CSE 140 Lecture 12
Combinational Standard Modules

CK Cheng

CSE Dept.

UC San Diego

Part III. Standard Modules

Interconnect Modules:

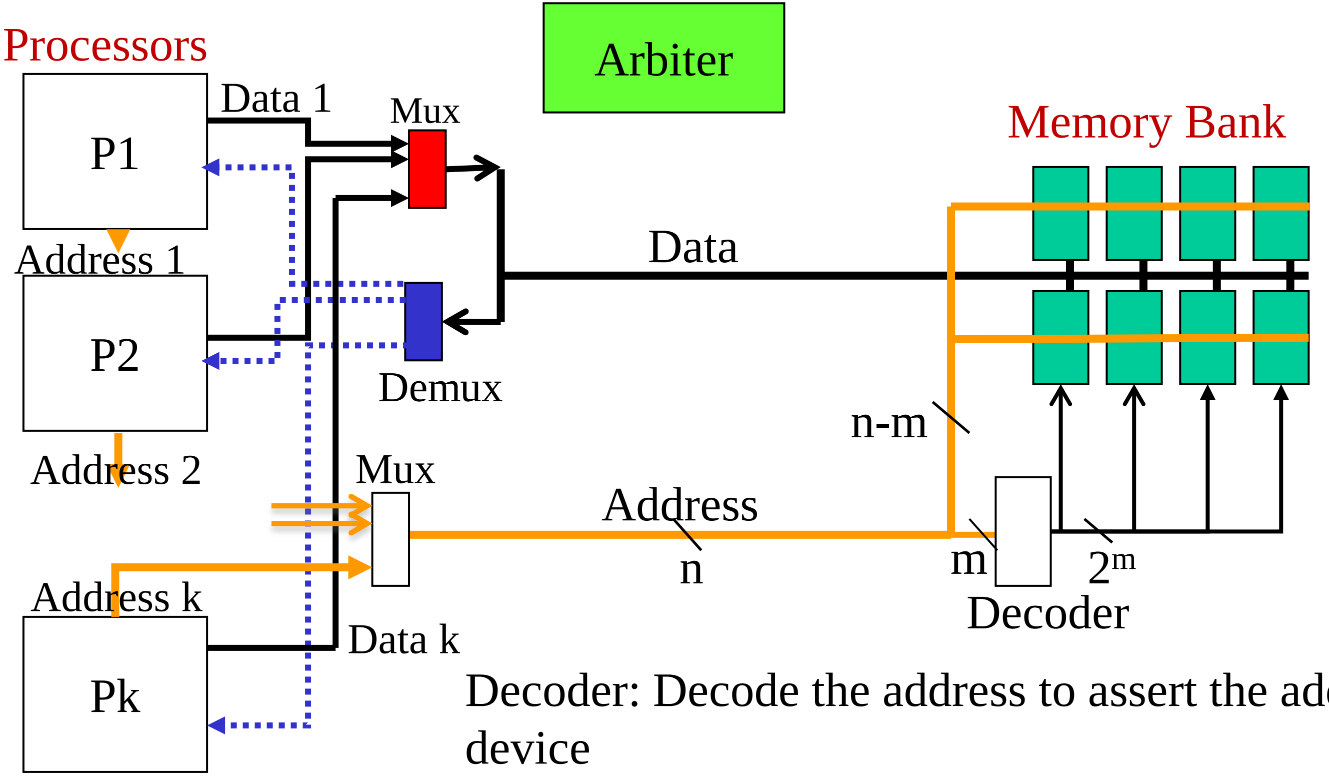
1. Decoder, 2. Encoder

3. Multiplexer, 4. Demultiplexer

Multiplexer

- Definition
- Logic Diagram
- Application

Interconnect: Decoder, Encoder, Mux, DeMux



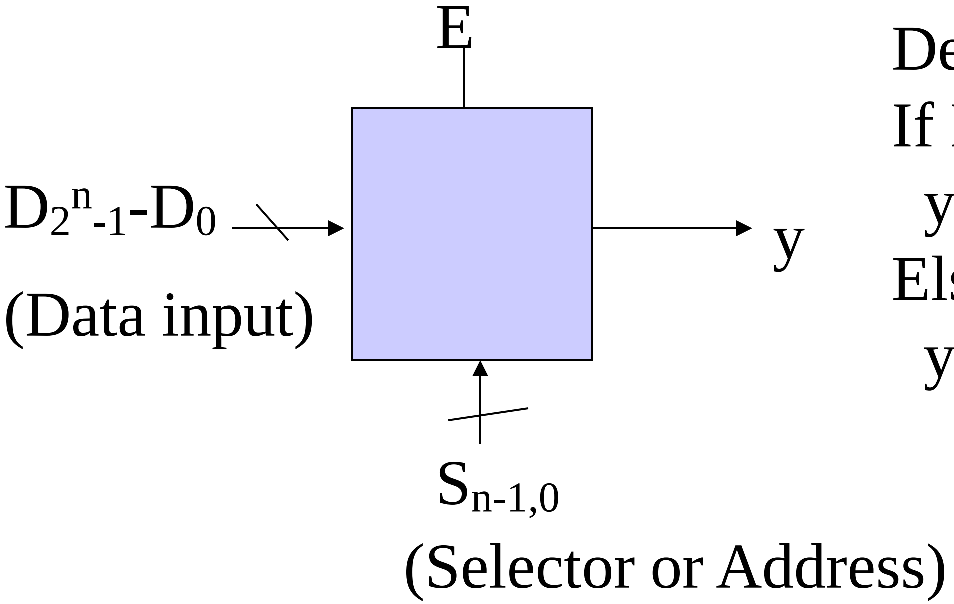
Decoder: Decode the address to assert the addressed device

Mux: Select the inputs according to the index addressed by the control signals

iClicker: 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.



Description

If $E = 1$

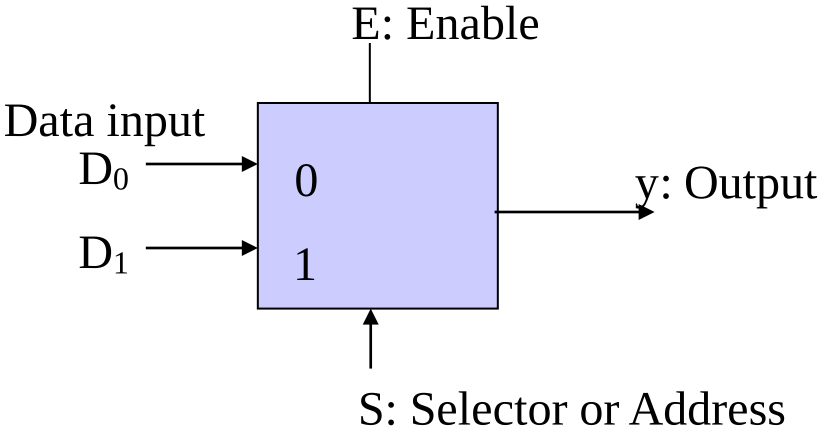
  $y = D_i$ where $i = (S_{n-1}, .. , S_0)$

Else

  $y = 0$

Multiplexer (Mux): Definition

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

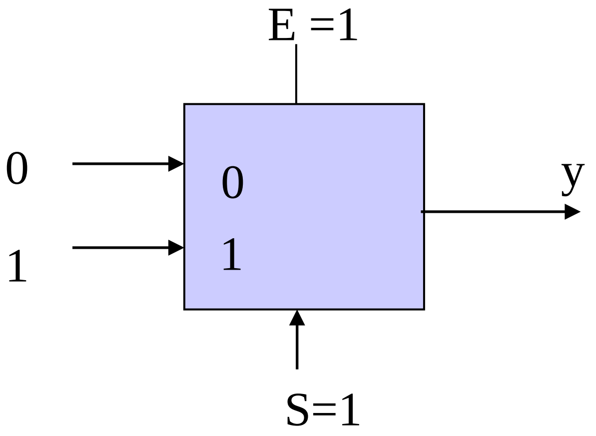


PI Q: What is the output of the following MUX?

A. 0

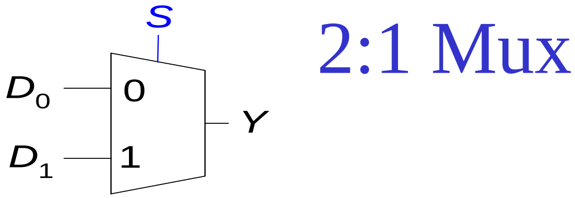
B. 1

C. Can’t say



Multiplexer (Mux): Definition

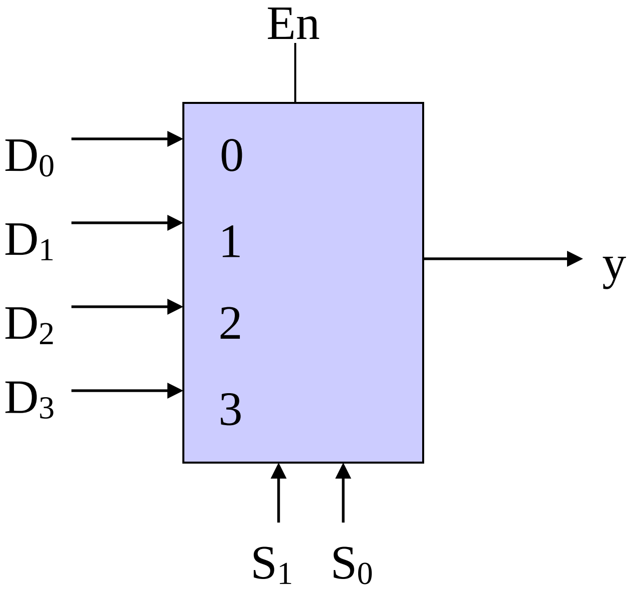
- Selects between one of N inputs to connect to the output.
- $\log_2 N$-bit select input – control input
- Example:



S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

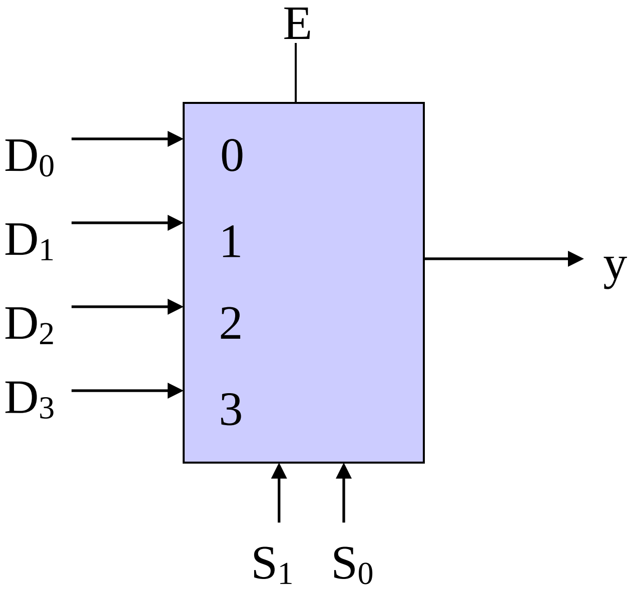
S	Y
0	D_0
1	D_1

Multiplexer Definition: Example



S_1	S_0	y

Multiplexer Definition: Example



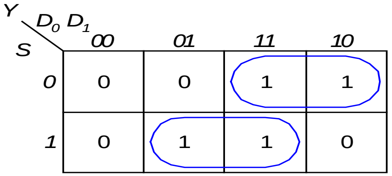
E=1:

If $D_0 = 0$ and $S_1S_0 = 00$ => $y = 0$

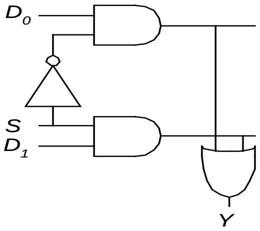
If $D_0 = 1$ and $S_1S_0 = 00$ => $y = 1$

Multiplexer: Logic Diagram

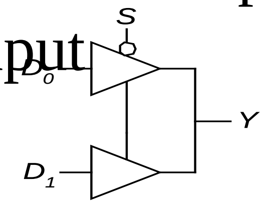
- Logic gates
  - Sum-of-products form



$Y = D_0\overline{S} + D_1S$



- Tristates
  - For an N-input mux, use N tristates
  - Turn on exactly one to select the appropriate input

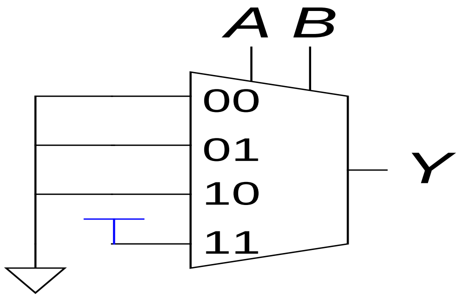


Multiplexer Application

- Mux for a Boolean function with truth table as input

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$$Y = AB$$

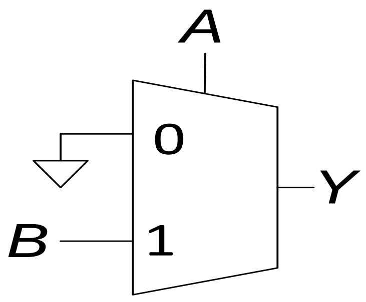


Multiplexer: Application

$Y = AB$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

A	Y
0	0
1	B



Multiplexer Application: universal set {Mux}

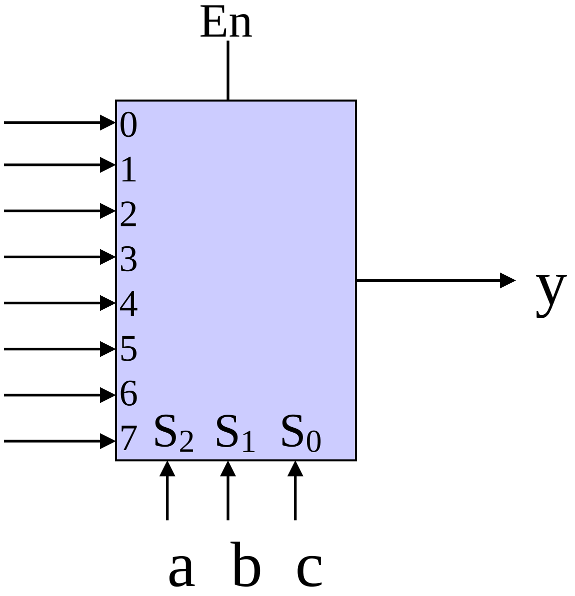
We use selector to decompose the function into smaller functions (less number of variables), which follows Shannon's expansion.

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

Multiplexer Application: universal set {Mux}

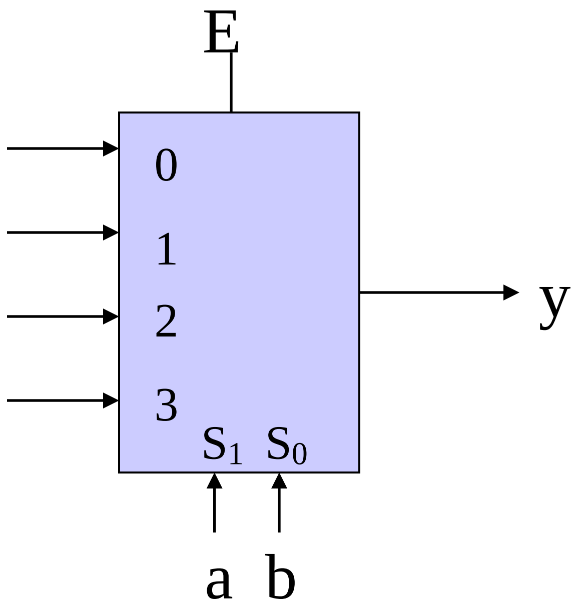
Example 1: Given f (a,b,c) = Σm (0,1,7) + $\Sigma d(2)$, implement with an 8-input Mux.

Id	a	b	c	f
0	0	0	0	1
1	0	0	1	1
2	0	1	0	-
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1



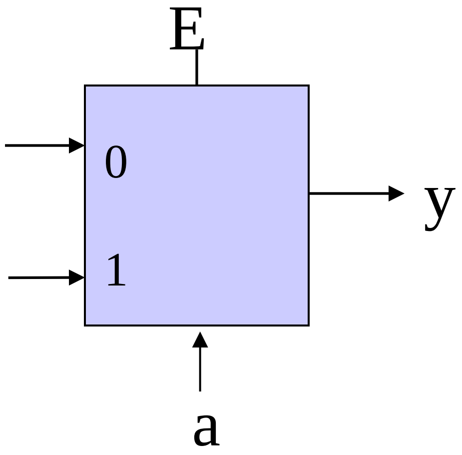
Example 2: Given f (a,b,c) = $\Sigma m\,(0,1,7) + \Sigma d(2)$, implement with 4-input Muxes.

a	b	c = 0	c = 1	D (c)
0	0			D_0 (c) =
0	1			D_1 (c) =
1	0			D_2 (c) =
1	1			D_3 (c) =



Example 3: Given f (a,b,c) = Σm (0,1,7) + Σd(2), implement with 2-input Muxes.

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



Example 3: Given f (a,b,c) = Σm (0,1,7) + Σd(2), implement with 2-input Muxes.

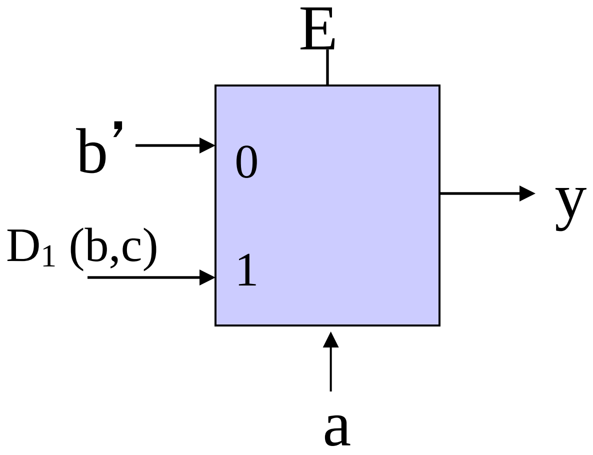
a	00	01	10	11	D (b,c)
0	1	1	-	0	D_0 (b,c)
1	0	0	0	1	D_1 (b,c)

D_0 (b,c) = b’

		b
	1	-
c	1	0

D_1 (b,c) = bc

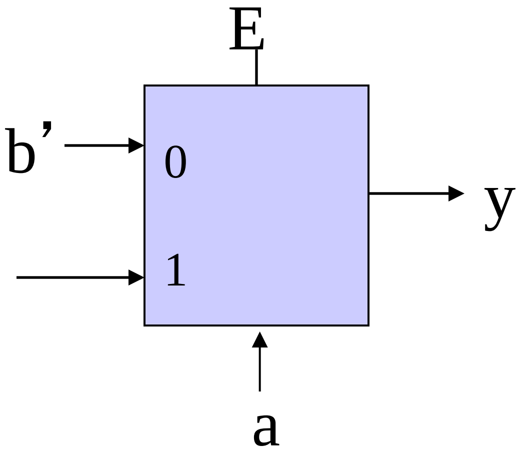
		b
	0	0
c	0	1



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

D_1 (b,c)

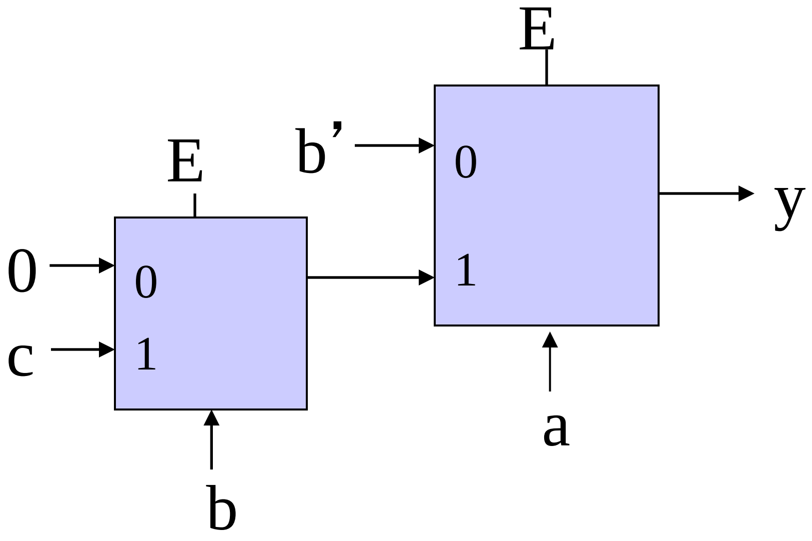
b	c = 0	c = 1	
0	0	0	$l_1(0) = 0$
1	0	1	$l_1(c) = c$



Example 3: Given f (a,b,c) = Σm (0,1,7) + Σd(2), implement with 2-input Muxes.

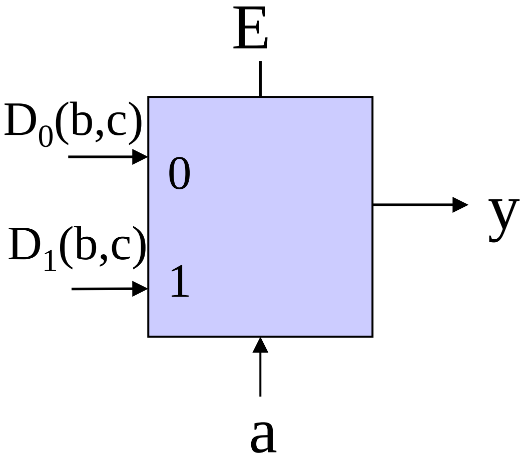
D_1 (b,c)

b	c = 0	c = 1	
0	0	0	$l_1(0) = 0$
1	0	1	$l_1(c) = c$

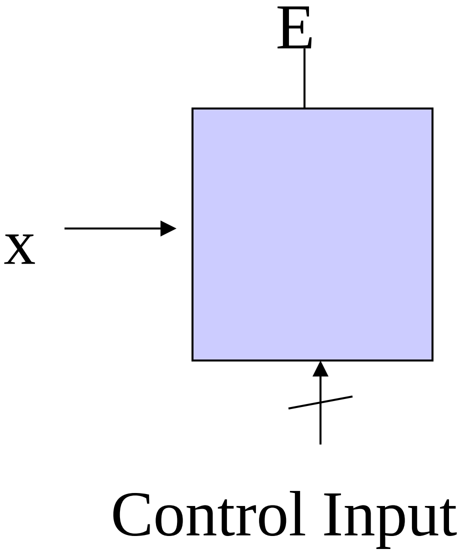


Example 4: Given f (a,b,c) = Σm (0,2,4,7) + Σd(3,5), implement with 2-input Muxes.

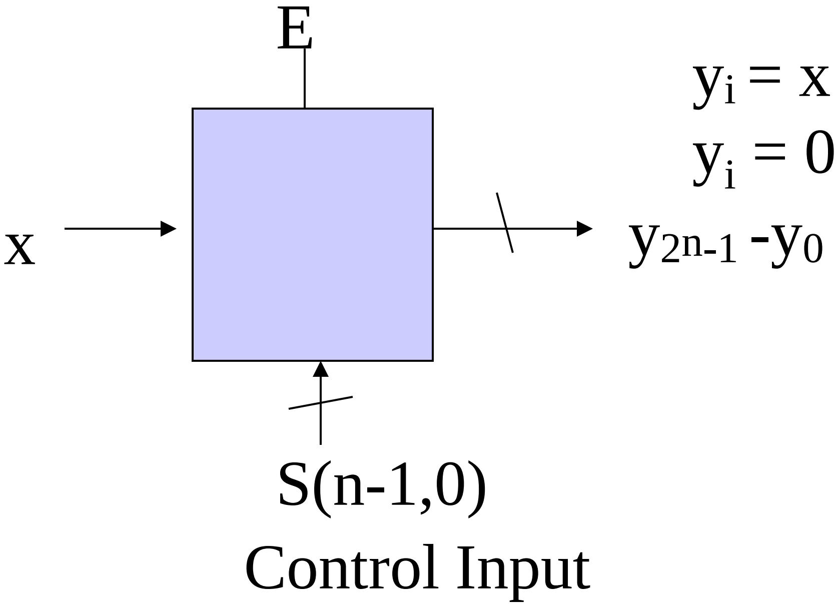
f (a \ bc)	00	01	10	11	
0	1	0	1	-	$D_0(b,c)$
1	1	-	0	1	$D_1(b,c)$



4. Demultiplexers



4. Demultiplexers



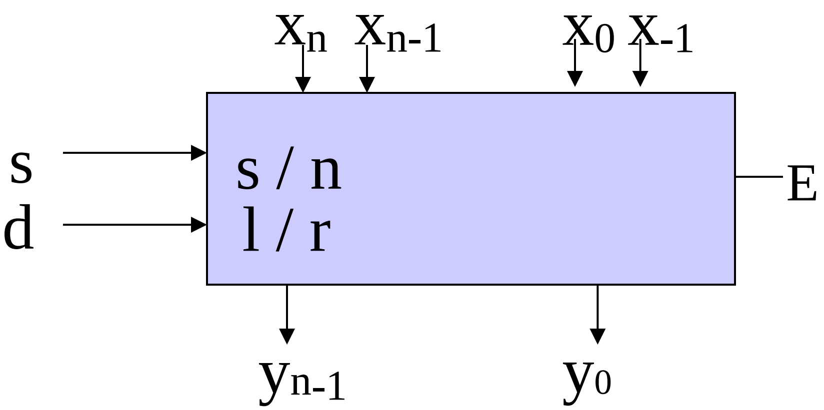
$y_i = x$ if $i = (S_{n-1}, .. , S_0)$ & E=1

$y_i = 0$ otherwise

Shifters

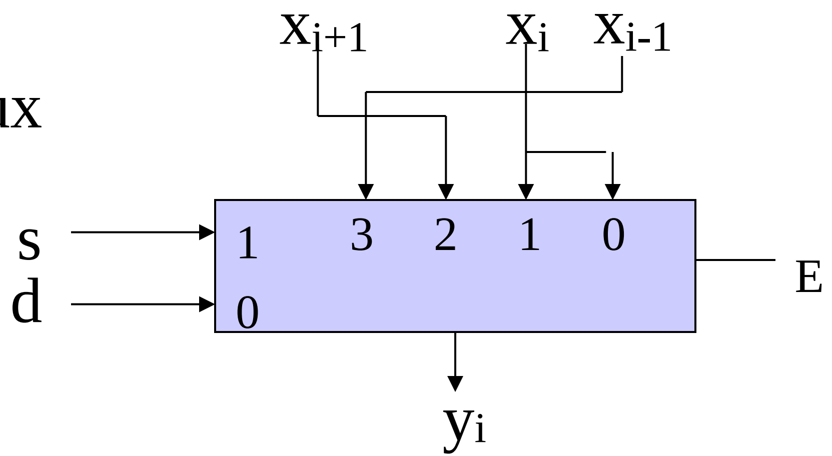
- **Logical shifter:** shifts value to left or right and fills empty spaces with 0's
  - Ex: 11001 >> 2 = 00110
  - Ex: 11001 << 2 = 00100
- **Arithmetic shifter:** same as logical shifter, but on right shift, fills empty spaces with the old most significant bit (msb).
  - Ex: 11001 >>> 2 = **11**110
  - Ex: 11001 <<< 2 = 00100
- **Rotator:** rotates bits in a circle, such that bits shifted off one end are shifted into the other end
  - Ex: 11001 ROR 2 = 01110
  - Ex: 11001 ROL 2 = 00111

Shifter

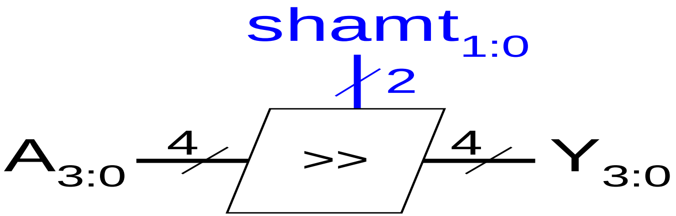


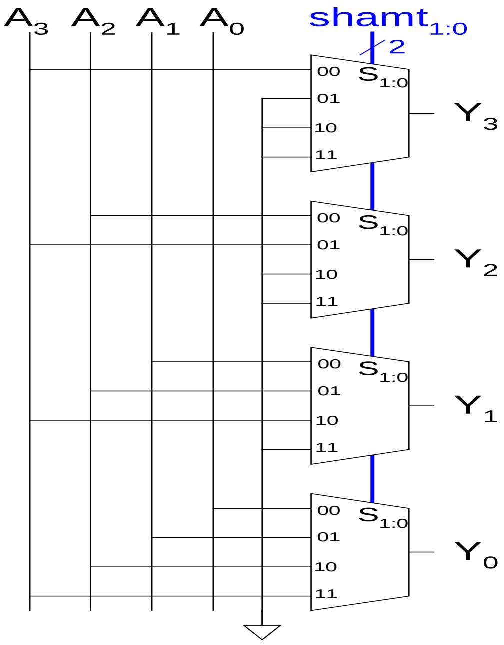
$$
\begin{aligned}
y_i &= x_{i-1} \text{ if } E = 1,\ s = 1, \text{ and } d = L \\
&= x_{i+1} \text{ if } E = 1,\ s = 1, \text{ and } d = R \\
&= x_i \quad \text{if } E = 1,\ s = 0 \\
&= 0 \quad \text{if } E = 0
\end{aligned}
$$

Can be implemented with a mux

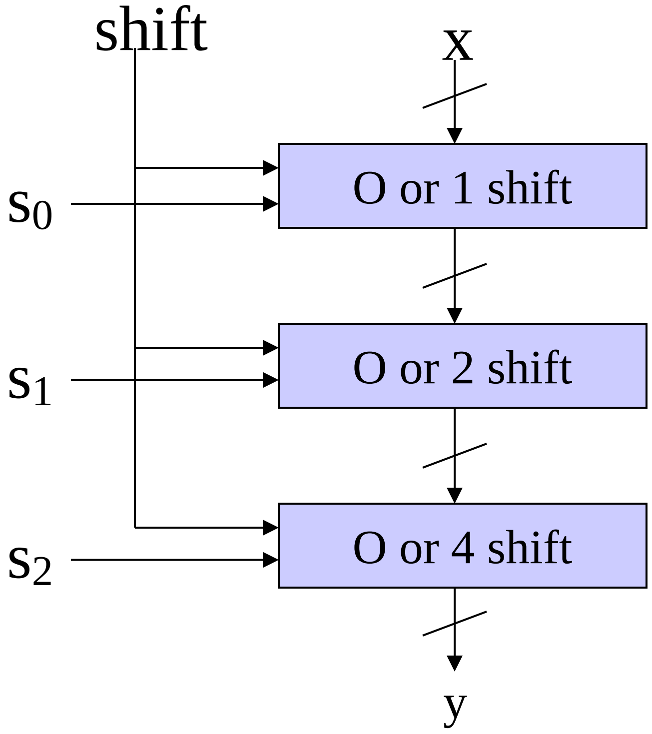


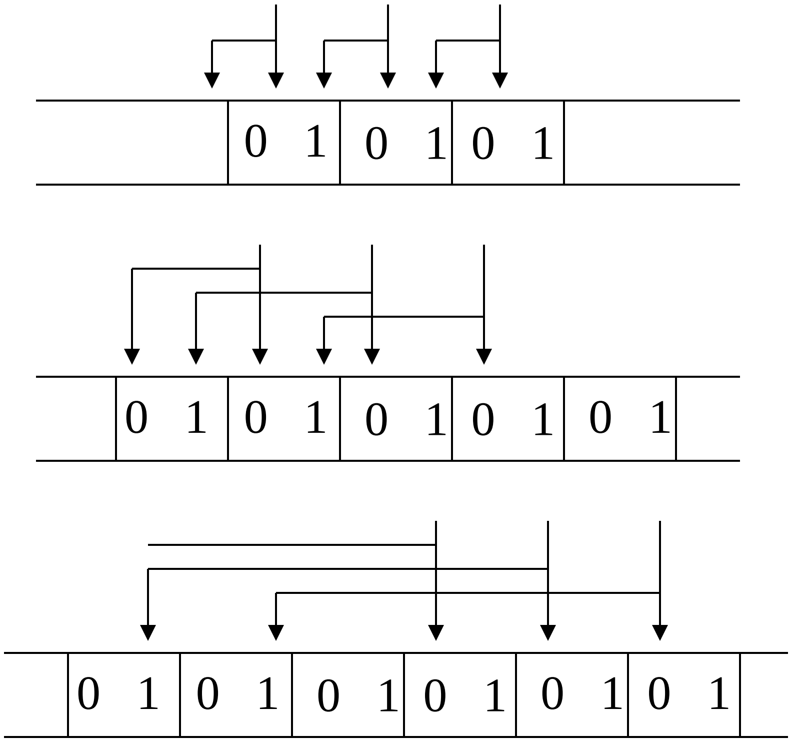
Shifter Design





Barrel Shifter





Shifters as Multipliers and Dividers

- A left shift by N bits multiplies a number by 2^N
  - Ex: 00001 << 2 = 00100 ($1 \times 2^2 = 4$)
  - Ex: 11101 << 2 = 10100 ($-3 \times 2^2 = -12$)

- The arithmetic right shift by N divides a number by 2^N
  - Ex: 01000 >>> 2 = 00010 ($8 \div 2^2 = 2$)
  - Ex: 10000 >>> 2 = 11100 ($-16 \div 2^2 = -4$)