3.6. Boolean Discrete Event Modeling of Circuit Structures

Gürkan Uygur Sebastian M. Sattler

3.6.1. Different Abstraction Levels for Modeling

Considering circuit structures on the transistor level is not only suitable for analog design, simulation, and validation, but also allows discrete event abstraction of the digital signal flow on a given structure. This abstraction can be represented as a Directed Graph (DG),

$$\mathrm{DG} = (N, E) \ ,$$

specifying a network (DG) with a set of nodes N and set of directed edges $E \subseteq N \times N$. Sometimes a DG is also denoted as a DiGraph.

In more concrete terms of a Signal Flow Graph (SFG), the nodes $n \in N$ represent instantaneous signal information on pins, ports and wires respectively, labeled as n, and the edges $(n, n') \in E$ specify the signal flow from the source node n to the target node n'. By considering an edge (n, n') as a string nn' of length two, concatenating two edges

$$(n_1 n_1') \cdot (n_0 n_0') \in N^*$$

and strings of nodes

$$(n_1 \cdots n_1') \cdot (n_0 \cdots n_0') \in N^*$$

of arbitrary length, respectively, along a path or cycle results in the SFG. This is a most formally abstract structural view of an arbitrary circuit.

One of its advantages is that it provides a topological approach observing and analyzing structural properties. In general, a circuit structure can be divided into combinational and sequential parts: a circuit structure implements a combinational logic, if the structure does not contain any simple cycle, that's to say an elementary circuit [160].

On the other hand, the structure has to contain at least one simple cycle—in a topological manner of speaking, a topological hole [406]—for feeding-back its state information to realize a sequential logic. Of course, more precisely, we need to factor in more criteria to differ between combinational and sequential logic. For example, it needs to be checked whether and when an elementary circuit is alive, so to speak, whether and when its gates are open, such that the circuit is in a position to change its logical information content depending on the input information as well as the fed-backed situation.

An SFG can be considered in more concrete terms of a Signal Flow Plan (SFP), which consists of modules M^i representing (partially defined) functionalities

$$ {}^{i}f : {}^{i}\mathbf{x} \mapsto {}^{i}\mathbf{y} \ , $$

whereas the connections (pins, ports, wires) ${}^{i}(\mathbf{x}, \mathbf{y})$ of modules M^i with

$$ \begin{aligned} {}^{i}(\mathbf{x}, \mathbf{y}) &= ({}^{i}\mathbf{x}, {}^{i}\mathbf{y}) \\ &= \left({}^{i}x_{n-1}, \cdots, {}^{i}x_j, \cdots, {}^{i}x_1, {}^{i}x_0, {}^{i}y_{m-1}, \cdots, {}^{i}y_k, \cdots, {}^{i}y_1, {}^{i}y_0\right) \end{aligned} $$

and ${}^{i}x_j, {}^{i}y_k \in \{1, 0\}$ are associated with the nodes in SFG. Within this abstraction in general, the elementary functions can be composed to two types of functions, the state transition function implementing the sequential logic, and the output function implementing the combinational logic, termed as a Mealy machine [390].

Furthermore, Petri nets are one famous general purpose modeling technique for discrete event systems with lots of derivatives for special applications [200]. Signal Transition Graphs (STGs) are a widely used interpreted Petri net based formalism with transitions labeled with rising and falling edges for the description of asynchronously fed-backed circuits, with several existing approaches to their synthesis [278].

Transaction Level Modeling (TLM) together with the modeling language SystemC become more and more relevant in system design [64, 151, 369]. Both TLM as a modeling concept and SystemC as a software language build a powerful tool for system simulation, synthesis and verification. The trace theory applied on Petri nets [193] together with word replacement systems [36] provide a formal language based computational algebraic modeling and simulation framework.

3.6.2. Theoretical Foundation

Meaning of the Used Operators, Constants, and Expressions. In this subsection we introduce the propositional syntax which qualifies for warranting a unique and consistent relation between propositional (truth) expressions and specified circuit structure elements [134]. In this we profit from the basic language elements of Boolean Algebra: totally and partially defined implication and equivalence. The statement of grounds is inherent: a specification composes only truth statements which are enforced to be fulfilled. Thus, we take the point of view that everything which is not specified becomes undefined and termed with the symbol $*$. For total implication and equivalence we take the symbols $\rightarrow$ and $\leftrightarrow$, respectively, and for partial implication and equivalence we take the symbols $\Rightarrow$ and $\Leftrightarrow$, respectively. Let h^* stand for the Boolean expression which represents the disjunction of all undefined assignments [390]. We declare that t means "true" and f means "false" per definition.

Propositional Syntax for Structure Specification. Table 3.23 shows the deployed syntax and semantics for the implication and equivalence defined for a partial and total case. Table 3.24 defines the specification of proposition C for the same two cases. Based on the theoretical foundation given in Tables 3.23 and 3.24 we will introduce our propositional syntax for structure specification.

Table 3.23. Definition of the implication and equivalence for two cases

			partial		total	
A	B	C	$A \Rightarrow B$	$A \Leftrightarrow B$	$A \rightarrow B$	$A \leftrightarrow B$
f	f	f	t	t	t	t
f	t	f	t	$*$	t	f
t	f	f	$*$	$*$	f	f
t	t	f	t	t	t	t
f	f	t	t	t	t	t
f	t	t	t	$*$	t	f
t	f	t	$*$	$*$	f	f
t	t	t	t	t	t	t

Table 3.24. Partial and total specification of proposition C

			partial	total
A	B	C	$C \Leftarrow (A \Rightarrow B)$	$C \Leftrightarrow (A \rightarrow B)$
f	f	f	$*$	$*$
f	t	f	$*$	$*$
t	f	f	$*$	t
t	t	f	$*$	$*$
f	f	t	t	t
f	t	t	t	t
t	f	t	$*$	$*$
t	t	t	t	t

Declaration of Propositional Variables Respectively Logical Pins. We introduce the most elementary language component:

- in formal language view, this is the declaration of a literal;
- in specification view, this is the declaration of propositions without the predefined degree of truth, propositional variables without any predefined information content so to speak;
- in logic view, this is just a pin or wire without information about its predicate with regard to allowed or undefined logical values, and further, without information about predefined structural pin property, e.g., input, output, and state.

The syntax for declaring a literal, propositional variable and pin name, respectively, denoted as A, is termed as:

$$\mathrm{t} \Leftarrow A \ . \tag{3.79}$$

Obviously, $\mathrm{t} \Leftarrow A$ is a truth expression warranted by the tautological implication. Looking closer, we see that the conclusion t is always truth per definition, hence the premise becomes trivial. Thus, A gets declared as a propositional variable without any predefined degree of truth.

Partially Specified Truth (Tautological) and Proposition (Rule). The syntax for partially specifying truth proposition C is termed as

$$C \Leftarrow (A \Rightarrow B) \tag{3.80}$$

whereby the premise A is a literal or a conjunction of literals, and the conclusion B is a literal or a disjunction of literals. Further, the literal C becomes defined as an output pin with the predicate that C is specified to be true for ever; in other words, C is undefined to be false, $A \wedge \overline{B} \triangleq *$. With this in mind, we also denote $C \Leftarrow (A \Rightarrow B)$ as tautological proposition C.

Totally Specified Proposition (Test). The syntax for totally specified proposition C is given as

$$C \Leftrightarrow (A \rightarrow B) \tag{3.81}$$

whereby the premise A is a literal or a conjunction of literals, and the conclusion B is a literal or a disjunction of literals. Further, the previously declared literal C becomes defined as an output pin with the predicate that its degree of truth is determined by the test $A \rightarrow B$. With this in mind, we also denote $C \Leftrightarrow (A \rightarrow B)$ as test C.

3.6.3. Syntax for Discrete Event Modeling Based on Partially and Totally Specified Propositions

Specification of Function ${}^{i}f$. In terms of a Signal Flow Plan (SFP), for a module M^i with associated function ${}^{i}f : {}^{i}\mathbf{x} \mapsto {}^{i}\mathbf{y}$ and the input vector ${}^{i}\mathbf{x} = {}^{i}(\mathbf{p}, \mathbf{k})$:

- the propositional expression

$$ {}^{i}\mathbf{y} \Leftarrow ({}^{i}\mathbf{p} \Rightarrow {}^{i}\mathbf{k}) \tag{3.82}$$

  specifies the partially defined function

$$ {}^{i}\mathbf{y} = {}^{i}f({}^{i}\mathbf{p}, {}^{i}\mathbf{k}) = \overline{{}^{i}\mathbf{p}} \vee {}^{i}\mathbf{k} = \overline{{}^{i}\mathbf{p} \wedge \overline{{}^{i}\mathbf{k}}} \tag{3.83}$$

  whereas the complement of ${}^{i}\mathbf{y}$ is undefined, $\overline{{}^{i}\mathbf{y}} \triangleq *$; and

- the propositional expression

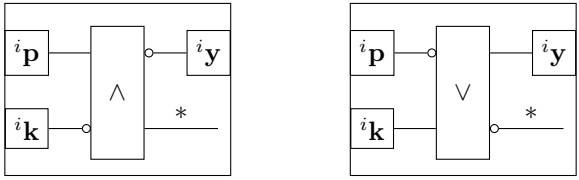
$$^i\mathbf{y} \Leftrightarrow (^i\mathbf{p} \rightarrow {}^i\mathbf{k}) \tag{3.84}$$

specifies the totally defined function

$$^i\mathbf{y} = {}^if(^i\mathbf{p}, {}^i\mathbf{k}) = \overline{^i\mathbf{p}} \vee {}^i\mathbf{k} = \overline{^i\mathbf{p} \wedge \overline{^i\mathbf{k}}} \tag{3.85}$$

with the complement of $^i\mathbf{y}$ is defined as $\overline{^i\mathbf{y}} = {}^i\mathbf{p} \wedge \overline{^i\mathbf{k}}$.

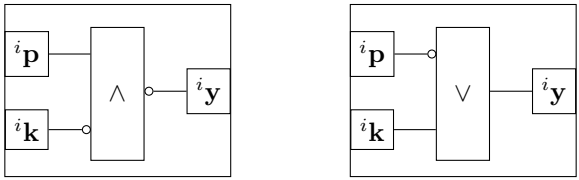
Figure 3.22 gives an overview of a partially respectively totally defined function with associated module view:



$^i\mathbf{y} = \neg(^i\mathbf{p} \wedge \neg {}^i\mathbf{k})$ $\quad$ $^i\mathbf{y} = \neg {}^i\mathbf{p} \vee {}^i\mathbf{k}$

$\overline{^i\mathbf{y}} \triangleq *$ $\quad$ $\overline{^i\mathbf{y}} \triangleq *$

(a) NAND and OR view of $^i\mathbf{y} \Leftarrow (^i\mathbf{p} \Rightarrow {}^i\mathbf{k})$



$^i\mathbf{y} = \neg(^i\mathbf{p} \wedge \neg {}^i\mathbf{k})$ $\quad$ $^i\mathbf{y} = \neg {}^i\mathbf{p} \vee {}^i\mathbf{k}$

$\overline{^i\mathbf{y}} = \quad {}^i\mathbf{p} \wedge \neg {}^i\mathbf{k}$ $\quad$ $\overline{^i\mathbf{y}} = \quad {}^i\mathbf{p} \wedge \neg {}^i\mathbf{k}$

(b) NAND and OR view of $^i\mathbf{y} \Leftrightarrow (^i\mathbf{p} \rightarrow {}^i\mathbf{k})$

Figure 3.22. Module M^i with input vector $^i\mathbf{x} = {}^i(\mathbf{p}, \mathbf{k})$ and output vector $^i\mathbf{y}$: (a) partially and (b) totally defined.

Module Concatenation by Pin Connection. The concatenations of modules in the SFP (e.g., $M^u \vdash M^v$) and the connections to the pri-

mary pins can be specified by the coupling function [390]:

$$k : \mathbf{x} \mapsto k(\mathbf{x}) \ .$$

To connect the input pin ${}^{v}x_i$ of M^v with the output pin ${}^{u}y_j$ of M^u, we term:

$${}^{v}x_i \Leftrightarrow (\mathrm{t} \rightarrow {}^{u}y_j) \ . \tag{3.86}$$

3.6.4. Use Case: Discrete Event Modeling and Composition of a CMOS Inverter

Modeling the Pull-Up Transistor. Figure 3.23 (a) shows the pull-up transistor of a CMOS inverter structure. The power supply pin V_{DD} is modeled to be clamped to truth:

$$\mathrm{t} \Rightarrow V_{\mathrm{DD}} \tag{3.87}$$

Hence, the pull-up output pin PU is modeled to be truth, if the low-active gate input $\bar{A}$ is assigned with digital 0, then its degree of truth is false; otherwise PU is electrically denoted as tri-state and digitally undefined. Figures 3.23 (b) and (c) show the associated truth table and block view. Figures 3.23 (d) and (e) state the module view and the specification:

- module M^1 with the pin encoding

$${}^{1}(\mathbf{x}, \mathbf{y}) = ({}^{1}x_1, {}^{1}x_0, {}^{1}y_0) = (V_{\mathrm{DD}}, \bar{A}, \bar{T}) \tag{3.88}$$

  and instantiation of the test $\bar{T}$

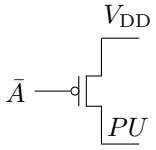
$$\bar{T} \Leftrightarrow (V_{\mathrm{DD}} \rightarrow \bar{A}) \ ; \tag{3.89}$$

- module M^2 with the pin encoding

$${}^{2}(\mathbf{x}, \mathbf{y}) = ({}^{2}x_1, {}^{2}x_0, {}^{2}y_0) = (\bar{T}, \mathrm{f}, PU) \tag{3.90}$$

  and instantiation of the specification PU

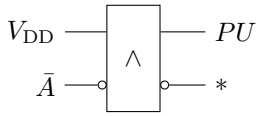
$$PU \Leftarrow (\bar{T} \Rightarrow \mathrm{f}) \ ; \text{and} \tag{3.91}$$



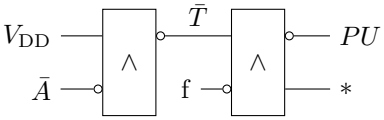
(a) circuit

V_{DD}	$\bar{A}$	PU
t	f	t
t	t	$*$

(b) truth table



(c) `AND` block view



(d) `NAND` module view

$$\bar{T} \Leftrightarrow (V_{DD} \to \bar{A}) \qquad {}^1y_0 \Leftrightarrow ({}^1x_1 \to {}^1x_0)$$
$$PU \Leftarrow (\bar{T} \Rightarrow \mathrm{f}) \qquad {}^2y_0 \Leftarrow ({}^2x_1 \Rightarrow {}^2x_0),\ {}^2x_0 \Rightarrow \mathrm{f}$$
$$M^1 \vdash M^2 \qquad {}^2x_0 \Leftrightarrow (\mathrm{t} \to {}^1y_0)$$

(e) partially specified function ${}^2y_0 = PU$

Figure 3.23. Pull-up transistor circuit of a CMOS inverter structure realizing: $PU = \neg\bar{A} \wedge V_{DD}$, $\overline{PU} \triangleq *$.

- the module concatenation $M^1 \vdash M^2$ by connecting ${}^2x_0$ with ${}^1y_0$:

$$ {}^2x_0 \Leftrightarrow (\mathrm{t} \to {}^1y_0) \ . \tag{3.92}$$

Table 3.25 gives an overview of the pin specification.

Table 3.25. Pin specification of the pull-up transistor

Syntax	Description
$V_{DD} \Rightarrow \mathrm{t}$	High-active pin V_{DD} at source
$\bar{A} \Rightarrow \mathrm{t}$	Low-active pin $\bar{A}$ at gate
$PU \Rightarrow \mathrm{t}$	High-active pin PU at drain
$\mathrm{t} \Rightarrow V_{DD}$	Power supply pin V_{DD} is clamped to truth

Modeling the Pull-Down Transistor. Figure 3.24 (a) shows the pull-down transistor of a CMOS inverter structure. The ground pin $\overline{GND}$ is modeled to be clamped to false:

$$\mathrm{f} \Leftarrow \overline{GND} \; . \tag{3.93}$$

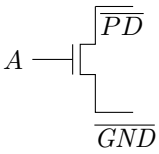
Hence, the pull-down output pin $\overline{PD}$ is modeled to be false if the high-active gate input A is assigned with digital 1, then its degree of truth is true; otherwise $\overline{PD}$ is electrically denoted as tri-state and digitally undefined. Figures 3.24 (b) and (c) show the associated truth table and block view. Figures 3.24 (d) and (e) state the module view and the specification:

- module M^3 with the pin encoding

$$^3(\mathbf{x}, \mathbf{y}) = (^3x_1, ^3x_0, ^3y_0) = (\overline{GND}, A, \overline{PD}) \; , \tag{3.94}$$

and instantiation of the specification $\overline{PD}$

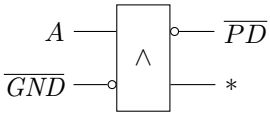
$$\overline{PD} \Leftarrow (A \Rightarrow \overline{GND}) \; . \tag{3.95}$$



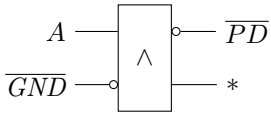
(a) circuit

$\overline{GND}$	A	$\overline{PD}$
f	f	$*$
f	t	f

(b) truth table



(c) `NAND` block view



(d) `NAND` module view

$$\overline{PD} \Leftarrow (A \Rightarrow \overline{GND}) \qquad {}^3y_0 \Leftarrow ({}^3x_1 \Rightarrow {}^3x_0)$$

(e) partially specified function $^3y_0 = \overline{PD}$

Figure 3.24. Pull-down transistor circuit of a CMOS inverter structure realizing: $\overline{PD} = \neg\left(A \wedge \neg\overline{GND}\right)$, $\overline{\overline{PD}} \triangleq *$.

Table 3.26 gives an overview of the pin specification.

Table 3.26. Pin specification of the pull-down transistor

Syntax	Description
$\overline{GND} \Rightarrow \mathrm{t}$	Low-active pin $\overline{GND}$ at source
$A \Rightarrow \mathrm{t}$	High-active pin A at gate
$\overline{PD} \Rightarrow \mathrm{t}$	Low-active pin $\overline{PD}$ at drain
$\overline{GND} \Rightarrow \mathrm{f}$	Ground pin $\overline{GND}$ is clamped to false

Parallel Composition of Pull-Up and Pull-Down. Figure 3.25 (a) shows the CMOS inverter composed of pull-up and pull-down circuits by electrically wiring the (tri-state) output pins PU and $\overline{PD}$. Further, the primary input pin A is connected with the low-active gate input pin $\bar{A}$ of the pull-up transistor and the high-active gate input pin A of the pull-down transistor. The composition of the electrical level becomes immediately modeled by parallelly composing the concatenated modules $M^1 \vdash M^2$ of the pull-up circuit and the module M^3 of the pull-down circuit at Module M^4:

$$\begin{aligned} M^2 &\vdash M^4 \ , \\ M^3 &\vdash M^4 \ . \end{aligned} \tag{3.96}$$

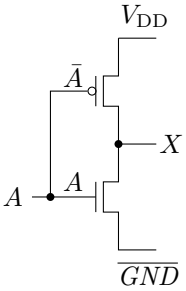
At this point, it is essential to consider that the partially specified proposition $\overline{PD}$, which *positively* encodes the digital value 0, must not be *switched* to the Boolean value 0, such that the Boolean superimposing ($\vee$) of the components $(PU, \overline{PD})$ implementing the CMOS inverter consistently holds:

$$\neg\bar{A} \wedge \mathrm{V_{DD}} \ \vee \ \neg \left(A \wedge \neg\overline{GND}\right) \Rightarrow \neg\bar{A} \ \vee \ \neg A \Rightarrow \bar{A} \Rightarrow X \ . \tag{3.97}$$

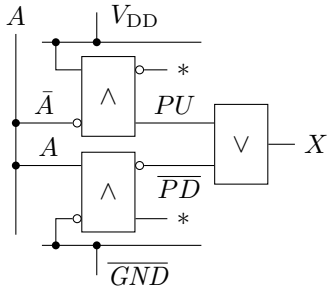
Figure 3.25 (e) states the specification of the module composition resulting in the totally defined CMOS inverter:

- module M^4 with the pin encoding

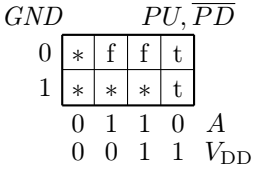
$$^4(\mathbf{x}, \mathbf{y}) = (^4x_1, {}^4x_0, {}^4y_0) = (\overline{PD}, PU, X) \tag{3.98}$$



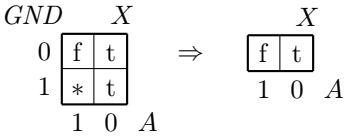
(a) circuit



(b) OR block view



(c) parallelly composed functions



(d) resolved inverter function

$$X \Leftrightarrow (\neg\overline{PD} \rightarrow PU)$$
$$M^2 \vdash M^4$$
$$M^3 \vdash M^4$$

$${}^4y_0 \Leftrightarrow (\neg {}^4x_1 \rightarrow {}^4x_0)$$
$${}^4x_0 \Leftrightarrow (\mathrm{t} \rightarrow {}^2y_0)$$
$${}^4x_1 \Leftrightarrow (\mathrm{t} \rightarrow {}^3y_0)$$

(e) specified parallel composition of pull-up and pull-down

Figure 3.25. CMOS inverter parallelly composed of pull-up and pull-down transistors: $X = PU \vee \overline{PD}$.

and instantiation of the test X

$$X \Leftrightarrow (\neg\overline{PD} \rightarrow PU) \; ; \tag{3.99}$$

- module concatenation $M^2 \vdash M^4$ by connecting ${}^4x_0$ with ${}^2y_0$:

$${}^4x_0 \Leftrightarrow (\mathrm{t} \rightarrow {}^2y_0) \; ; \tag{3.100}$$

- module concatenation $M^3 \vdash M^4$ by connecting ${}^4x_1$ with ${}^3y_0$:

$${}^4x_1 \Leftrightarrow (\mathrm{t} \rightarrow {}^3y_0) \; ; \tag{3.101}$$

- declaration of the primary input pin A of M^0 with the pin encoding

$$ {}^0\mathbf{x} = ({}^0x_0) = A \tag{3.102} $$

and the pin declaration

$$ \mathrm{t} \Leftarrow A \; ; \tag{3.103} $$

- module concatenation $M^0 \vdash M^1$ by connecting ${}^1x_0$ with ${}^0x_0$:

$$ {}^1x_0 \Leftrightarrow (\mathrm{t} \rightarrow {}^0x_0) \; ; \tag{3.104} $$

- module concatenation $M^0 \vdash M^3$ by connecting ${}^3x_1$ with ${}^0x_0$:

$$ {}^3x_1 \Leftrightarrow (\mathrm{t} \rightarrow {}^0x_0) \; . \tag{3.105} $$

3.6.5. Results

In general, circuit structures are electrically composed of tri-state components. In digital abstraction, a tri-state output pin can be considered as an output variable of partial logic function, which is undefined in the case that the output pin is in a so-called high impedance state. The problem definition is to deploy a suitable and qualified abstraction level for discrete event modeling of such circuit systems. The most abstraction is given by modeling with Directed Graphs (DGs), respectively Signal Flow Graphs (SFGs). Signal Flow Plan (SFP) modeling is more concrete but a still sufficient abstraction level for the given problem definition.

In this work we developed a propositional syntax for language based building blocks, respectively modules of partial as well as total functionality. The deployed propositional syntax provides a declaration of high-active respectively low-active pins. The building blocks allow us to specify local input respectively output pins as well as the signal flow on these pins within a module. Additionally, the propositional syntax supports module concatenation by connecting pins. Thus, we presented our theoretical foundation for the syntactical specification

of an SFP. The advantage of the SFP considered is that it allows each specified module to proceed directly to Boolean functions.

As a use case, we demonstrated the modeling of a Complementary Metal-Oxide-Semiconductor (CMOS) inverter. For this, we modeled the pull-up circuit and pull-down circuit, and provided different abstraction levels and brought them into relation. Further, we presented our module composition resulting in the overall SFP modeling of the use case structure. It is notable that the resulting SFP represents uniquely and consistently all of the syntactically specified structure elements. Hence, the SFP exhibits all necessary information for extracting the abstract SFG level for structural analysis.