Digital Logic Systems

Recitation 7: Foundation of Combinational Circuits, Trees, Masks

Guy Even Moti Medina

School of Electrical Engineering Tel-Aviv Univ.

April 30, 2019

Foundations of Combinational Circuits: Building Blocks

The building blocks of combinational circuits:

- Combinational gates
- Wires and nets
- I/O ports

Wires and nets

A wire is a connection between two terminals (e.g., an output of one gate and an input of another gate). In the zero-noise model, the signals at both ends of a wire are identical.

Very often we need to connect several terminals (i.e., inputs and outputs of gates) together. We could, of course, use any set of edges (i.e., wires) that connects these terminals together. Instead of specifying how the terminals are physically connected together, we use nets.

Definition

A net is a subset of terminals that are connected by wires. The fan-out of a net N is the number of input terminals that are contained in N.

Example

We may draw a net in any way that we find convenient or aesthetic. The interpretation of the drawing is that terminals that are connected by lines or curves constitute a net.

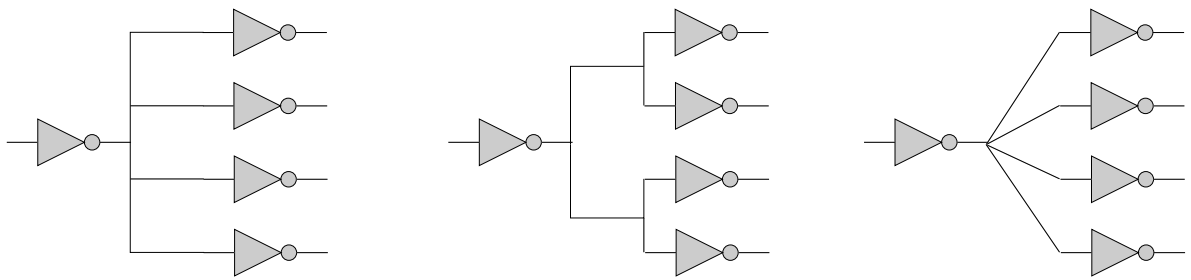


Figure: Three equivalent nets.

Simple nets

The following definition captures the type of nets we would like to use. We call these nets *simple*.

Definition

A net N is *simple* if (i) N is fed by exactly one output terminal, and (ii) N feeds at least one input terminal.

A simple net N that is fed by the output terminal t and feeds the input terminals $\{t_i\}_{i\in I}$ can be modeled by the wires $\{w_i\}_{i\in I}$. Each wire w_i connects t and t_i. In fact, since information flows in one direction, we may regard each wire w_i as a directed edge $t \to t_i$. To simplify the discussion, we model simple nets by a "star" of wires emanating from a common output terminal.

Each such wire connects an output terminal of a gate to input terminal of a gate. Thus, a full description of a wire is of the form $(g_1, t_1) \longrightarrow (g_2, t_2)$, where t_1 is an output terminal of gate g_1 and t_2 is an input terminal of gate t_2.

Input/Output gates

Definition (input and output gates)

An *input gate* is a gate with zero inputs and a single output.
An *output gate* is a gate with one input and zero outputs.

Input Gate Output Gate

- Inputs from the "external world" are fed to a circuit via input gates.
- Outputs to the "external world" are fed by the circuit via output gates.
- an input gate is labeled (IN, x_i), where x_i is the name of the signal along the wire that emanates from it.
- an output gate is labeled (OUT, y_i), where y_i is the name of the signal along the wire that enters it.

Combinational Gates

- inputs and outputs of a gate are often referred to as *terminals*, *ports*, or even *pins*.
- fan-in of a gate g = number of input terminals of g (i.e., the number of bits in the domain of the Boolean function that specifies the functionality of g).
- The basic gates that we consider are: inverter (NOT-gate), OR-gate, NOR-gate, AND-gate, NAND-gate, XOR-gate, NXOR-gate, multiplexer (MUX). All these gates have a single output.
- basic gates have constant fan-in (1 for inverter, 3 for MUX only, 2 for all the others). We usually assume that these basic gates have the same constant t_{pd}. This assumption is not precise.
- fan-out $\neq$ the number of output ports.

Combinational Circuit - definition

For simplicity, we assume that Γ contains combinational gates with a single output terminal, two input terminals, and implement commutative Boolean functions.
Let IO denote a library that contains two special types of gates: input-gates (IN, x_i) and output-gates (OUT, y_j).

Definition

A combinational circuit C is a pair (G, π), where $G = (V, E)$ is a directed acyclic graph and $\pi : V \to \Gamma \cup IO$ is a labeling function such that:

1. $\pi(v) \in IO$ iff v is a source or a sink in G.
2. For every vertex v, the in-degree of v equals the fan-in of $\pi(v)$.
3. The restriction of π to sources and sinks is one-to-one. (Namely, the names of input-gates and output-gates are distinct.)

Metrics

We judge the combinational circuits by:

- **Cost** - sum of all the gates' costs
- **Propagation Delay** - sum of t_{pd} of all components along the critical path

Cost

- Let $C = (G, \pi)$ denote a combinational circuit where $G = (V, E)$ is a directed graph and $\pi : V \to \Gamma \cup IO$ is a labeling.
- Let $c : \Gamma \cup IO \to \mathbb{R}^{\geq 0}$ denote a cost function. Usually, input-gates and output-gates have zero cost.

Definition

The cost of C is defined by

$$c(C) \stackrel{\triangle}{=} \sum_{v \in V} c(\pi(v)).$$

Propagation delay

The propagation delays $t_{pd}(v)$ are computed by Algorithm SIM$(C, \vec{x})$.

Definition

The propagation delay of C is defined by

$$t_{pd}(C) \stackrel{\triangle}{=} \max_{v \in V} t_{pd}(v).$$

We often refer to the propagation delay of a combinational circuit as its depth or simply its delay.

Definition

The propagation delay of a path p in G is defined as

$$t_{pd}(p) \stackrel{\triangle}{=} \sum_{v \in p} t_{pd}(\pi(v)).$$

Propagation delay example (skipped in lecture slides 10-11)

Example

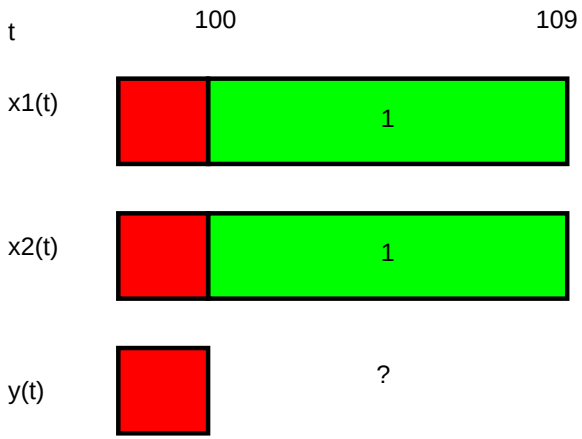
Consider an and-gate with inputs $x_1(t)$ and $x_2(t)$ and an output $y(t)$. Given: $t_{pd} = 2$, $t_{cont} = 0$. seconds. (All time units are in seconds in this example, so units will not be mentioned anymore in this example).

Propagation delay example (skipped in lecture slides 10-11)

Example

Consider an and-gate with inputs $x_1(t)$ and $x_2(t)$ and an output $y(t)$. Given: $t_{pd} = 2$, $t_{cont} = 0$. seconds. (All time units are in seconds in this example, so units will not be mentioned anymore in this example).

The inputs equal 1 during the interval $[100, 109]$. When is the gate consistent?

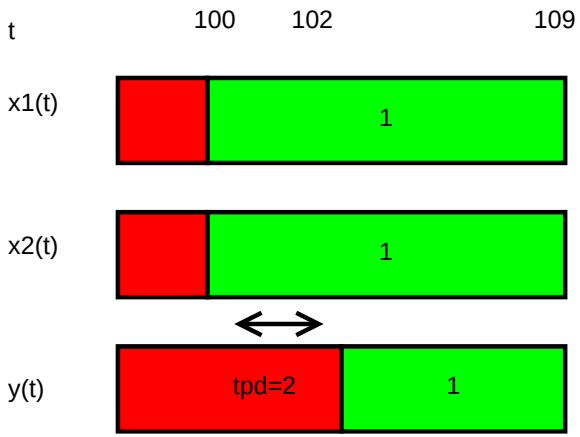


Propagation delay example (skipped in lecture slides 10-11)

Example

Consider an and-gate with inputs $x_1(t)$ and $x_2(t)$ and an output $y(t)$. Given: $t_{pd} = 2$, $t_{cont} = 0$. seconds. (All time units are in seconds in this example, so units will not be mentioned anymore in this example).

The inputs equal 1 during the interval $[100, 109]$. When is the gate consistent? $y(t) = 1$ in the interval [102,109]

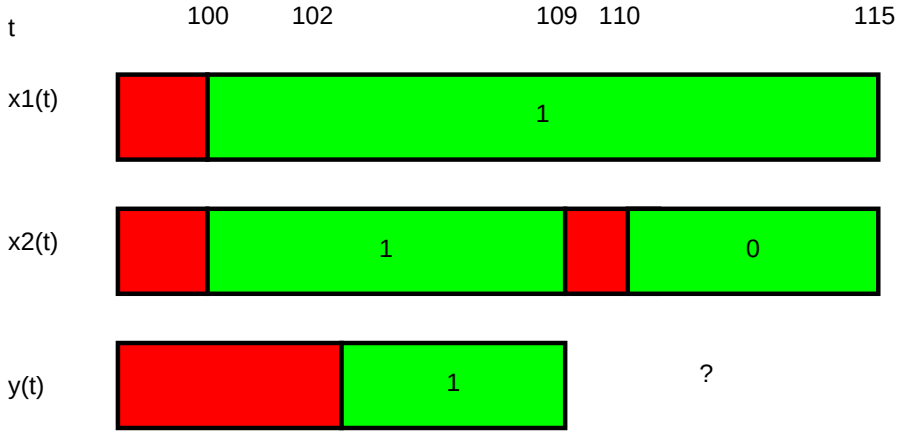


Propagation delay example (skipped in lecture slides 10-11)

Example

Consider an and-gate with inputs $x_1(t)$ and $x_2(t)$ and an output $y(t)$. Given: $t_{pd} = 2$, $t_{cont} = 0$. seconds. (All time units are in seconds in this example, so units will not be mentioned anymore in this example).

$x_1(t) = 1$ during the interval $(109, 115]$, $x_2(t)$ = non-logical during the interval $(109, 110)$, and $x_2(t) = 0$ during the interval $[110, 115]$. What can we say about $y(t)$?

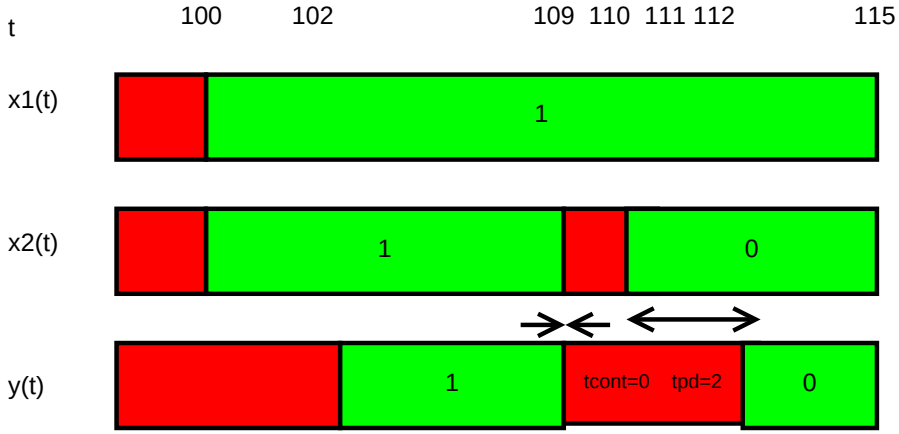


Propagation delay example (skipped in lecture slides 10-11)

Example

Consider an and-gate with inputs $x_1(t)$ and $x_2(t)$ and an output $y(t)$. Given: $t_{pd} = 2$, $t_{cont} = 0.$ seconds. (All time units are in seconds in this example, so units will not be mentioned anymore in this example).

$x_1(t) = 1$ during the interval $(109, 115]$, $x_2(t) =$ non-logical during the interval $(109, 110)$, and $x_2(t) = 0$ during the interval $[110, 115]$. What can we say about $y(t)$? $= 0$ in $t \in [112, 115]$

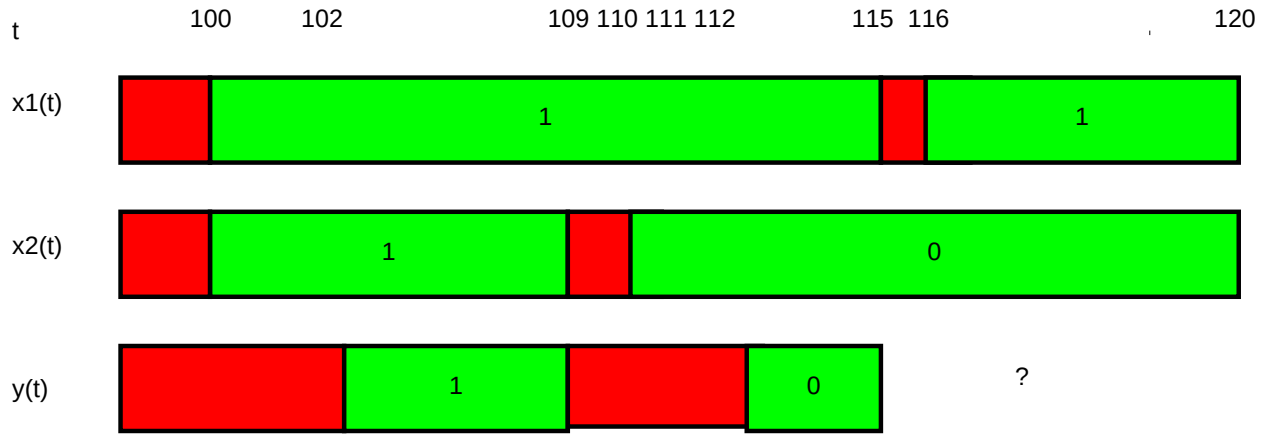


Propagation delay example (skipped in lecture slides 10-11)

Example

Consider an and-gate with inputs $x_1(t)$ and $x_2(t)$ and an output $y(t)$. Given: $t_{pd} = 2$, $t_{cont} = 0$. seconds. (All time units are in seconds in this example, so units will not be mentioned anymore in this example).

$x_2(t)$ remains stable during the interval $[110, 120]$, $x_1(t)$ becomes non-logical during the interval $(115, 116)$, and $x_1(t)$ equals 1 again during the interval $[116, 120]$. $y(t) =$

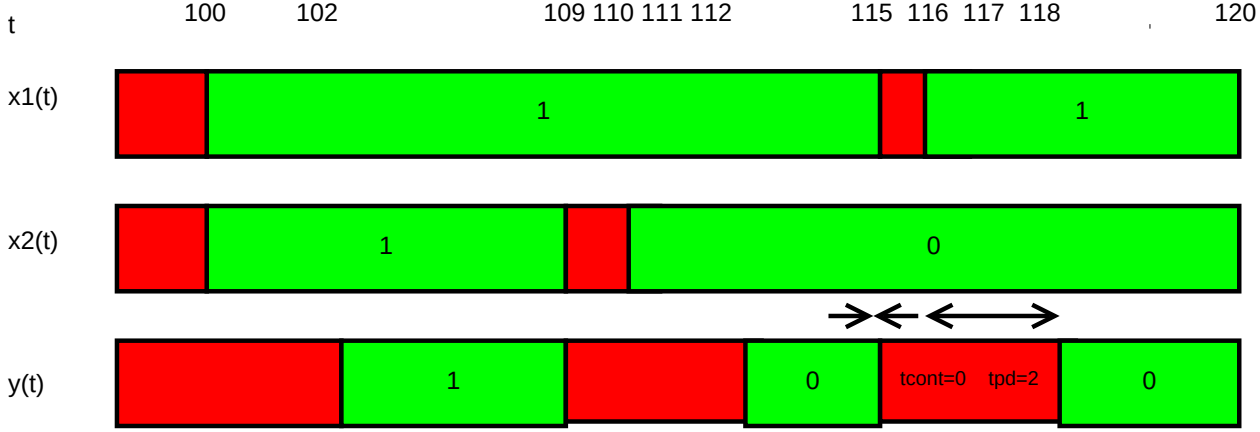


Propagation delay example (skipped in lecture slides 10-11)

Example

Consider an and-gate with inputs $x_1(t)$ and $x_2(t)$ and an output $y(t)$. Given: $t_{pd} = 2$, $t_{cont} = 0.$ seconds. (All time units are in seconds in this example, so units will not be mentioned anymore in this example).

$x_2(t)$ remains stable during the interval $[110, 120]$, $x_1(t)$ becomes non-logical during the interval $(115, 116)$, and $x_1(t)$ equals 1 again during the interval $[116, 120]$. $y(t) = 0$ in $t \in [118, 120]$



Critical paths

Algorithm SIM$(C, \vec{x})$ computes the largest delay of a path in G.

Claim (9)

$$t_{pd}(C) = \max\left\{t_{pd}(p) \mid p \text{ is a path in } G\right\}$$

Definition

Let $C = (G, \pi)$ denote a combinational circuit. A path p in G is critical if $t_{pd}(p) = t_{pd}(C)$.

We focus on critical paths that are maximal (i.e., cannot be further augmented). This means that maximal critical paths begin in an input-gate and end in an output-gate.

SIM - Algorithm for a simulation of a combinational circuit

Algorithm 11.1 SIM$(C, \vec{x})$ - An algorithm for simulating the combinational circuit $C = (V, N, \pi)$ with respect an input vector $\vec{x}$.

$(v_1, v_2, \ldots, v_n) \leftarrow TS(DG(C))$ {topological sorting of $DG(C)$}

For $i = 1$ to n do

switch $deg_{in}(v_i)$

case $deg_{in}(v_i) = 0$: $\{\pi(v_i) = (\text{IN}, x_j)\}$

- Let x_j denote the name of v_i before topological sorting.
- Set $f_{v_i}(\vec{x}) \triangleq x_j$ and $t_{pd}(v_i) \triangleq 0$.

case $deg_{in}(v_i) = 1$:

If $\{\pi(v_i) = \text{NOT}\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = \text{NOT}(f_{v_j}(\vec{x}))$ and $t_{pd}(v_i) = t_{pd}(v_j) + t_{pd}(\text{NOT})$.

If $\{\pi(v_i) = (\text{OUT}, y)\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = f_{v_j}(\vec{x})$ and $t_{pd}(v_i) = t_{pd}(v_j)$.

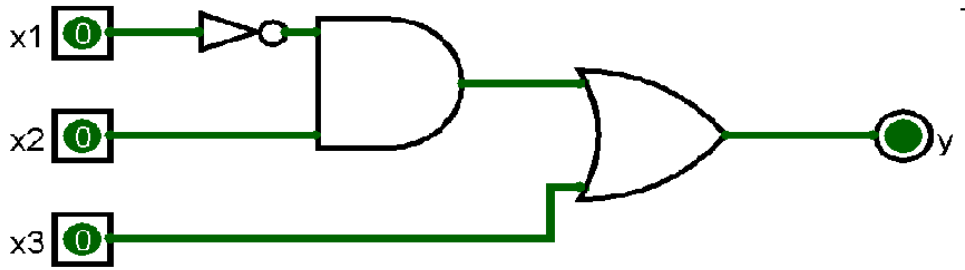
case $deg_{in}(v_i) = 2$:

- Let $v_j \longrightarrow v_i$ and $v_k \longrightarrow v_i$ denote the arcs that enter v_i.
- Set $f_{v_i}(\vec{x}) = B_{\pi(v_i)}(f_{v_j}(\vec{x}), f_{v_k}(\vec{x}))$, and $t_{pd}(v_i) = \max\{t_{pd}(v_j), t_{pd}(v_k)\} + t_{pd}(\pi(v_i))$.

Example - SIM algorithm

Consider the following circuit

Algorithm 11.1 SIM$(C, \vec{x})$ - An algorithm for simulating the combinational circuit $C = (V, N, \pi)$ with respect an input vector $\vec{x}$.



$(v_1, v_2, \ldots, v_n) \leftarrow TS(DG(C))$

For $i = 1$ to n do

switch $deg_{in}(v_i)$

case $deg_{in}(v_i) = 0$: $\{\pi(v_i) = (\text{IN}, x_j)\}$

- Let x_j denote the name of v_i before topological sorting.
- Set $f_{v_i}(\vec{x}) \triangleq x_j$ and $t_{pd}(v_i) \triangleq 0$.

case $deg_{in}(v_i) = 1$:

If $\{\pi(v_i) = \text{NOT}\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = \text{NOT}(f_{v_j}(\vec{x}))$ and $t_{pd}(v_i) = t_{pd}(v_j) + t_{pd}(\text{NOT})$.

If $\{\pi(v_i) = (\text{OUT}, y)\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = f_{v_j}(\vec{x})$ and $t_{pd}(v_i) = t_{pd}(v_j)$.

case $deg_{in}(v_i) = 2$:

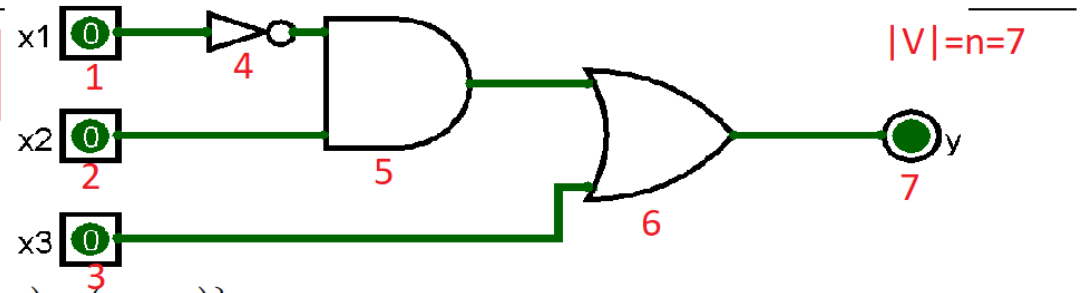
- Let $v_j \longrightarrow v_i$ and $v_k \longrightarrow v_i$ denote the arcs that enter v_i.
- Set $f_{v_i}(\vec{x}) = B_{\pi(v_i)}(f_{v_j}(\vec{x}), f_{v_k}(\vec{x}))$, and $t_{pd}(v_i) = \max\{t_{pd}(v_j), t_{pd}(v_k)\} + t_{pd}(\pi(v_i))$.

Example - SIM algorithm

First step - run topological sorting

Algorithm 11.1 SIM$(C, \vec{x})$ - An algorithm for simulating the combinational circuit $C = (V, N, \pi)$ with respect an input vector $\vec{x}$.

$(v_1, v_2, \ldots, v_n) \leftarrow TS(DG(C))$



For $i = 1$ to n do

switch $deg_{in}(v_i)$

case $deg_{in}(v_i) = 0$: $\{\pi(v_i) = (\text{IN}, x_j)\}$

- Let x_j denote the name of v_i before topological sorting.
- Set $f_{v_i}(\vec{x}) \triangleq x_j$ and $t_{pd}(v_i) \triangleq 0$.

case $deg_{in}(v_i) = 1$:

If $\{\pi(v_i) = \text{NOT}\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = \text{NOT}(f_{v_j}(\vec{x}))$ and $t_{pd}(v_i) = t_{pd}(v_j) + t_{pd}(\text{NOT})$.

If $\{\pi(v_i) = (\text{OUT}, y)\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = f_{v_j}(\vec{x})$ and $t_{pd}(v_i) = t_{pd}(v_j)$.

case $deg_{in}(v_i) = 2$:

- Let $v_j \longrightarrow v_i$ and $v_k \longrightarrow v_i$ denote the arcs that enter v_i.
- Set $f_{v_i}(\vec{x}) = B_{\pi(v_i)}(f_{v_j}(\vec{x}), f_{v_k}(\vec{x}))$, and $t_{pd}(v_i) = \max\{t_{pd}(v_j), t_{pd}(v_k)\} + t_{pd}(\pi(v_i))$.

Example - SIM algorithm

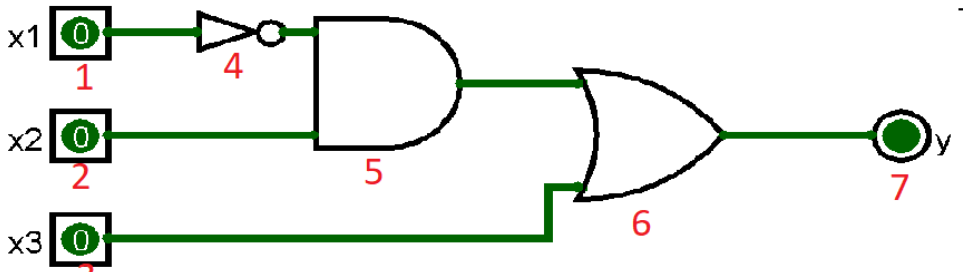
Begin iterating according to topological order

Algorithm 11.1 SIM$(C, \vec{x})$ - An algorithm for simulating the combinational circuit $C = (V, N, \pi)$ with respect an input vector $\vec{x}$.

$(v_1, v_2, \ldots, v_n) \leftarrow TS(DG(C))$

For $i = 1$ to n do i=1

switch $deg_{in}(v_i)$



case $deg_{in}(v_i) = 0$: $\{\pi(v_i) = (\text{IN}, x_j)\}$

- Let x_j denote the name of v_i before topological sorting.
- Set $f_{v_i}(\vec{x}) \triangleq x_j$ and $t_{pd}(v_i) \triangleq 0$.

case $deg_{in}(v_i) = 1$:

If $\{\pi(v_i) = \text{NOT}\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = \text{NOT}(f_{v_j}(\vec{x}))$ and $t_{pd}(v_i) = t_{pd}(v_j) + t_{pd}(\text{NOT})$.

If $\{\pi(v_i) = (\text{OUT}, y)\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = f_{v_j}(\vec{x})$ and $t_{pd}(v_i) = t_{pd}(v_j)$.

case $deg_{in}(v_i) = 2$:

- Let $v_j \longrightarrow v_i$ and $v_k \longrightarrow v_i$ denote the arcs that enter v_i.
- Set $f_{v_i}(\vec{x}) = B_{\pi(v_i)}(f_{v_j}(\vec{x}), f_{v_k}(\vec{x}))$, and $t_{pd}(v_i) = \max\{t_{pd}(v_j), t_{pd}(v_k)\} + t_{pd}(\pi(v_i))$.

Example - SIM algorithm

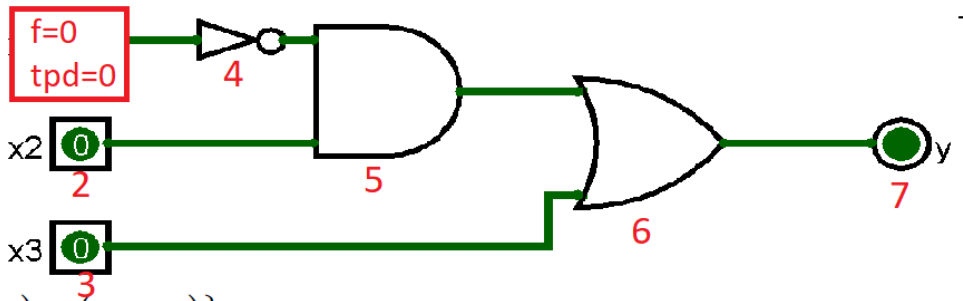
Input x1 is evaluated as a zeros-delay identity function

Algorithm 11.1 SIM$(C, \vec{x})$ - An algorithm for simulating the combinational circuit $C = (V, N, \pi)$ with respect an input vector $\vec{x}$.

$(v_1, v_2, \ldots, v_n) \leftarrow TS(DG(C))$

For $i = 1$ to n do

switch $deg_{in}(v_i)$



case $deg_{in}(v_i) = 0$: $\{\pi(v_i) = (\text{IN}, x_j)\}$

- Let x_j denote the name of v_i before topological sorting.
- Set $f_{v_i}(\vec{x}) \triangleq x_j$ and $t_{pd}(v_i) \triangleq 0$.

case $deg_{in}(v_i) = 1$:

If $\{\pi(v_i) = \text{NOT}\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = \text{NOT}(f_{v_j}(\vec{x}))$ and $t_{pd}(v_i) = t_{pd}(v_j) + t_{pd}(\text{NOT})$.

If $\{\pi(v_i) = (\text{OUT}, y)\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = f_{v_j}(\vec{x})$ and $t_{pd}(v_i) = t_{pd}(v_j)$.

case $deg_{in}(v_i) = 2$:

- Let $v_j \longrightarrow v_i$ and $v_k \longrightarrow v_i$ denote the arcs that enter v_i.
- Set $f_{v_i}(\vec{x}) = B_{\pi(v_i)}(f_{v_j}(\vec{x}), f_{v_k}(\vec{x}))$, and $t_{pd}(v_i) = \max\{t_{pd}(v_j), t_{pd}(v_k)\} + t_{pd}(\pi(v_i))$.

Example - SIM algorithm

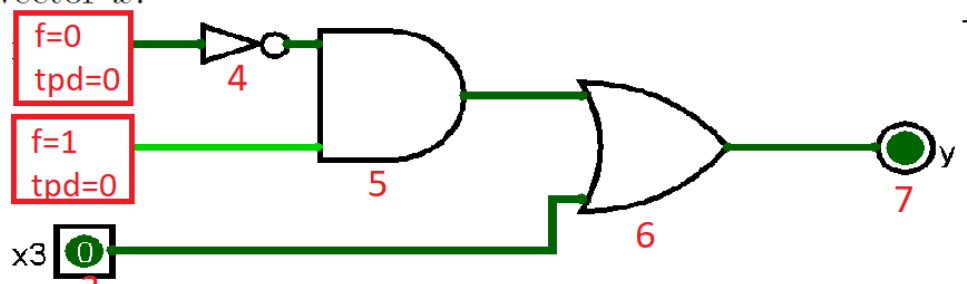
Input x2 is evaluated as a zeros-delay identity function

Algorithm 11.1 SIM$(C, \vec{x})$ - An algorithm for simulating the combinational circuit $C = (V, N, \pi)$ with respect an input vector $\vec{x}$.

$(v_1, v_2, \ldots, v_n) \leftarrow TS(DG(C))$

For $i = 1$ to n do

switch $deg_{in}(v_i)$



case $deg_{in}(v_i) = 0$: $\{\pi(v_i) = (\text{IN}, x_j)\}$

- Let x_j denote the name of v_i before topological sorting.
- Set $f_{v_i}(\vec{x}) \triangleq x_j$ and $t_{pd}(v_i) \triangleq 0$.

case $deg_{in}(v_i) = 1$:

If $\{\pi(v_i) = \text{NOT}\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = \text{NOT}(f_{v_j}(\vec{x}))$ and $t_{pd}(v_i) = t_{pd}(v_j) + t_{pd}(\text{NOT})$.

If $\{\pi(v_i) = (\text{OUT}, y)\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = f_{v_j}(\vec{x})$ and $t_{pd}(v_i) = t_{pd}(v_j)$.

case $deg_{in}(v_i) = 2$:

- Let $v_j \longrightarrow v_i$ and $v_k \longrightarrow v_i$ denote the arcs that enter v_i.
- Set $f_{v_i}(\vec{x}) = B_{\pi(v_i)}(f_{v_j}(\vec{x}), f_{v_k}(\vec{x}))$, and $t_{pd}(v_i) = \max\{t_{pd}(v_j), t_{pd}(v_k)\} + t_{pd}(\pi(v_i))$.

Example - SIM algorithm

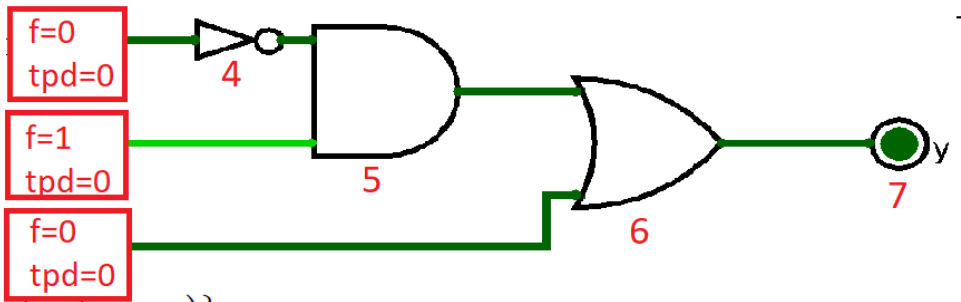
Input x3 is evaluated as a zeros-delay identity function

Algorithm 11.1 SIM$(C, \vec{x})$ - An algorithm for simulating the combinational circuit $C = (V, N, \pi)$ with respect an input vector $\vec{x}$.

$(v_1, v_2, \ldots, v_n) \leftarrow TS(DG(C))$

For $i = 1$ to n do

switch $deg_{in}(v_i)$



case $deg_{in}(v_i) = 0$: $\{\pi(v_i) = (\text{IN}, x_j)\}$

- Let x_j denote the name of v_i before topological sorting.
- Set $f_{v_i}(\vec{x}) \triangleq x_j$ and $t_{pd}(v_i) \triangleq 0$.

case $deg_{in}(v_i) = 1$:

If $\{\pi(v_i) = \text{NOT}\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = \text{NOT}(f_{v_j}(\vec{x}))$ and $t_{pd}(v_i) = t_{pd}(v_j) + t_{pd}(\text{NOT})$.

If $\{\pi(v_i) = (\text{OUT}, y)\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = f_{v_j}(\vec{x})$ and $t_{pd}(v_i) = t_{pd}(v_j)$.

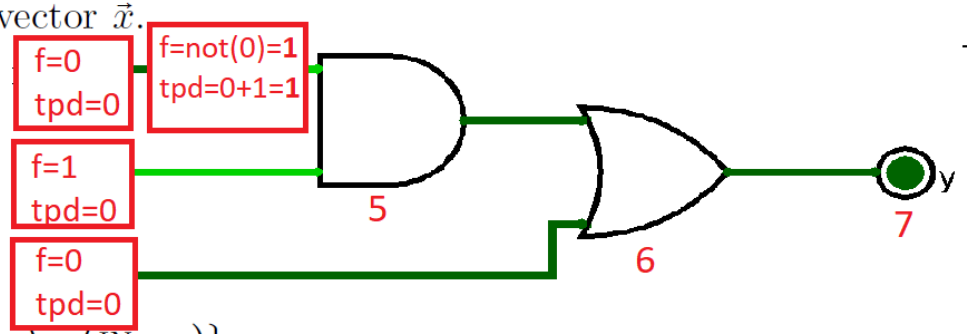
case $deg_{in}(v_i) = 2$:

- Let $v_j \longrightarrow v_i$ and $v_k \longrightarrow v_i$ denote the arcs that enter v_i.
- Set $f_{v_i}(\vec{x}) = B_{\pi(v_i)}(f_{v_j}(\vec{x}), f_{v_k}(\vec{x}))$, and $t_{pd}(v_i) = \max\{t_{pd}(v_j), t_{pd}(v_k)\} + t_{pd}(\pi(v_i))$.

Example - SIM algorithm

Inverter is evaluated

Algorithm 11.1 SIM$(C, \vec{x})$ - An algorithm for simulating the combinational circuit $C = (V, N, \pi)$ with respect an input vector $\vec{x}$.



$(v_1, v_2, \ldots, v_n) \leftarrow TS(DG(C))$

For $i = 1$ to n do i=4

switch $deg_{in}(v_i)$

case $deg_{in}(v_i) = 0$: $\{\pi(v_i) = (\text{IN}, x_j)\}$

input vector=010

- Let x_j denote the name of v_i before topological sorting.
- Set $f_{v_i}(\vec{x}) \triangleq x_j$ and $t_{pd}(v_i) \triangleq 0$.

case $deg_{in}(v_i) = 1$:

If $\{\pi(v_i) = \text{NOT}\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = \text{NOT}(f_{v_j}(\vec{x}))$ and $t_{pd}(v_i) = t_{pd}(v_j) + t_{pd}(\text{NOT})$.

If $\{\pi(v_i) = (\text{OUT}, y)\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = f_{v_j}(\vec{x})$ and $t_{pd}(v_i) = t_{pd}(v_j)$.

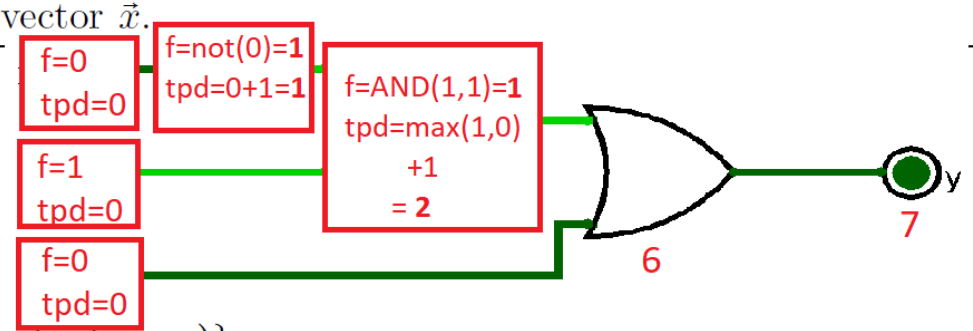
case $deg_{in}(v_i) = 2$:

- Let $v_j \longrightarrow v_i$ and $v_k \longrightarrow v_i$ denote the arcs that enter v_i.
- Set $f_{v_i}(\vec{x}) = B_{\pi(v_i)}(f_{v_j}(\vec{x}), f_{v_k}(\vec{x}))$, and $t_{pd}(v_i) = \max\{t_{pd}(v_j), t_{pd}(v_k)\} + t_{pd}(\pi(v_i))$.

Example - SIM algorithm

AND2 gate is evaluated

Algorithm 11.1 SIM$(C, \vec{x})$ - An algorithm for simulating the combinational circuit $C = (V, N, \pi)$ with respect an input vector $\vec{x}$.



$(v_1, v_2, \ldots, v_n) \leftarrow TS(DG(C))$

For $i = 1$ to n do i=5

switch $deg_{in}(v_i)$

case $deg_{in}(v_i) = 0$: $\{\pi(v_i) = (\text{IN}, x_j)\}$

input vector=010

- Let x_j denote the name of v_i before topological sorting.
- Set $f_{v_i}(\vec{x}) \triangleq x_j$ and $t_{pd}(v_i) \triangleq 0$.

case $deg_{in}(v_i) = 1$:

If $\{\pi(v_i) = \text{NOT}\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = \text{NOT}(f_{v_j}(\vec{x}))$ and $t_{pd}(v_i) = t_{pd}(v_j) + t_{pd}(\text{NOT})$.

If $\{\pi(v_i) = (\text{OUT}, y)\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = f_{v_j}(\vec{x})$ and $t_{pd}(v_i) = t_{pd}(v_j)$.

case $deg_{in}(v_i) = 2$:

- Let $v_j \longrightarrow v_i$ and $v_k \longrightarrow v_i$ denote the arcs that enter v_i.
- Set $f_{v_i}(\vec{x}) = B_{\pi(v_i)}(f_{v_j}(\vec{x}), f_{v_k}(\vec{x}))$, and $t_{pd}(v_i) = \max\{t_{pd}(v_j), t_{pd}(v_k)\} + t_{pd}(\pi(v_i))$.

Example - SIM algorithm

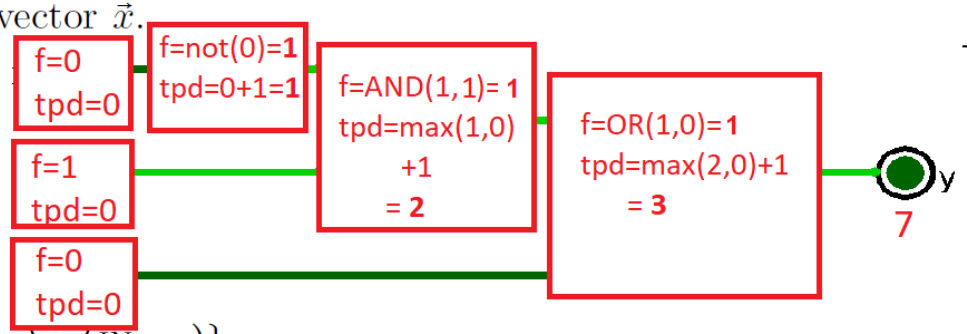
OR2 gate is evaluated

Algorithm 11.1 SIM$(C, \vec{x})$ - An algorithm for simulating the combinational circuit $C = (V, N, \pi)$ with respect an input vector $\vec{x}$.

$(v_1, v_2, \ldots, v_n) \leftarrow TS(DG(C))$

For $i = 1$ to n do

switch $deg_{in}(v_i)$



case $deg_{in}(v_i) = 0$: $\{\pi(v_i) = (\text{IN}, x_j)\}$

- Let x_j denote the name of v_i before topological sorting.
- Set $f_{v_i}(\vec{x}) \triangleq x_j$ and $t_{pd}(v_i) \triangleq 0$.

case $deg_{in}(v_i) = 1$:

If $\{\pi(v_i) = \text{NOT}\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = \text{NOT}(f_{v_j}(\vec{x}))$ and $t_{pd}(v_i) = t_{pd}(v_j) + t_{pd}(\text{NOT})$.

If $\{\pi(v_i) = (\text{OUT}, y)\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = f_{v_j}(\vec{x})$ and $t_{pd}(v_i) = t_{pd}(v_j)$.

case $deg_{in}(v_i) = 2$:

- Let $v_j \longrightarrow v_i$ and $v_k \longrightarrow v_i$ denote the arcs that enter v_i.
- Set $f_{v_i}(\vec{x}) = B_{\pi(v_i)}(f_{v_j}(\vec{x}), f_{v_k}(\vec{x}))$, and $t_{pd}(v_i) = \max\{t_{pd}(v_j), t_{pd}(v_k)\} + t_{pd}(\pi(v_i))$.

Example - SIM algorithm

Output gate y is evaluated as a zero delay identity function

Algorithm 11.1 SIM$(C, \vec{x})$ - An algorithm for simulating the combinational circuit $C = (V, N, \pi)$ with respect an input vector $\vec{x}$.

$(v_1, v_2, \ldots, v_n) \leftarrow TS(DG(C))$

For $i = 1$ to n do

switch $deg_{in}(v_i)$

case $deg_{in}(v_i) = 0$: $\{\pi(v_i) = (\text{IN}, x_j)\}$

- Let x_j denote the name of v_i before topological sorting.
- Set $f_{v_i}(\vec{x}) \triangleq x_j$ and $t_{pd}(v_i) \triangleq 0$.

case $deg_{in}(v_i) = 1$:

If $\{\pi(v_i) = \text{NOT}\}$, then

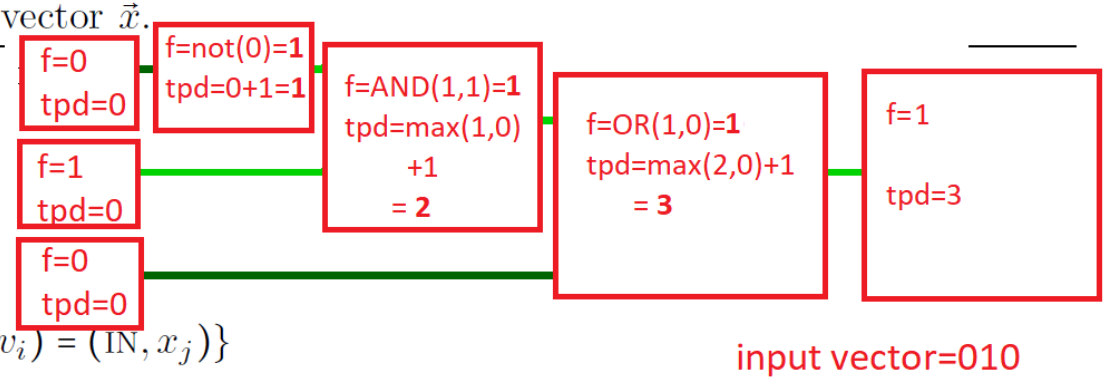
- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = \text{NOT}(f_{v_j}(\vec{x}))$ and $t_{pd}(v_i) = t_{pd}(v_j) + t_{pd}(\text{NOT})$.

If $\{\pi(v_i) = (\text{OUT}, y)\}$, then

- Let $v_j \longrightarrow v_i$ denote the arc that enters v_i.
- Set $f_{v_i}(\vec{x}) = f_{v_j}(\vec{x})$ and $t_{pd}(v_i) = t_{pd}(v_j)$.

case $deg_{in}(v_i) = 2$:

- Let $v_j \longrightarrow v_i$ and $v_k \longrightarrow v_i$ denote the arcs that enter v_i.
- Set $f_{v_i}(\vec{x}) = B_{\pi(v_i)}(f_{v_j}(\vec{x}), f_{v_k}(\vec{x}))$, and $t_{pd}(v_i) = \max\{t_{pd}(v_j), t_{pd}(v_k)\} + t_{pd}(\pi(v_i))$.



Question 1: How hard is the critical path computation?

Recall Claim 9

Claim (9)

$$t_{pd}(C) = \max\left\{t_{pd}(p) \mid p \text{ is a path in } G\right\}$$

The number of paths can be exponential in n. Does this mean that we cannot compute the propagation delay of a combinational circuit in linear time?

Answer: Homework

Question 2 - A common counting problem

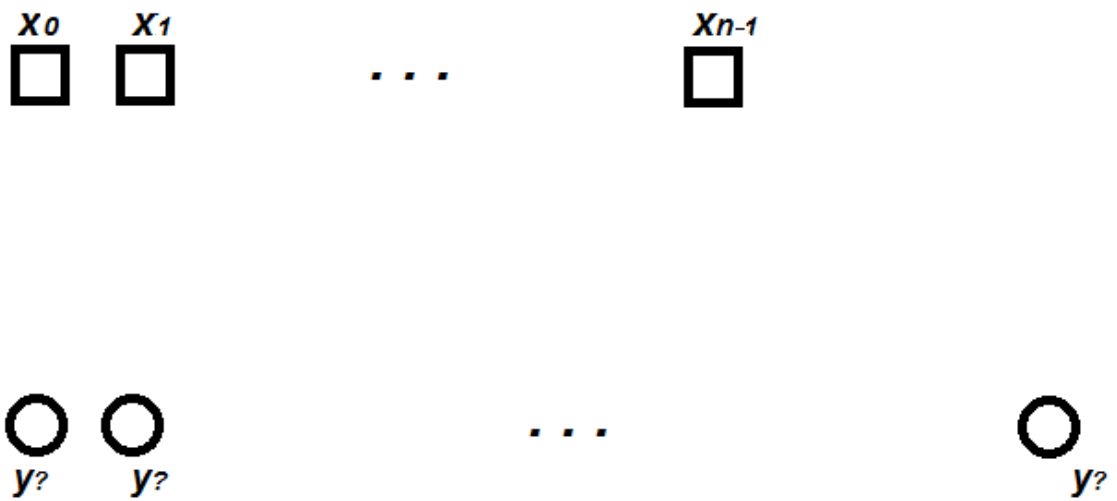
Example

One wants to have all the possible couples of x_i, x_j to go through a XOR gate and reach a y_k. What would be the cost of such a circuit? What is the propagation delay?

Question 2 - A common counting problem

Example

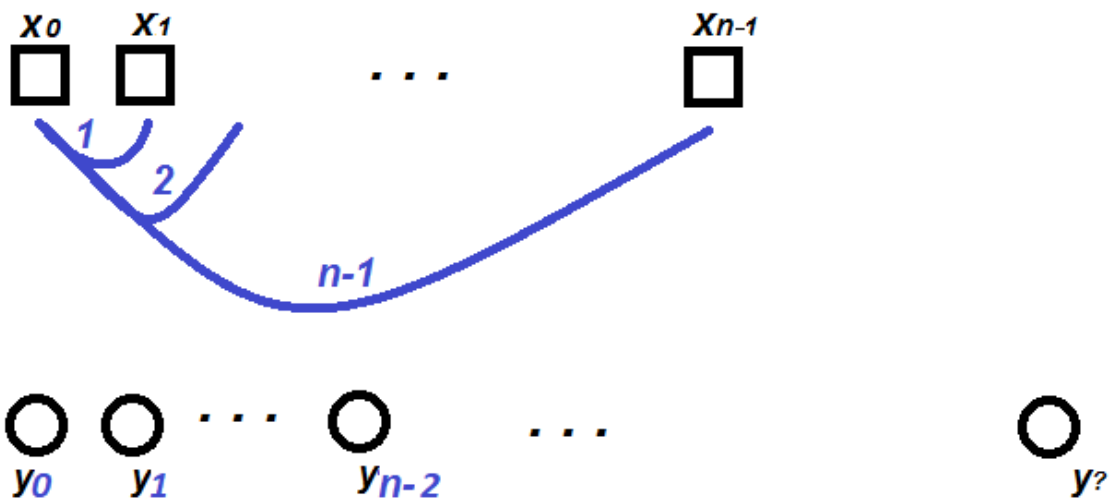
One wants to have all the possible couples of x_i, x_j to go through a XOR gate and reach a y_k. What would be the cost of such a circuit? What is the propagation delay?



Question 2 - A common counting problem

Example

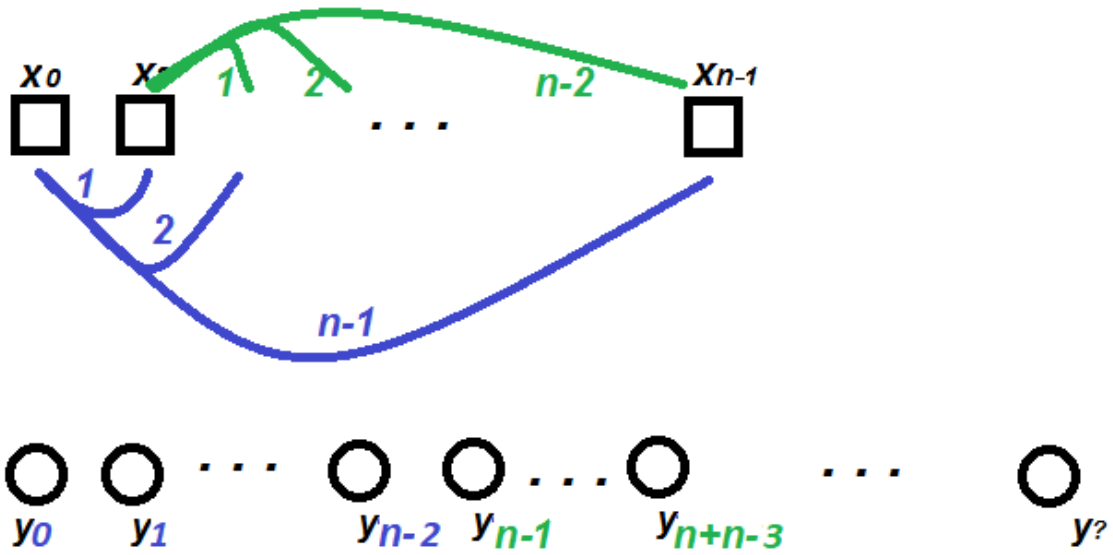
One wants to have all the possible couples of x_i, x_j to go through a XOR gate and reach a y_k. What would be the cost of such a circuit? What is the propagation delay?



Question 2 - A common counting problem

Example

One wants to have all the possible couples of x_i, x_j to go through a XOR gate and reach a y_k. What would be the cost of such a circuit? What is the propagation delay?



All the gates are summed up to:

$$c(n) = \sum_{i=0}^{n-1} i \underbrace{=}_{\text{arithmetic series}} 0 \cdot n + 1 \cdot \frac{(n-1)\cdot n}{2} = \Theta(n^2)$$

Associative Boolean functions

Definition

A Boolean function $f : \{0,1\}^2 \to \{0,1\}$ is *associative* if

$$f(f(\sigma_1, \sigma_2), \sigma_3) = f(\sigma_1, f(\sigma_2, \sigma_3)),$$

for every $\sigma_1, \sigma_2, \sigma_3 \in \{0,1\}$.

If $f(x_1, x_2)$ is an associative Boolean function, then one could define f_n in many equivalent ways, as summarized in the following claim.

Claim

If $f : \{0,1\}^2 \to \{0,1\}$ *is an associative Boolean function, then*

$$f_n(x_1, x_2, \ldots x_n) = f(f_{n-k}(x_1, \ldots, x_{n-k}), f_k(x_{n-k+1}, \ldots, x_n)),$$

for every $n \geq 2$ *and* $k \in [1, n-1]$.

Associative Boolean functions

Among the 16 existing binary ($n = 2$ variables) Boolean operators - 8 which are *associative*:

1. $f(x, y) = \mathrm{AND}(x, y)$
2. $f(x, y) = \mathrm{OR}(x, y)$
3. $f(x, y) = \mathrm{XOR}(x, y)$
4. $f(x, y) = \mathrm{NXOR}(x, y)$
5. $f(x, y) = x$
6. $f(x, y) = y$
7. $f(x, y) = 1$
8. $f(x, y) = 0$

Associative Boolean functions

Among the 16 existing binary ($n = 2$ variables) Boolean operators - 8 which are *associative*:

1. $f(x, y) = \text{AND}(x, y)$
2. $f(x, y) = \text{OR}(x, y)$
3. $f(x, y) = \text{XOR}(x, y)$
4. $f(x, y) = \text{NXOR}(x, y)$
5. $f(x, y) = x$
6. $f(x, y) = y$
7. $f(x, y) = 1$
8. $f(x, y) = 0$

Important Remark

For higher number of inputs $n > 2$, these functions can be implemented using by a **tree-structured** combinational circuit.

Associative Boolean functions

Among the 16 existing binary ($n = 2$ variables) Boolean operators - 8 which are *associative*:

1. $f(x, y) = \text{AND}(x, y)$
2. $f(x, y) = \text{OR}(x, y)$
3. $f(x, y) = \text{XOR}(x, y)$
4. $f(x, y) = \text{NXOR}(x, y)$
5. $f(x, y) = x$
6. $f(x, y) = y$
7. $f(x, y) = 1$
8. $f(x, y) = 0$

Important Remark

For higher number of inputs $n > 2$, these functions can be implemented using by a **tree-structured** combinational circuit.

Example

OR-Tree(n), AND-Tree(n), XOR-Tree(n). Where n is # inputs

Trees can have large delay

- A tree with n inputs can have a linear delay $\Theta(n)$ if not balanced properly.

Trees can have large delay

- A tree with n inputs can have a linear delay $\Theta(n)$ if not balanced properly.
- We want to construct trees with logarithmic delay $\Theta(log(n))$.

Trees can have large delay

- A tree with n inputs can have a linear delay $\Theta(n)$ if not balanced properly.
- We want to construct trees with logarithmic delay $\Theta(log(n))$.
- Solution: Balanced-Tree(n) algorithm:
  1. Case of $n = 1$ is trivial, just return the variable itself.
  2. If $n \geq 2$ then
     1. let a, b be a **balanced partition of** n
     2. T_a^* = Balanced-Tree(a)
     3. T_b^* = Balanced-Tree(b)
     4. Connect the roots of T_a^*, T_b^* to a new (fan-in=2) root.

Balanced partitions

Definition

Two positive integers a, b are a balanced partition of n if:

1. $a + b = n$, and
2. $\max\{\lceil \log_2 a \rceil, \lceil \log_2 b \rceil\} \leq \lceil \log_2 n \rceil - 1$.

Claim (This is how you pick balanced (a,b) pairs)

If $n = 2^k - r$, *where* $0 \leq r < 2^{k-1}$, *then the set of balanced partitions is*

$$P \triangleq \{(a, b) \mid 2^{k-1} - r \leq a \leq 2^{k-1} \text{ and } b = n - a\}.$$

Corollary (12.10)

The propagation delay of a balanced OR*-tree*(n) *is* $\lceil \log_2 n \rceil \cdot t_{pd}(\mathrm{OR})$.

Bitwise operations

Logic instructions are commonly used to set (turn on) or clear (reset, turn off) individual bits within a word without affecting other bits.

AND-mask

By ANDing a value with a deliberately designed constant, called a "bit mask" we can clear/preserve specific bits.
Zeros in the mask clear the corresponding value.
Ones in the mask preserve the corresponding value.

value 10010100
mask 00001111

---

result 00000100

Masks and bitwise operations

OR-mask

OR operation is used to preserve/set bits.
Zeros in the mask preserve the corresponding value.
Ones in the mask set the corresponding value to 1.

value 10010100
mask 00001111

---

result 10011111

XOR-mask

An XOR can be used to preserve/negate specific bits.
Zeros in the mask preserve the corresponding value.
Ones in the mask flip the corresponding value.

value 10010100
mask 00001111

---

result 10011011

Masks - Example

Example

$PB(n)$ is a combinational circuit defined as follows:

Input: $x, m \in \{0,1\}^n$

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

Functionality:

$$y \triangleq OR(\{x_i | m_i = 1\})$$

Note: $OR(\varnothing) \triangleq 0$

Masks - Example

Example

$PB(n)$ is a combinational circuit defined as follows:

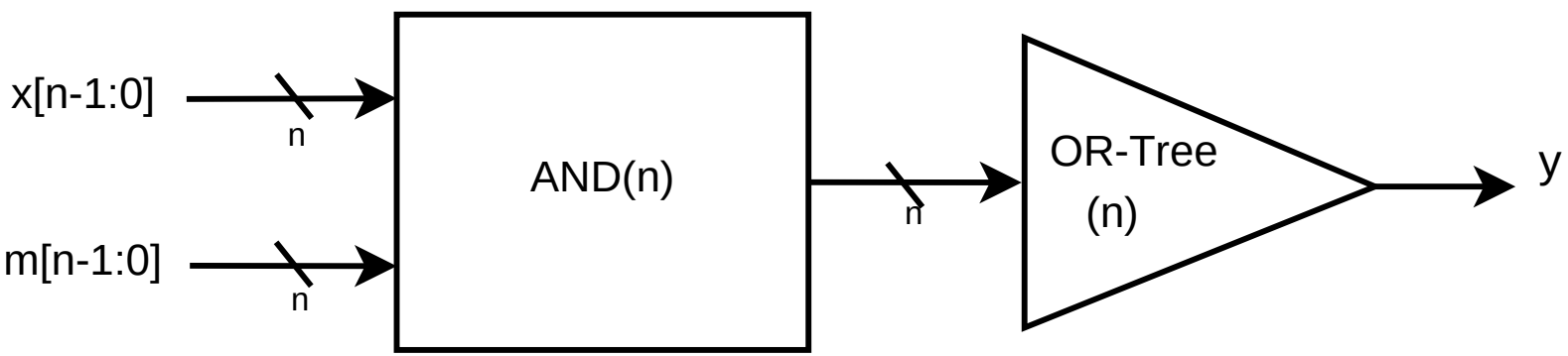
Input: $x, m \in \{0,1\}^n$

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

Functionality:

$$y \triangleq OR(\{x_i | m_i = 1\})$$

Note: $OR(\varnothing) \triangleq 0$



Circuit parameters vs inputs

Example

Consider the following circuit Toy(n)

Input: $x \in \{0,1\}^n$

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

Functionality: Let $p = \langle x[n-1:0] \rangle$. The output y respects the following:

$$y = 1 \Leftrightarrow p = 2^{n-1}$$

n is the Toy(n)'s parameter

It is a hardwired constant of a particular circuit. It cannot be changed during the circuit's operation, but can be used as a design parameter.

p is the Toy(n)'s input value

It changes during the operation of the circuit, as a function of the input x. Cannot be used as a design parameter.