Chapter 2

Foundations of combinational circuits

In this chapter we define and study combinational circuits. Our goal is to prove two theorems: (A) Every Boolean function can be implemented by a combinational circuit, and (B) Every combinational circuit implements a Boolean function.

2.1 Boolean functions

Let $\{0,1\}^n$ denote the set of n-bit strings. A Boolean function is defined as follows.

Definition 4 *A function* $f : \{0,1\}^n \rightarrow \{0,1\}^k$ *is called a* Boolean function.

2.2 Gates as implementations of Boolean functions

A gate is a device that has inputs and outputs. The inputs and outputs of a gate are often referred to as *terminals*, *ports*, or even *pins*. In combinational gates, the relation between the logical values of the outputs and the logical values of the inputs is specified by a Boolean function. It takes some time till the logical values of the outputs of a gate properly reflect the value of the Boolean function. We say that a gate is *consistent* if this relation holds. To simplify notation, we consider a gate G with 2 inputs, denoted by x_1, x_2, and a single output, denoted by y. We denote the digital signal at terminal x_1 by $x_1(t)$. The same notation is used for the other terminals. Consistency is defined formally as follows:

Definition 5 *A gate G is* consistent with a Boolean function f at time t *if the input values are digital at time t and*

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

The propagation delay is the amount of time that elapses till a gate becomes consistent. The following definition defines when a gate implements a Boolean function with propagation delay t_{pd}.

Definition 6 *A gate G implements a Boolean function $f : \{0,1\}^2 \rightarrow \{0,1\}$ with propagation delay t_{pd} if the following holds.*

For every $\sigma_1, \sigma_2 \in \{0,1\}$, *if* $x_i(t) = \sigma_i$, *for* $i = 1, 2$, *during the interval* $[t_1, t_2]$, *then*

$$\forall t \in [t_1 + t_{pd}, t_2] \;:\; y(t) = f(\sigma_1, \sigma_2).$$

The following remarks should be well understood before we continue:

1. The above definition can be stated in a more compact form. Namely, a gate G implements a Boolean function $f : \{0,1\}^n \to \{0,1\}$ with propagation delay t_{pd} if stability of the inputs of G in the interval $[t_1, t_2]$ implies that the gate G is consistent with f in the interval $[t_1 + t_{pd}, t_2]$.

2. If $t_2 < t_1 + t_{pd}$, then the statement in the above definition is empty. It follows that the inputs of a gate must be stable for at least a period of t_{pd}, otherwise, the gate need not reach consistency.

3. Assume that the gate G is consistent at time t_2, and that at least one input is not stable in the interval (t_2, t_3). We do not assume that the output of G remains stable after t_2. The *contamination delay* of a gate is the amount of time that the output of a consistent gate remains stable after its inputs stop being stable. Throughout this course, unless stated otherwise, we will make the most "pessimistic" assumption about the contamination delay. Namely, we will assume that the contamination delay is zero.

4. If a gate G implements a Boolean function $f : \{0,1\}^n \to \{0,1\}$ with propagation delay t_{pd}, then G also implements a Boolean function $f : \{0,1\}^n \to \{0,1\}$ with propagation delay t', for every $t' \geq t_{pd}$. It follows that it is legitimate to use upper bounds on the actual propagation delay. Pessimistic assumptions should not render a circuit incorrect.

   In fact, the actual exact propagation delay is very hard to compute. It depends on $x(t)$ (i.e how fast does the input change?). This is why we resort to upper bounds on the propagation delays.

2.3 Building blocks

The building blocks of combinational circuits are gates and wires. In fact, we will need to consider *nets* which are generalizations of wires.

Gates. A gate, as seen in Definition 6 is a device that implements a Boolean function. The *fan-in* of a gate G is the number of inputs 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 will be using as building blocks for combinational circuits have a constant fan-in (i.e. at most $2-3$ input ports). The basic gates that we consider are: inverter (NOT-gate), OR-gate, NOR-gate, AND-gate, NAND-gate, XOR-gate, NXOR-gate, multiplexer (MUX).

The input ports of a gate G are denoted by the set $\{in(G)_i\}_{i=1}^n$, where n denotes the fan-in of G. The output ports of a gate G are denoted by the set $\{out(G)_i\}_{i=1}^k$, where k denotes the number of output ports of G.

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 7 *A* net *is a subset of terminals that are connected by wires.*

In the digital abstraction we assume that the signals all over a net are identical (why?). The *fan-out* of a net N is the number of input terminals that are connected by N.

The issue of drawing nets is a bit confusing. Figure 2.1 depicts three different drawings of the same net. All three nets contain an output terminal of an inverter and 4 input terminals of inverters. However, the nets are drawn differently. Recall that the definition of a net is simply a subset of terminals. 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.

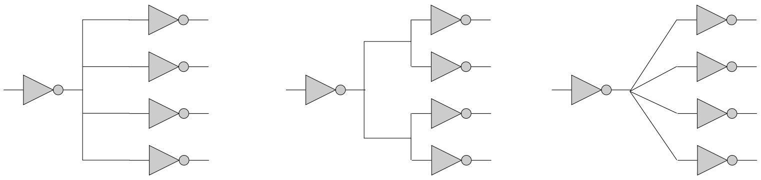

Figure 2.1: Three equivalent nets.

Consider a net N. We would like to define the digital signal $N(t)$ for the whole net. The problem is that due to noise (and other reasons) the analog signals at different terminals of the net might not equal each other. This might cause the digital interpretations of analog signals at different terminals of the net to be different, too. We solve this problem by defining $N(t)$ to logical only if there is a consensus among all the digital interpretations of analog signals at different terminals of the net. Namely, $N(t)$ is zero (one) if the digital values of all the analog signals along the net are zero (one). If there is no consensus, then $N(t)$ is non-logical. Recall that, in the bounded-noise model, different thresholds are used to interpret the digital values of the analog signals measured in input and output terminals.

We say that a net N *feeds* an input terminal t if the input terminal t is in N. We say that a net N is *fed* by an output terminal t if t is in N. Figure 2.2 depicts a an output terminal that feeds a net and an input terminal that is fed by a net. The notion of feeding and being fed implies a direction according to which information "flows"; namely, information is "supplied" by output terminals and is "consumed" by input terminals. From an electronic point of view, in "pure" CMOS gates, output terminals are connected via resistors either the the ground or the the power. Input terminals are connected only to capacitors.

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

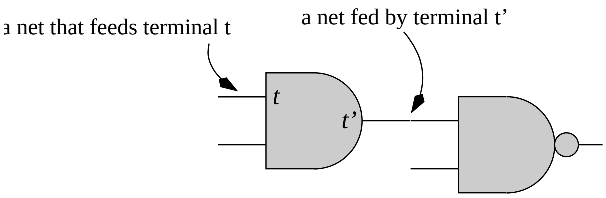

Figure 2.2: A terminal that is fed by a net and a terminal that feeds a net.

Definition 8 *A net N is* simple *if (a) N is fed by exactly one output terminal, and (b) 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 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 $t \to t_i$.

It follows that a circuit, all the nets of which are simple, may be modeled by a directed graph. We define this graph in the following definition.

Definition 9 *Let C denote a circuit, all the nets of which are simple. The directed graph $DG(C)$ is defined as follows. The vertices of the graph $DG(C)$ are the gates of C. The directed edges correspond to the the wires as follows. Consider a simple net N fed by an output terminal t that feeds the input terminals $\{t_i\}_{i \in I}$. The directed edges that correspond to N are $u \to v_i$, where u is the gate that contains the output terminal t and v_i is the gate that contains the input terminal t_i.*

Note that the information of which terminal is connected to each wire is not maintained in the graph $DG(C)$. One could of course label each endpoint of an edge in $DG(C)$ with the name of the terminal the edge is connected to.

2.4 Combinational circuits

Question 8 *Consider the circuits depicted in Figure 2.3. Can you explain why these are not valid combinational circuits?*

Before we define combinational circuits it is helpful to define two types of special gates: an input gate and an output gate. The purpose of these gates is to avoid endpoints in nets that seem to be not connected (for example, all the nets in the circuit on the right in Figure 2.3 have endpoints that are not connected to a gate).

Definition 10 (input and output gates) *An* input gate *is a gate with zero inputs and a single input. An* output gate *is a gate with one input and zero outputs.*

Figure 2.4 depicts an input gate and an output gate. Inputs from the "external world" are fed to a circuit via input gates. Similarly, outputs to the "external world" are fed by the circuit via output gates.

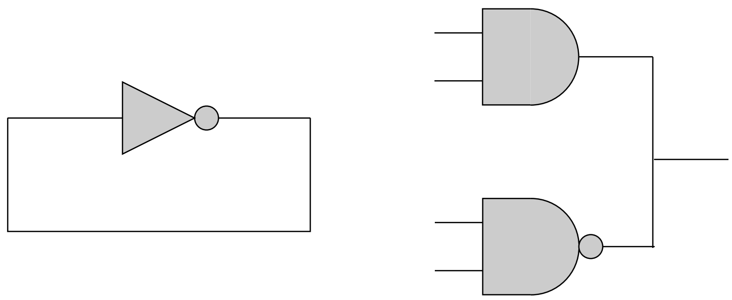

Figure 2.3: Two examples of non-combinational circuits.

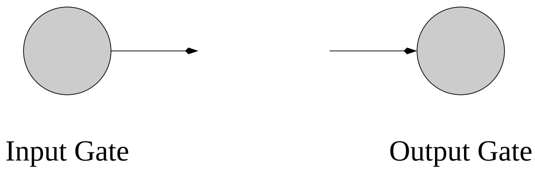

Figure 2.4: An input gate and an output gate

Consider a fixed set of gate-types (e.g. inverter, NAND-gate, etc.); we often refer to such a set of gate-types as a *library*. We associate with every gate-type in the library the number of inputs, the number of outputs, and the Boolean function that specifies its functionality.

Let $\mathcal{G}$ denote the set of gates in a circuit. Every gate $G \in \mathcal{G}$ is an *instance* of a gate from the library. Formally, the *gate-type* of a gate G indicates the library element that corresponds to G (e.g. "the gate-type of G is an inverter"). To simplify the discussion, we simply refer to a gate G as an inverter instead of saying that its gate-type is an inverter.

We now present a syntactic definition of combinational circuits.

Definition 11 (syntactic definition of combinational circuits) *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.*

Note that Definition 11 is independent of the gate types. One need not even know the gate-type of each gate to determine whether a circuit is combinational. Moreover, the question of whether a circuit is combinational is a purely topological question (i.e. are the interconnections between gates legal?).

Question 9 *Which conditions in the syntactic definition of combinational circuits are violated by the circuits depicted in Figure 2.3?*

We list below a few properties that explain why the syntactic definition of combinational circuits is so important. In particular, these properties show that the syntactic definition of combinational circuits implies well defined semantics.

1. Completeness: for every Boolean function f, there exists a combinational circuit that implements f. We leave the proof of this property as an exercise for the reader.

2. Soundness: every combinational circuit implements a Boolean function. Note that it is NP-Complete to decide if the Boolean function that is implemented by a given combinational circuit with one output ever gets the value 1.

3. Simulation: given the digital values of the inputs of a combinational circuit, one can simulate the circuit in linear time. Namely, one can compute the digital values of the outputs of the circuit that are output by the circuit once the circuit becomes consistent.

4. Delay analysis: given the propagation delays of all the gates in a combinational circuit, one can compute in linear time an upper bound on the propagation delay of the circuit. Moreover, computing tighter upper bounds is again NP-Complete.

The last three properties are proved in the following theorem by showing that in a combinational circuit every net implements a Boolean function of the inputs of the circuit.

Theorem 2 (Simulation theorem of combinational circuits) *Let $C = \langle \mathcal{G}, \mathcal{N} \rangle$ denote a combinational circuit that contains k input gates. Let $\{x_i\}_{i=1}^k$ denote the output terminals of the input gates in C. Assume that 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 $t \in [t_1 + t_{pd}(N), t_2]$.

We can simplify the statement of Theorem 2 by considering each net $N \in \mathcal{N}$ as an output of a combinational circuit with k inputs. The theorem then states that every net implements a Boolean function with an appropriate propagation delay.

We use $\vec{x}(t)$ to denote the vector $x_1(t), \ldots, x_k(t)$.

Proof: Let n denote the number of gates in $\mathcal{G}$ and m the number of nets in $\mathcal{N}$. The directed graph $DG(C)$ is acyclic. It follows that we can topologically sort the vertices of $DG(C)$. Let $v_1, v_2, \ldots, v_n$ denote the set of gates $\mathcal{G}$ according to the topological order. (This means that if there is a directed path from v_i to v_j in $DG(C)$, then $i < j$.) We assume, without loss of generality, that $v_1, \ldots, v_k$ is the set of input gates.

Let $\mathcal{N}_i$ denote the subset of nets in $\mathcal{N}$ that are fed by gate v_i. Note that if v_i is an output gate, then $\mathcal{N}_i$ is empty. Let $e_1, e_2, \ldots, e_m$ denote an ordering of the nets in $\mathcal{N}$ such that nets in $\mathcal{N}_i$ precede nets in $\mathcal{N}_{i+1}$, for every $i < n$. In other words, we first list the nets fed by gate v_1, followed by a list of the nets fed by gate v_2, etc.

Having defined a linear order on the gates and on the nets, we are now ready to prove the theorem by induction on m (the number of nets).

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

Induction Basis: We prove the induction basis for $m' = k$. Consider an $i \leq k$. Note that, for every $i \leq k$, e_i is fed by the input gate v_i. Let x_i denote the output terminal of v_i. It follows that the digital signal along e_i always equals the digital signal $x_i(t)$. Hence we define f_{e_i} to be simply the projection on the ith component, namely $f_{e_1}(\sigma_1, \ldots, \sigma_k) = \sigma_i$. The propagation delay $t_{pd}(e_i)$ is zero. The induction basis follows.

Induction Step: Assume that the induction hypothesis holds for $m' < m$. We wish to prove that it also holds for $m' + 1$. Consider the net $e_{m'+1}$. Let v_i denote the gate that feeds the net $e_{m'+1}$. To simplify notation, assume that the gate v_i has two terminals that are fed by the nets e_j and e_k, respectively. The ordering of the nets guarantees that $j, k \leq m'$. By the induction hypothesis, the net e_j (resp., e_k) implements a Boolean function f_{e_j} (resp., f_{e_k}) with propagation delay $t_{pd}(e_j)$ (resp., $t_{pd}(e_k)$). This implies that both inputs to gate v_i are stable during the interval

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

Gate v_i implements a Boolean function f_{v_i} with propagation delay $t_{pd}(v_i)$. It follows that 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

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

We define $f_{e_{m'+1}}$ to be the Boolean function obtained by the composition of Boolean functions $f_{e_{m'+1}}(\vec{\sigma}) = f_{v_i}(f_{e_j}(\vec{\sigma}), f_{e_k}(\vec{\sigma}))$. We define $t_{pd}(e_{m'+1})$ to be $\max\{t_{pd}(e_j), t_{pd}(e_k)\} + t_{pd}(v_i)$, and the induction step follows. □

The digital abstraction allows us to assume that the signal corresponding to every net in a combinational circuit is logical (provided that the time that elapses since the inputs become stable is at least the propagation delay). This justifies the convention of identifying a net with the digital value of the net.

The proof of Theorem 2 leads to two related algorithms. One algorithm simulates a combinational circuit, namely, given a combinational circuit and a Boolean assignment to the inputs $\vec{x}$, the algorithm can compute the digital signal of every net after a sufficient amount of time elapses. The second algorithm computes the propagation delay of each net. Of particular interest are the nets that feed the output gates of the combinational circuit. Hence, we 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.

The algorithms are very easy. For convenience we describe them as one joint algorithm. First, the directed graph $DG(C)$ is constructed (this takes linear time). Then the gates are sorted in topological order (this also takes linear time). This order also induced an order on the nets. Now a sequence of *relaxation* steps take place for nets $e_1, e_2, \ldots, e_m$. In a relaxation step the propagation delay of a net e_i two computations take place:

1. The Boolean value of e_i is set to
$$f_{v_j}(\vec{I}_{v_j}),$$
where v_j is the gate that feeds the net e_i and $\vec{I}_{v_j}$ is the binary vector that describes the values of the nets that feed gate v_j.

2. The propagation delay of the gate that feeds e_i is set to
$$t_{pd}(e_i) \leftarrow t_{pd}(v_j) + \max\{t_{pd}(e')\}_{\{e' \text{ feeds } v_j\}}.$$

If the number of outputs of each gate is constant, then the total amount of time spend in the relaxation steps is linear, and hence the running time of this algorithm is linear. (Note that the input length is the number of gates plus the sum of the sizes of the nets.)

Question 10 *Prove that the total amount of time spent in the relaxation steps is linear if the fan-in of each gate is constant (say, at most* 3*).*

Note that it is not true that each relaxation step can be done in constant time if the fan-in of the gates is not constant. Can you still prove linear running time if the fan-in of the gates is not constant but the number of outputs of each gate is constant?

Can you suggest a slight weakening of this restriction which still maintains a linear running time?

2.5 Cost and propagation delay

In this section we define the cost and propagation delay of a combinational circuit.

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

Definition 12 *The cost of a combinational circuit* $C = \langle \mathcal{G}, \mathcal{N} \rangle$ *is defined by*
$$c(C) \triangleq \sum_{G \in \mathcal{G}} c(G).$$

The following definition defined the propagation delay of a combinational circuit.

Definition 13 *The propagation delay of a combinational circuit $C = \langle \mathcal{G}, \mathcal{N} \rangle$ is defined by*

$$t_{pd}(C) \stackrel{\triangle}{=} \max_{N \in \mathcal{N}} t_{pd}(N).$$

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

Definition 14 *A sequence $p = \{v_0, v_1, \ldots, v_k\}$ of gates from $\mathcal{G}$ is a* path *in a combinational circuit $C = \langle \mathcal{G}, \mathcal{N} \rangle$ if p is a path in the directed graph $DG(C)$.*

The propagation delay of a path p is defined as

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

The proof of the following claim follows directly from the proof of Theorem 2.

Claim 3 *The propagation delay of a combinational circuit $C = \langle \mathcal{G}, \mathcal{N} \rangle$ equals*

$$t_{pd}(C) = \max_{\textit{paths } p} t_{pd}(p)$$

Paths, the delay of which equals the propagation delay of the circuit, are called *critical paths*.

Question 11

1. *Describe a combinational circuit with n gates that has at least $2^{n/2}$ paths. Can you describe a circuit with 2^n different paths?*
2. *In Claim 3 the propagation delay of a combinational circuit is defined to be the maximum delay of a path in the circuit. The number of paths can be exponential in n. How can we compute the propagation delay of a combinational circuit in linear time?*

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. They referred to these figures as Motorola and Venus. Table 2.1 summarizes the normalized costs and delays in these technologies.

2.6 Syntax and semantics

In this chapter we have used both explicitly and implicitly the terms *syntax* and *semantics*. These terms are so fundamental that they deserve a section.

The term semantics (in our context) refers to the function that a circuit implements. Often, the semantics of a circuit is referred to as the *functionality* or even the *behavior* of the circuit. In general, the semantics of a circuit is a formal description that relates the outputs of the circuit to the inputs of the circuit. In the case of combinational circuits, semantics are described by Boolean functions. Note that in non-combinational circuits, the

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

Table 2.1: Costs and delays of gates

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

The term syntax refers to a formal set of rules that govern how "grammatically correct" circuits are constructed from smaller circuits (just as sentences are built of words). In the syntactic definition of combinational circuits the functionality (or gate-type) of each gate is not important. The only part that matters is that the rules for connecting gates together are followed. Following syntax in itself does not guarantee that the resulting circuit is useful. Following syntax is, in fact, a restriction that we are willing to accept so that we can enjoy the benefits of well defined functionality, simple simulation, and simple timing analysis. The restriction of following syntax rules is a reasonable choice since every Boolean function can be implemented by a syntactically correct combinational circuit.

2.7 Summary

Combinational circuits were formally defined in this chapter. We started by considering the basic building blocks: gates and wires. Gates are simply implementations of Boolean functions. The digital abstraction enables a simple definition of what it means to implement a Boolean function f. Given a propagation delay t_{pd} and stable inputs whose digital value is $\vec{x}$, the digital values of the outputs of a gate equal $f(\vec{x})$ after t_{pd} time elapses.

Wires are used to connect terminals together. Bunches of wires are used to connect multiple terminals to each other and are called nets. Simple nets are nets in which the direction in which information flows is well defined; from output terminals of gates to input terminals of gates.

The formal definition of combinational circuits turns out to be most useful. It is a syntactic definition that only depends on the topology of the circuit, namely, how the terminals of the gates are connected. One can check in linear time whether a given circuit is indeed a combinational circuit. Even though the definition ignores functionality, one can compute in linear time the digital signals of every net in the circuit. Moreover, one can also compute in linear time the propagation delay of every net.

Two quality measures are defined for every combinational circuit: cost and propagation delay. The cost of a combinational circuit is the sum of the costs of the gates in the circuit. The propagation delay of a combinational is the maximum delay of a path in the circuit.