# Digital Logic Design: a rigorous approach ©

## Chapter 14: Shifters

part 1 - shifters

Guy Even Moti Medina

School of Electrical Engineering Tel-Aviv Univ.

May 13, 2020

Book Homepage:
http://www.eng.tau.ac.il/~guy/Even-Medina

# Preliminary questions:

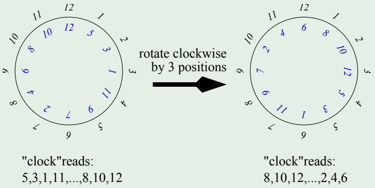
1. Which types of shifts are you familiar with in your favorite programming language? What is the differences between these shifts? Why do we need different types of shifts?
2. How are these shifts executed in a microprocessor?
3. Should shifters be considered to be combinational circuits? After all, they simply "move bits around" and do not compute "new bits".

C/python: A << 5 A >> 5

# Cyclic Shifters

## Example

- assume that we place the bits of $a[1:12]$ on a wheel.
- $a[1]$ is at one o'clock, $a[2]$ is at two o'clock, etc.
- rotate the wheel, and read the bits in clockwise order starting from one o'clock and ending at twelve o'clock.
- the resulting string is a cyclic shift of $a[1:12]$.



# Definition of a Cyclic Shifter

We denote ($a$ mod $b$) by $\mathrm{mod}(a, b)$.

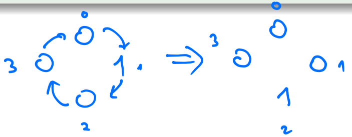
## Definition

The string $b[n-1:0]$ is a cyclic left shift by $i$ positions of the string $a[n-1:0]$ if

$$\forall j : \quad b[j] = a[\mathrm{mod}(j-i, n)].$$

## Example

Let $a[3:0] = 0010$. A cyclic left shift by one position of $\vec{a}$ is the string $0100$. A cyclic left shift by 3 positions of $\vec{a}$ is the string $0001$.



# Definition of $\text{BARREL-SHIFTER}(n)$

## Definition

A $\text{BARREL-SHIFTER}(n)$ is a combinational circuit defined as follows:

**Input:** $x[n-1:0] \in \{0,1\}^n$ and $sa[k-1:0] \in \{0,1\}^k$ where $k = \lceil \log_2 n \rceil$.

**Output:** $y[n-1:0] \in \{0,1\}^n$.

**Functionality:** $\vec{y}$ is a cyclic left shift of $\vec{x}$ by $\langle \vec{sa} \rangle$ positions. Formally,

$$\forall j \in [n-1:0]: \quad y[j] = x[\mathrm{mod}(j - \langle \vec{sa} \rangle, n)].$$

We often refer to the input $\vec{x}$ as the data input and to the input $\vec{sa}$ as the shift amount input. **To simplify the discussion, we assume that $n$ is a power of 2, namely, $n = 2^k$.**

We break the task of designing a barrel shifter into smaller sub-tasks of shifting by powers of two. We define this sub-task formally as follows.

A CLS($n, 2^i$) is a combinational circuit that implements a cyclic left shift by zero or $2^i$ positions depending on the value of its select input.

**Definition**

A CLS($n, i$) is a combinational circuit defined as follows:

Input: $x[n-1:0]$ and $s \in \{0,1\}$.

Output: $y[n-1:0]$.

Functionality:

$$\forall j \in [n-1:0]: \quad y[j] = x[\mathrm{mod}(j - s \cdot i, n)].$$

$$y[j] = \begin{cases} x[j] & \text{if } s=0 \\ x[\mathrm{mod}(j-i,n)] & \text{if } s=1 \end{cases}$$

# Subtask: $\textsc{cls}(n, i)$ Implementation

A $\textsc{cls}(n, i)$ is quite simple to implement since:

- $y[j]$ is either $x[j]$ or $x[\mathrm{mod}(j - i, n)]$.
- So all one needs is a $\textsc{mux}$-gate to select between $x[j]$ or $x[\mathrm{mod}(j - i, n)]$.
- The selection is based on the value of $s$.
- It follows that the delay of $\textsc{cls}(n, i)$ is the delay of a $\textsc{mux}$, and the cost is $n$ times the cost of a $\textsc{mux}$.

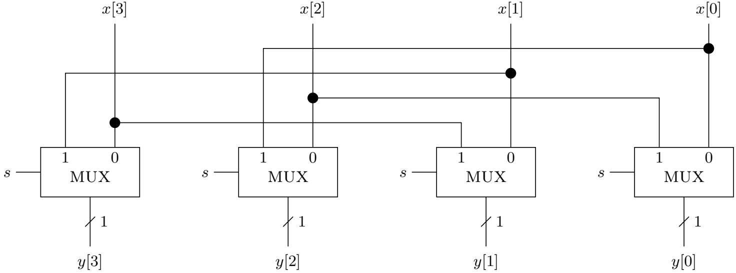
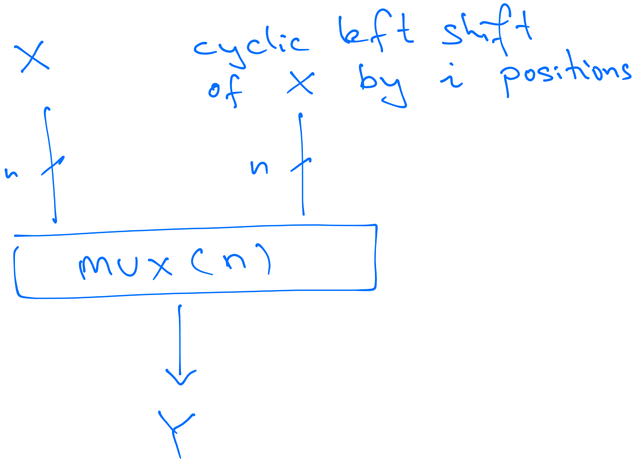



Figure: A row of multiplexers implement a $\textsc{cls}(4, 2)$.

# CLS(n, i)



# Back to BARREL-SHIFTER($n$)

- The design of a BARREL-SHIFTER($n$) is based on CLS($n, 2^i$) shifters.
- The implementation is based on $k$ levels of CLS($n, 2^i$), for $i \in [k-1:0]$.
- The $i$th level is controlled by $sa[i]$.

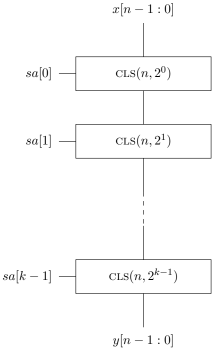



Figure: A BARREL-SHIFTER($n$) built of $k$ levels of CLS($n, 2^i$) ($n = 2^k$).

# Correctness

## Observation

*For every $x, q \in \mathbb{Z}$,*

exercise

$$\mathrm{mod}(x, n) = \mathrm{mod}(x + qn, n).$$

## Observation

*If $\alpha = \mathrm{mod}(a, n)$ and $\beta = \mathrm{mod}(b, n)$, then*

* CH 5 (bin. repr)
slide #5

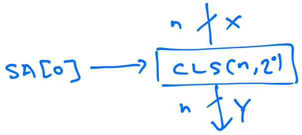
$$\mathrm{mod}(a - b, n) = \mathrm{mod}(\alpha - \beta, n)\ .$$

## Claim

*The barrel shifter design depicted in the previous slide is correct.*

## Proof.

Prove by induction on $i$, that output of $\textsc{cls}(n, 2^i)$ equals the cyclic left shift of $x$ by $\langle sa[i:0] \rangle$. $\square$

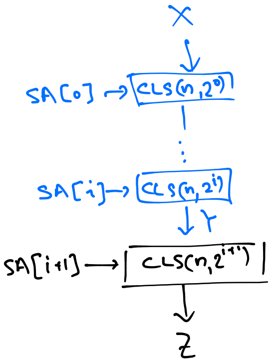



**basis:**

$i = 0$ $\quad Y$ = shift of $X$ by SA[0] positions

**hyp:**

$Y$ = shift of $X$ by $\langle SA[i:0] \rangle$ pos.




**step:**

$Z$ = shift of $Y$ by $2^{i+1} \cdot SA[i+1]$ pos.

= shift of $X$ by $\langle SA[i:0] \rangle + 2^{i+1} \cdot SA[i+1]$ pos. = shift by $\langle SA[i+1:0] \rangle$ pos. ☒

# Cost & Delay

## Claim

*The cost and delay of* $\text{BARREL-SHIFTER}(n)$ *satisfy:*

$$c(\text{BARREL-SHIFTER}(n)) = n\log_2 n \cdot c(\text{MUX})$$
$$d(\text{BARREL-SHIFTER}(n)) = \log_2 n \cdot d(\text{MUX}).$$

## Proof.

Follows from the fact that the design consists of $\log_2 n$ levels of $\text{CLS}(n, 2^i)$ shifters. □

# The cone of the Barrel Shifter

Consider the output $y[0]$ of BARREL-SHIFTER$(n)$ .

**Claim**

*The cone of the Boolean function implemented by the output $y[0]$ contains at least $n$ elements.*

**Corollary**

*The delay of* BARREL-SHIFTER$(n)$ *is asymptotically optimal.*

$x[i] \in cone(y[0])$ : $\langle SA \rangle = n - i \Rightarrow y[0] = x[i]$

# Logical Shift

## Definition

The binary string $y[n-1:0]$ is a logical left shift by $\ell$ positions of the binary string $x[n-1:0]$ if

$\boxed{x}\,00000$

left shift

$$y[i] \triangleq \begin{cases} 0 & \text{if } i < \ell \\ x[i-\ell] & \text{if } \ell \le i < n. \end{cases}$$

## Example

$y[3:0] = 0100$ is a logical left shift of $x[3:0] = 1001$ by $\ell = 2$ positions. When we apply a logical left shift to $x[n-1:0]$ by $\ell$ positions, we obtain the string $x[n-1-\ell:0] \circ 0^{\ell}$.

## Fast multiplication

In binary representation, logical shifting to the left by $s$ positions corresponds to multiplying by $2^s$ followed by modulo $2^n$.

# Logical Shifters (cont.)

## Definition

The binary string $y[n-1:0]$ is a logical right shift by $\ell$ positions of the binary string $x[n-1:0]$ if

MSB LSB

0000

right

$$y[i] \triangleq \begin{cases} 0 & \text{if } i \geq n-\ell \\ x[i+\ell] & \text{if } 0 \leq i < n-\ell. \end{cases}$$

## Example

$y[3:0] = 0010$ is a logical right shift of $x[3:0] = 1001$ by $\ell = 2$ positions. When we apply a logical right shift to $x[n-1:0]$ by $\ell$ positions, we obtain the string $0^{\ell} \circ x[n-1:\ell]$.

## Fast division

In binary representation, logical shifting to the right by $s$ positions corresponds to the integer part of the quotient after division by $2^s$.

# Notation.

- Let $\textsc{lls}(\vec{x}, i)$ denote the logical left shift of $\vec{x}$ by $i$ positions.
- Let $\textsc{lrs}(\vec{x}, i)$ denote the logical right shift of $\vec{x}$ by $i$ positions.

# A bi-directional logical shifter

## Definition

A L-SHIFT$(n)$ is a combinational circuit defined as follows:

Input:

- $x[n-1:0] \in \{0,1\}^n$,
- $sa[k-1:0] \in \{0,1\}^k$, where $k = \lceil \log_2 n \rceil$, and
- $\ell \in \{0,1\}$.

$\ell=1$ left shift
$\ell=0$ right shift

Output: $y[n-1:0] \in \{0,1\}^n$.

Functionality: The output $\vec{y}$ satisfies

$$\vec{y} \triangleq \begin{cases} \text{LLS}(\vec{x}, \langle \vec{sa} \rangle) & \text{if } \ell = 1, \\ \text{LRS}(\vec{x}, \langle \vec{sa} \rangle) & \text{if } \ell = 0. \end{cases}$$

## Question

Design a bi-directional shifter using a left shifter and a right shifter (and select the answer based on $\ell$).

# A bi-directional logical shifter (cont.)

**Example**

- let $x[3:0] = 0010$.
- If $sa[1:0] = 10$ and $\ell = 1$, then $\textsc{l-shift}(4)$ outputs $y[3:0] = 1000$.
- If $\ell = 0$, then the output equals $y[3:0] = 0000$.

# Implementation

As in the case of cyclic shifters, we break the task of designing a logical shifter into sub-tasks of logical shifts by powers of two.

## Definition

An $\text{LBS}(n, i)$ is a combinational circuit defined as follows:

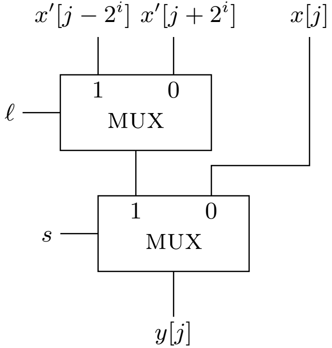
**Input:** $x[n-1:0]$ and $s, \ell \in \{0, 1\}$.

**Output:** $y[n-1:0]$.

**Functionality:** The output $\vec{y}$ satisfies

$$\vec{y} \triangleq \begin{cases} \vec{x} & \text{if } s = 0, \\ \text{LLS}(\vec{x}, i) & \text{if } s = 1 \text{ and } \ell = 1, \\ \text{LRS}(\vec{x}, i) & \text{if } s = 1 \text{ and } \ell = 0. \end{cases}$$

The role of the input $s$ in is to determine if a shift (in either direction) takes place at all. If $s = 0$, then $y[j] = x[j]$, and no shift takes place. If $s = 1$, then the direction of the shift is determined by $\ell$.



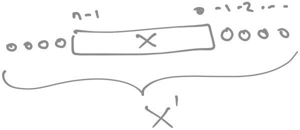




Figure: A bit-slice of an implementation of $\text{LBS}(n, 2^i)$.

**question**

Design a bi-directional logical shifter $\text{L-SHIFT}(n)$ by cascading $\text{LBS}(n, 2^i)$ shifters.

# Reduction of right shift to left shift

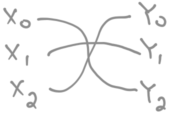
**Definition**

Let $rev : \{0,1\}^* \to \{0,1\}^*$ denote the function that reverses strings. Formally:

$$rev(A_{n-1}, \ldots, A_1, A_0) = (A_0, A_1, \ldots, A_n).$$

Reversing a string can be implemented with zero cost and zero delay. All one needs to do is connect input $A[i]$ to the output $B[n-i]$.

$rev(011) = 110$

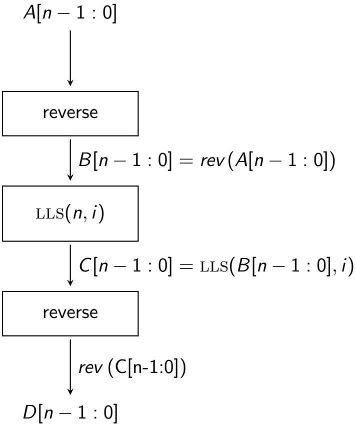



zero cost
zero delay

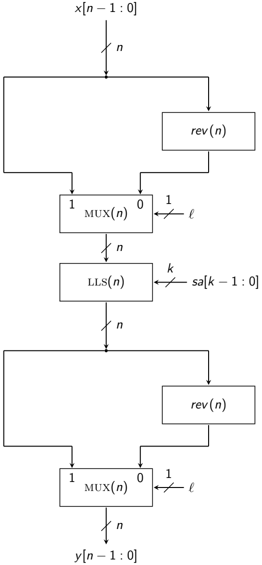
# Reduction of right shift to left shift (cont.)

**Claim**

$$\text{LRS}(\vec{x}, i) = rev(\text{LLS}(rev(\vec{x}), i)).$$



bi-directional
logical
shifter
(left/right)

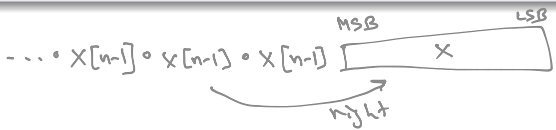


Arithmetic shifters are used for shifting binary strings that represent signed integers in two's complement representation. **Since left shifting is the same in logical shifting and in arithmetic shifting, we discuss only right shifting** (i.e., division by a power of 2).

### Definition

The binary string $y[n-1:0]$ is an arithmetic right shift by $\ell$ positions of the binary string $x[n-1:0]$ if the following holds:

$$y[i] \triangleq \begin{cases} x[n-1] & \text{if } i \geq n-\ell \\ x[i+\ell] & \text{if } 0 \leq i < n-\ell. \end{cases}$$



**Example**

- $y[3:0] = 0010$ is an arithmetic shift of $x[3:0] = 0101$ by $\ell = -1$ positions.
- On the other hand, $y[3:0] = 1110$ is an arithmetic shift of $x[3:0] = 1001$ by $\ell = -2$ positions.
- When we apply an arithmetic shift by $\ell < 0$ positions to $x[n-1:0]$, we obtain the string $x[n-1]^{\ell} \circ x[n-1:\ell]$.

**Notation.**
Let $\text{ARS}(\vec{x}, i)$ denote the arithmetic right shift of $\vec{x}$ by $i$ positions.

# An arithmetic right shifter

## Definition

An $\text{ARITH-SHIFT}(n)$ is a combinational circuit defined as follows:

**Input:** $x[n-1:0] \in \{0,1\}^n$ and $sa[k-1:0] \in \{0,1\}^k$, where $k = \lceil \log_2 n \rceil$.

**Output:** $y[n-1:0] \in \{0,1\}^n$.

**Functionality:** The output $\vec{y}$ is a (sign-extended) arithmetic right shift of $\vec{x}$ by $\langle \vec{sa} \rangle$ positions. Formally,

$$y[n-1:0] \triangleq \text{ARS}(x[n-1:0], \langle \vec{sa} \rangle).$$

## Example

Let $x[3:0] = 1001$. If $sa[1:0] = 10$, then $\text{ARITH-SHIFT}(4)$ outputs $y[3:0] = 1110$.

## question

Design an arithmetic right shifter $\text{ARITH-SHIFT}(n)$.