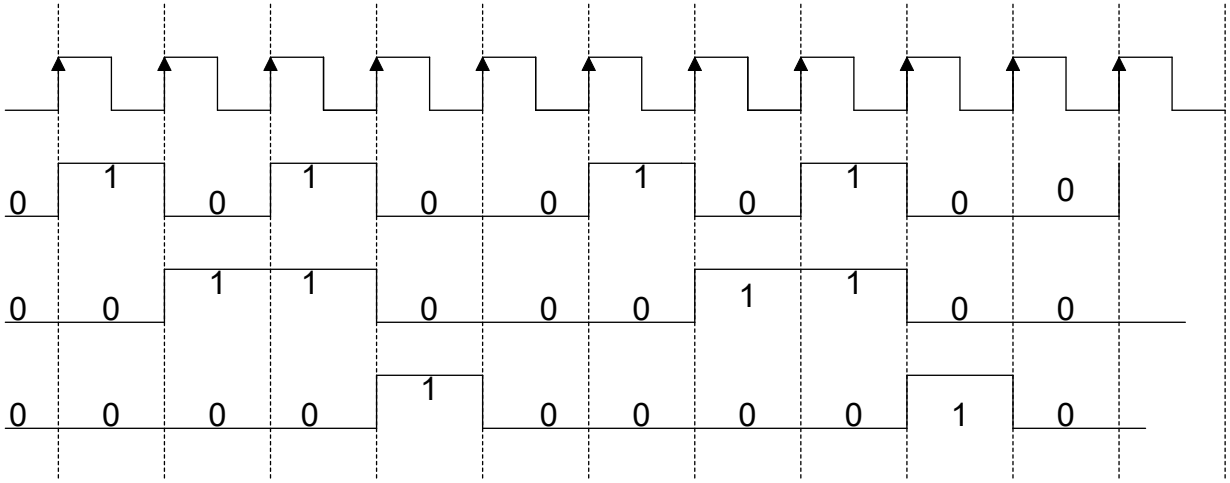
A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1

This will have a frequency of 1KHz but it will not have the same waveform

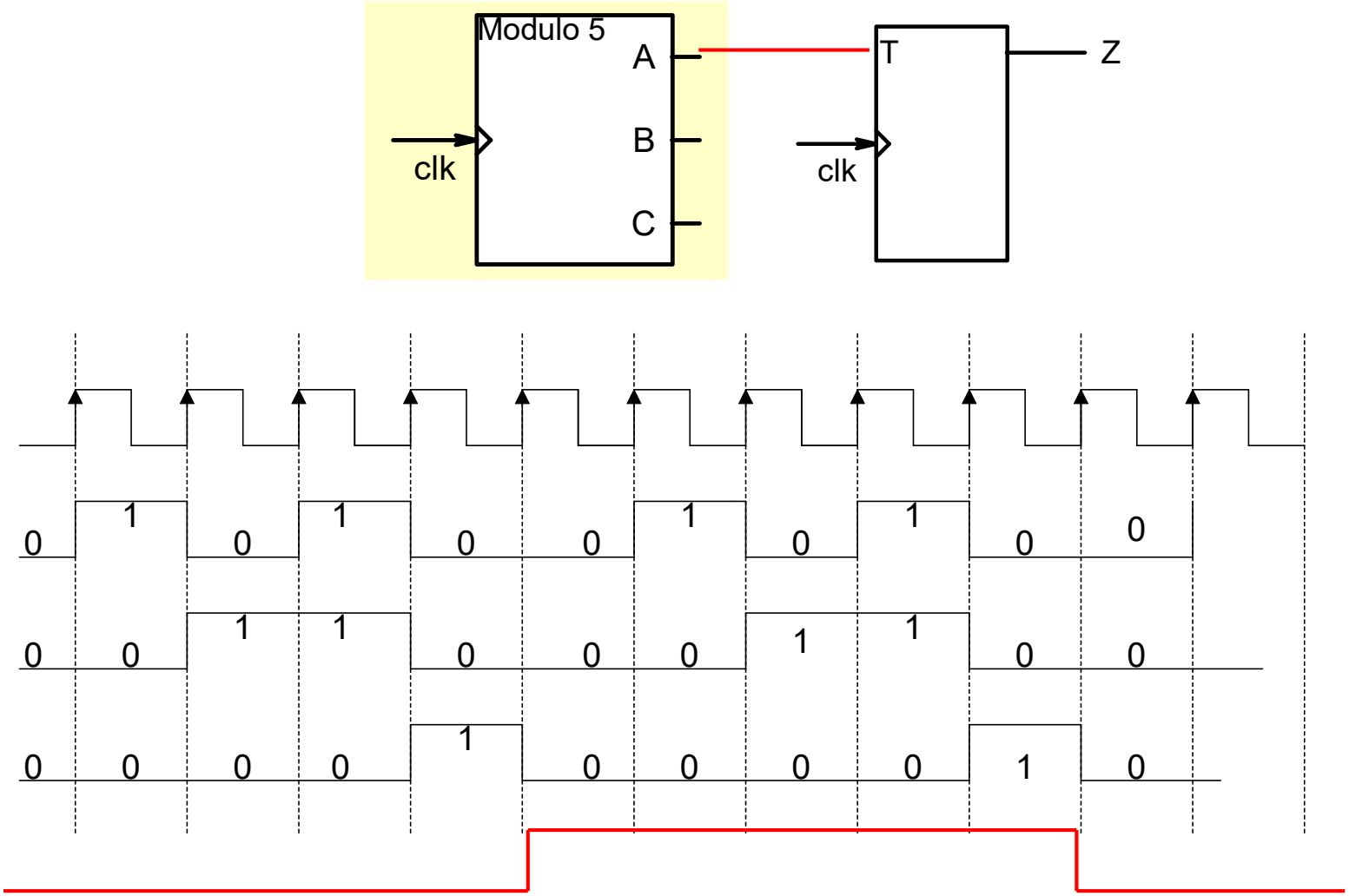
Example From a frequency of 10KHz, generate a signal of frequency 1KHz

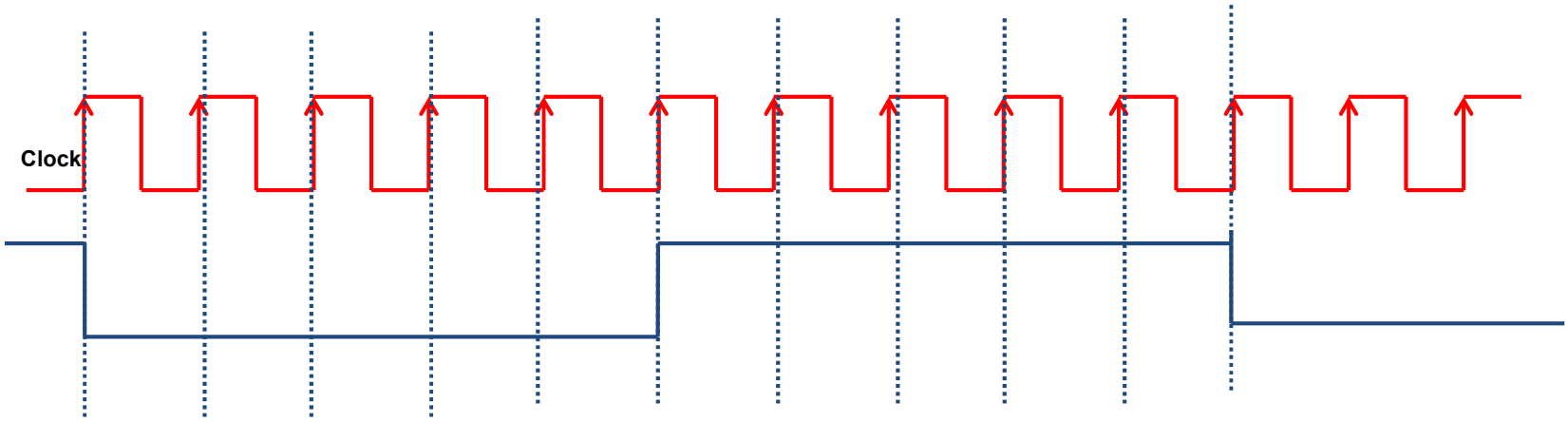


A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0



The idea is to generate a divide by 5 counter first and then divide it again by 2 to get the required waveform

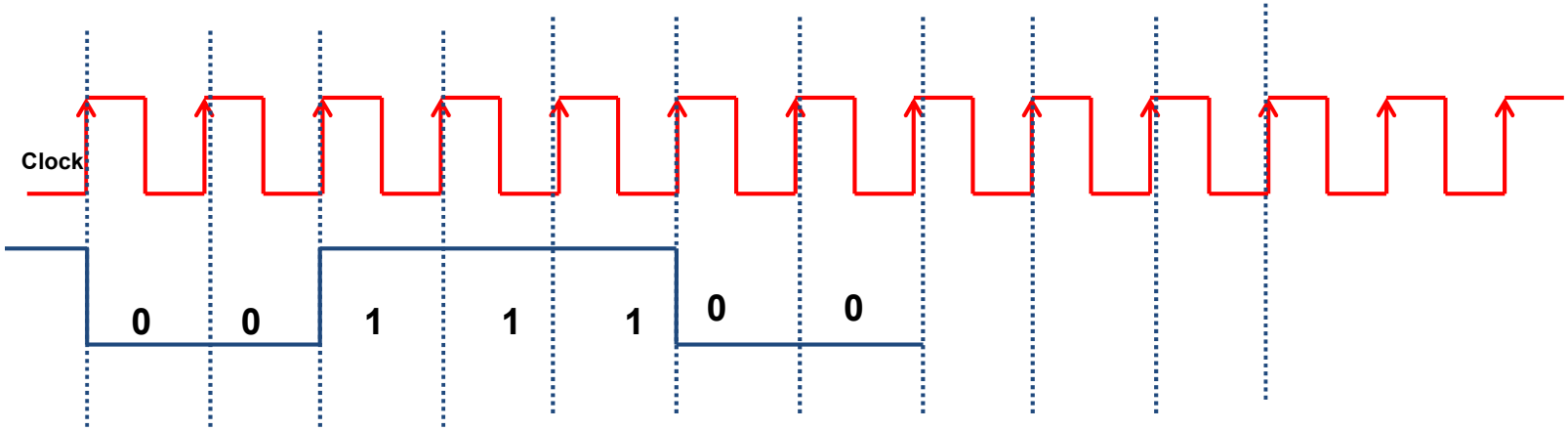

Modulo 5
A
B
C
clk
T
clk
Z
0 1 0 1 0 0 1 0 1 0 0
0 0 1 1 0 0 0 1 1 0 0
0 0 0 0 1 0 0 0 0 1 0



Alternatively design a divide by 10 counter with the following states

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

Example From a frequency of 10KHz, generate the following signal of frequency 2KHz



A divide by 5 counter is required that has 5 states.

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

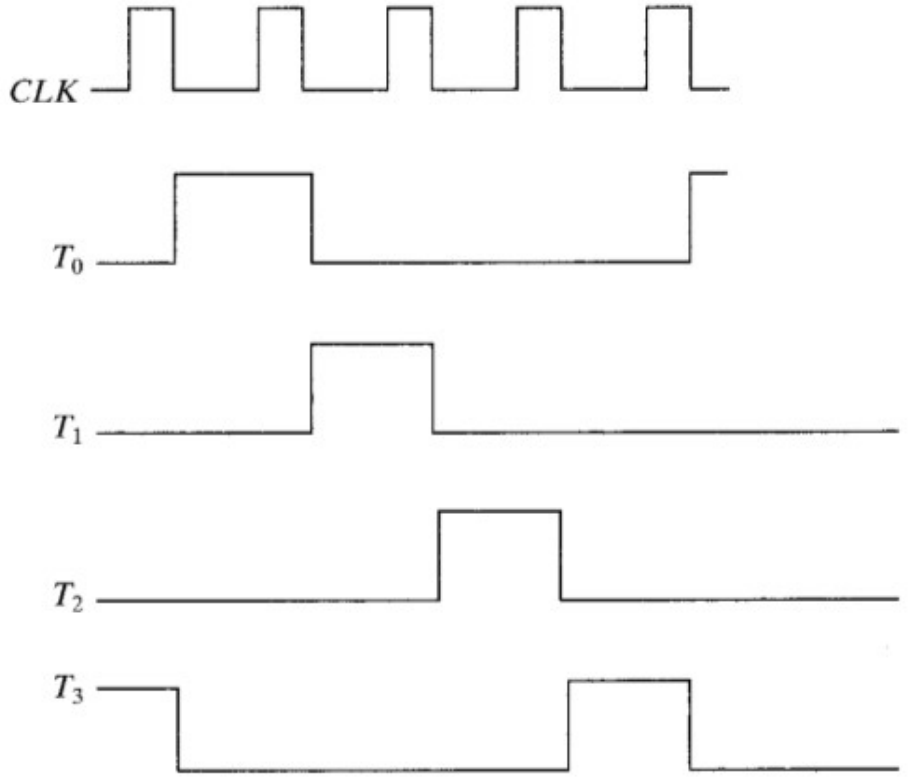
Modify

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

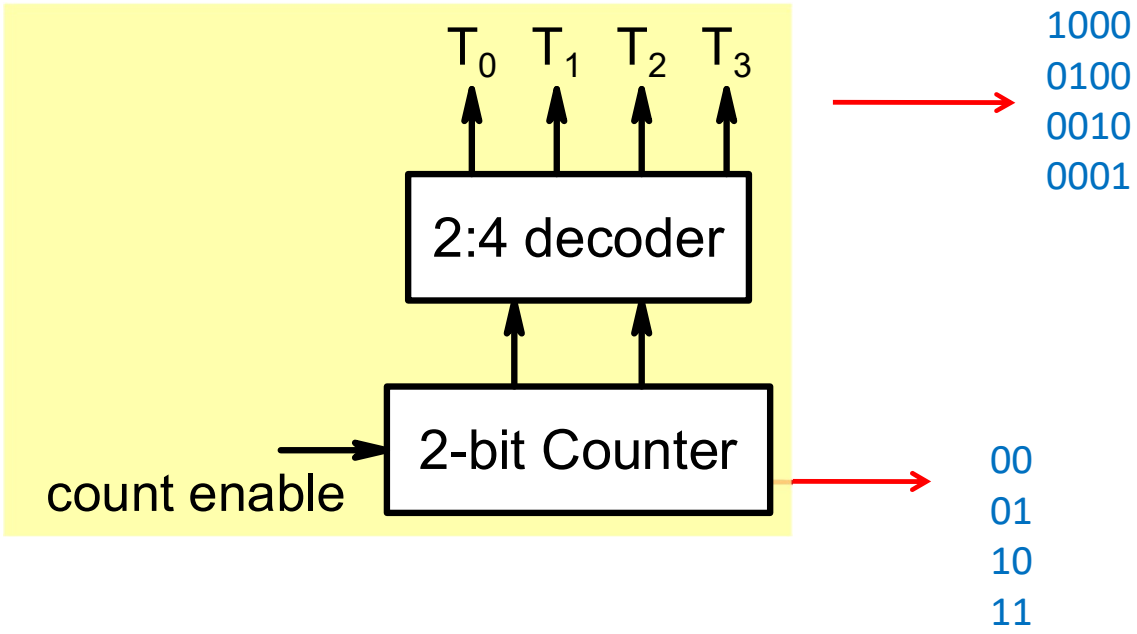
A will give the required waveform.

Ring Counter

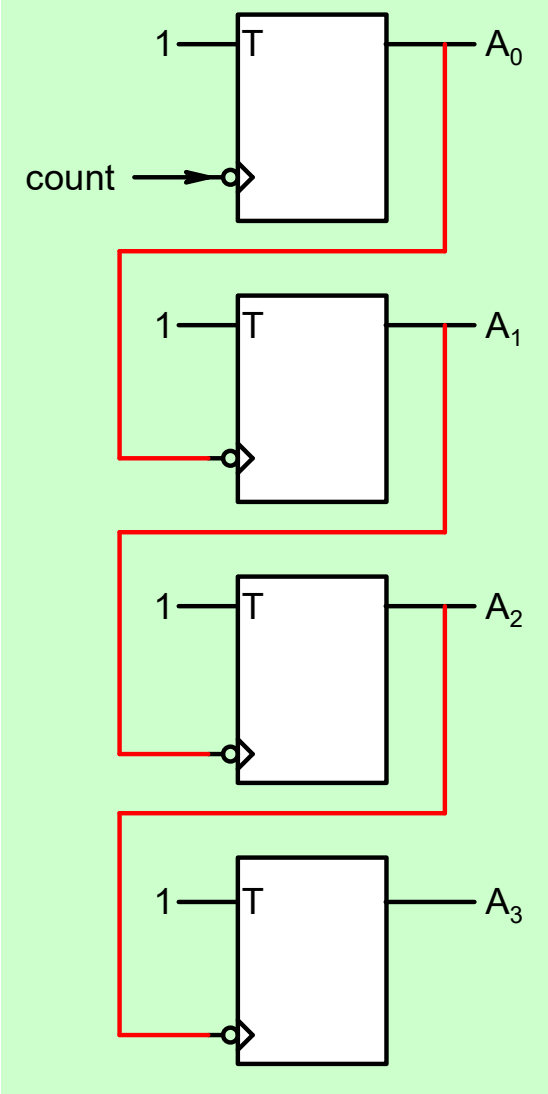
T_3	T_2	T_1	T_0
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0



Alternative Implementation



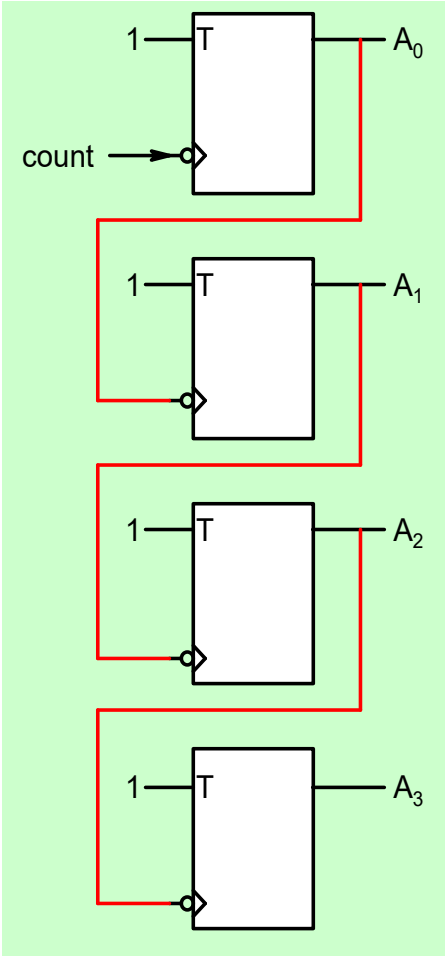
Ripple Counter



T FF toggles when T = 1; otherwise Hold state

Clock is negative edge Triggered

Ripple Counter



0 1 2 3 4 5 ------15

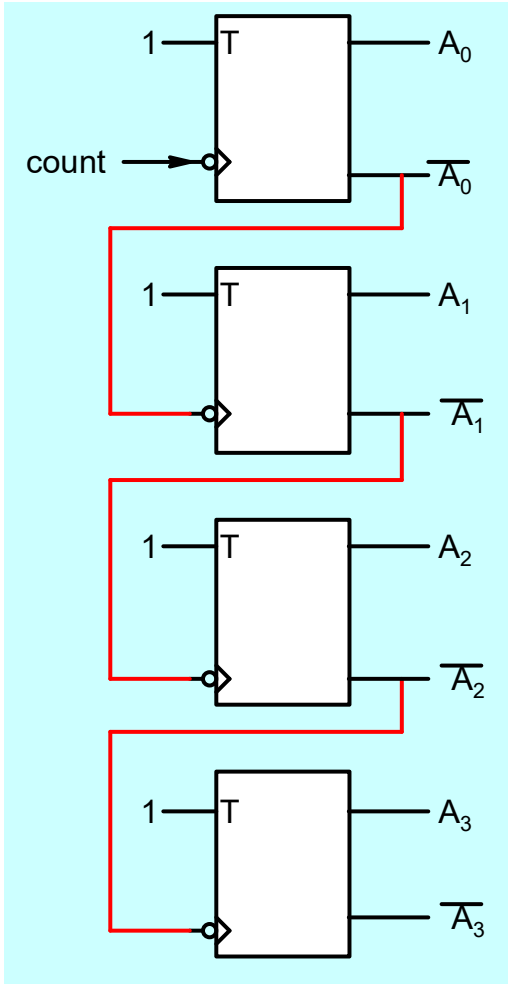
0 1 0 1 0 1 ------1 0

0 0 1 1 0 0 ------1 0

0 0 0 0 1 1 ------1 0

0 0 0 0 0 0 ------1 0

Ripple Down Counter



0 1 0

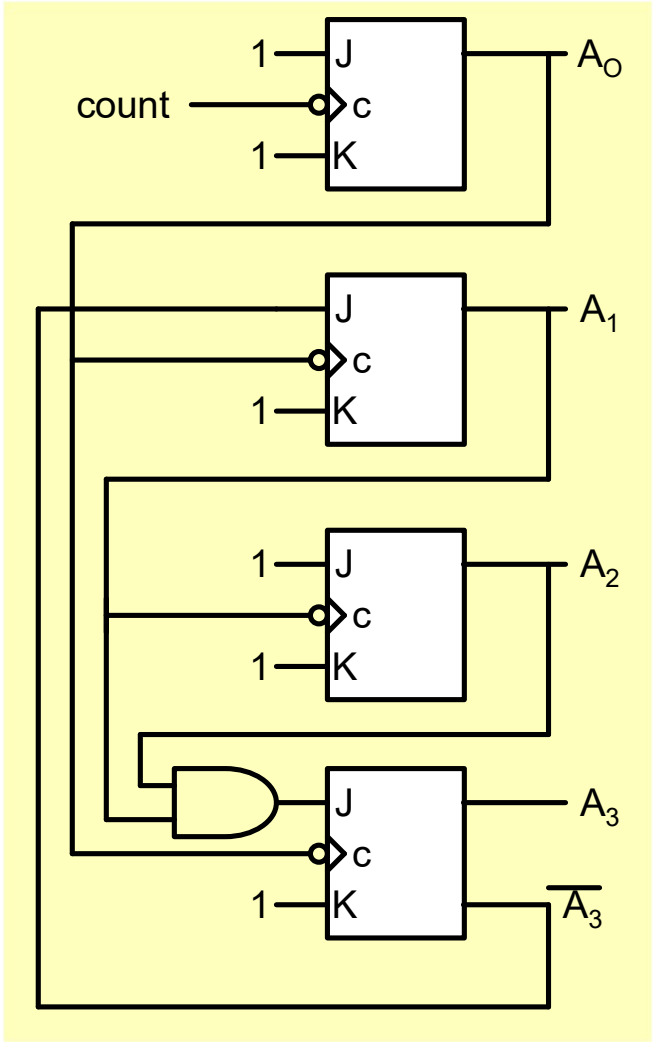
0 1 1

0 1 1

0 1 1

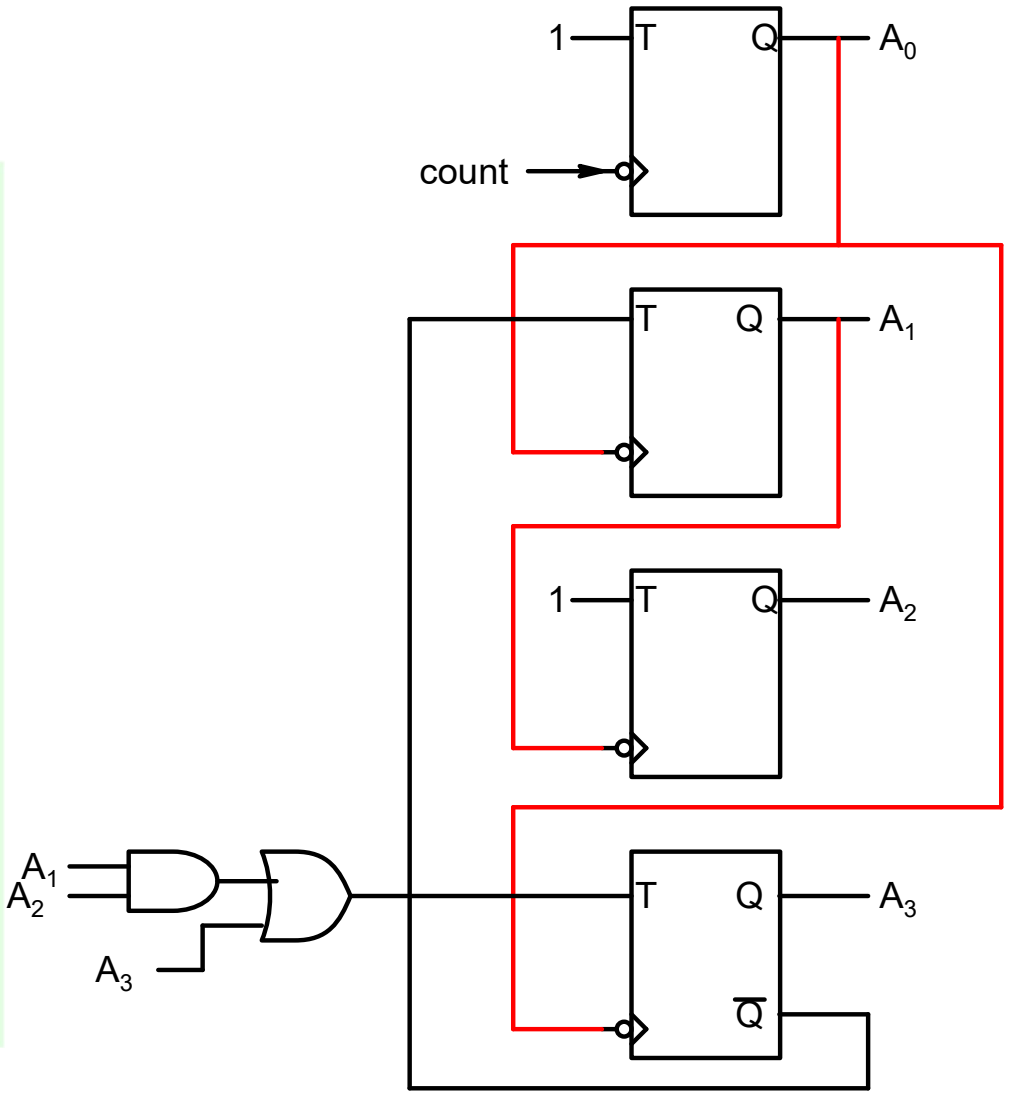
BCD Ripple Counter

A_3	A_2	A_1	A_0
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



BCD Ripple Counter

A_3	A_2	A_1	A_0
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



Cascading of BCD counters