

CSE 140 Lecture 12

Standard Combinational Modules

CK Cheng
CSE Dept.
UC San Diego

1

Part III. Standard Modules

Interconnect Modules:

1. Decoder, 2. Encoder
- 3. Multiplexer, 4. Demultiplexer**

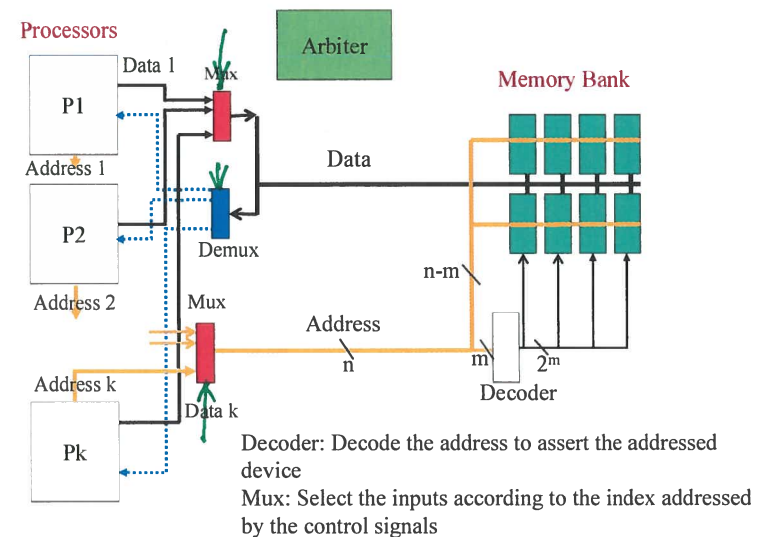
2

Multiplexer

- Definition
- Logic Diagram
- Application

3

Interconnect: Decoder, Encoder, Mux, DeMux

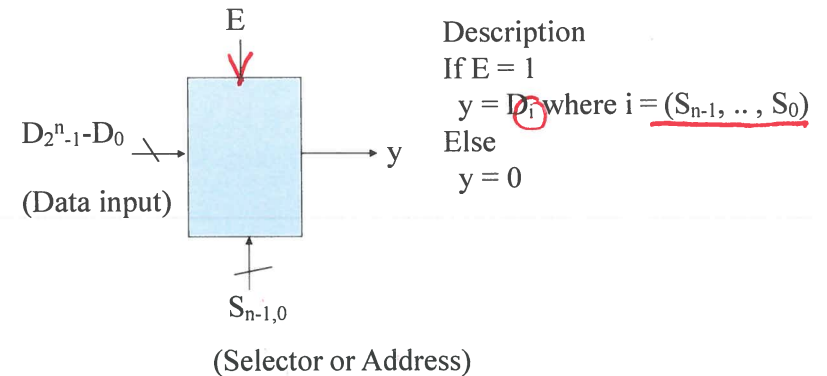


4

Multiplexer Definition

- A. A device that interleaves two or more activities
- B. A communications device that combines several signals for transmission over a single medium
- C. A logic circuit that sends one of several inputs out over a single output channel.**
- D. The circuit that uses a common communications channel for sending two or more messages or signals.
- E. All of the above

3. Mux (Multiplexer) Definition: A digital module that selects one of data inputs according to the binary address of the selector.

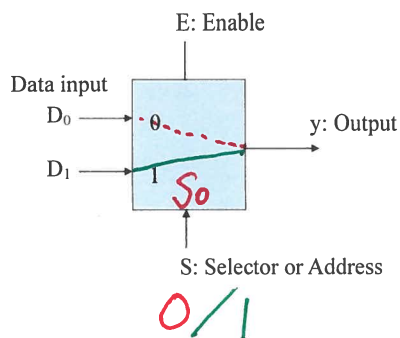


5

6

Multiplexer (Mux): Definition

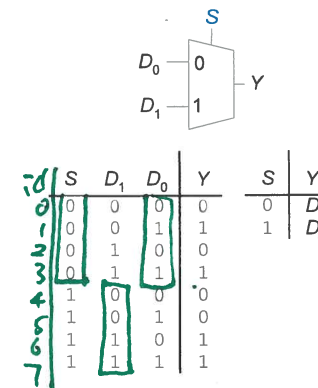
- Selects between one of N inputs to connect to the output.
- $\log_2 N$ -bit select input – control input



7

Multiplexer (Mux): Definition

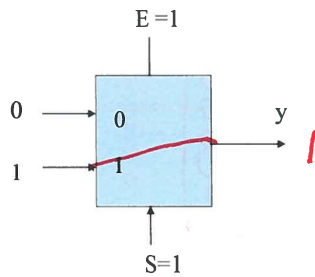
- Selects between one of N inputs to connect to the output.
- $\log_2 N$ -bit select input – control input
- Example: **2:1 Mux**



8

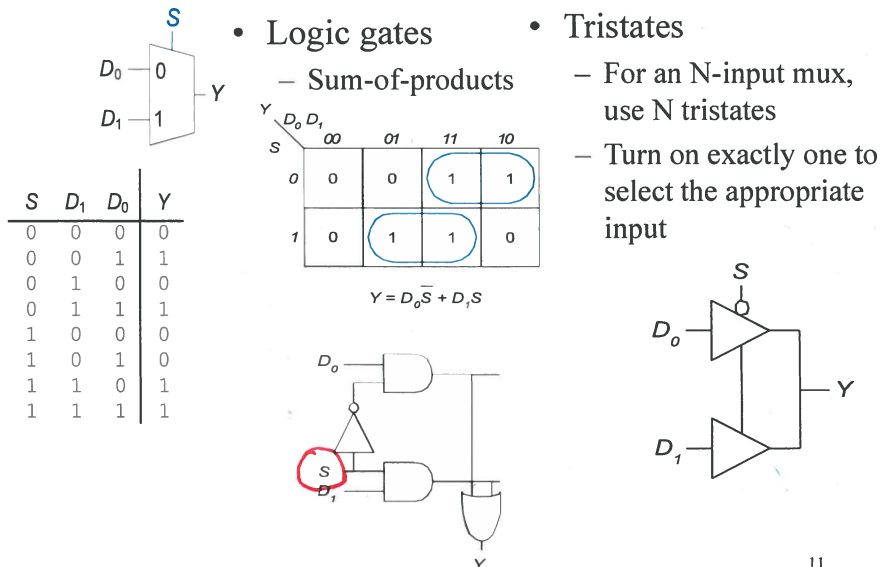
What is the output of the following MUX?

- A.0
- B.1
- C.Can't say



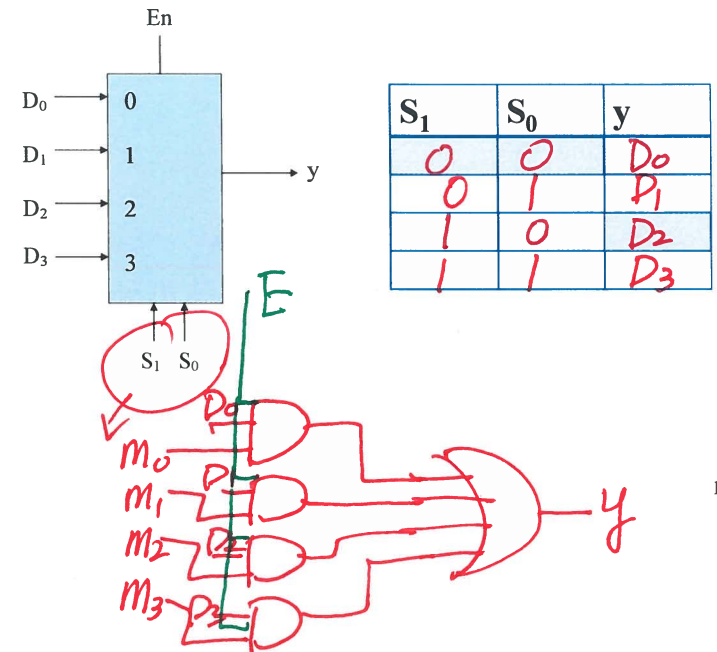
9

Multiplexer: Logic Diagram



11

Multiplexer Definition: 4-input mux



10

Multiplexer Application: universal set {Mux}

We use selector to decompose the function into smaller functions (less number of variables), which follows

Shannon's expansion.

We simplify the decomposed functions using **K-map**, which follows **consensus theorem**.

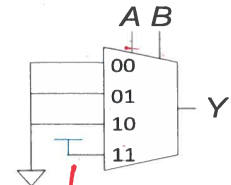
12

Multiplexer Application

Intel
Xilinx
Universal Set

FPGA CBL Combinational Building Block

- Mux for a Boolean function with truth table as input
- Building blocks of FPGA (Field Programmable Gate Array).



For the logic diagram on left, output Y is

- A. AB
- B. (AB)'
- C. A+B
- D. (A+B)'
- E. None of the above

AB	Y
00	0
01	0
10	0
11	1

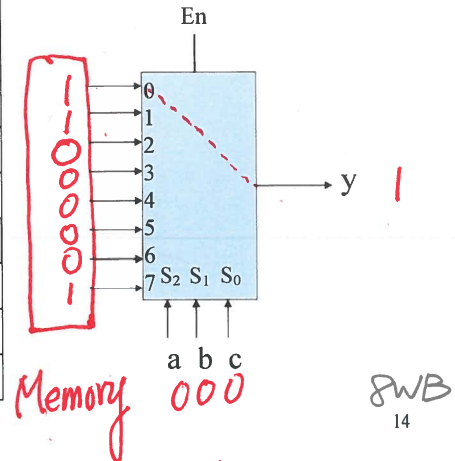
$\Rightarrow AB$

13

Multiplexer Application: universal set {Mux}

Example 1: Given $f(a,b,c) = \sum m(0,1,7) + \sum d(2)$, implement with an 8-input Mux.

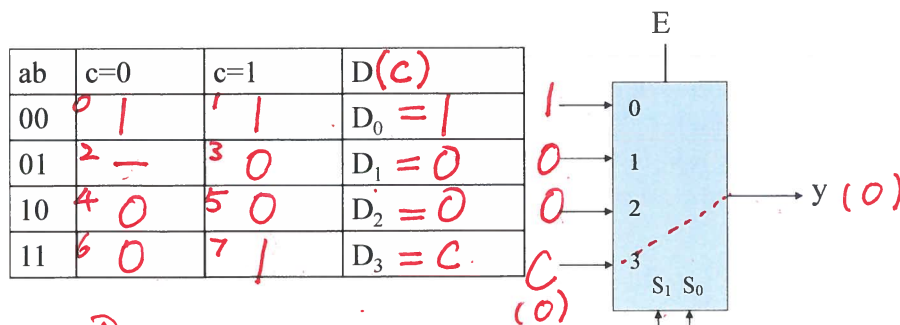
id	abc	f
0	000	1
1	001	1
2	010	-
3	011	0
4	100	0
5	101	0
6	110	0
7	111	1



All inputs have to be connected to either 0 or 1

14

Example 2: Given $f(a,b,c) = \sum m(0,1,7) + \sum d(2)$, implement with 4-input Muxes.

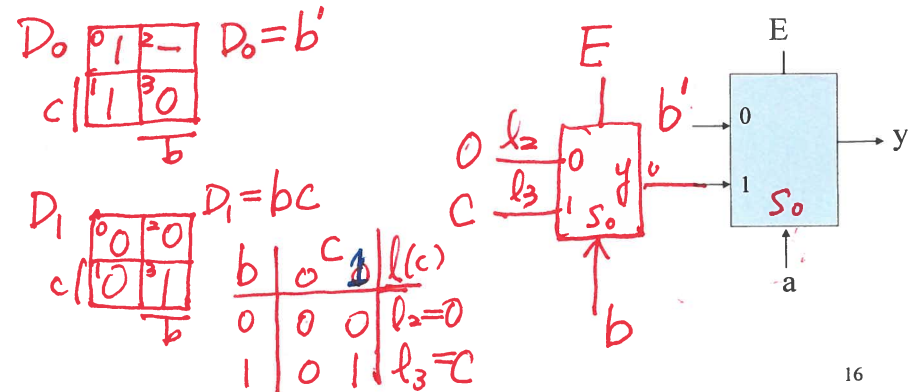


Ex. $(abc) = (110) \Rightarrow y = 0$
 $(abc) = (111) \Rightarrow y = 1$

15

Example 3: Given $f(a,b,c) = \sum m(0,1,7) + \sum d(2)$, implement with 2-input Muxes.

a\bc	00	01	10	11	D(b,c)
0	1	1	-	0	D_0
1	0	0	0	1	D_1



16

Example 3: Given $f(a,b,c) = \sum m(0,1,7) + \sum d(2)$, implement with 2-input Muxes.

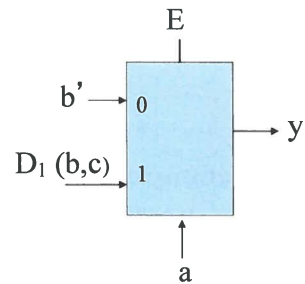
a\bc	00	01	10	11	D(b,c)
0	1	1	-	0	D_0
1	0	0	0	1	D_1

$D_0(b,c) = b'$

c=0	1	-
c=1	1	0
	b=0	b=1

$D_1(b,c) = bc$

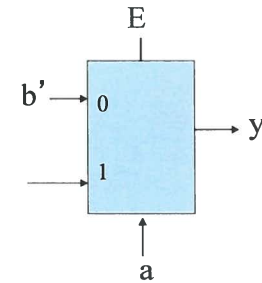
c=0	0	0
c=1	0	1
	b=0	b=1



Example 3: Given $f(a,b,c) = \sum m(0,1,7) + \sum d(2)$, implement with 2-input Muxes.

$D_1(b,c)$

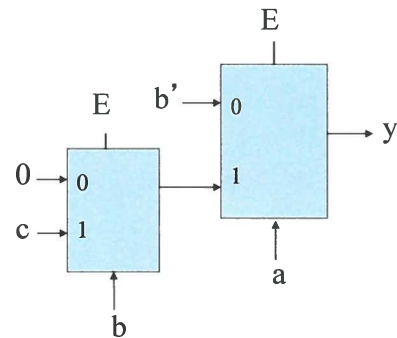
b\c	0	1	D
0	0	0	$D_0 =$
1	0	1	$D_1 =$



Example 3: Given $f(a,b,c) = \sum m(0,1,7) + \sum d(2)$, implement with 2-input Muxes.

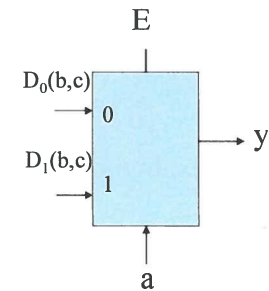
$D_1(b,c)$

b\c	0	1	D
0	0	0	$D_0 = 0$
1	0	1	$D_1 = c$

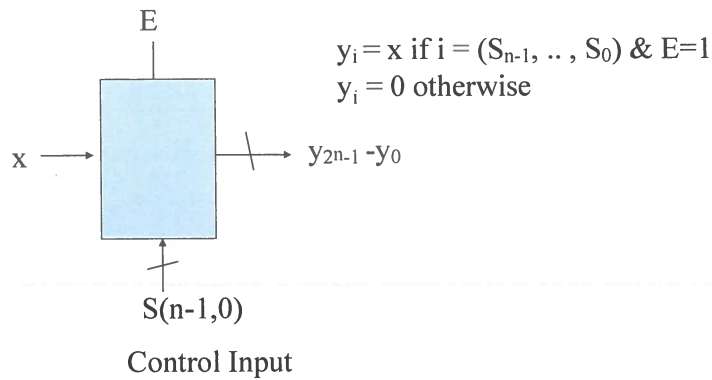


Example 4: Given $f(a,b,c) = \sum m(0,2,4,7) + \sum d(3,5)$, implement with 2-input Muxes.

a\bc	00	01	10	11	D
0	1	0	1	-	D_0
1	1	-	0	1	D_1



4. Demultiplexers



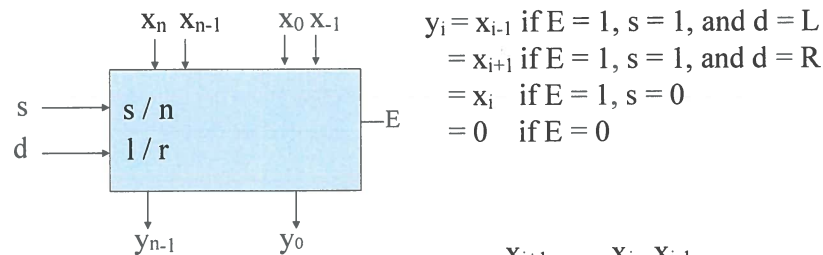
21

Shifters

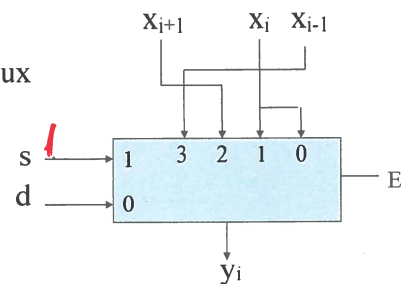
- Logical shifter:** shifts value to left or right and fills empty spaces with 0's
 - Ex: $11001 \gg 2 = 00110$
 - Ex: $11001 \ll 2 = 00100$
- Arithmetic shifter:** same as logical shifter, but on right shift, fills empty spaces with the old most significant bit (msb).
 - Ex: $11001 \ggg 2 = 11110$
 - Ex: $11001 \lll 2 = 00100$
- Rotator:** rotates bits in a circle, such that bits shifted off one end are shifted into the other end
 - Ex: $11001 \text{ ROR } 2 = 01110$
 - Ex: $11001 \text{ ROL } 2 = 00111$

22

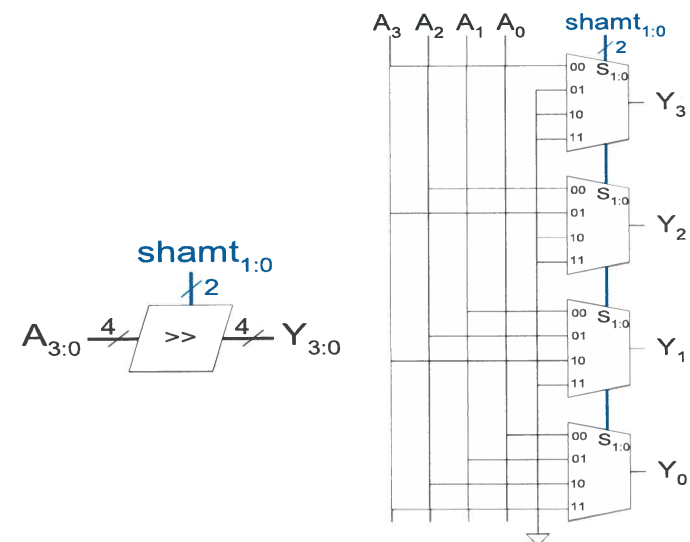
Shifter



Can be implemented with a mux

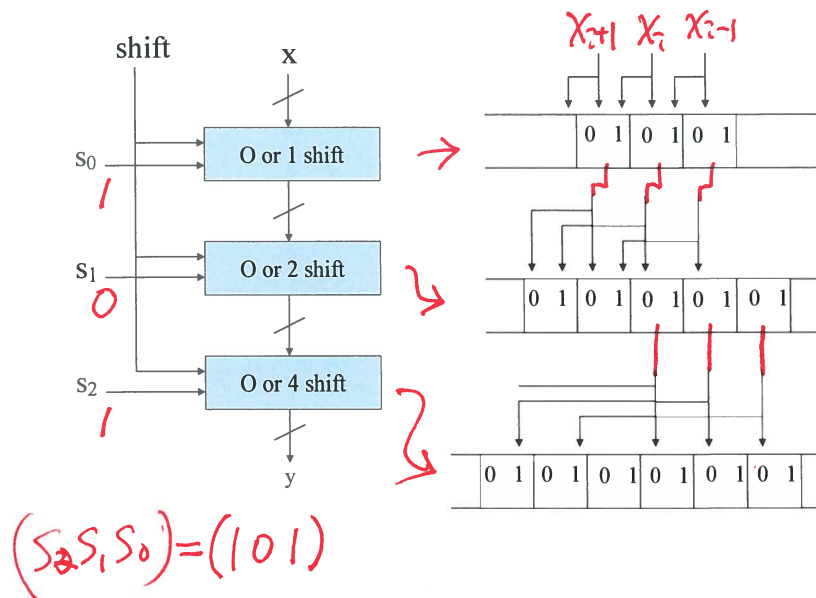


Shifter Design



24

Barrel Shifter



Shifters as Multipliers and Dividers

- A left shift by N bits multiplies a number by 2^N
 - Ex: $00001 \ll 2 = 00100$ ($1 \times 2^2 = 4$)
 - Ex: $11101 \ll 2 = 10100$ ($-3 \times 2^2 = -12$)
- The arithmetic right shift by N divides a number by 2^N
 - Ex: $01000 \ggg 2 = 00010$ ($8 \div 2^2 = 2$)
 - Ex: $10000 \ggg 2 = 11100$ ($-16 \div 2^2 = -4$)

Standard Sequential Modules

CS 140 Lecture 15 Sequential Modules

Professor CK Cheng
CSE Dept.
UC San Diego

1. Serial Adders
2. Serial Multipliers
3. Register
4. Counter

1

2

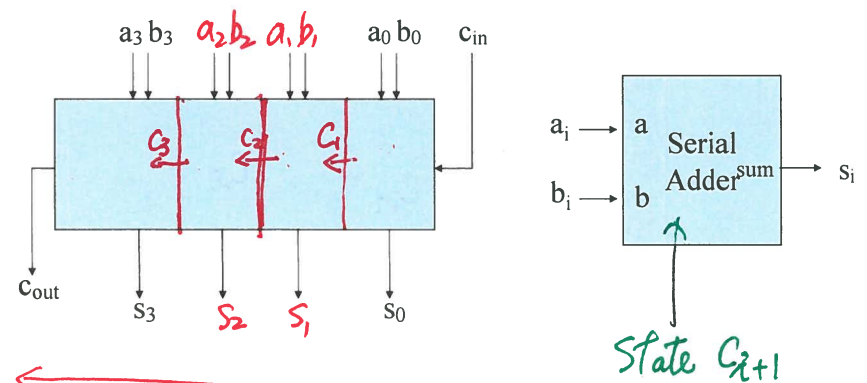
Motivation for Serial Adders and Multipliers

- Tradeoff of silicon area and system performance
 - Perform process in a series of time
- Utilization of FPGA architecture
 - Slice operation bitwise
- Metrics of Cost, Speed, and Power
- Ad: Cheaper hardware, Fit for FPGA architecture, Pipelining for excellent throughput
- Dis: Longer latency

3

Serial Adder: Perform serial bit-addition

At time i , read a_i and b_i . Produce s_i and c_{i+1}
Internal state stores c_i . Carry bit c_0 is set as c_{in}

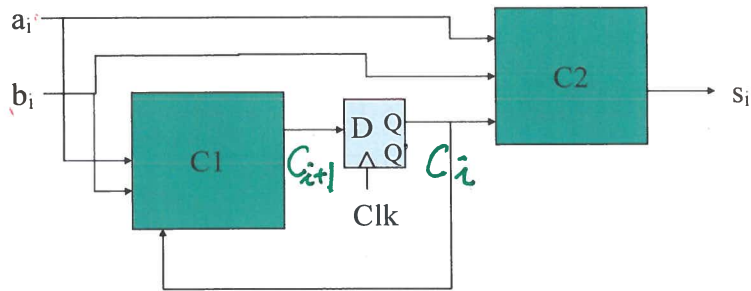


Space

Time

4

Serial Adder using D F-F



Feed a_i and b_i and generate s_i at time i .
Where is c_i and c_{i+1} ?

$$\begin{array}{r} 5 \\ + 7 \\ \hline 12 \end{array}$$

$$\begin{array}{r} 1100 \\ + 1111 \\ \hline 11000 \end{array}$$

$c_{in}=0$

5

Serial Adder using a D Flip-Flop

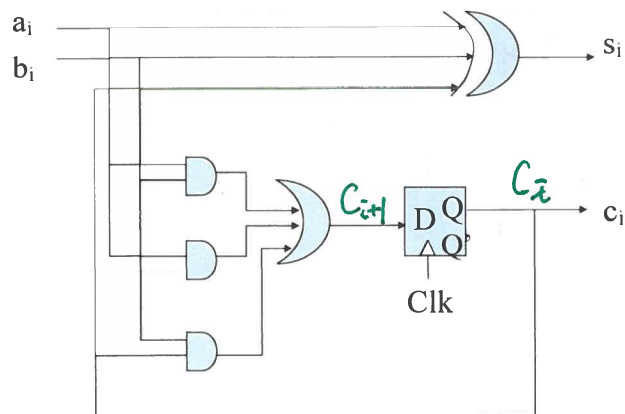
id	a_i	b_i	c_i	c_{i+1}	s_i
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

$$D=c_{i+1}$$

$$Q=c_i$$

6

Serial Adder using a D Flip-Flop Logic Diagram



7

Multiplication using Serial Addition

$$3 \times 5 = 15$$

$$\begin{array}{r} 011 \\ \times 101 \\ \hline 011 \\ 000 \\ + 011 \\ \hline 01111 \end{array}$$

$$\begin{array}{r} a_2 a_1 a_0 \\ \times b_2 b_1 b_0 \\ \hline a_2 b_0 a_1 b_0 a_0 b_0 \\ a_2 b_1 a_1 b_1 a_0 b_1 \\ + a_2 b_2 a_1 b_2 a_0 b_2 \\ \hline m_5 m_4 m_3 m_2 m_1 m_0 \end{array}$$

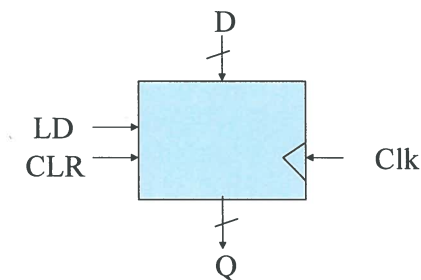
For $m=A \times B$, set $m^{(0)}=0$

At time i , perform $m^{(i+1)}=m^{(i)}+A b_i 2^i$

$\uparrow$
 i th bit of B

8

Register



CLR: initialize the content

Priority: CLR
2. LD

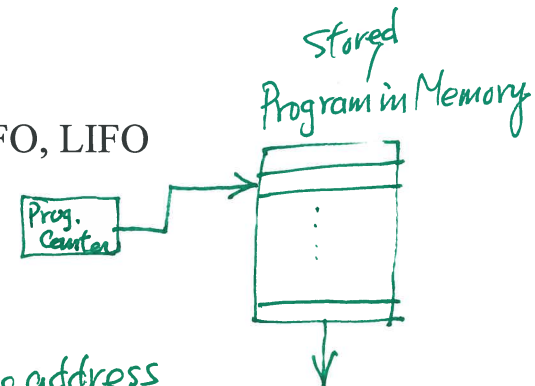
$$Q(t+1) = (0, 0, \dots, 0) \\ = D \\ = Q(t)$$

if CLR = 1
if LD = 1 and CLR = 0
if LD = 0 and CLR = 0

9

Counter

- Program Counter
- Address Keeper: FIFO, LIFO
- Clock Divider
- Sequential Machine



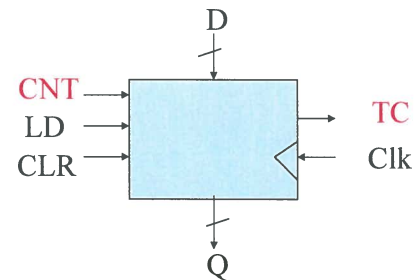
Program Counter:
Increment the address
by one word (instruction)
as a default.

10

Counter

- Modulo-n Counter
- Modulo Counter ($m < n$)
- Counter (a-to-b)
- Counter of an Arbitrary Sequence
- Cascade Counter

Modulo-n Counter



Priority
1. CLR
2. LD
3. CNT

$$Q(t+1) = (0, 0, \dots, 0) \quad \text{if CLR} = 1 \\ = D \quad \text{if LD} = 1 \text{ and CLR} = 0 \\ = (Q(t)+1) \bmod n \quad \text{if LD} = 0, \text{ CNT} = 1 \text{ and CLR} = 0 \\ = Q(t) \quad \text{if LD} = 0, \text{ CNT} = 0 \text{ and CLR} = 0$$

$$TC = 1 \quad \text{if } Q(t) = n-1 \text{ and CNT} = 1 \\ = 0 \quad \text{otherwise}$$

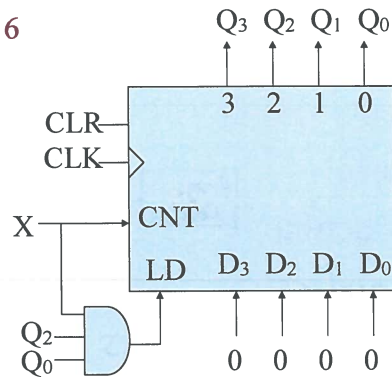
11

12

Modulo-m Counter ($m < n$)

Given a mod 16 counter, construct
a mod-m counter ($0 < m < 16$) with AND, OR, NOT gates

$m = 6$



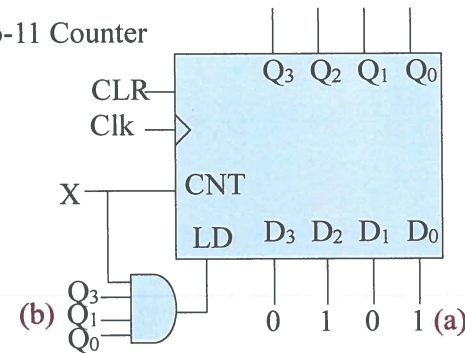
Set LD = 1 when $X = 1$ and $(Q_3Q_2Q_1Q_0) = (0101)$, ie $m-1$

13

Counter (a-to-b)

Given a mod 16 counter, construct an a-to-b counter
($0 \leq a \leq b \leq 15$)

A 5-to-11 Counter



Set LD = 1 when $X = 1$ and $(Q_3Q_2Q_1Q_0) = b$ (in this case, 1011)

14

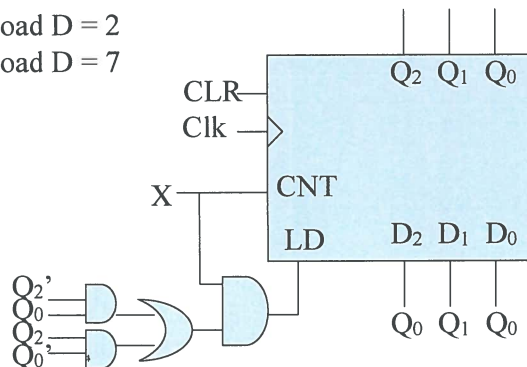
Counter of an Arbitrary Sequence

Given a mod 8 counter, construct
a counter with sequence 0 1 5 6 2 3 7

When $Q = 1$, load $D = 5$

When $Q = 6$, load $D = 2$

When $Q = 3$, load $D = 7$



15

Counter of an Arbitrary Sequence

Given a mod 8 counter, construct
a counter with sequence 0 1 5 6 2 3 7

Id	$Q_2Q_1Q_0$	LD	D_2	D_1	D_0
0	000	0	-	-	-
1	001	1	1	0	1
2	010	0	-	-	-
3	011	1	1	1	1
4	100	-	-	-	-
5	101	0	-	-	-
6	110	1	0	1	0
7	111	0	-	-	-

K Mapping LD and D ,
we get

$$LD = Q_2'Q_0 + Q_2Q_0'$$

$$D_2 = Q_0$$

$$D_1 = Q_1$$

$$D_0 = Q_0$$

16