Chapter 5: Selectors and Shifters

Computer Structure - Spring 2004

Dr. Guy Even

Tel-Aviv Univ.

Goals

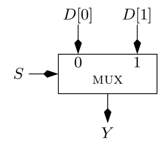
- Selectors:
  - review definition of multiplexer.
  - build $(n:1)$-multiplexers.
- Shifters:
  - Cyclic shifter (Barrel shifter)
  - Logical Shifter
  - Arithmetic Shifter

Multiplexer

DEF: A MUX-gate (also known as a $(2:1)$-multiplexer) is a combinational gate that has three inputs $D[0], D[1], S$ and one output Y. The functionality is defined by

$$Y = \begin{cases} D[0] & \text{if } S = 0 \\ D[1] & \text{if } S = 1. \end{cases}$$

Equivalently: $Y = D[S]$



Selectors

DEF: An (n:1)-MUX is a combinational circuit defined as follows:

Input: $D[n-1:0]$ and $S[k-1:0]$ where $k = \lceil \log_2 n \rceil$.

Output: $Y \in \{0,1\}$.

Functionality:

$$Y = D[\langle \vec{S} \rangle].$$

Example: Let $n = 4$, $D[3:0] = 0101$, and $S[1:0] = 11$. The output Y should be 1.

- $\vec{D}$ - data input
- $\vec{S}$ - select input
- simplify: assume that n is a power of 2, namely, $n = 2^k$.

Implementation of (n:1)-MUX

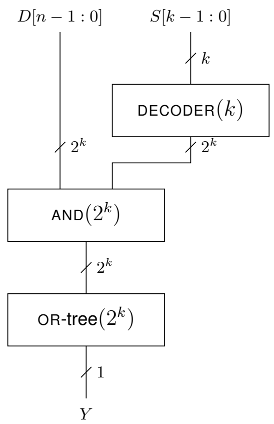
We will present two implementations:

- a decoder based implementation
- a tree-like implementation

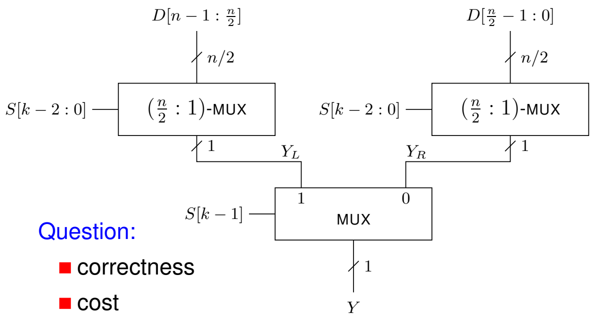
(n:1)-MUX : a decoder based implementation

Question:

- correctness
- cost
- delay
- asymptotic optimality



(n:1)-MUX : a tree-like implementation



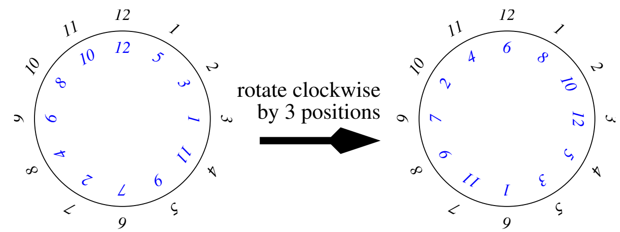
Question:

- correctness
- cost
- delay
- asymptotic optimality

Which design is better?

- both designs are asymptotically optimal.
- based on the tables of Müller & Paul, the tree-like design is better.
- decision is based on specific gate costs in the technology one uses.
- fast MUX-gates in CMOS (transmission gates) do not restore the signals well.
  $\Rightarrow$ long paths consisting only of MUX-gates are not allowed.

Cyclic shift - example



"clock"reads:
5,3,1,11,...,8,10,12

"clock"reads:
8,10,12,...,2,4,6

Cyclic shift - definition

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

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

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.

Barrel Shifter

DEF: A BARREL-SHIFTER(n) is a combinational circuit defined as follows:

Input: $x[n-1:0]$ and $sa[k-1:0]$ where $k = \lceil \log_2 n \rceil$.

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

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

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

- $\vec{x}$ - data input
- $\vec{sa}$ - shift amount input
- simplify - assume that n is a power of 2, namely, $n = 2^k$.

CLS(n, i) - Cyclic Left Shift by 2^i positions

DEF: 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:

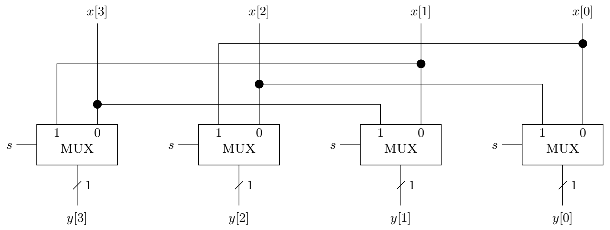
$$\forall j \in [n-1:0]: \quad y[j] = x[\text{mod}(j - s \cdot 2^i, n)].$$

Equivalently,

$$y[j] = \begin{cases} x[j] & \text{if } s = 0 \\ x[\text{mod}(j - 2^i, n)] & \text{if } s = 1. \end{cases}$$

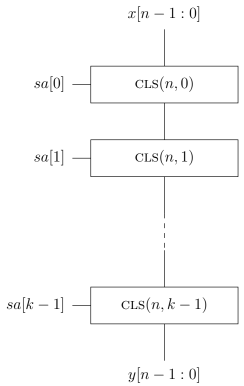
$\Rightarrow$ can implement CLS(n, i) with a row of n MUX-gates.

CLS$(4, 1)$



Evident that a CLS(n, i) requires a lot of area for the wires. Our model does not capture routing cost.

BARREL-SHIFTER(n) - a chain of CLS(n, i)



BARREL-SHIFTER(n) - correctness

Define the strings $y_i[n-1:0]$, for $0 \le i \le k-1$, recursively as follows:

$$y_0[n-1:0] \leftarrow \text{CLS}_{n,0}(x[n-1,0], sa[0])$$
$$y_{i+1}[n-1:0] \leftarrow \text{CLS}_{n,i+1}(y_i[n-1,0], sa[i+1])$$

Claim: $y_{k-1}[n-1:0]$ is a cyclic left shift of $x[n-1:0]$ by $\langle sa[k-1:0]\rangle$ positions.

Proof: Induction. $k = 0$ - trivial because CLS$(n, 0)$ shifts by zero/one position.

induction step

- $$y_i[j] = \text{CLS}_{n,i}(y_{i-1}[n-1,0], sa[i])[j] \quad \text{(by definition of } y_i\text{)}$$
  $$= y_{i-1}[\text{mod}(j - 2^i \cdot sa[i], n)] \quad \text{(by definition of CLS}_{n,i}\text{)}.$$
- Let $\ell = \text{mod}(j - 2^i \cdot sa[i], n)$.
- Ind. Hyp. $\Rightarrow y_{i-1}[\ell] = x[\text{mod}(\ell - \langle sa[i-1:0]\rangle, n)$.
- Note that

$$\text{mod}(\ell - \langle sa[i-1:0]\rangle, n) = \text{mod}(j - 2^i \cdot sa[i] - \langle sa[i-1:0]\rangle, n)$$
$$= \text{mod}(j - \langle sa[i:0]\rangle, n).$$

- Therefore $y_i[j] = x[\text{mod}(j - \langle sa[i:0]\rangle, n)]$, and the claim follows.

Logical Shifting - motivation

- Used for shifting binary strings that represent unsigned integers in binary representation.
- Shifting to the left by s positions corresponds to

$$\langle \vec{y}\rangle \leftarrow \text{mod}(\langle \vec{x}\rangle \cdot 2^s, 2^n).$$

- Shifting to the right by s positions corresponds to

$$\langle \vec{y}\rangle \leftarrow \left\lfloor \frac{\langle \vec{x}\rangle}{2^s} \right\rfloor.$$

Bi-Directional Logical Shifter - definition

A LOG-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
- $\ell \in \{0, 1\}$.

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

Functionality: If $\ell = 1$, then logical left shift as follows:

$$y[n-1:0] \triangleq x[n-1-\langle \vec{sa}\rangle : 0] \cdot 0^{\langle \vec{sa}\rangle}.$$

If $\ell = 0$, then logical right shift as follows:

$$y[n-1:0] \triangleq 0^{\langle \vec{sa}\rangle} \cdot x[n-1 : \langle \vec{sa}\rangle].$$

Bi-Directional Logical Shifter - example

Example: Let $x[3:0] = 0010$. If $sa[1:0] = 10$ and $\ell = 1$, then LOG-SHIFT(4) outputs $y[3:0] = 1000$. If $\ell = 0$, then the output equals $y[3:0] = 0000$.

Bi-Directional Logical Shifter - 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.
- Loosely speaking, an LBS(n,i) is a logical bi-directional shifter that outputs one of three possible strings:
  - the input shifted to the left by 2^i positions, or
  - the input shifted to the right by 2^i positions, or
  - the input without shifting.

  We now formally define this circuit....

LBS(n,i) - definition

DEF: An 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: Define $x'[n-1+2^i : -2^i] \in \{0,1\}^{n+2\cdot 2^i}$ as follows:

$$x'[j] \triangleq \begin{cases} x[j] & \text{if } n < j \leq 0 \\ 0 & \text{otherwise.} \end{cases}$$

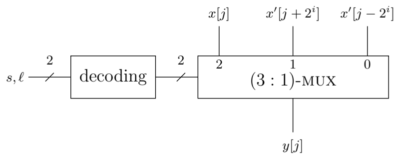
The value of the output $y[n-1:0]$ is specified by

$$\forall j \in [n-1:0]: \quad y[j] = x'[j + (-1)^{\ell} \cdot s \cdot 2^i].$$

$y[j] = x'[j + (-1)^{\ell} \cdot s \cdot 2^i]$

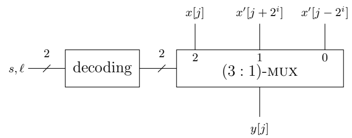
- $x'[n-1+2^i : -2^i] = 0^{2^i} \cdot x[n-1:0] \cdot 0^{2^i}$.
- ℓ - determines if the shift is a left shift or a right shift. If $\ell = 1$ then $(-1)^{\ell} = -1$, and the shift is a left shift (since increasing indexes from $j - 2^i$ to j has the effect of a left shift).
- s - determines if a shift (in either direction) takes place at all. If $s = 0$, then $y[j] = x[j]$, and no shift takes place.

A bit-slice of an implementation of LBS(n,i)



1. $(3:1)$-MUX. Implemented either by a “pruned” tree-like construction or we can simply consider a $(3:1)$-MUX as a basic gate. Simple circuit ⇒ best option can be easily determined based on the technology at hand.
2. decoding circuit - not a decoder! Decoding of s and ℓ causes the $(3:1)$-MUX to select the correct input.

A bit-slice of an implementation of LBS(n,i)



Question: This question deals with various aspects and details concerning the design of a logical shifter.

1. Design a “pruned” tree-like $(3:1)$-MUX.
2. Design the decoding box.
3. Show how LBS(n,i) circuits can be cascaded to obtain a LOG-SHIFT(n).
   Hint: follow the design of a BARREL-SHIFTER(n).

Arithmetic Shifters - motivation

- Used for shifting binary strings that represent signed integers in two's complement representation.
- logical left shifting = arithmetic left shifting.
- Arithmetic right shifting corresponds to dividing by a power of 2 (with sign extension).

Arithmetic right shifter - definition

DEF: An 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,

$$y[n-1:0] \triangleq x[n-1]^{\langle \vec{sa} \rangle} \cdot x[n-1:\langle \vec{sa} \rangle].$$

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

Arithmetic right shifter - implementation

Question: Consider the definitions of CLS(n,i) and LBS(n,i). Suggest an analogous definition ARS(n,i) for arithmetic right shift (i.e., modify the definition of $\vec{x}'$ and use $(2:1)$-MUXs). Suggest an implementation of an arithmetic right shifter based on cascading ARS$(n.i)$ circuits.

Further questions

Question: Design a bi-directional cyclic shifter. Such a shifter is like a cyclic left shifter but has an additional input $\ell \in \{0,1\}$ that indicates the direction of the required shift. Hint: Consider reducing a cyclic right shift to a cyclic left shifter. To simplify the reduction you may assume that $n = 2^k - 1$ (hint: use one's complement negation). Suggest a simple reduction in case $n = 2^k$ (hint: avoid explicit subtraction!).

Further questions - cont.

Question: CPUs often support all three types of shifting: cyclic, logical, and arithmetic shifting.

1. Write a complete specification of a shifter that can perform all three types of shifts.
2. Propose an implementation of such a shifter.

Summary

- $(n:1)$-multiplexers:
  - definition.
  - two implementations: decoder based & tree-like.
  - both designs are optimal.
- three types of shifts: cyclic, logical, and arithmetic shifts.
- Design method: cascade a logarithmic number of shifters (with parameter i) that either perform a shift by 2^i positions or no shift at all.