Available online at www.sciencedirect.com
SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science 153 (2006) 23–33

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

ESBC: an application for computing stabilization bounds

Alessandro Avellone[a] Mauro Ferrari[b,1] Camillo Fiorentini[c]
Guido Fiorino[a] Ugo Moscato[a]

[a] *Dipartimento di Metodi Quantitativi per le Scienze Economiche Aziendali, Università Milano-Bicocca, piazza dell'Ateneo Nuovo 1, 20126 Milano, Italy*

[b] *Dipartimento di Informatica e Comunicazione, Università degli Studi dell'Insubria, via Mazzini 5, 21100 Varese, Italy*

[c] *Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, via Comelico 39, 20135 Milano, Italy*

Abstract

We describe the application ESBC to perform the timing analysis of a combinatorial circuit. The circuit is described by formulas of Classical Logic and the delays of propagation of the signals in a gate are represented by a kind of valuation form semantics. ESBC computes the exact stabilization times at which the output signals stabilize.

Keywords: Valuation form semantics, timing analysis, intermediate logics

1 Introduction

Valuation forms provide an intensional semantics of formulas inspired to the BHK explanation of constructive connectives (see e.g., [3,8]). Roughly speaking, a valuation form for a formula A is an "object" providing a *constructive justification* for the truth of A. In recent years, several families of valuation forms have been devised to formalize different problems in logic and computer science. As an example we cite [6] where valuation forms are studied from a purely logical point of view; [5] where they are used to formalize databases;

[1] Contacting author: `mauro.ferrari@uninsubria.it`.

doi:10.1016/j.entcs.2005.08.004

[4] where valuation forms characterize stabilization bounds in combinatorial circuits and [7] where valuation forms provide the semantics of CML, a Constructive Modeling Language for Object Oriented systems.

One of the interesting aspects of valuation forms is that they are compatible with classical semantics. In the framework of formal methods one can use formulas with the intended classical meaning to formalize the system at hand and valuation forms to characterize some intensional property of the system. As an example, in the context of timing analysis formulas describe the usual functional behavior of the circuit components while valuation forms are the functions describing the delays of the circuit components. We remark that in this context valuation forms provide a *partial* semantical justification of the truth of the formulas since they only characterize the correctness of the circuit *up to* stabilization bounds.

All the above mentioned families of valuation forms characterize constructive superintuitionistic logics with constructive negation and any of these logics is a superset of the logic **E** studied in [6]. This means that the natural deduction calculus $\textsc{Nd}_{\mathbf{E}}$ for logic **E** devised in [6] is valid for all these semantics. This allows us to define a parametric extraction procedure that, selected the family of valuation forms of interest, extracts from an $\textsc{Nd}_{\mathbf{E}}$ proof of $A_1, \ldots, A_n \vdash B$ a function computing the valuation form justifying B, having as inputs the valuation forms for $A_1, \ldots, A_n$.

In this paper we describe the stabilization bound semantics and an application built to compute and evaluate the valuation forms describing stabilization bounds of combinatorial circuits. This application consists of a C++ theorem prover for the propositional logic **E** based on tableau calculi (fully described in [1]), a Java package translating tableau proofs into $\textsc{Nd}_{\mathbf{E}}$ proofs, a Java package extracting the function computing the stabilization bound from a correctness $\textsc{Nd}_{\mathbf{E}}$-proof of the circuit and the Java package which evaluates the function.

The paper is structured as follows: in Section 2 we discuss the stabilization bounds semantics, in Section 3 we briefly describe the tableau calculus, the theorem prover and the translation into $\textsc{Nd}_{\mathbf{E}}$-proofs and finally in Section 4 we describe the implementation of the valuation form extractor and provide an example.

2 Exact Stabilization Bounds

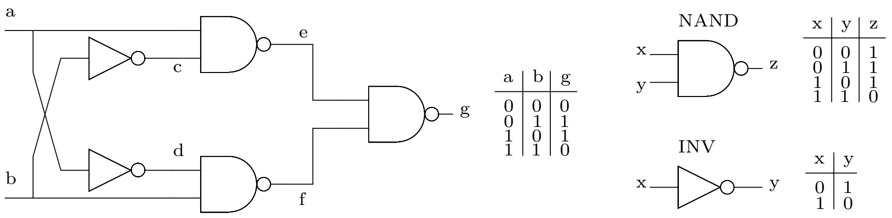
In the logical approach to circuit analysis a semantics represents an abstraction from the physical details. To give an example, let us consider the gates INV and NAND of Figure 1. Their behavior is specified by the following formulas of classical logic

$$\mathrm{INV}(x,y) \equiv (x \to \sim y) \wedge (\sim x \to y)$$
$$\mathrm{NAND}(x,y,z) \equiv (x \wedge y \to \sim z) \wedge (\sim x \vee \sim y \to z)$$

Indeed, the truth table of $\mathrm{INV}(x,y)$ represents the input/output behavior of the INV gate having x as input and y as output. Analogously, NAND(x,y,z) represents the NAND gate, where x and y are the inputs and z is the output. Similarly, the classical behavior of the XOR circuit can be specified by the formula

$$\mathrm{XOR}(x,y,z) \equiv ((x \wedge \sim y) \vee (\sim x \wedge y) \to z) \wedge ((x \wedge y) \vee (\sim x \wedge \sim y) \to \sim z)$$

where x and y represent the inputs and z the output.



a	b	g
0	0	0
0	1	1
1	0	1
1	1	0

x	y	z
0	0	1
0	1	1
1	0	1
1	1	0

x	y
0	1
1	0

Fig. 1. The XOR circuit and its components

Classical semantics allows us to study the input/output behavior of combinatorial circuits, but does not allow us to represent temporal information about the stabilization properties of the circuits. A more realistic description of the XOR circuit of Figure 1 should consider the instant at which the signals become stable and the delays in the propagation of signals; e.g., an "informal" characterization of the behavior of the above circuit should be as follows:

(a stable to 1 at t_1) and (b stable to 0 at t_2) **or** (a stable to 0 at t_1) and (b stable to 1 at t_2)	$\Rightarrow$	(g stable to 1 at $F(t_1,t_2)$)
(a stable to 1 at t_1) and (b stable to 1 at t_2) **or** (a stable to 0 at t_1) and (b stable to 0 at t_2)	$\Rightarrow$	(g stable to 0 at $G(t_1,t_2)$)

where F and G are functions from $\mathbf{N}^2$ to $\mathbf{N}$ and $\mathbf{N}$ represents discrete time.

To formalize these aspects, we recall the main notions introduced in [4]. A *signal* is a discrete timed boolean function $\sigma \in \mathbf{N} \to \mathbf{B}$. A *circuit* is characterized by a set of *observables* $\mathbf{S} = \{a, b, c_1, c_2, \dots\}$ (the atomic formulas of our language); for instance, to represent the XOR circuit of Figure 1, we need the set of observables $\{a, b, c, d, e, f, g\}$ representing the connections between the gates of the circuit. A *waveform* is a map $V \in \mathbf{S} \to (\mathbf{N} \to \mathbf{B})$ associating with every observable a signal. A waveform represents an observable property of a circuit C, whereas an observable *behavior* of C is described by a set of waveforms. A signal $V(a)$ is *stable* to 1 at time t iff $V(a)(k) = 1$ for all $k \geq t$; similarly, $V(a)$ is stable to 0 at t iff $V(a)(k) = 0$ for all $k \geq t$. We only consider *eventually stable* waveforms V, namely: for every $a \in \mathbf{S}$, there is t such that the signal $V(a)$ is stable at time t (to 0 or to 1). Figure 2 describes a possible eventually stable waveform for the NAND circuit. Here, the input signal $V(x)$ stabilizes to 1 at time t_4, while the input signal $V(y)$ stabilizes to 1 at time t_5; the output signal $V(z)$ stabilizes to 0 at time t_6, with a certain delay with respect to the time t_5 where both the inputs are stable to 1.

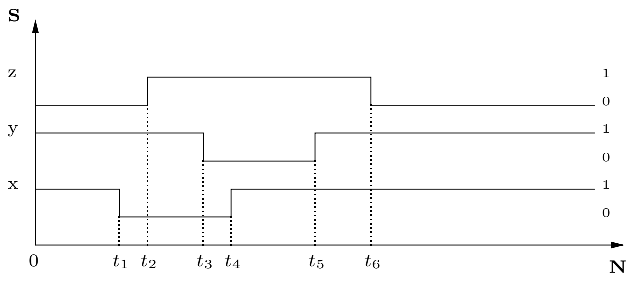


Fig. 2. A waveform for NAND

To express *stabilization properties* of waveforms and behaviors we use a propositional language $\mathcal{L}_\mathbf{S}$ based on a denumerable set of *observables* $\mathbf{S}$. Formulas of $\mathcal{L}_\mathbf{S}$ are inductively defined as follows: for every $a \in \mathbf{S}$, a is an *atomic* formula of $\mathcal{L}_\mathbf{S}$; if $A, B \in \mathcal{L}_\mathbf{S}$, then $A \wedge B$, $A \vee B$, $A \rightarrow B$ and $\sim A$ belong to $\mathcal{L}_\mathbf{S}$. Temporal information is represented by stabilization bounds which is a variant of the valuation form semantics introduced in [6]. Let A be a formula of $\mathcal{L}_\mathbf{S}$; the set of *stabilization bounds* $\lceil A \rceil$ for A is inductively defined on the structure of A as follows:

- If $A = a$ or $A = \sim a$, with $a \in \mathbf{S}$, then $\lceil A \rceil = \mathbf{N}$.
- $\lceil B \wedge C \rceil = \lceil B \rceil \times \lceil C \rceil$.
- $\lceil A_1 \vee A_2 \rceil = \lceil A_1 \rceil \oplus \lceil A_2 \rceil = \{(i, \alpha_i) \mid i \in \{1, 2\} \text{ and } \alpha_i \in \lceil A_i \rceil\}$.
- $\lceil B \rightarrow C \rceil = \{f \mid f : \lceil B \rceil \rightarrow \lceil C \rceil\}$.
- $\lceil \sim (A_1 \wedge A_2) \rceil = \lceil \sim A_1 \rceil \oplus \lceil \sim A_2 \rceil = \{(i, \alpha_i) \mid i \in \{1, 2\} \text{ and } \alpha_i \in \lceil \sim A_i \rceil\}$.
- $\lceil \sim (B \vee C) \rceil = \lceil \sim B \rceil \times \lceil \sim C \rceil$.
- $\lceil \sim (B \rightarrow C) \rceil = \lceil B \rceil \times \lceil \sim C \rceil$.
- $\lceil \sim\sim B \rceil = \lceil B \rceil$.

Intuitively, a stabilization bound $\alpha \in \lceil A \rceil$ intensionally represents a set of waveforms that validate A for the “same reasons” and with the “same delay bounds”. The main concern of timing analysis is to determine the *exact* time instant where an output signal of a circuit becomes stable, and this is performed by computing exact stabilization bounds. Let A be a formula, let $\alpha \in \lceil A \rceil$ and let V be an eventually stable waveform; α is *exact for V and A* if one of the following conditions holds:

- $A = a$ and $\alpha = \min\{t \mid V(a)$ is stable to 1 at $t\}$.
- $A = \sim a$ and $\alpha = \min\{t \mid V(a)$ is stable to 0 at $t\}$.
- $A = B \wedge C$, $\alpha = (\beta, \gamma)$, β is exact for B and V, and γ is exact for C and V.
- $A = B_1 \vee B_2$, $\alpha = (i, \beta_i)$, with $i \in \{1, 2\}$, and β_i is exact for V and B_i.
- $A = B \to C$ and, for all $\beta \in \lceil B \rceil$, if β is exact for V and B, then $\alpha(\beta)$ is exact for V and C.
- $A = \sim(B_1 \wedge B_2)$, $\alpha = (i, \beta_i)$, with $i \in \{1, 2\}$, and β_i is exact for V and $\sim B_i$.
- $A = \sim(B \vee C)$, $\alpha = (\beta, \gamma)$, β is exact for V and $\sim B$, γ is exact for V and $\sim C$.
- $A = \sim(B \to C)$, $\alpha = (\beta, \gamma)$, β is exact for V and B, γ is exact for V and $\sim C$.
- $A = \sim\sim B$ and α is exact for V and B.

To give an example, let us consider the above INV and NAND gates. A stabilization bound for INV(x, y) is a pair of functions from **N** to **N**. Let us assume that the INV gate has the following observable behavior: if the signal $V(x)$ stabilizes to 1 at t, then the signal $V(y)$ stabilizes to 0 at $t + \delta_0$; if $V(x)$ stabilizes to 0 at t, then $V(y)$ stabilizes to 1 at $t + \delta_1$. In our semantical framework, this is described by the exact stabilization bound $(f^-_{\text{INV}}, f^+_{\text{INV}})$ for V and INV(x, y) defined as follows:

$$f^-_{\text{INV}}(t) = t + \delta_0 \qquad f^+_{\text{INV}}(t) = t + \delta_1$$

Similarly, a stabilization bound for NAND(x, y, z) is a pair of functions

$$f^-_{\text{NAND}} : \mathbf{N} \times \mathbf{N} \to \mathbf{N} \qquad f^+_{\text{NAND}} : \mathbf{N} \oplus \mathbf{N} \to \mathbf{N}$$

An exact stabilization bound $(f^-_{\text{NAND}}, f^+_{\text{NAND}})$ for the waveform V of Figure 2 and NAND(x, y, z) must satisfy

$$f^-_{\text{NAND}}((t_4, t_5)) \ = \ t_6$$

which formalizes the fact that, if $V(x)$ stabilizes to 1 at t_4 and $V(y)$ stabilizes to 1 at t_5, then $V(z)$ stabilizes to 0 at t_6.

To give a picture of our extraction algorithm, let C be a circuit consisting of the gates $G_1, \dots, G_n$, where each G_i is described by a formula A_i, and let B

$$\frac{\sim A_i}{\sim(A_1 \wedge A_2)}\ \mathrm{I}{\sim}\wedge_i\ i\in\{1,2\} \qquad \frac{\sim(A \wedge B) \quad \begin{array}{c}[\sim A]\\ \vdots\ \pi_1\\ C\end{array} \quad \begin{array}{c}[\sim B]\\ \vdots\ \pi_2\\ C\end{array}}{C}\ \mathrm{E}{\sim}\wedge$$

$$\frac{A}{\sim\sim A}\ \mathrm{I}{\sim\sim} \qquad \frac{\sim\sim A}{A}\ \mathrm{E}{\sim\sim} \qquad \frac{\sim A \quad \sim B}{\sim(A \vee B)}\ \mathrm{I}{\sim}\vee \qquad \frac{\sim(A_1 \vee A_2)}{\sim A_i}\ \mathrm{E}{\sim}\vee_i\ i\in\{1,2\}$$

$$\frac{A \quad \sim B}{\sim(A \rightarrow B)}\ \mathrm{I}{\sim}{\rightarrow} \qquad \frac{\sim(A \rightarrow B)}{A}\ \mathrm{E}{\sim}{\rightarrow_1} \qquad \frac{\sim(A \rightarrow B)}{\sim B}\ \mathrm{E}{\sim}{\rightarrow_2}$$

Table 1
The negation rules of the calculus $\mathrm{N{\small D}}_{\mathbf{E}}$

be the formula describing the input/output behavior of C (see the examples above). A formal correctness verification of the circuit amounts to providing a classical proof $\pi : A_1, \ldots, A_n \vdash B$, where $A_1, \ldots, A_n$ are the open assumptions of π and B is the proved formula. Moreover, let V_i ($1 \leq i \leq n$) be the waveform corresponding to the observable behavior of G_i and let α_i be an exact stabilization bound for A_i and V_i (namely, α_i describes the behavior of G_i); finally, let V be the waveform corresponding to the observable behavior of C (V describes the temporal information about the input/output signals of the whole circuit). To determine V, it suffices to compute an exact stabilization bound for V and B. As fully described in [4], this can be accomplished by considering proofs of the constructive calculus $\mathrm{N{\small D}}_{\mathbf{E}}$ obtained by adding to the usual natural deduction calculus for intuitionistic logic the rules of Table 1, where we put between square brackets the discharged assumptions.

As a matter of fact, the main result of [4] states:

Theorem 2.1 *Let $\pi : A_1, \ldots, A_n \vdash B$ be a proof of the calculus $\mathrm{N{\small D}}_{\mathbf{E}}$. There is a recursive function $F_\pi : \lceil A_1 \rceil \times \cdots \times \lceil A_n \rceil \rightarrow \lceil B \rceil$ such that, for all $\alpha_1 \in \lceil A_1 \rceil$, $\ldots$, $\alpha_n \in \lceil A_n \rceil$ and for every eventually stable waveform V, if α_1 is an exact stabilization bound for V and A_1, $\ldots$, α_n is an exact stabilization bound for V and A_n, then $F_\pi(\alpha_1, \ldots, \alpha_n)$ is an exact stabilization bound for V and B.*

The function F_π is defined according to the structure of π. Here we only provide some cases. If $\pi : A \vdash A$ only consists of an assumption introduction, F_π is the identity function. If π is the proof

$$\frac{\begin{array}{c}A_1, \ldots, A_k\\ \vdots\ \pi_1\\ B\end{array} \qquad \begin{array}{c}A_{k+1}, \ldots, A_n\\ \vdots\ \pi_2\\ C\end{array}}{B \wedge C}\ \mathrm{I}\wedge$$

then $F_\pi(\underline{\alpha}) = (F_{\pi_1}(\alpha_1,\ldots,\alpha_k), F_{\pi_2}(\alpha_{k+1},\ldots,\alpha_n))$.

If π terminates with an implication introduction

$$\dfrac{\begin{array}{c}A_1,\ldots,A_n,[B]\\ \vdots\,{\scriptstyle \pi_1}\\ C\end{array}}{B\to C}\,{\scriptstyle \to\mathrm{I}}$$

then $F_\pi(\underline{\alpha})$ is the function $f : \lceil B\rceil \to \lceil C\rceil$ such that $f(\beta) = F_{\pi_1}(\underline{\alpha},\beta)$.

If π is the proof

$$\dfrac{\begin{array}{c}A_1,\ldots,A_k\\ \vdots\,{\scriptstyle \pi_1}\\ B\end{array}\qquad\begin{array}{c}A_{k+1},\ldots,A_n\\ \vdots\,{\scriptstyle \pi_2}\\ B\to C\end{array}}{C}\,{\scriptstyle \to\mathrm{E}}$$

then $F_\pi(\underline{\alpha}) = F_{\pi_2}(\alpha_{k+1},\ldots,\alpha_n)(F_{\pi_1}(\alpha_1,\ldots,\alpha_k))$.

3 Building Nd$_\mathbf{E}$ proofs

To apply our extraction procedure, we need a Nd$_\mathbf{E}$ proof of correctness of the circuit. However, it is well-known that natural deduction calculi are not adequate for proof search. For this reason we use a tableau based theorem prover to build up the correctness proof and then we translate it into a Nd$_\mathbf{E}$ proof. In this section we give a quick overview of both the tableau calculus and the translation rules.

Our theorem prover implements the tableau calculus Tab of [1] which has the "same deductive power" of Nd$_\mathbf{E}$. Differently from natural deduction, tableaux are goal-oriented calculi; this feature makes them suitable for automated deduction (see [2] for an account of the wide of applicability of tableau systems).

The tableau calculus Tab uses an object language with the signs $\mathbf{T}$, $\mathbf{F}$, $\mathbf{F_c}$ and $\mathbf{T_c}$ and is equivalent to Nd$_\mathbf{E}$ in the following sense:

- $\pi : A_1,\ldots,A_n \vdash B \in$ Nd$_\mathbf{E}$ iff there exists a tableau proof in Tab of $\{\mathbf{T}A_1,\ldots,\mathbf{T}A_n,\mathbf{F}B\}$.

The depth of every proof table is linearly bounded by the length of the input formulas, moreover there exists an "efficient" strategy in the application of the rules which strongly bounds the backtracking. The theorem prover described in [1] implements an $O(n^2)$-SPACE proof search procedure for proofs of Tab.

To simplify the translation from tableau proofs into natural deduction proofs, we add to the calculus $\mathrm{ND}_{\mathbf{E}}$ the following cut-rules

$$\dfrac{\begin{matrix} A_1,\dots,A_k \\ \vdots\,\pi_1 \\ B\end{matrix} \qquad \begin{matrix} A_{k+1},\dots,A_n,[B] \\ \vdots\,\pi_2 \\ C\end{matrix}}{C}\,\mathrm{Cut}_1$$

$$\dfrac{\begin{matrix} A_1,\dots,A_k \\ \vdots\,\pi_1 \\ B\end{matrix} \qquad \begin{matrix} A_{k+1},\dots,A_h \\ \vdots\,\pi_2 \\ C\end{matrix} \qquad \begin{matrix} A_{h+1},\dots,A_n,[B],[C] \\ \vdots\,\pi_2 \\ D\end{matrix}}{D}\,\mathrm{Cut}_2$$

Here we give two examples of translation. Let $\Gamma = \{H_1,\dots,H_n\}$ and let $S = \{\mathbf{T}H_1,\dots,\mathbf{T}H_n\}$.

The rule $\frac{S,\mathbf{F}C,\mathbf{T}(A\wedge B)}{S,\mathbf{F}C,\mathbf{T}A,\mathbf{T}B}\mathbf{T}\wedge$, is translated as

$$\dfrac{\dfrac{A\wedge B}{A}\,\mathrm{E}\wedge_1 \qquad \dfrac{A\wedge B}{B}\,\mathrm{E}\wedge_2 \qquad \begin{matrix}\Gamma,[A],[B]\\ \vdots\,\pi \\ C\end{matrix}}{C}\,\mathrm{Cut}_2$$

The rule $\frac{S,\mathbf{F}D,\mathbf{T}(A\wedge B\to C)}{S,\mathbf{F}D,\mathbf{T}(A\to(B\to C))}\mathbf{T}{\to}\wedge$ is translated as

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[A]_1 \quad [B]_2}{A\wedge B}\,\mathrm{I}\wedge \qquad A\wedge B\to C}{C}\,\mathrm{E}{\to}}{B\to C}\,\mathrm{I}{\to}(2)}{A\to(B\to C)}\,\mathrm{I}{\to}(1) \qquad \begin{matrix}\Gamma,[A\to(B\to C)]_3\\ \vdots\,\pi\\ D\end{matrix}}{D}\,\mathrm{Cut}_1(3)$$

In the above proofs, the number beside the rule name indicate the points where the assumptions are discharged.

The translation of the proof generated by the tableau theorem prover (implemented in C++ language) is performed by a JAVA package of the application.

4 The system ESBC

In this section we discuss the implementation issues. The system software ESBC performs all the steps to compute stabilization bounds of combinatorial circuits discussed in the previous sections. It consists of the following independent modules: (i) the tableau theorem prover, (ii) the translator from proofs of TAB into proofs of $\mathrm{ND}_{\mathbf{E}}$, (iii) the tool which computes stabilization bounds exploiting proofs produced by (ii).

As a summarizing example, let us consider the Full Adder Circuit of Figure 3. The gates in the circuit are represented by the formulas in the set:

$$\begin{aligned}\mathcal{C} = \{&\mathrm{XOR}(y,z,a), \mathrm{XOR}(a,x,s), \mathrm{AND}(y,z,b), \mathrm{AND}(x,y,c),\\ &\mathrm{AND}(x,z,d), \mathrm{OR}(b,c,e), \mathrm{OR}(e,d,r)\}\end{aligned}$$

The behavior of AND, OR and XOR gates are specified by the formulas:

$$\begin{aligned}\mathrm{AND}(x,y,z) &\equiv (x \wedge y \to z) \wedge (\sim x \vee \sim y \to \sim z)\\ \mathrm{OR}(x,y,z) &\equiv (\sim x \wedge \sim y \to \sim z) \wedge (x \vee y \to z)\\ \mathrm{XOR}(x,y,z) &\equiv ((x \wedge \sim y) \vee (\sim x \wedge y) \to z) \wedge ((x \wedge y) \vee (\sim x \wedge \sim y) \to \sim z)\end{aligned}$$

The input/output behavior of the signals of the circuit is specified by the formula:

$$\begin{aligned}S \equiv\ &(\sim x \wedge \sim y \wedge \sim z \to \sim s \wedge \sim r) \wedge (\sim x \wedge \sim y \wedge z \to s \wedge \sim r) \wedge\\ &(\sim x \wedge y \wedge \sim z \to s \wedge \sim r) \wedge (\sim x \wedge y \wedge z \to \sim s \wedge r) \wedge\\ &(x \wedge \sim y \wedge \sim z \to s \wedge \sim r) \wedge (x \wedge \sim y \wedge z \to \sim s \wedge r) \wedge\\ &(x \wedge y \wedge \sim z \to \sim s \wedge r) \wedge (x \wedge y \wedge z \to s \wedge r)\end{aligned}$$

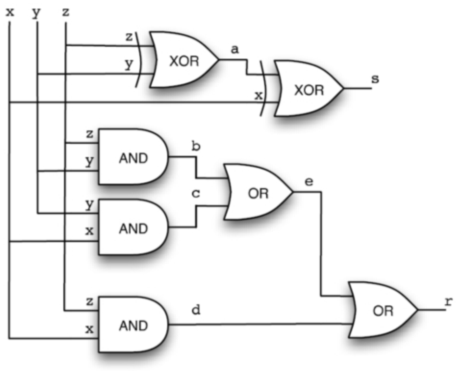


Fig. 3. The Full Adder Circuit

Firstly, we search for a proof $\pi : \mathcal{C} \vdash S$ of $\mathrm{ND}_{\mathbf{E}}$, by using modules (i) and (ii) of ESBC. Then, (iii) uses π to compute stabilization bounds of the circuit. We have to provide the stabilization bound of the elementary gates. As discussed in Section 2, a stabilization bound for $\mathrm{AND}(x,y,z)$ is a pair of functions $(f^-_{\mathrm{AND}}, f^+_{\mathrm{AND}})$, where

$$f^-_{\mathrm{AND}} : \mathbf{N} \oplus \mathbf{N} \to \mathbf{N} \qquad f^+_{\mathrm{AND}} : \mathbf{N} \times \mathbf{N} \to \mathbf{N}$$

Let us assume that the output signal of the AND gate stabilizes to 1 with delay 2 after the stabilization to 1 of both the input signals and that it stabilizes to 0 with delay 5 after the stabilization to 0 of either of the input signals. Then, we have to set

$$f^-_{\text{AND}}(i,t) = t+5 \qquad f^+_{\text{AND}}(t_1,t_2) = max\{t_1,t_2\}+2, \quad i \in \{1,2\}$$

Similarly, a stabilization bound for OR(x,y,z) is a pair of functions $(f^-_{\text{OR}}, f^+_{\text{OR}})$, where

$$f^-_{\text{OR}} : \mathbf{N} \times \mathbf{N} \to \mathbf{N} \qquad f^+_{\text{OR}} : \mathbf{N} \oplus \mathbf{N} \to \mathbf{N}$$

We assume that behavior of OR gate corresponds to the following exact stabilization bound:

$$f^-_{\text{OR}}(t_1,t_2) = max\{t_1,t_2\}+6 \qquad f^+_{\text{OR}}(i,t) = t+10, \quad i \in \{1,2\}$$

Finally, a stabilization bound for XOR(x,y,z) is a pair of functions $(f^-_{\text{XOR}}, f^+_{\text{XOR}})$, where

$$f^-_{\text{XOR}} : (\mathbf{N} \times \mathbf{N}) \oplus (\mathbf{N} \times \mathbf{N}) \to \mathbf{N} \qquad f^+_{\text{XOR}} : (\mathbf{N} \times \mathbf{N}) \oplus (\mathbf{N} \times \mathbf{N}) \to \mathbf{N}$$

We choose f^-_{XOR} and f^+_{XOR} as follows:

$$f^-_{\text{XOR}}(i,(t_1,t_2)) = max\{t_1,t_2\}+1, \quad i \in \{1,2\}$$
$$f^+_{\text{XOR}}(i,(t_1,t_2)) = max\{t_1,t_2\}+5, \quad i \in \{1,2\}$$

We compute the exact stabilization bounds of output signals assuming that input signals are stable at time 0. By applying the above exact stabilization bounds to the function F_π associated with π, ESBC produces the results given in the following table where for every input we put in evidence the truth value of the signal and for every output both the truth value and the stabilization bounds are provided. For instance, if the signals $V(x), V(y)$ and $V(z)$ stabilizes to 1 at time 0, we have that $V(s)$ stabilizes to 1 at time 6 and $V(r)$ stabilizes to 1 at time 12 (see the last row of the table).

x	y	z	s	r
false	false	false	(false,10)	(false,17)
false	false	true	(true,2)	(false,17)
false	true	false	(true,2)	(false,17)
false	true	true	(false,10)	(true,22)
true	false	false	(true,6)	(false,17)
true	false	true	(false,6)	(true,12)
true	true	false	(false,6)	(true,22)
true	true	true	(true,6)	(true,12)

References

[1] A. Avellone, C. Fiorentini, G. Fiorino, and U. Moscato. A space efficient implementation of a tableau calculus for a logic with a constructive negation. In J. Marcinkowski and A. Tarlecki, editors, *Computer Science Logic, 18th International Workshop, CSL 2004*, volume 3210 of *Lecture Notes in Computer Science*, pages 488–502, 2004.

[2] M. D'Agostino, D.M. Gabbay, R. Hähnle, and J. Posegga, editors. *Handbook of tableau methods.* Kluwer Academic Press, 1999.

[3] M.A.E. Dummett. *Elements of Intuitionism.* Claredon Press, Oxford, 1977.

[4] M. Ferrari, C. Fiorentini, and M. Ornaghi. Extracting exact time bounds from logical proofs. In A. Pettorossi, editor, *Logic Based Program Synthesis and Transformation, 11th International Workshop, LOPSTR 2001, Selected Papers*, volume 2372 of *Lecture Notes in Computer Science*, pages 245–265. Springer-Verlag, 2002.

[5] P. Miglioli, U. Moscato, and M. Ornaghi. A constructive logic approach to database theory. In *Logic Programming-Proceeding of the First Russian Conference on Logic Programming*, pages 302–321, 1990.

[6] P. Miglioli, U. Moscato, M. Ornaghi, S. Quazza, and G. Usberti. Some results on intermediate constructive logics. *Notre Dame Journal of Formal Logic*, 30(4):543–562, 1989.

[7] M. Ornaghi, M. Benini, M. Ferrari, C. Fiorentini, and A. Momigliano. A constructive modeling language for object oriented information systems. *CLASE*, 2005.

[8] A.S. Troelstra. Aspects of constructive mathematics. In J. Barwise, editor, *Handbook of Mathematical Logic.* North-Holland, 1977.