Chapter 2: Foundations of combinational structures

Computer Structure
&
Intro. to Digital Computers

Dr. Guy Even

Tel-Aviv Univ.

Goals

- define combinational circuits.
- prove that every Boolean function can be implemented by a combinational circuit.
- prove that every combinational circuit implements a Boolean function.
- present an algorithm for simulating a combinational circuit.
- present an algorithm for analyzing the delay of a combinational circuit.

Boolean functions

$\{0,1\}^n$ - the set of n-bit strings.

A Boolean function - a function $f : \{0,1\}^n \rightarrow \{0,1\}^k$.

n: input length

k: output length

Gates

- A gate is a device with inputs and outputs. Inputs/outputs often called terminals, ports, or even pins.
- For simplicity, assume a gate G with two inputs x_1, x_2 and one output y.
- The functionality of a gate G is specified by a Boolean function $f : \{0,1\}^2 \rightarrow \{0,1\}$. (must allow gate time to compute f! recall definition of static transfer function.)
- Number of inputs is called fan-in of the gate.

Def: The gate G is consistent with a Boolean function f at time t if the input values are digital at time t and

$$y(t) = f(x_1(t), x_2(t)).$$

Gates - cont.

propagation delay t_{pd} -the amount of time that elapses till G becomes consistent.

Def: A gate G implements a Boolean function $f : \{0, 1\}^2 \rightarrow \{0, 1\}$ with propagation delay t_{pd} if:

$$x_1, x_2 \text{ stable in } [t_1, t_2]$$
$$\Rightarrow$$
$$G \text{ is consistent with } f \text{ in the interval } [t_1 + t_{pd}, t_2].$$

remarks:

- If $t_2 < t_1 + t_{pd}$, then definition is empty.
- If $x_i(t)$ is not stable in (t_2, t_3), then $y(t)$ need not remain stable after t_2.

Contamination delay

Contamination delay - the amount of time that the output of a consistent gate remains stable after its inputs stop being stable.

Contamination delay tells us how fast an output “reacts” to a change in the input

We we will assume that the contamination delay is zero.

Combinational circuits - building blocks

Combinational circuits are built of gates and wires & nets.

- Gates: inverter (NOT-gate), OR-gate, NOR-gate, AND-gate, NAND-gate, XOR-gate, NXOR-gate, multiplexer (MUX). (all have fan-in ≤ 3)
- Wires & Nets: connect between terminals of gates.

Wires & Nets

Wires connect points to each other. Very often we need to connect several terminals (i.e. inputs and outputs of gates) together.

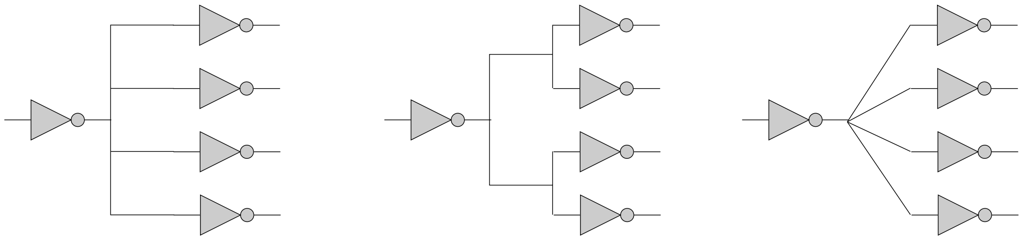
Ignore how connections are actually made.

Net - subset of terminals that are connected by wires. In the digital abstraction we assume that the signals all over a net are identical (why?).

fan-out of a net N - the number of input terminals that are connected by N.

Drawing nets

Three different drawings of the same net (of fan-out 4). 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.



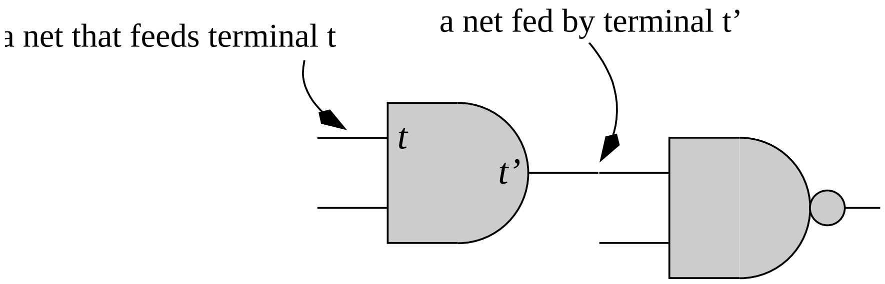
Digital signals for nets

We would like to define the digital signal $N(t)$ for a whole net N.
Noise creates different analog signals along the net.
Define $N(t)$ to logical only if there is a consensus among all the digital interpretations of analog signals at different terminals of the net.
In other words:

- $N(t)$ is zero if the digital values of all the analog signals along the net are zero.
- $N(t)$ is one if the digital values of all the analog signals along the net are one.
- If there is no consensus, then $N(t)$ is non-logical.

Directions in nets

A net N feeds an input terminal t if the input terminal t in N.
A net N is fed by an output terminal t if t is in N.



Information is “supplied” by output terminals and is “consumed” by input terminals.

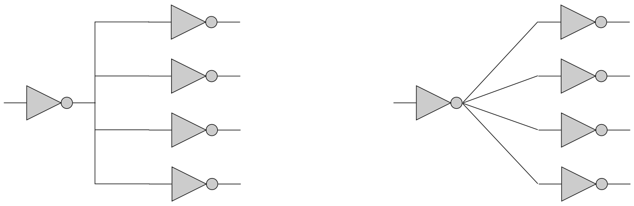
In “pure” CMOS gates, output terminals are connected via resistors either to the ground (low voltage) or to the power (high voltage). Input terminals are connected only to capacitors.

Simple nets

Def: A net N is simple if:

1. N is fed by exactly one output terminal, and
2. N feeds at least one input terminal.

- Consider a simple net $N = \{t_{out}, t_1, t_2, \ldots, t_k\}$, where t_{out} is an output terminal, and $\{t_i\}_{i=1}^k$ are input terminals.
- N can be modeled by a “star” of wires $\{w_i\}_{i \in I}$. Each wire w_i connects t_{out} and t_i. We may regard each wire w_i as a directed edge $t_{out} \to t_i$.

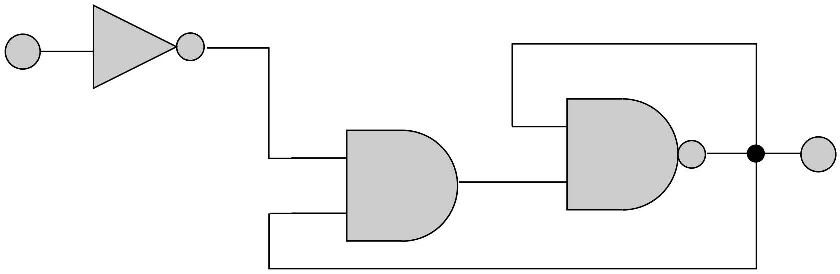


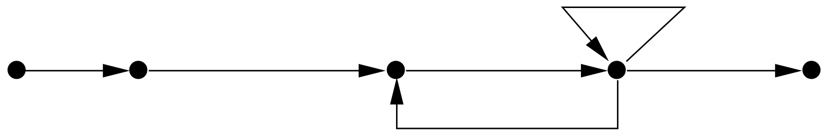
Directed graph corresponding to simple nets

If every every net N in a circuit C is simple, then we can model C by a directed graph.

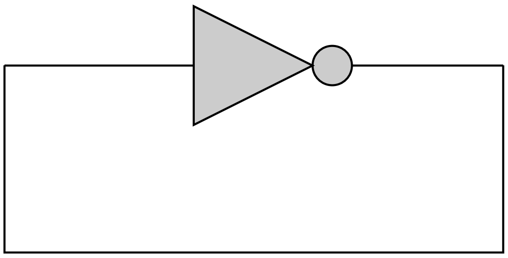
- $DG(C)$ - a directed graph.
- Nodes - gates of C.
- Directed edges - directed edge $u \to v$ if there is a net N such that: (i) an output terminal of gate u feeds N, and (ii) an input terminal of v is fed by N.

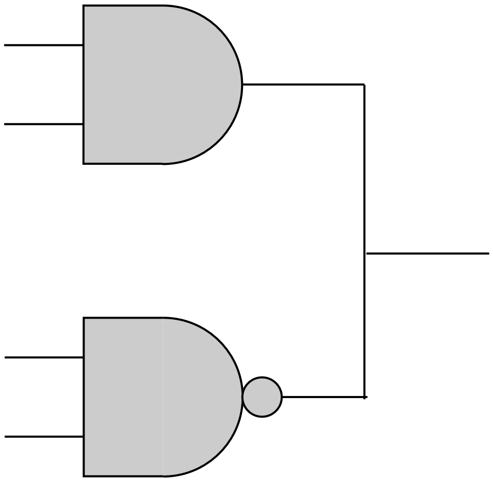
Example of a circuit C and a directed graph $DG(C)$





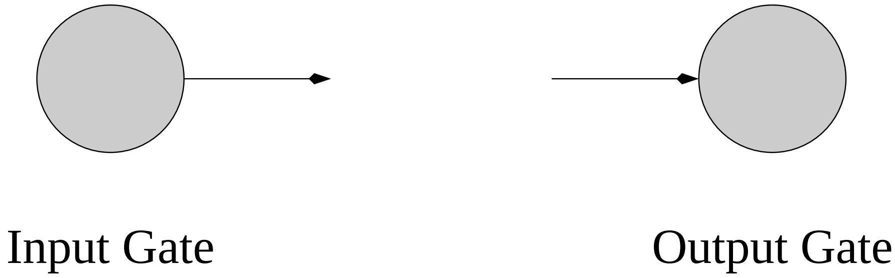
Are these circuits combinational circuits?





Input gates & output gates

Input and output gates model communication with the “external world”. Solve the problem of “hanging” wires.



- input gate - a gate with zero inputs and a single input.
- output gate - a gate with one input and zero outputs.

Syntactic definition of combinational circuits

Def: A combinational circuit is a pair $C = \langle \mathcal{G}, \mathcal{N} \rangle$ that satisfies the following conditions:

1. $\mathcal{G}$ is a set of gates.
2. $\mathcal{N}$ is a set of nets over terminals of gates in $\mathcal{G}$.
3. Every terminal t of a gate $G \in \mathcal{G}$ belongs to exactly one net $N \in \mathcal{N}$.
4. Every net $N \in \mathcal{N}$ is simple.
5. The directed graph $DG(C)$ is acyclic.

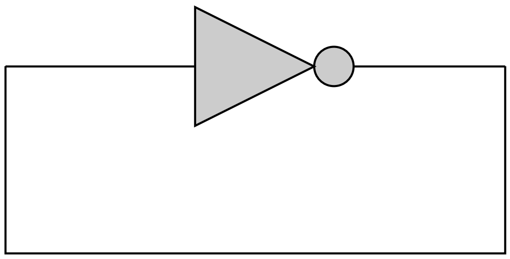
Syntactic definition - remarks

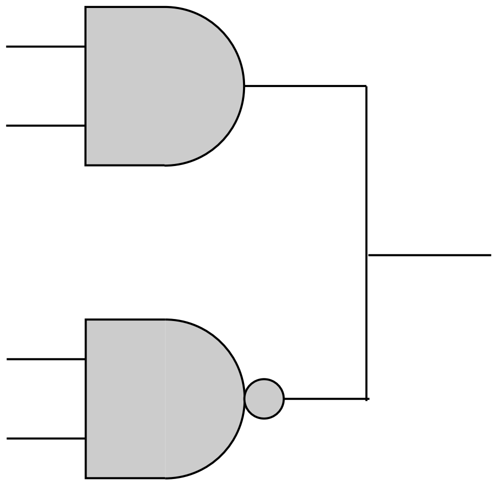
Definition of combinational circuits is independent of the gate types (e.g. inverter, NAND-gate, etc.). The question of whether a circuit is combinational is a purely topological question (i.e. are the interconnections between gates legal?).

syntax - “grammar” rules for forming compound circuits from simple circuits.

Back to “bad” examples...

Which conditions in the syntactic definition of combinational circuits are violated by the “bad” circuits?





homework: design an efficient algorithm to check if a given circuit is combinational.

Combinational circuits: Syntax ⇒ Semantics

- Completeness: for every Boolean function f, there exists a combinational circuit that implements f (exercise).
- Soundness: every combinational circuit implements a Boolean function. (NP-Complete to decide if a given combinational circuit ever outputs a 1.)
- Simulation: given the digital values of the inputs of a combinational circuit, one can simulate the circuit in linear time.
- Delay analysis: given the propagation delays of all the gates in a combinational circuit, one can compute in linear time the propagation delay of the circuit (upper bound).

Simulation theorem of combinational circuits

- $C = \langle \mathcal{G}, \mathcal{N} \rangle$ - a combinational circuit with k input gates.
- $\{x_i\}_{i=1}^k$ - digital input signals
- $[t_1, t_2]$ - a sufficiently long interval of time.

Theorem: If the digital signals $\{x_i(t)\}_{i=1}^k$ are stable during the interval $[t_1, t_2]$, then, for every net $N \in \mathcal{N}$ there exist:

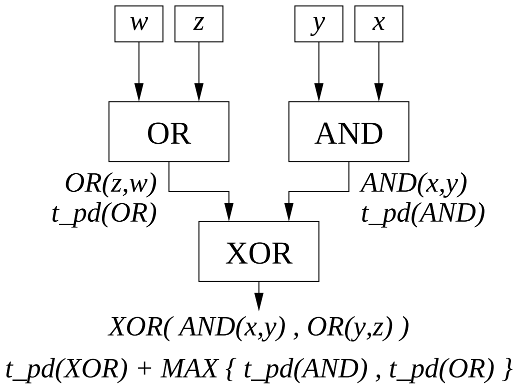
1. a Boolean function $f_N : \{0,1\}^k \rightarrow \{0,1\}$, and
2. a propagation delay $t_{pd}(N)$

such that

$$N(t) = f_N(x_1(t), x_2(t), \ldots, x_k(t)),$$

for every $t \in [t_1 + t_{pd}(N), t_2]$.

Example - simulation of combinational circuit



- process nets according to topological order (i.e. u before v if there is an edge $u \rightarrow v$ in $DG(C)$).
- assign Boolean function to each net.
- assign t_{pd} to each net.

Proof of Simulation Theorem

Notation:

- $\vec{x}(t)$ - the vector $x_1(t), \ldots, x_k(t)$.
- $v_1, v_2, \ldots, v_n$ - topological order of vertices (gates) in $DG(C)$.
- WLOG: $v_1, \ldots, v_k$ are the input gates.
- $x_i(t)$ is the digital signal output by v_i (for $1 \leq i \leq k$).
- $\mathcal{N}_i$ - subset of nets in $\mathcal{N}$ that are fed by gate v_i.
- $e_1, e_2, \ldots, e_m$ - ordering of the nets in $\mathcal{N}$ such that nets in $\mathcal{N}_i$ precede nets in $\mathcal{N}_{i+1}$.
- Note that e_1 is fed by v_1, ..., e_k is fed by v_k.

Proof - Induction hypothesis

For every $i \leq m'$ there exist:

1. a Boolean function $f_{e_i} : \{0,1\}^k \rightarrow \{0,1\}$, and
2. a propagation delay $t_{pd}(e_i)$

such that the network e_i implements the Boolean function

$$f_{e_i} : \{0,1\}^k \rightarrow \{0,1\}$$

with propagation delay $t_{pd}(e_i)$.

Proof - Induction basis

Instead of proving for $m' = 1$, we prove for $m' = k$.

Consider an $i \leq k$. The net e_i is fed by v_i, and the digital signal corresponding to e_i is $x_i(t)$.

$\Longrightarrow$ define

$$f_{e_1}(\sigma_1, \ldots, \sigma_k) = \sigma_i.$$
$$t_{pd}(e_i) = 0.$$

now to induction step...

Proof - Induction step

Ind. Hyp.$(m') \quad \Longrightarrow \quad$ Ind. Hyp.$(m'+1)$.

Focus on $e_{m'+1}$:

- let v_i denote the gate that feeds $e_{m'+1}$.
- for simplicity: assume that v_i has 2 inputs fed by the nets e_j & e_k, respectively.
- Topological ordering $\Rightarrow j, k \leq m'$.
- Ind. Hyp. $\Rightarrow$:
  - e_j implements a Boolean function f_{e_j} with $t_{pd}(e_j)$.
  - e_k implements a Boolean function f_{e_k} with $t_{pd}(e_k)$.
- $\Rightarrow$ both inputs to gate v_i are stable during the interval

$$[t_1 + \max\{t_{pd}(e_j), t_{pd}(e_k)\}, t_2].$$

Proof - Ind. step - cont.

- Gate v_i implements a Boolean function f_{v_i} with propagation delay $t_{pd}(v_i)$.
- $\Rightarrow$ the output of gate v_i equals

$$f_{v_i}(f_{e_j}(\vec{x}(t)), f_{e_k}(\vec{x}(t)))$$

during the interval

$$[t_1 + \max\{t_{pd}(e_j), t_{pd}(e_k)\} + t_{pd}(v_i), t_2].$$

- Define

$$f_{e_{m'+1}}(\vec{\sigma}) = f_{v_i}(f_{e_j}(\vec{\sigma}), f_{e_k}(\vec{\sigma})).$$
$$t_{pd}(e_{m'+1}) = \max\{t_{pd}(e_j), t_{pd}(e_k)\} + t_{pd}(v_i).$$

QED

Simulation theorem - Corollaries

- simulation algorithm
- timing analysis algorithm
- may regard a combinational circuit as a “macro-gate”. All instances of the same combinational circuit implement the same Boolean function and have the same propagation delay.

very simple algorithms...

Simulation and timing-analysis algorithm

- construct the directed graph $DG(C)$.
- sort gates in topological order .
- order the nets $e_1, e_2, \ldots, e_m$.
- For $i = 1$ to m do:
  - Let v_j denote the gate that feeds e_i.
  - $$val(e_i) \leftarrow f_{v_j}\left(\{val(e_k)\}_{e_k \text{ feeds } v_j}\right)$$
    $$t_{pd}(e_i) \leftarrow t_{pd}(v_j) + \max\{t_{pd}(e_k)\}_{e_k \text{ feeds } v_j}.$$

Complexity: linear if each gate has a single output terminal. (why?)

Quality measures of combinational gates

- Suppose C_1 and C_2 are combinational circuits that compute the same Boolean function. How do we decide which one is better? We use two criteria:
- Cost
- Propagation delay

Cost

We associate a cost with every gate. We denote the cost of a gate G by $c(G)$.

Def: The cost of a combinational circuit $C = \langle \mathcal{G}, \mathcal{N} \rangle$ is defined by

$$c(C) \triangleq \sum_{G \in \mathcal{G}} c(G).$$

Propagation delay

We associate a propagation delay with every gate. We denote the propagation delay of a gate G by $t_{pd}(G)$.

Def: The propagation delay of a combinational circuit $C = \langle \mathcal{G}, \mathcal{N} \rangle$ is defined by

$$t_{pd}(C) \triangleq \max_{N \in \mathcal{N}} t_{pd}(N).$$

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

Delays of paths

- path - a sequence $p = \{v_0, v_1, \ldots, v_k\}$ of gates that form a path in the directed graph $DG(C)$.
- delay of a path p -

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

Claim:

$$t_{pd}(C) = \max\{t_{pd}(p) : \text{paths } p\}.$$

critical path - a path p that satisfies $t_{pd}(p) = t_{pd}(C)$.

Q: Number of paths can be exponential. How were we able to compute $max\{t_{pd}(p) : \text{paths } p\}$?

Example: gate costs and delays

Müller and Paul compiled the following costs and delays of gates. These figures were obtained by considering ASIC libraries of two technologies and normalizing them with respect to the cost and delay of an inverter.

Gate	Motorola		Venus	
	cost	delay	cost	delay
INV	1	1	1	1
AND,OR	2	2	2	1
NAND, NOR	2	1	2	1
XOR, NXOR	4	2	6	2
MUX	3	2	3	2

Syntax & Semantics

- semantics - function that a circuit implements. Also called functionality or even the behavior of the circuit.

  In general, formal description that relates

  $$\text{input values} \longmapsto \text{output values}.$$

  In non-combinational circuits, the output depends not only on the current inputs, so semantics cannot be described simply by a Boolean function.

- syntax - a formal set of rules that govern how "grammatically correct" circuits are constructed from smaller circuits (just as sentences are built of words).
  - syntax $\not\Longrightarrow$ useful circuit (e.g. adder).
  - syntax $\Longrightarrow$ well defined functionality, simple simulation, & simple timing analysis.

Summary

- gates - implement simple Boolean functions
- nets & wires - used to connect terminals of gates
- formal (syntactic) definition of combinational gates
- combinational gates are easy to:
  - recognize
  - simulate
  - analyze (propagation delay)
- quality criteria: cost & delay