Digital Logic Systems

Recitation 9: The Cone of a Boolean function & Selectors

Guy Even Moti Medina

School of Electrical Engineering Tel-Aviv Univ.

December 27, 2011

Definitions

Definition (Restricted Boolean functions)

Let $f : \{0,1\}^n \to \{0,1\}$ denote a Boolean function. Let $\sigma \in \{0,1\}$. The Boolean function $g : \{0,1\}^{n-1} \to \{0,1\}$ defined by

$$g(w_0, \ldots, w_{n-2}) \stackrel{\triangle}{=} f(w_0, \ldots, w_{i-1}, \sigma, w_i, \ldots, w_{n-2})$$

is called the *restriction* of f with $x_i = \sigma$. We denote it by $f_{\restriction x_i = \sigma}$.

Example

Consider the Boolean function $f(\vec{x}) = \mathrm{XOR}_n(x_1, \ldots, x_n)$. The restriction of f with $x_n = 1$ is the Boolean function

$$\begin{aligned} f_{\restriction x_n = 1}(x_1, \ldots, x_{n-1}) &\stackrel{\triangle}{=} \mathrm{XOR}_n(x_1, \ldots, x_{n-1}, 1) \\ &= \mathrm{INV}(\mathrm{XOR}_{n-1}(x_1, \ldots, x_{n-1})). \end{aligned}$$

when does a function depend on an input?

Definition

A Boolean function $f : \{0,1\}^n \to \{0,1\}$ depends on its ith input if

$$f_{\restriction x_i=0} \neq f_{\restriction x_i=1}.$$

Example

Consider the Boolean function $f(\vec{x}) = \mathrm{XOR}_2(x_1, x_2)$. The function f depends on the ith input for $i = 2$. Indeed, $f_{\restriction x_2=1}(x_1) = \mathrm{NOT}(x_1)$ and $f_{\restriction x_2=0}(x_1) = x_1$.

The cone of a function

Definition (Cone of a Boolean function)

The cone of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is defined by

$$cone(f) \stackrel{\triangle}{=} \{i : f_{\restriction x_i=0} \neq f_{\restriction x_i=1}\}.$$

Example

The cone of the Boolean function $f(\vec{x}) = \mathrm{XOR}_2(x_1, x_2)$ equals $\{1,2\}$ because XOR depends on both inputs.

Flipping bits

Definition

Let $flip_i : \{0,1\}^n \to \{0,1\}^n$ be the Boolean function defined by $flip_i(\vec{x}) \triangleq \vec{y}$, where

$$y_j \triangleq \begin{cases} x_j & \text{if } j \neq i \\ \text{NOT}(x_j) & \text{if } i = j. \end{cases}$$

Claim

Let $f : \{0,1\}^n \to \{0,1\}^k$ denote a Boolean function. Then,

$$i \in \text{cone}(f) \iff \exists \vec{v} \in \{0,1\}^n : f(v) \neq f(flip_i(\vec{v})).$$

Claim

The Boolean function OR_n depends on all its inputs, namely,

$$|\text{cone}(\text{OR}_n)| = n.$$

Example

Example

Consider the following Boolean function:

$$f(\vec{x}) = \begin{cases} 0 & \text{if } \sum_i x_i < 3 \\ 1 & \text{otherwise.} \end{cases}$$

Suppose that one reveals the input bits one by one. As soon as 3 ones are revealed, one can determine the value of $f(\vec{x})$. Nevertheless, the function $f(\vec{x})$ depends on all its inputs, and hence, $cone(f) = \{1, \ldots, n\}$.

Linear Cost Lower Bound Theorem

Theorem

Let C denote a combinational circuit that implements a Boolean function $f : \{0,1\}^n \to \{0,1\}$. If the fan-in of every gate in $\mathcal{C}$ is at most 2, then

$$c(C) \geq |\text{cone}(f)| - 1.$$

This implies:

Corollary

Let C_n denote a combinational circuit that implements OR_n. Then,

$$c(C_n) \geq n - 1.$$

Logarithmic Delay Lower Bound Theorem

Theorem

Let $C = (G, \pi)$ denote a combinational circuit that implements a non-constant Boolean function $f : \{0,1\}^n \to \{0,1\}$. If the fan-in of every gate in C is at most k, then

$$t_{pd}(C) \geq \log_k |\text{cone}(f)|.$$

This implies a lower bound on the delay of combinational circuits that implement OR_n.

Corollary

Let C_n denote a combinational circuit that implements OR_n. Let k denote the maximum fan-in of a gate in C_n. Then

$$t_{pd}(C_n) \geq \lceil \log_k n \rceil .$$

Definition

A MUX-gate is a combinational gate that has three inputs $D[0], D[1], S$ and one output Y. The functionality is defined by

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

Note that we could have used the shorter expression $Y = D[S]$ to define the functionality of a MUX-gate.

n-bit selector

Definition

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

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

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

Functionality:

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

To simplify the discussion, we will assume in this chapter that n is a power of 2, namely, $n = 2^k$.

Example

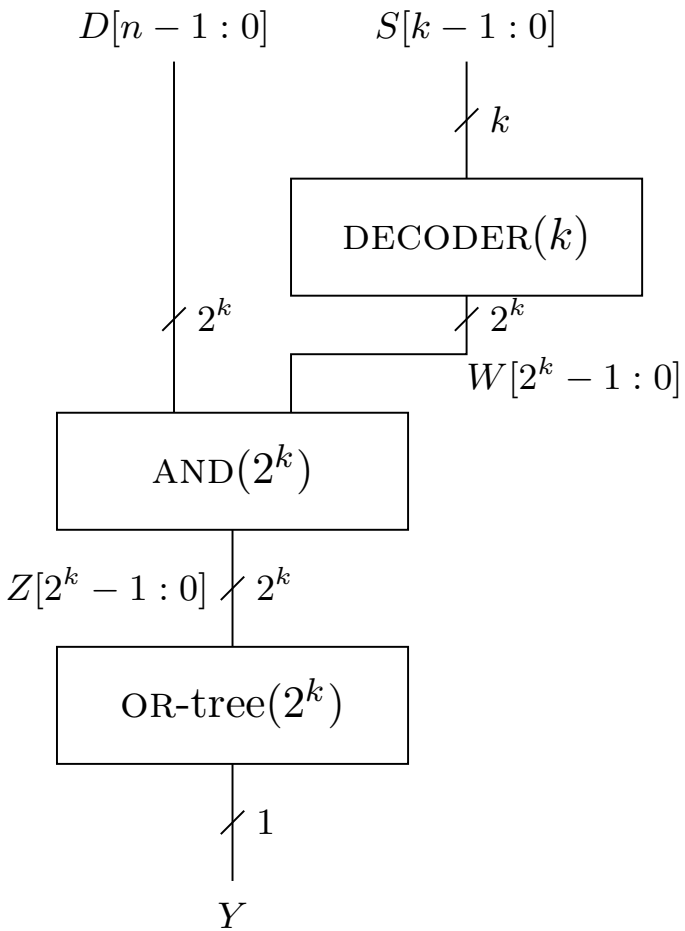
Let $n = 4$ and $D[3:0] = 0101$. If $S[1:0] = 00$, then $Y = D[0] = 1$. If $S[1:0] = 01$, then $Y = D[1] = 0$.

Implementation

We describe two implementations of (n:1)-MUX.

- translate the number $\langle \vec{S} \rangle$ to 1-out-of-n representation (using a decoder).
- tree based.

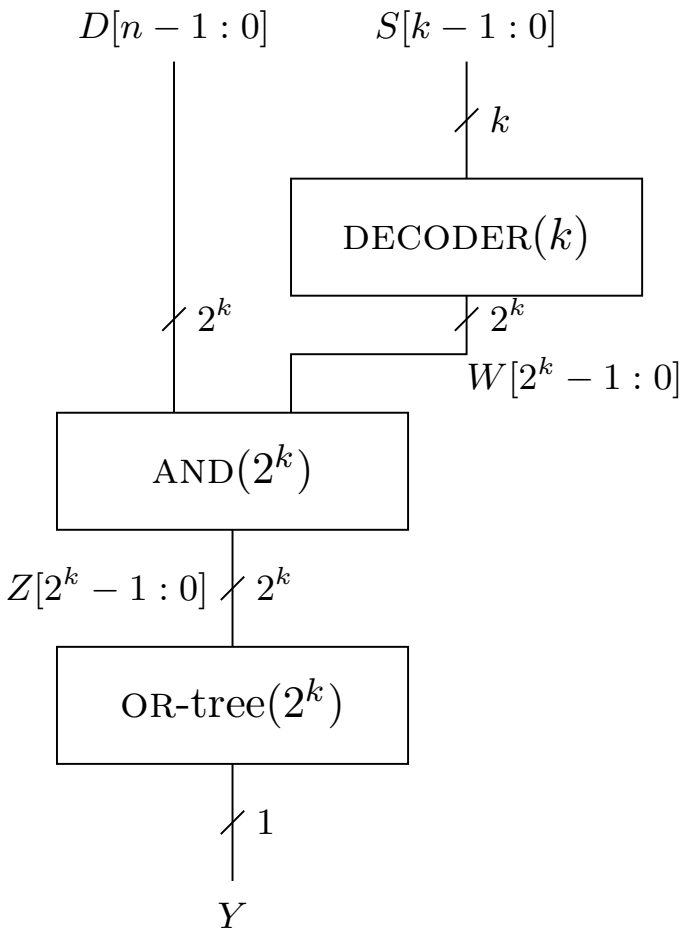
decoder based (n:1)-MUX



Claim

The (n:1)-MUX design is correct.

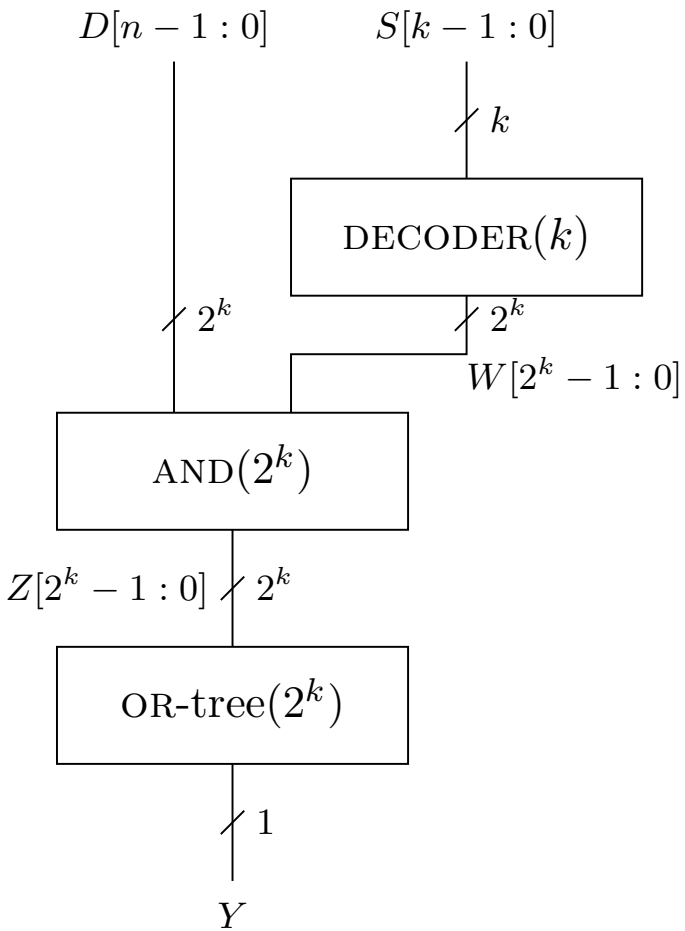
decoder based (n:1)-MUX - cost



Claim

The cost of the (n:1)-MUX design is $\Theta(n)$.

decoder based (n:1)-MUX - delay



Claim

The delay of the (n:1)-MUX design is $\Theta(\log n)$.

(n:1)-MUX - lower bounds

Claim

The cone of the Boolean function implemented by a $(n:1)$*-*MUX *circuit contains at least n elements.*

Consider combinational circuits with with at most two input terminals per gate (or any constant).

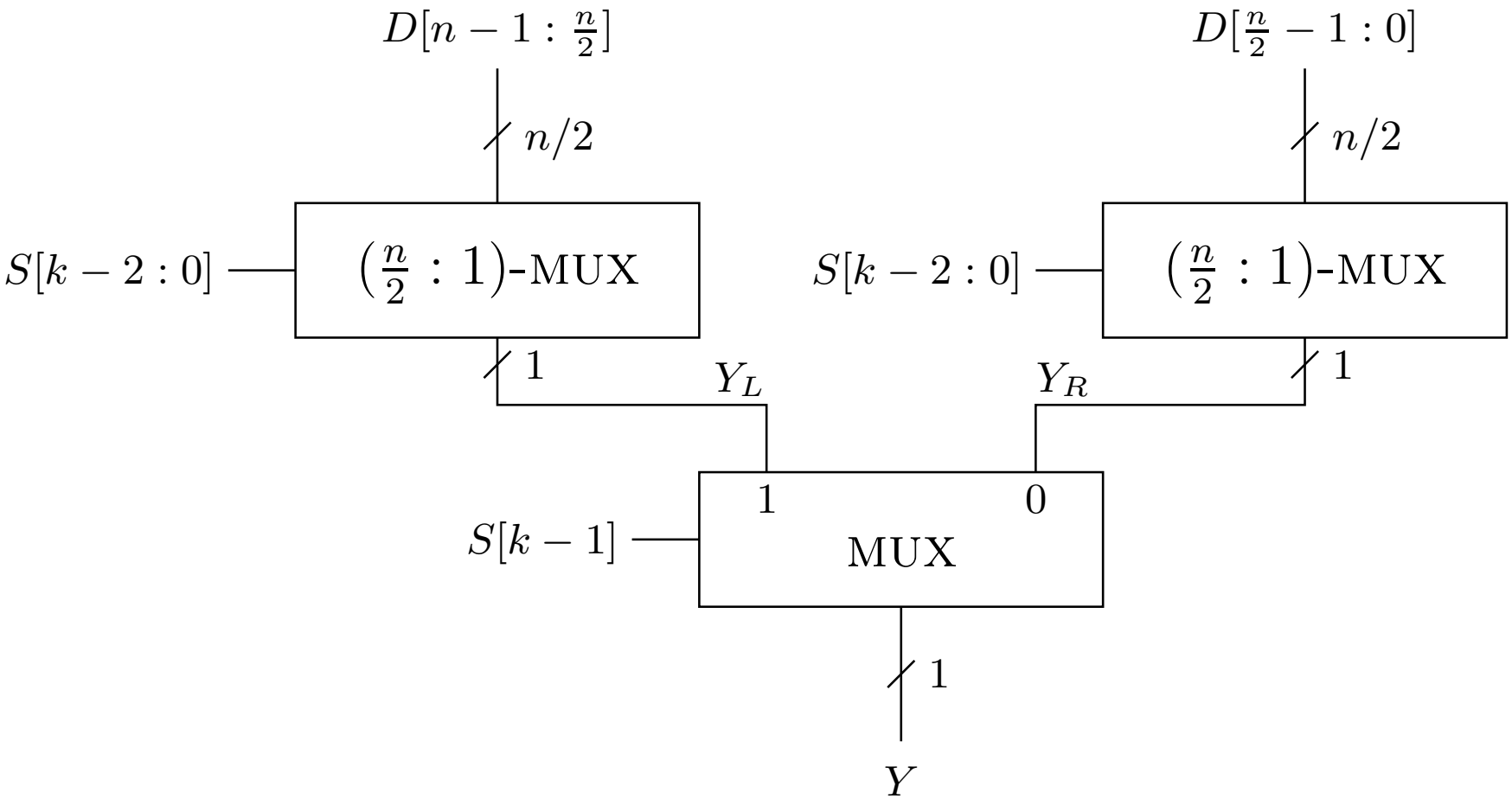
Corollary

*The cost of the (n:1)-*MUX *design is asymptotically optimal.*

Corollary

*The delay of the (n:1)-*MUX *design is asymptotically optimal.*

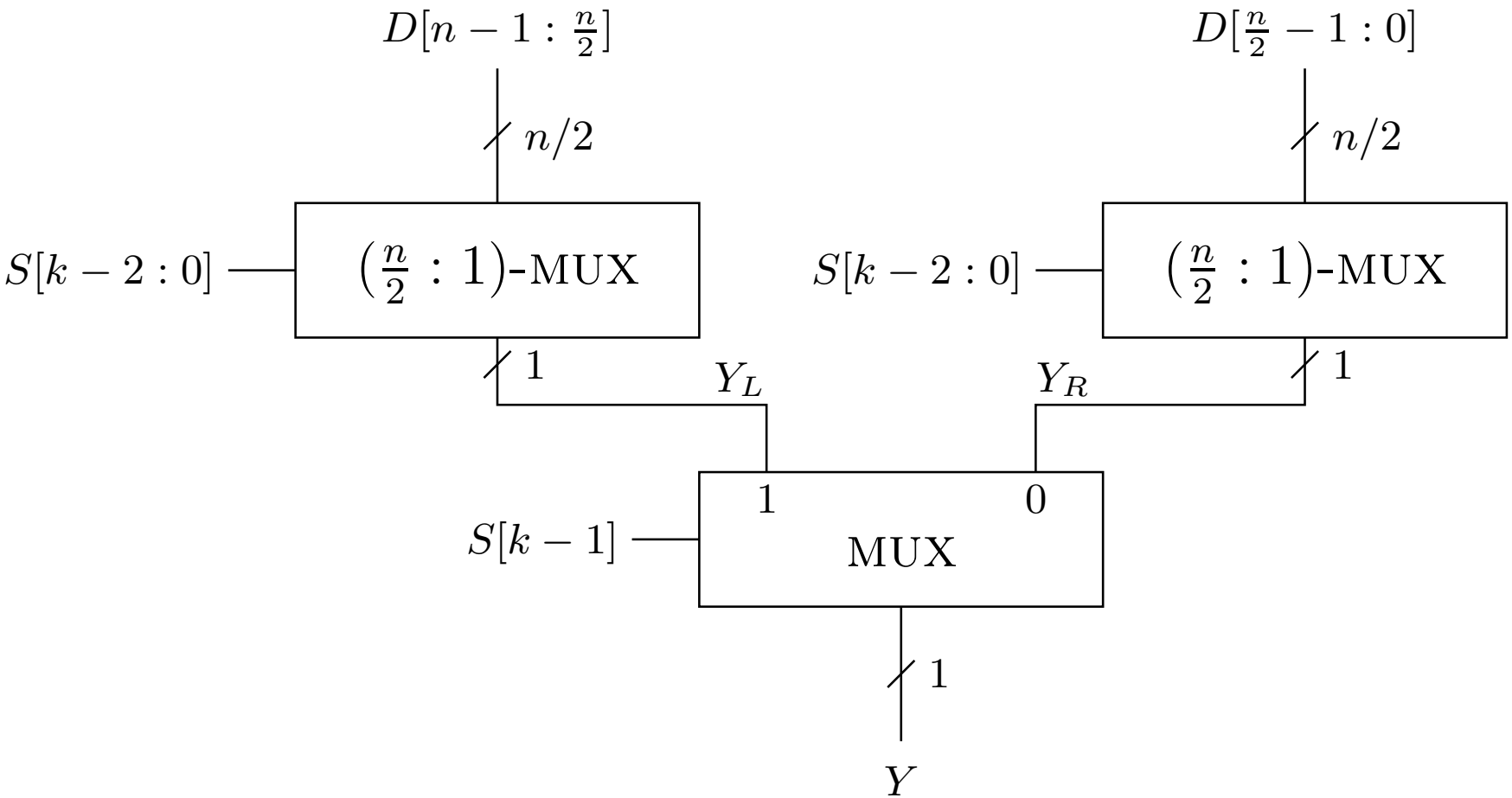
tree based (n:1)-MUX



Claim

The (n:1)-MUX design is correct.

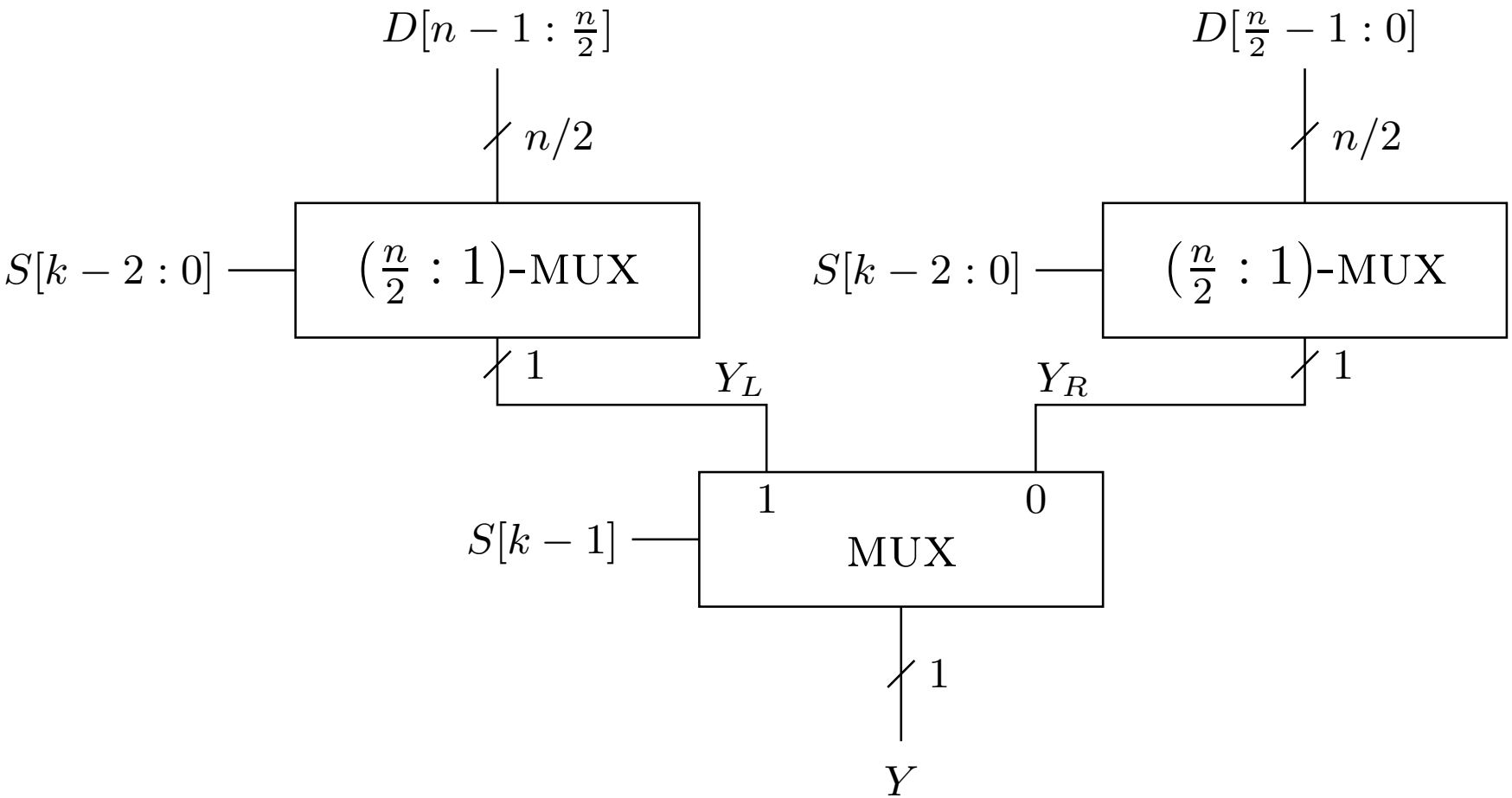
tree based (n:1)-MUX - cost



Claim

The cost of the (n:1)-MUX design is $\Theta(n)$.

tree based (n:1)-MUX - delay



Claim

The delay of the (n:1)-MUX design is $\Theta(\log n)$.