UNIVERSITE D'ORLEANS

Faculté des Sciences

LIFO

Laboratoire d'Informatique Fondamentale d'Orléans
4, rue Léonard de Vinci, BP 6759
F-45067 Orléans Cedex 2
FRANCE

Rapport de Recherche

www : http://www.univ-orleans.fr/SCIENCES/LIFO/

A conditional rewrite-based method for designing combinational logic circuits.

N. Andrianarivelo, W. Bousdira, J. Chabin, Z. Maazouzi.
Université d'Orléans, LIFO

Rapport N^o **98-04**

Abstract

We present a conditional rewrite-based method to design not trivial combinational logic circuits using only Small Scale Integrated (SSI) elements. For this, we develop a maximal unit strategy method, using built-in algebras. In the present work, we assume that natural integers and boolean algebra are built-in. The formalism of constraint is used and the search space is reduced by suppressing rules according to a new goal-oriented deletion rule criteria.

1 Introduction

Logic circuits make up the basis of all electronic devices. Thus, correct, complete and automated methods to design circuits are necessary. To the best of our knowledge, the only work dealing with logic circuit design through rewriting techniques is the one decribed in [WOLB91]. But when constraints on the gates numbers are to be considered for the circuit to design, ad hoc and unnatural techniques are to be involved in the method mentionned above. Such method does not make it efficient enough.

Here we propose both an improvement and extension of the method mentionned above. It is more general and works the same as most of software for logic circuit design. Among of them are Abel, Palasm, OrCad, ...

For most of the latter, for every solution proposed by the software, simulations are to be performed both to check the correctness of the solution and to have an idea on its real behavior. Thus it is quite time-consuming. Users may use test vectors for the correctness verification as well.

In our case, tests for correctness are no longer necessary. All the inference rules of our inference system are theoritically correct, thus - assuming that they are correctly implemented - tests for verification are not necessary.

On the other hand, formalizing circuits by rewrite rules are both *natural* and efficient since Presburger's Arithmetic and boolean algebra are built-in. Furthermore, both simplification mechanisms - included in the rewriting approach - and an original goal-oriented deletion rule will reduce considerably the search space. Endless researches are totally suppressed in our approach.

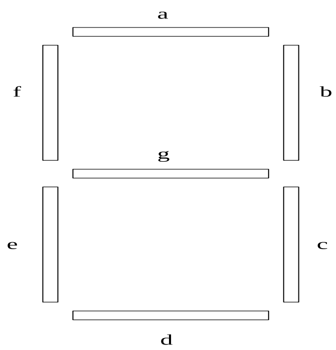
In this paper, we suppose readers are familiar with basic notions in conditional and constrained rewriting. For further details see [DJ90]. We use the same notation as in the latter. The formalism of constraint used here is also the usual one. For more details see [KKR90], and [Smo89]. All the notations used here are the same as in our previous work [ABT96]. Let us recall that a constrained conditional rule is noted $\mathcal{C} \wedge L \Rightarrow s = t$, in which $\mathcal{C}$ is a list of constraints to be solved by using algorithms in the predefined algebras, or by unification, and L is a list of equations to be solved by rewriting techniques. When the L part does not exist then we call it an equation.

We suppose also readers are familiar to all the usual notations and principles in logic circuit design. For more details, see [Wak91].

The layout of this paper is as follows: in Section 2, we present one of the well-known combinational circuits. It will be considered as the example to illustrate the whole concept developed all through this paper. In Section 3, we present our design method by formalizing the inference rules and the proof process. Section 4 is devoted to the example; we show the main steps of the proof process in order to get one solution, and then we conclude.

2 The decoder with 7-segments

This device is to display 0,1, ...,15 in hexadecimal forms. Its inputs are 4-binary numbers, and its outputs are 7 segments (lights) a,b,c,d,e,f,g represented as follows:



Digital display table with 7-segments

	Inputs				Outputs						
	D	C	B	A	a	b	c	d	e	f	g
0	ff	ff	ff	ff	tt	tt	tt	tt	tt	tt	ff
1	ff	ff	ff	tt	ff	tt	tt	ff	ff	ff	ff
2	ff	ff	tt	ff	tt	tt	ff	tt	tt	ff	tt
3	ff	ff	tt	tt	tt	tt	tt	tt	ff	ff	tt
4	ff	tt	ff	ff	ff	tt	tt	ff	ff	tt	tt
5	ff	tt	ff	tt	tt	ff	tt	tt	ff	tt	tt
6	ff	tt	tt	ff	ff	ff	tt	tt	tt	tt	tt
7	ff	tt	tt	tt	tt	tt	tt	ff	ff	ff	ff
8	tt	ff	ff	ff	tt	tt	tt	tt	tt	tt	tt
9	tt	ff	ff	tt	tt	tt	tt	ff	ff	tt	tt
A	tt	ff	tt	ff	ff	ff	ff	tt	tt	ff	tt
b	tt	ff	tt	tt	ff	ff	tt	tt	ff	ff	tt
C	tt	tt	ff	ff	ff	tt	ff	ff	ff	tt	tt
d	tt	tt	ff	tt	tt	ff	ff	tt	ff	tt	tt
E	tt	tt	tt	ff	ff	ff	ff	tt	tt	tt	tt
F	tt	tt	tt	tt	ff	ff	ff	ff	ff	ff	ff

Table 1 : the thruth table of the decoder with 7 segments.

For instance, to display 5, which is 0101 in binary form, the outputs a, f, g, c and d must be at the state *tt* (for *true*) and all the other outputs must be at the state *ff* (for *false*). The initial truth table of the decoder is represented by Table 1. The latter is from [Wak91]. Nothing is displayed for the decimal value 15.

3 The architecture of the method

It is the following:

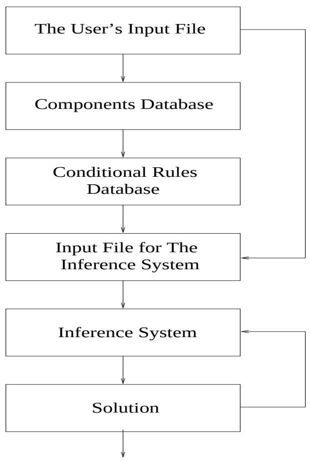


Figure 1: The architecture of our expert system

Let us precise the contents of each entity in the above schema.

1. *The User's Input File* contains:
   (a) the description of the circuit to design in the form of a truth table,
   (b) the list of electronic components by which the circuit is to be designed.

2. *The Components Database* is a list of all the available electronic components, with their basic characteristics.
3. *The Conditional Rules Database* contains all the conditional rules schematizing all the gates contained in the components database.
4. *The Input File for The Inference System* is the *real* input file for the inference system and it is deduced automatically from the three previous entities. It is formed by *constrained* and conditional rules and a negative clause to represent the circuit to design.
5. The ouput file contains the results of the design process.

The truth table given by the user may be *reduced* by usual methods of *minimization* like those of Kargnaugh or McCluskey. Now let us comment the above architecture.

From the input file of the user, we consult our electronic components database to extract from the database of conditional rules only what we need. To those conditional rules will be added a purely negative clause describing the outputs of the circuit to design, also deduced from the user's input file. Then, we infer in our inference system until we get a solution. If the given solution is not convenient for the user, we infer again in our system to get another solution.

In this section we begin by precising the contents in each entity of the architecture of Figure 1. Then we explain how our method works. After that, we give our deletion criteria to reduce the search space, and we give a result of correctness and completeness of our method.

3.1 The electronic components database

For this part, the electronic components database is the list of all the SSI basic elements. For instance, the '7427' chip of Texas Instruments is an element of it. On such kind of chips, the main information which is of interest for us are : it contains 3 NOR gates with 3 inputs. The list of basic components we deal with is the one contained in [Wak91].

3.2 The user's input file

It contains both the description of the circuit in term of truth table and the list of components to be used. For the example of the design of the decoder we may get :

- The output columns a, b, c, d, e, f, and g of Table 1,
- the list of components is : 1 of 7404, 2 of 7400, 1 of 4072, 1 of 7408, 2 of 7410, 2 of 4075, and 2 of 7411. For instance, the chip 7404 contains 6 NOT gates. All such necessary characterics - the numbers of included gates - of all those chips are in our components database.

3.3 The conditional rules database

It contains all the 15 rules coding the 15 kinds of gates. The rule AND2 denotes the AND gate with 2 inputs, AND3 denotes the AND gate with 3 inputs, then by the same way, the remaining gates are noted AND4, OR2, OR3, OR4, NOT, NAND2, NAND3, NAND4, NAND8, NOR2, NOR3, NOR4, NOR8. For instance, the following rule represents the AND gate with 2 inputs (noted AND2)

$$(AND2) \quad l_s = g(l_1, l_2) \wedge s_1 = f(1, l_1, l_2, m_1^1, m_1^2) + 1 \wedge \cdots \wedge s_{15} = f(15, l_1, l_2, m_{15}^1, m_{15}^2) \wedge$$

$$F\$(out(\overrightarrow{x}), l_1, gates(m_1^1, \ldots, m_{15}^1)) = tt \wedge F\$(out(\overrightarrow{y}), l_2, gates(m_1^2, \ldots, m_{15}^2)) = tt$$

$$\Rightarrow F\$(out(and(\overrightarrow{x}, \overrightarrow{y}), l_s, gates(s_1, \ldots, s_{15})) = tt$$

Let us make clear the meaning of the terms and their arguments in this rule.

1. $\overrightarrow{x}$, $\overrightarrow{y}$, ... denote finite ordered sequences of tt and ff.

2. *not*, *and* and *or* denote the usual logic connectors. $and(\overrightarrow{x}, \overrightarrow{y})$ denote the sequence $and(x_1, y_1)$ $and(x_2, y_2), \ldots,$ $x_i, y_i, \ldots$ are boolean variables. *and* and *or* are considered AC (Associative-Commutative).
3. The two inputs of the gate AND2 are represented by the two equations in the tail of the rule: $F\$(out(\overrightarrow{x}), l_1, gates(m_1^1, \ldots, m_{15}^1)) = tt \wedge F\$(out(\overrightarrow{y}), l_2, gates(m_1^2, \ldots, m_{15}^2)) = tt$

   Inputs and outputs are represented by terms with $F\$$ at their top. In such terms, the first subterm $out(\ldots)$ is the real input, the second one is a list to trace the history of the considered output, then the last one $gates(\ldots)$ is to manage the gates numbers to get the considered output.
4. For all term $gates(g_1, g_2, \ldots, g_{15})$, in which all g_i is a natural integer, we have: g_1 is the number of AND2 gate to get the associated output, g_2 is the same for AND3, g_3 is for AND4, g_4 is for OR2, g_5 is for OR3, g_6 is for OR4, g_7 is for NOT, g_8 is for NAND2, g_9 is for NAND3, g_{10} is for NAND4, g_{11} is for NAND8, g_{12} is for NOR2, g_{13} is for NOR3, g_{14} is for NOR4 and g_{15} is for NOR8.
5. f is a function written by ourselves to compute the numbers of gates to get the associated output to the rule. It is considered as a predefined function. Its algorithm is in appendix B.
6. g is also a function written by ourselves to update the list of all the used rules. Thus, it is also considered as a predefined function. Its algorithm is in appendix B.

Rules for AND3, AND4, OR2, OR3, OR4, NOT, NAND2, NAND3, NAND4, NAND8, NOR2, NOR3, NOR4 and NOR8 are also included in our database. They are detailed in appendix A.

Let us point out that the above notation is slightly distinct from the one used in the first step of this work presented at UNIF'97 [ABCM97].

3.4 The input file for the inference system

From the three entities above, we automatically compute the input file for our inference system. It contains two main parts: i) an initial set of conditional rules and demodulators, ii) an inequation representing the outputs of the circuit to design. Let us make each of these two sets clear.

3.5 The initial set of conditional rules

First, it contains all the rules representing the inputs of the circuit to design. For the example of the decoder, we get:

(1) Input D:

$$F\$(out(ff, ff, ff, ff, ff, ff, ff, ff, tt, tt, tt, tt, tt, tt, tt, tt), [1], gates(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)) = tt$$

(2) Input C:

$$F\$(out(ff, ff, ff, ff, tt, tt, tt, tt, ff, ff, ff, ff, tt, tt, tt, tt), [2], gates(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)) = tt$$

(3) Input B:

$$F\$(out(ff, ff, tt, tt, ff, ff, tt, tt, ff, ff, tt, tt, ff, ff, tt, tt), [3], gates(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)) = tt$$

(4) Input A:

$$F\$(out(ff, tt, ff, tt, ff, tt, ff, tt, ff, tt, ff, tt, ff, tt, ff, tt), [4], gates(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)) = tt$$

By means of both the list of components and the database of conditional rules, we extract from the latter the conditional rules we need. For the example of the decoder with 7 segments, NOT, AND2, AND3, OR3, OR4, NAND2, NAND3 are the only gates contained in the list of components given by the user, thus we only need the conditional rules associated to them. They are:

(5) The rule for NOT:

$$l_s = g(l) \wedge s_1 = f(1, l, m_1) \wedge \cdots \wedge s_7 = f(7, l, m_7) + 1 \wedge \cdots \wedge s_{15} = f(15, l, m_{15}) \wedge$$
$$F\$(out(x_1, \ldots, x_{16}), l, gates(m_1, \ldots, m_{15})) = \ tt \Rightarrow F\$(out(not(x_1), \ldots, not(x_{16})), l_s, gates(s_1, \ldots, s_{15})) = \ tt$$

(6) The rule for AND2:

$$l_s = g(l_1, l_2) \wedge s_1 = f(1, l_1, l_2, m_1^1, m_1^2) + 1 \wedge \ldots \wedge s_{15} = f(15, l_1, l_2, m_{15}^1, m_{15}^2) \wedge$$
$$F\$(out(x_1, \ldots, x_{16}), l_1, gates(m_1^1, \ldots, m_{15}^1) = tt \wedge F\$(out(y_1, \ldots, y_{16}), l_2, gates(m_1^2, \ldots, m_{15}^2) = tt$$
$$\Rightarrow F\$(out(and(x_1, y_1), \ldots, and(x_{16}, y_{16})), l_s, gates(s_1, \ldots, s_{15})) = tt$$

(7) The rule for AND3:

$$l_s = g(l_1, l_2, l_3) \wedge s_1 = f(1, l_1, l_2, l_3, m_1^1, m_1^2, m_1^3) \wedge s_2 = f(2, l_1, l_2, l_3, m_2^1, m_2^2, m_2^3) + 1 \wedge \ldots \wedge$$
$$s_{15} = f(15, l_1, l_2, l_3, m_{15}^1, m_{15}^2, m_{15}^3) \wedge$$
$$F\$(out(x_1, \ldots, x_{16}), l_1, gates(m_1^1, \ldots, m_{15}^1) = \ tt \wedge F\$(out(y_1, \ldots, y_{16}), l_2, gates(m_1^2, \ldots, m_{15}^2) = tt \wedge$$
$$F\$(out(z_1, \ldots, z_{16}), l_3, gates(m_1^3, \ldots, m_{15}^3) = \ tt$$
$$\Rightarrow F\$(out(and(x_1, y_1, z_1), \ldots, and(x_{16}, y_{16}, z_{16})), l_s, gates(s_1, \ldots, s_{15})) = \ tt$$

(8) The rule for OR3:

$$l_s = g(l_1, l_2, l_3) \wedge s_1 = f(1, l_1, l_2, l_3, m_1^1, m_1^2, m_1^3) \wedge \ldots \wedge s_5 = f(2, l_1, l_5, l_3, m_5^1, m_5^2, m_5^3) + 1 \wedge \ldots \wedge$$
$$s_{15} = f(15, l_1, l_2, l_3, m_{15}^1, m_{15}^2, m_{15}^3) \wedge$$
$$F\$(out(x_1, \ldots, x_{16}), l_1, gates(m_1^1, \ldots, m_{15}^1) = tt \wedge F\$(out(y_1, \ldots, y_{16}), l_2, gates(m_1^2, \ldots, m_{15}^2) = \ tt \wedge$$
$$F\$(out(z_1, \ldots, z_{16}), l_3, gates(m_1^3, \ldots, m_{15}^3) = \ tt \Rightarrow$$
$$F\$(out(and(x_1, y_1, z_1), \ldots, and(x_{16}, y_{16}, z_{16})), l_s, gates(s_1, \ldots, s_{15})) = \ tt$$

(9) The rule for OR4:

$$l_s = g(l_1, l_2, l_3, l_4) \wedge s_1 = f(1, l_1, l_2, l_3, l_4, m_1^1, m_1^2, m_1^3, m_1^4) \wedge \ldots \wedge s_6 = f(6, l_1, l_2, l_3, l_4, m_6^1, m_6^2, m_6^3, m_6^4) + 1 \wedge \ldots \wedge$$
$$s_{15} = f(15, l_1, l_2, l_3, l_4, m_{15}^1, m_{15}^2, m_{15}^3, m_{15}^4) \wedge F\$(out(x_1, \ldots, x_{16}), l_1, gates(m_1^1, \ldots, m_{15}^1) = \ tt \wedge$$
$$F\$(out(y_1, \ldots, y_{16}), l_2, gates(m_1^2, \ldots, m_{15}^2) = \ tt \wedge F\$(out(z_1, \ldots, z_{16}), l_3, gates(m_1^3, \ldots, m_{15}^3) = \ tt \wedge$$
$$F\$(out(t_1, \ldots, t_{16}), l_4, gates(m_1^4, \ldots, m_{15}^4) = \ tt \Rightarrow$$
$$F\$(out(or(x_1, y_1, z_1, t_1), \ldots, or(x_{16}, y_{16}, z_{16}, t_{16})), l_s, gates(s_1, \ldots, s_{15})) = \ tt$$

(10) this for NAND2:

$$l_s = g(l_1, l_2) \wedge s_1 = f(1, l_1, l_2, m_1^1, m_1^2) \wedge \ldots \wedge s_8 = f(8, l_1, l_2, m_8^1, m_8^2) + 1 \wedge \ldots \wedge s_{15} = f(15, l_1, l_2, m_{15}^1, m_{15}^2) \wedge$$
$$F\$(out(x_1, \ldots, x_{16}), l_1, gates(m_1^1, \ldots, m_{15}^1) = \ tt \wedge F\$(out(y_1, \ldots, y_{16}), l_2, gates(m_1^2, \ldots, m_{15}^2) = \ tt$$
$$\Rightarrow F\$(out(nand(x_1, y_1), \ldots, nand(x_{16}, y_{16})), l_s, gates(s_1, \ldots, s_{15})) = tt$$

(11) This for NAND3 :

$$l_s = g(l_1,l_2,l_3) \wedge s_1 = f(1,l_1,l_2,l_3,m_1^1,m_1^2,m_1^3) \wedge \ldots \wedge s_9 = f(9,l_1,l_2,l_3,m_9^1,m_9^2,m_9^3)+1 \wedge \ldots \wedge s_{15} = f(15,l_1,l_2,l_3,m_{15}^1,m_{15}^2,m_{15}^3) \wedge F\$(out(x_1,\ldots,x_{16}),l_1,gates(m_1^1,\ldots,m_{15}^1) = tt \wedge F\$(out(y_1,\ldots,y_{16}),l_2,gates(m_1^2,\ldots,m_{15}^2) = tt \wedge F\$(out(z_1,\ldots,z_{16}),l_3,gates(m_1^3,\ldots,m_{15}^3) = tt \Rightarrow F\$(out(nand(x_1,y_1,z_1),\ldots,nand(x_{16},y_{16},z_{16})),l_s,gates(s_1,\ldots,s_{15})) = tt$$

Then, we have all the usual demodulators on 'and' and 'or' as well. They are :

(12) $not(ff) \rightarrow tt$ (13) $not(tt) \rightarrow ff$ (14) $and(ff,x) \rightarrow ff$ (15) $and(tt,x) \rightarrow x$
(16) $or(ff,x) \rightarrow x$ (17) $or(tt,x) \rightarrow tt$ (18) $nand(x,y) \longrightarrow not(and(x,y))$

Both *and* and *or* are declared AC (Associative-Commutative) symbols. Let us note that the rewrite rule $nor(x,y) \longrightarrow not(or(x,y))$ - although contained in our database of rules - is not included in the input file for the inference system for the present case.

3.6 The outputs of the circuit to design

The outputs of the circuit to design are represented by a negative clause with only one constrained equation at its tail. We call such clause an inequation.

Thanks to the list of components given by the user, we automatically compute the constraints on the number of gates of this inequation. As regards of the decoder, as the total number of AND2 gates contained in all the chips of the given list of Section 3.5 is 4, our first constraint (the one associated to the AND2 gate) is $f(1,l_1,\ldots,l_7,m_1^1,\ldots,m_1^7) \leq 4$. We do the same for all the other gates. Naturally for all non existing gates in the given input list of components, their total number is equal to 0.

The inequation derives from the output columns of the truth table given by the user. For this, a new symbol '$A\$$' is introduced so that all the outputs of the circuit to design is represented by a single term. Such outputs are the subterms of the $A\$(\ldots)$ term. The '$out(\ldots)$' first subterm of it is the first output column of the truth table and so on. For the example of the decoder with 7 segments, we get the following :

$$\begin{array}{l}
f(1,l_1,\ldots,l_7,m_1^1,\ldots,m_1^7) \leq 4 \wedge f(2,l_1,\ldots,l_7,m_2^1,\ldots,m_2^7) \leq 6 \wedge f(3,l_1,\ldots,l_7,m_3^1,\ldots,m_3^7) \leq 0 \wedge \\
f(4,l_1,\ldots,l_7,m_4^1,\ldots,m_4^7) \leq 0 \wedge f(5,l_1,\ldots,l_7,m_5^1,\ldots,m_5^7) \leq 6 \wedge f(6,l_1,\ldots,l_7,m_6^1,\ldots,m_6^7) \leq 2 \wedge \\
f(7,l_1,\ldots,l_7,m_7^1,\ldots,m_7^7) \leq 6 \wedge f(8,l_1,\ldots,l_7,m_8^1,\ldots,m_8^7) \leq 8 \wedge f(9,l_1,\ldots,l_7,m_9^1,\ldots,m_9^7) \leq 6 \wedge \\
f(10,l_1,\ldots,l_7,m_{10}^1,\ldots,m_{10}^7) \leq 0 \wedge \ldots \wedge f(15,l_1,\ldots,l_7,m_{15}^1,\ldots,m_{15}^7) \leq 0 \wedge \\
A\$(F\$(out(tt,ff,tt,tt,ff,tt,ff,tt,tt,tt,ff,ff,ff,tt,ff,ff),l_1,gates(m_1^1,\ldots,m_{15}^1)), \\
\quad F\$(out(tt,tt,tt,tt,tt,ff,ff,tt,tt,tt,ff,ff,tt,ff,ff,ff),l_2,gates(m_1^2,\ldots,m_{15}^2)), \\
\quad F\$(out(tt,tt,ff,tt,tt,tt,tt,tt,tt,tt,ff,tt,ff,ff,ff,ff),l_3,gates(m_1^3,\ldots,m_{15}^3)), \\
\quad F\$(out(tt,ff,tt,tt,ff,tt,tt,ff,tt,ff,tt,tt,ff,tt,tt,ff),l_4,gates(m_1^4,\ldots,m_{15}^4)), \\
\quad F\$(out(tt,ff,tt,ff,ff,ff,tt,ff,tt,ff,tt,ff,ff,ff,tt,ff),l_5,gates(m_1^5,\ldots,m_{15}^5)), \\
\quad F\$(out(tt,ff,ff,ff,tt,tt,tt,ff,tt,tt,ff,ff,tt,tt,tt,ff),l_6,gates(m_1^6,\ldots,m_{15}^6)), \\
\quad F\$(out(ff,ff,tt,tt,tt,tt,tt,ff,tt,tt,tt,tt,tt,tt,tt,ff),l_7,gates(m_1^7,\ldots,m_{15}^7))) = tt \Rightarrow
\end{array}$$

and finally, we have :

$$A\$(tt,tt,tt,tt,tt,tt,tt,tt,tt,tt,tt,tt,tt,tt,tt,tt) \rightarrow tt$$

for the simplification to the empty clause.

3.7 The inference system

All the inference rules of the maximal-unit-strategy described in [ABT96] are included in our system. The main inference rules we use are *deduction of new rules by Conditional-narrowing, and simplifying conditional rules and inequations.* Let us recall the first one. Let us start by precising that terms the form of $F\$(\ldots)$ or $A\$(\ldots)$ are called terms of the high hierarchy.

Definition 1 Conditional-narrowing
If (1) $\mathcal{C} \wedge s[u] = t \wedge L \Rightarrow A = B$ *is a conditional rule, u a non-variable subterm of s in the high hierarchy, and* (2) $\mathcal{C}' \Rightarrow\ v = w$ *an equation, then* $\mathcal{C} \wedge \mathcal{C}' \wedge \mathcal{C}'' \wedge u = v \wedge s[w] = t \wedge L \Rightarrow A = B$ *is a conditional-narrowing consequence from (1) and (2).* $\mathcal{C}''$ *is* $s = t \not< a = b\ \forall a = b \in L$ *and* $\mathcal{C} \wedge \mathcal{C}' \wedge \mathcal{C}'' \wedge u = v$ *is satisfiable.*

Note that there is no unification in the deduction process in this case ; only matching operation is performed.

Let us point out that this inference rule works between an inequation (a pure negative *conditional rule*) and an equation as well. Thus, in our approach, we may have several inequations to deal with. But for all $i \in \{1, \ldots, 15\}$, for all the inequations we have the same $\mathcal{C}_i$. A new goal-oriented deletion criteria is added to our inference system to improve its efficiency. Let us present it.

3.8 The deletion criteria

The approach described below is mainly inspired from the work of Anantharaman S. and Andrianarivelo N. in [AA90]. We start by making clear the new notion of *measure*, then the new deletion criteria will be stated.

Definition 2
1. *Let be* $\mathbf{G}$ *the set of all ground terms with top operator '*gates*'.* $\forall i \in \{1, \ldots, 15\}$ *let us define* $meas_i$ *(for measure i),* $meas_i : \mathbf{G} \longrightarrow \mathbf{N}$ *such that* $\forall t \in \mathbf{G}$, $meas_i(t) = t|_i$, *the subterm of t at the position i.*
2. *Let us extend this notion on ground terms whose top operator is* $F\$$.
   $meas_i(F\$(out(\overrightarrow{x}), l, gates(\overrightarrow{m}))) = meas_i(gates(\overrightarrow{m}))\ \ \forall i \in \{1, \ldots, 15\}$. *Naturally,* $\overrightarrow{m}$ *denotes a finite and ordered sequence of natural integers.*
3. *Let us extend this notion on ground equations of the form* $s = tt$ *where the top operator of s is* $F\$$. *We say :* $\forall i \in \{1, \ldots, 15\} : meas_i(s = tt) = meas_i(s)$.
4. *Let us extend* $meas_i$ *on ground inequation. For this, let us consider an inequation the form of:*

   $$\begin{array}{l}[0](f(1, l_1, \ldots, l_r, m_1^1, \ldots, m_1^r) \leq \mathcal{C}_1 \wedge \cdots \wedge f(15, l_1, \ldots, l_r, m_{15}^1, \ldots, m_{15}^r) \leq \mathcal{C}_{15} \wedge \\ A\$(F\$(out(\overrightarrow{s_1}), l_1, gates(m_1^1, \ldots, m_{15}^1)), F\$(out(\overrightarrow{s_2}), l_2, gates(m_1^2, \ldots, m_{15}^2)), \cdots, \\ F\$(out(\overrightarrow{s_r}), l_r, gates(m_1^r, \ldots, m_{15}^r))) =\ tt \Rightarrow\ )\end{array}$$

   We define $meas_i([0]) = \mathcal{C}_i,\ \forall i \in \{1, \ldots, 15\}$
5. *Let be NE a set of inequations. First, let us note that in this approach, during a process design all the computed inequations have the same value of* $\mathcal{C}_1$. *They also have the same value of* $\mathcal{C}_2$. *This is the case of* $\mathcal{C}_3, \mathcal{C}_4, \ldots, \mathcal{C}_{15}$ *as well. We define* $meas_i(NE) = \mathcal{C}_i\ \ \forall i \in \{1, \ldots, 15\}$

Definition 3 *Let be* $\mathcal{C}_i, i \in \{1, \ldots, 15\}$ *all the constraints in NE. An equation* $s = tt$*, where the top operator of s is $F\$$ is useful for NE, we say NE-useful, iff* $\forall i \in \{1, \ldots, 15\} : meas_i(s) \leq \mathcal{C}_i, \forall i \in \{1, 2, \ldots, 15\}$.

Theorem 1 *Let be NE a set of inequations representing a circuit to design. All not NE-useful equation can be deleted without losing the proof.*

Proof: Like in [ABT96], at every step of the design-process a *minimal* ground proof-tree is associated to each circuit-solution. For every gate, its total number in such proof-trees is smaller than the associated natural number $\mathcal{C}_i$ in the inequations. Since all the conditional rules *increase* the number of gates, all output - ground equation with $F\$$ as top operator - such that to get it, there exists one gate such that its total number exceeds the associated $\mathcal{C}_i$ in the inequations, cannot be *used* in any proof-trees leading to solutions.

Thus, by using the usual way to note inference rule, if **H** denotes a set of constrained and conditional rules, the following inference rule, called *delete*, will be added to our inference system:

delete:

$$\frac{\mathbf{H} \cup \{\mathcal{C} \Rightarrow s = tt\}}{\mathbf{H}}$$

If $\exists i \in \{1, \ldots, 15\}$ such that $meas_i(\mathcal{C} \Rightarrow s = tt) > meas_i(NE)$

Let us recall that NE is the set of all the inequations in **H**.

3.9 Correctness and completeness

Let us begin by making clear what *complete* means here. This is: if solutions exist we get all of them within a finite period of time, and if no solutions exist, within a finite period of time, we stop by pointing out to the users that there are no solutions. By assuming the usual fairness hypothesis, we get the following.

Theorem 2 *The design method decribed above is correct and complete.*

Proof: The correctness of this method is trivial. As regards completeness, it results straightly from that of the maximal-unit-strategy in [AA91] or [Der90] or [ABT96].

4 Example

Let us deal with the example of the decoder with 7 segments. We only consider output a. All the other outputs are processed similarly. We start with all the rules contained in the example of the section 'The input file for the inference system'. The main steps of the design process are the following.

By conditional-narrowing between 4 and 5, then simplifying by 12 and 13, we get:

(22) $F\$(out(tt, ff, tt, ff, tt, ff, tt, ff, tt, ff, tt, ff, tt, ff, tt, ff), [22, [4]], gates(0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)) = tt$

By conditional-narrowing between 3 and 5, then simplifying by 12 and 13 we get:

(23) $F\$(out(tt, tt, ff, ff, tt, tt, ff, ff, tt, tt, ff, ff, tt, tt, ff, ff), [23, [3]], gates(0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)) = tt$

By conditional-narrowing between 23 and 5, and simplifying by 12 and 13 we may get:

$F\$(out(ff, ff, tt, tt, ff, ff, tt, tt, ff, ff, tt, tt, ff, ff, tt, tt), g([23, [3]]), gates(0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0)) = tt$

By conditional-narrowing 5 in this latter, we may get

$F\$(out(tt, tt, ff, ff, tt, tt, ff, ff, tt, tt, ff, ff, tt, tt, ff, ff), g(g([23, [3]])), gates(0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0)) = tt$

Processing the same way, we get similar rules with 4, then 5, then 6 as the seventh arguments in the 'gates(...)' term. Thanks to the deletion rule, rules with 7, 8, ... as the seventh arguments will not be derived!

By conditional-narrowing between 22 and 10

(24) $l_s = g([22,[4]], l_2) \wedge s_1 = f(1,[22,[4]], l_2, 0, m_1^2) \wedge \ldots \wedge s_8 = f(8,[22,[4]], l_2, 0, m_8^2) + 1 \wedge \ldots \wedge$
$s_{15} = f(15,[22,[4]], l_2, 0, m_{15}^2) \wedge F\$(out(y_1, \ldots, y_{16}), l_2, gates(m_1^2, \ldots, m_{15}^2)) = tt \Rightarrow$
$F\$(out(nand(tt, y_1), nand(ff, y_2), \ldots, nand(ff, y_{16}), l_s, gates(s_1, \ldots, s_{15}) = tt$

By conditional-narrowing 1 into 10

(25) $l_s = (l_1,[1]) s_1 = f(1, l_1, [1], m_1^2, 0) \wedge \ldots \wedge s_8 = f(8, l_1, [1], m_1^2, 0) + 1 \wedge \ldots \wedge s_{15} = f(15, l_1, [1], m_1^2, 0) \wedge$
$F\$(out(x_1, \ldots, x_{16}), l_1, gates(m_1^1, \ldots, m_{15}^1)) = tt \Rightarrow$
$F\$(out(nand(ff, x_1), nand(ff, x_2), \ldots, nand(tt, x_{16}), l_s, gates(s_1, \ldots, s_{15})) = tt$

By conditional narrowing 22 into 9

(26) $l_s = g([22,[4]], l_2, l_3, l_4) \wedge s_1 = f(1,[22,[4]], l_2, l_3, l_4, 0, m_1^2, m_1^3, m_1^4) \wedge \ldots \wedge s_6 = f(6,[22,[4]], l_2, l_3, l_4, 1, m_6^2, m_6^3, m_6^4) +$
$1 \wedge \ldots \wedge s_{15} = f(15,[22,[4]], l_2, l_3, l_4, 0, m_{15}^2, m_{15}^3, m_{15}^4) \wedge F\$(out(y_1, \ldots, y_{16}), l_2, gates(m_1^2, \ldots, m_{15}^2) =$
$tt \wedge F\$(out(z_1, \ldots, z_{16}), l_3, gates(m_1^3, \ldots, m_{15}^3) = tt \wedge F\$(out(t_1, \ldots, t_{16}), l_4, gates(m_1^4, \ldots, m_{15}^4) = tt \Rightarrow$
$F\$(out(or(tt, y_1, z_1, t_1), \ldots, or(ff, y_{16}, z_{16}, t_{16})), l_s, gates(s_1, \ldots, s_{15})) = tt$

By conditional-narrowing 3 into 25, and simplifying by 18

(27) $F\$(out(tt, tt, tt, tt, tt, tt, tt, tt, tt, tt, ff, ff, tt, tt, ff, ff), [27,[3],[1]], gates(0,0,0,0,0,0,0,1,0,0,0,0,0,0,0)) = tt$

By conditional-narrowing 2 into 26

(28) $l_s = g([22,[4]],[2], l_3, l_4) \wedge s_1 = f(1,[22,[4]],[2], l_3, l_4, 0, 0, m_1^3, m_1^4) \wedge \ldots \wedge s_6 = f(6,[22,[4]],[2], l_3, l_4, 1, 0, m_6^3, m_6^4) +$
$1 \wedge \ldots \wedge s_{15} = f(15,[22,[4]],[2], l_3, l_4, 0, m_{15}^2, m_{15}^3, m_{15}^4) \wedge F\$(out(z_1, \ldots, z_{16}), l_3, gates(m_1^3, \ldots, m_{15}^3) = tt \wedge$
$F\$(out(t_1, \ldots, t_{16}), l_4, gates(m_1^4, \ldots, m_{15}^4) = tt \Rightarrow$
$F\$(out(or(tt, ff, z_1, t_1), \ldots, or(ff, tt, z_{16}, t_{16})), l_s, gates(s_1, \ldots, s_{15})) = tt$

By condtional-narrowing 2 into 24 and simplifying by 18

(29) $F\$(out(tt, tt, tt, tt, ff, tt, ff, tt, tt, tt, tt, tt, ff, tt, ff, tt), [29,[22,[4]],[2]], gates(0,0,0,0,0,0,1,1,0,0,0,0,0,0,0)) = tt$

By conditional-narrowing 29 into 7

(30) $l_s = g([29,[22,[4]],[2]], l_2, l_3) \wedge s_1 = f(1,[29,[22,[4]],[2]], l_2, l_3, 0, m_1^2, m_1^3) \wedge$
$s_2 = f(2,[29,[22,[4]],[2]], l_2, l_3, 0, m_2^2, m_2^3) + 1 \wedge \ldots \wedge s_{15} = f(15,[29,[22,[4]],[2]], l_2, l_3, 0, m_{15}^2, m_{15}^3) \wedge$
$F\$(out(y_1, \ldots, y_{16}), l_2, gates(m_1^2, \ldots, m_{15}^2) = tt \wedge F\$(out(z_1, \ldots, z_{16}), l_3, gates(m_1^3, \ldots, m_{15}^3) = tt \Rightarrow$
$F\$(out(and(tt, y_1, z_1), \ldots, and(tt, y_{16}, z_{16})), l_s, gates(s_1, \ldots, s_{15})) = tt$

By conditional-narrowing 3 into 28

(31) $l_s = g([22,[4]],[2],[3], l_4) \wedge s_1 = f(1,[22,[4]],[2],[3], l_4, 0, 0, 0, m_1^4) \wedge \ldots \wedge$
$s_6 = f(6,[22,[4]],[2],[3], l_4, 1, 0, 0, m_6^4) + 1 \wedge \ldots \wedge s_{15} = f(15,[22,[4]],[2],[3], l_4, 0, 0, 0, m_{15}^4) \wedge$
$F\$(out(t_1, \ldots, t_{16}), l_4, gates(m_1^4, \ldots, m_{15}^4) = tt \Rightarrow$
$F\$(out(or(tt, ff, ff, t_1), \ldots, or(ff, tt, tt, t_{16})), l_s, gates(s_1, \ldots, s_{15})) = tt$

By conditional-narrowing 27 into 30

(32) $l_s = g([29,[22,[4]],[2]],[27,[3],[1]],l_3) \wedge s_1 = f(1,[29,[22,[4]],[2]],[27,[3],[1]],l_3,0,0,m_1^3)\wedge$
$s_2 = f(2,[29,[22,[4]],[2]],[27,[3],[1]],l_3,0,0,m_2^3){+}1\wedge\ldots\wedge s_{15} = f(15,[29,[22,[4]],[2]],[27,[3],[1]],l_3,0,0,m_{15}^3)\wedge$
$F\$(out(z_1,\ldots,z_{16}),l_3,gates(m_1^3,\ldots,m_{15}^3) = tt \Rightarrow$
$F\$(out(and(tt,tt,z_1),\ldots,and(tt,ff,z_{16})),l_s,gates(s_1,\ldots,s_{15})) = tt$

By conditional-narrowing 1 into 31 and by simplifying with 16, 17

(33) $F\$(out(tt,ff,tt,tt,tt,tt,tt,tt,tt,tt,tt,tt,tt,tt,tt,tt),[33,[22,[4]],[2],[3],[1]],$
$gates(0,0,0,0,0,1,1,0,0,0,0,0,0,0,0)) = tt$

By conditional-narrowing 33 into 32

(34) $F\$(out(tt,ff,tt,tt,ff,tt,ff,tt,tt,tt,ff,ff,ff,tt,ff,ff),[34,[29,[2]],[27,[3],[1]],[33,[22,[4]],[2],[3],[1]]],$
$gates(0,1,0,0,0,1,2,2,0,0,0,0,0,0,0)) = tt$

Simplifying 0 by 34, we get *the first subterm* with F\$ of the inequation, which is to capture the output a of the circuit

(35)

$$
\begin{array}{c}
f(1,[34,[29,[2]],[27,[3],[1]],[33,[22,[4]],[2],[3],[1]]],l_2,\ldots,l_7,0,m_1^2,\ldots,m_1^7) \le 4\wedge \\
f(2,[34,[29,[2]],[27,[3],[1]],[33,[22,[4]],[2],[3],[1]]],l_2,\ldots,l_7,1,m_2^2,\ldots,m_2^7) \le 6\wedge \\
f(3,[34,[29,[2]],[27,[3],[1]],[33,[22,[4]],[2],[3],[1]]],l_2,\ldots,l_7,0,m_3^2,\ldots,m_3^7) \le 0\wedge \\
f(4,[34,[29,[2]],[27,[3],[1]],[33,[22,[4]],[2],[3],[1]]],l_2,\ldots,l_7,0,m_4^2,\ldots,m_4^7) \le 0\wedge \\
f(5,[34,[29,[2]],[27,[3],[1]],[33,[22,[4]],[2],[3],[1]]],l_2,\ldots,l_7,0,m_5^2,\ldots,m_5^7) \le 6\wedge \\
f(6,[34,[29,[2]],[27,[3],[1]],[33,[22,[4]],[2],[3],[1]]],l_2,\ldots,l_7,1,m_6^2,\ldots,m_6^7) \le 2\wedge \\
f(7,[34,[29,[2]],[27,[3],[1]],[33,[22,[4]],[2],[3],[1]]],l_2,\ldots,l_7,2,m_7^2,\ldots,m_7^7) \le 6\wedge \\
f(8,[34,[29,[2]],[27,[3],[1]],[33,[22,[4]],[2],[3],[1]]],l_2,\ldots,l_7,2,m_8^2,\ldots,m_8^7) \le 8\wedge \\
f(9,[34,[29,[2]],[27,[3],[1]],[33,[22,[4]],[2],[3],[1]]],l_2,\ldots,l_7,0,m_9^2,\ldots,m_9^7) \le 6\wedge \\
f(10,[34,[29,[2]],[27,[3],[1]],[33,[22,[4]],[2],[3],[1]]],l_2,\ldots,l_7,0,m_{10}^2,\ldots,m_{10}^7) \le 0\wedge \\
f(11,[34,[29,[2]],[27,[3],[1]],[33,[22,[4]],[2],[3],[1]]],l_2,\ldots,l_7,0,m_{11}^2,\ldots,m_{11}^7) \le 0\wedge \\
f(12,[34,[29,[2]],[27,[3],[1]],[33,[22,[4]],[2],[3],[1]]],l_2,\ldots,l_7,0,m_{12}^2,\ldots,m_{12}^7) \le 0\wedge \\
f(13,[34,[29,[2]],[27,[3],[1]],[33,[22,[4]],[2],[3],[1]]],l_2,\ldots,l_7,0,m_{13}^2,\ldots,m_{13}^7) \le 0\wedge \\
f(14,[34,[29,[2]],[27,[3],[1]],[33,[22,[4]],[2],[3],[1]]],l_2,\ldots,l_7,0,,m_{14}^2,\ldots,m_{14}^7) \le 0\wedge \\
f(15,[34,[29,[2]],[27,[3],[1]],[33,[22,[4]],[2],[3],[1]]],l_2,\ldots,l_7,0,,m_{15}^2,\ldots,m_{15}^7) \le 0\wedge
\end{array}
$$

$$
\begin{array}{l}
A\$(tt, \\
\quad F\$(out(tt,tt,tt,tt,tt,ff,ff,tt,tt,tt,ff,ff,tt,ff,ff,ff),l_2,gates(m_1^2,\ldots,m_{15}^2)), \\
\quad F\$(out(tt,tt,ff,tt,tt,tt,tt,tt,tt,tt,ff,tt,ff,ff,ff,ff),l_3,gates(m_1^3,\ldots,m_{15}^3)), \\
\quad F\$(out(tt,ff,tt,tt,ff,tt,tt,ff,tt,ff,tt,tt,ff,tt,tt,ff),l_4,gates(m_1^4,\ldots,m_{15}^4)), \\
\quad F\$(out(tt,ff,tt,ff,ff,ff,tt,ff,tt,ff,tt,ff,ff,ff,tt,ff),l_5,gates(m_1^5,\ldots,m_{15}^5)), \\
\quad F\$(out(tt,ff,ff,ff,tt,tt,tt,ff,tt,tt,ff,ff,tt,tt,tt,ff),l_6,gates(m_1^6,\ldots,m_{15}^6)), \\
\quad F\$(out(ff,ff,tt,tt,tt,tt,tt,ff,tt,tt,tt,tt,tt,tt,tt,ff),l_7,gates(m_1^7,\ldots,m_{15}^7))) = tt \Rightarrow
\end{array}
$$

Thanks to a similar process, we get all the other subterms with F\$, and as all the final constraint obtained is satisfiable, we get the refutation simplifying with the rule $A\$(tt,\ldots,tt) \rightarrow tt$. Like in [WOLB91], the circuit to design is given by both the history of the refutation and the list associated to the final output.

5 Conclusion

We have presented a method correct and complete to design small sizes circuits by using basic electronic components. Compared with McCune's team work, our method doest not require any ad hoc and subtle techniques. It is also more general. A new goal-oriented deletion criteria is included and so, search spaces are reduced tremendously.

Thanks to the correctness of our method, simulations are no longer *necessary* to check if the proposed circuit is correct or not. Tests performed through vectors are no longer necessary to ensure the correctness of the proposed circuit !

Users need not master any notion of rewriting in our case. They only have to give the truth table of the circuit to design and the list of electronic components to be used. All the remaining work, especially the construction of the actual input file for the inference system, is carried out fully automatically by our expert system. Naturally, the truth table may be only partially defined.

This is a constructive method, thus theorically, we can generate all the solutions if the users want this. Our method is loop-proof. If solutions exist, theoretically, by the fairness hypothesis, we get them within a finite period of time, and if there is not any solution, within a finite period of time we terminate our design by telling the users that no solution using their list of components exists. In this case, they can change their initial list.

For each solution, we are in a position to provide important informations to users to help them to make a decision. Among them is the occupation rate of each chip. Thanks to this, electronicians may have some idea on the actual behavior of the proposed circuit.

Presently, electronicians use basic electronic chips less and less, because it is space-consuming and thus this makes the signal propagation slower. Those chips are being replaced by Field Programmable Logic Array (FPLA). We have also developed a method for designing FPLA, and one can find our results in[ABCM98a, ABCM98b].

Extending our methods to deal with sequential circuits are next steps of our works as well.

References

[AA90] S. Anantharaman and N. Andrianarivelo. Heuristical Criteria in Refutational Theorem Proving. In *Proc. DISCO'90*, volume 541, pages 184–193, Pisa, Italia, 1990.

[AA91] N. Anantharaman and N. Andrianarivelo. A semi-decision procedure in Horn Theories. Technical report, Laboratoire d'Informatique Fondamentale d'Orléans–Laboratoire de Recherche en Informatique d'Orsay, 1991.

[ABCM97] N. Andrianarivelo, W. Bousdira, J. Chabin, and Z. Maazouzi. A conditional Rewrite based Method for designing Circuits. Orléans, France, 1997. 11th International Workshop on Unification.

[ABCM98a] N. Andrianarivelo, W. Bousdira, J. Chabin, and Z. Maazouzi. Designing FPLA Combinational Circuits by Conditional Rewriting. In H. Prade, editor, *Wiley publisher*, pages 373–377, Brighton, UK, 1998. 13th European Conference on Artificial Intelligence.

[ABCM98b] N. Andrianarivelo, W. Bousdira, J. Chabin, and Z. Maazouzi. Designing FPLA combinational circuits by conditional rewriting. Technical Report 98-1, Laboratoire d'Informatique Fondamentale d'Orléans, 1998.

[ABT96] N. Andrianarivelo, W. Bousdira, and J-M. Talbot. On theorem-proving in Horn theories with built-in algebras. In J. Calmet, J.A. Campbell, and J. Pfalzgraf, editors, *Lecture Notes in Computer Science*, volume 1138, pages 320–338, Steyr, Austria, 1996. Third International Conference on Artificial Intelligence and Symbolic Computation.

[Der90] N. Dershowitz. A maximal-literal strategy for Horn Clauses. In S. Kaplan and M. Okada, editors, *2nd International Workshop on Conditional Term Rewriting Systems*, volume 516 of *Lecture Notes in Computer Science*, pages 143–154. Springer-Verlag, June 1990.

[DJ90] N. Dershowitz and J-P. Jouannaud. Rewriting Systems. In J. Van Leuven, editor, *Handbook of Theoretical Computer Science*. Elsevier Science Publishers North-Holland, 1990.

[KKR90] C. Kirchner, H. Kirchner, and M. Rusinowitch. Deduction with Symbolic Constraints. *Revue Française d'Intelligence Artificielle*, 4(3):9–52, 1990. Special issue on Automatic Deduction.

[Smo89] G. Smolka. *Logic Programming over Polymorphically Order-Sorted Types.* PHD Thesis, Universitat Kaiserslautern, FB Informatik, West Germany, 1989.

[Wak91] John F. Wakerly. Digital design principles and practices. Prentice Hall International Editions, 1991.

[WOLB91] L. Wos, R. Overbeek, E. Lusk, and J. Boyle. Automated Reasoning, Introduction and Applications. McGraw-Hill, 1991.

APPENDIX A

Here, we present all the conditional rules denoting all the gates found in the chips of electronic components database.

AND2

$$
\begin{array}{l}
l_s = g(l_1, l_2) \wedge s_1 = f(1, l_1, l_2, m_1^1, m_1^2) + 1 \wedge \cdots \wedge s_{15} = f(15, l_1, l_2, m_{15}^1, m_{15}^2) \wedge \\
F\$(out(\overrightarrow{x}), l_1, gates(m_1^1, \ldots, m_{15}^1)) = tt \wedge \\
F\$(out(\overrightarrow{y}), l_2, gates(m_1^2, \ldots, m_{15}^2)) = tt \\
\Rightarrow F\$(out(and(\overrightarrow{x}, \overrightarrow{y})), l_s, gates(s_1, \ldots, s_{15})) = tt
\end{array}
$$

AND3

$$
\begin{array}{l}
l_s = g(l_1, l_2, l_3) \wedge s_1 = f(1, l_1, l_2, l_3, m_1^1, m_1^2, m_1^3) \wedge s_2 = f(2, l_1, l_2, l_3, m_2^1, m_2^2, m_2^3) + 1 \wedge \\
\cdots \wedge s_{15} = f(15, l_1, l_2, l_3, m_{15}^1, m_{15}^2, m_{15}^3) \wedge \\
F\$(out(\overrightarrow{x}), l_1, gates(m_1^1, \ldots, m_{15}^1)) = tt \wedge \\
F\$(out(\overrightarrow{y}), l_2, gates(m_1^2, \ldots, m_{15}^2)) = tt \wedge \\
F\$(out(\overrightarrow{z}), l_3, gates(m_1^3, \ldots, m_{15}^3)) = tt \\
\Rightarrow F\$(out(and(\overrightarrow{x}, \overrightarrow{y}, \overrightarrow{z})), l_s, gates(s_1, \ldots, s_{15})) = tt
\end{array}
$$

AND4

$$
\begin{array}{l}
l_s = g(l_1, l_2, l_3, l_4) \wedge s_1 = f(1, l_1, l_2, l_3, l_4, m_1^1, m_1^2, m_1^3, m_1^4) \wedge \\
\cdots \wedge s_3 = f(3, l_1, l_2, l_3, l_4, m_3^1, m_3^2, m_3^3, m_3^4) + 1 \wedge \cdots \\
\wedge s_{15} = f(15, l_1, l_2, l_3, l_4, m_{15}^1, m_{15}^2, m_{15}^3, m_{15}^4) \wedge \\
F\$(out(\overrightarrow{x}), l_1, gates(m_1^1, \ldots, m_{15}^1)) = tt \wedge \\
F\$(out(\overrightarrow{y}), l_2, gates(m_1^2, \ldots, m_{15}^2)) = tt \wedge \\
F\$(out(\overrightarrow{z}), l_3, gates(m_1^3, \ldots, m_{15}^3)) = tt \wedge \\
F\$(out(\overrightarrow{t}), l_4, gates(m_1^4, \ldots, m_{15}^4)) = tt \\
\Rightarrow F\$(out(and(\overrightarrow{x}, \overrightarrow{y}, \overrightarrow{z}, \overrightarrow{t})), l_s, gates(s_1, \ldots, s_{15})) = tt
\end{array}
$$

OR2

$$
\begin{array}{l}
l_s = g(l_1, l_2) \wedge s_1 = f(1, l_1, l_2, m_1^1, m_1^2) \wedge \cdots \\
\cdots \wedge s_4 = f(4, l_1, l_2, m_4^1, m_4^2) + 1 \wedge \cdots \\
\cdots \wedge s_{15} = f(15, l_1, l_2, m_{15}^1, m_{15}^2) \wedge \\
F\$(out(\overrightarrow{x}), l_1, gates(m_1^1, \ldots, m_{15}^1)) = tt \wedge \\
F\$(out(\overrightarrow{y}), l_2, gates(m_1^2, \ldots, m_{15}^2)) = tt \\
\Rightarrow F\$(out(or(\overrightarrow{x}, \overrightarrow{y})), l_s, gates(s_1, \ldots, s_{15})) = tt
\end{array}
$$

OR3

$$
\begin{array}{l}
l_s = g(l_1, l_2, l_3) \wedge s_1 = f(1, l_1, l_2, l_3, m_1^1, m_1^2, m_1^3) \wedge \cdots \\
\cdots \wedge s_5 = f(5, l_1, l_2, l_3, m_5^1, m_5^2, m_5^3) + 1 \wedge \cdots \\
\cdots \wedge s_{15} = f(15, l_1, l_2, l_3, m_{15}^1, m_{15}^2, m_{15}^3) \wedge \\
F\$(out(\overrightarrow{x}), l_1, gates(m_1^1, \ldots, m_{15}^1)) = tt \wedge \\
F\$(out(\overrightarrow{y}), l_2, gates(m_1^2, \ldots, m_{15}^2)) = tt \wedge \\
F\$(out(\overrightarrow{z}), l_3, gates(m_1^3, \ldots, m_{15}^3)) = tt \\
\Rightarrow F\$(out(or(\overrightarrow{x}, \overrightarrow{y}, \overrightarrow{z})), l_s, gates(s_1, \ldots, s_{15})) = tt
\end{array}
$$

OR4	$l_s = g(l_1, l_2, l_3, l_4) \wedge s_1 = f(1, l_1, l_2, l_3, l_4, m_1^1, m_1^2, m_1^3, m_1^4) \wedge$ $\cdots \wedge s_6 = f(6, l_1, l_2, l_3, l_4, m_6^1, m_6^2, m_6^3, m_6^4) + 1 \wedge \cdots$ $\cdots \wedge s_{15} = f(15, l_1, l_2, l_3, l_4, m_{15}^1, m_{15}^2, m_{15}^3, m_{15}^4) \wedge$ $F\$(out(\overrightarrow{x}), l_1, gates(m_1^1, \ldots, m_{15}^1)) = tt \wedge$ $F\$(out(\overrightarrow{y}), l_2, gates(m_1^2, \ldots, m_{15}^2)) = tt \wedge$ $F\$(out(\overrightarrow{z}), l_3, gates(m_1^3, \ldots, m_{15}^3)) = tt \wedge$ $F\$(out(\overrightarrow{t}), l_4, gates(m_1^4, \ldots, m_{15}^4)) = tt$ $\Rightarrow F\$(out(or(\overrightarrow{x}, \overrightarrow{y}, \overrightarrow{z}, \overrightarrow{t})), l_s, gates(s_1, \ldots, s_{15})) = tt$

NOT	$l_s = g(l) \wedge s_1 = f(1, l, m_1) \wedge$ $\cdots \wedge s_7 = f(7, l, m_7) + 1 \wedge \cdots$ $\cdots s_{15} = f(15, l, m_{15}) \wedge$ $F\$(out(\overrightarrow{x}), l, gates(m_1, \ldots, m_{15})) = tt$ $\Rightarrow F\$(out(not(\overrightarrow{x})), l_s, gates(s_1, \ldots, s_{15})) = tt$

NAND2	$l_s = g(l_1, l_2) \wedge s_1 = f(1, l_1, l_2, m_1^1, m_1^2) \wedge \cdots$ $\cdots \wedge s_8 = f(8, l_1, l_2, m_8^1, m_8^2) + 1 \wedge \cdots$ $\cdots \wedge s_{15} = f(15, l_1, l_2, m_{15}^1, m_{15}^2) \wedge$ $F\$(out(\overrightarrow{x}), l_1, gates(m_1^1, \ldots, m_{15}^1)) = tt \wedge$ $F\$(out(\overrightarrow{y}), l_2, gates(m_1^2, \ldots, m_{15}^2)) = tt$ $\Rightarrow F\$(out(nand(\overrightarrow{x}, \overrightarrow{y})), l_s, gates(s_1, \ldots, s_{15})) = tt$

NAND3	$l_s = g(l_1, l_2, l_3) \wedge s_1 = f(1, l_1, l_2, l_3, m_1^1, m_1^2, m_1^3) \wedge \cdots$ $\cdots \wedge s_9 = f(9, l_1, l_2, l_3, m_9^1, m_9^2, m_9^3) + 1 \wedge \cdots$ $\cdots \wedge s_{15} = f(15, l_1, l_2, l_3, m_{15}^1, m_{15}^2, m_{15}^3) \wedge$ $F\$(out(\overrightarrow{x}), l_1, gates(m_1^1, \ldots, m_{15}^1)) = tt \wedge$ $F\$(out(\overrightarrow{y}), l_2, gates(m_1^2, \ldots, m_{15}^2)) = tt \wedge$ $F\$(out(\overrightarrow{z}), l_3, gates(m_1^3, \ldots, m_{15}^3)) = tt$ $\Rightarrow F\$(out(nand(\overrightarrow{x}, \overrightarrow{y}, \overrightarrow{z})), l_s, gates(s_1, \ldots, s_{15})) = tt$

AND4	$l_s = g(l_1, l_2, l_3, l_4) \wedge s_1 = f(1, l_1, l_2, l_3, l_4, m_1^1, m_1^2, m_1^3, m_1^4) \wedge$ $\cdots \wedge s_{10} = f(10, l_1, l_2, l_3, l_4, m_{10}^1, m_{10}^2, m_{10}^3, m_{10}^4) + 1 \wedge \cdots$ $\wedge s_{15} = f(15, l_1, l_2, l_3, l_4, m_{15}^1, m_{15}^2, m_{15}^3, m_{15}^4) \wedge$ $F\$(out(\overrightarrow{x}), l_1, gates(m_1^1, \ldots, m_{15}^1)) = tt \wedge$ $F\$(out(\overrightarrow{y}), l_2, gates(m_1^2, \ldots, m_{15}^2)) = tt \wedge$ $F\$(out(\overrightarrow{z}), l_3, gates(m_1^3, \ldots, m_{15}^3)) = tt \wedge$ $F\$(out(\overrightarrow{t}), l_4, gates(m_1^4, \ldots, m_{15}^4)) = tt$ $\Rightarrow F\$(out(nand(\overrightarrow{x}, \overrightarrow{y}, \overrightarrow{z}, \overrightarrow{t})), l_s, gates(s_1, \ldots, s_{15})) = tt$

NAND8

$$\begin{array}{l}
l_s = g(l_1,\ldots,l_8) \wedge s_1 = f(1, l_1,\ldots,l_8, m_1^1,\ldots,m_1^8)\wedge \\
\cdots \wedge s_{11} = f(11, l_1,\ldots,l_8, m_{11}^1,\ldots,m_{11}^8) + 1 \wedge \cdots \\
\wedge s_{15} = f(15, l_1,\ldots,l_8, m_{15}^1,\ldots,m_{15}^8)\wedge \\
F\$(out(\overrightarrow{x_1}), l_1, gates(m_1^1,\ldots,m_{15}^1)) = tt \wedge \vdots \\
F\$(out(\overrightarrow{x_8}), l_8, gates(m_1^8,\ldots,m_{15}^8)) = tt \\
\Rightarrow F\$(out(nand(\overrightarrow{x_1},\ldots,\overrightarrow{x_8})), l_s, gates(s_1,\ldots,s_{15})) = tt
\end{array}$$

NOR2

$$\begin{array}{l}
l_s = g(l_1,l_2) \wedge s_1 = f(12, l_1, l_2, m_1^1, m_1^2) \wedge \cdots \\
\cdots \wedge s_{12} = f(12, l_1, l_2, m_{12}^1, m_{12}^2) + 1 \wedge \cdots \\
\cdots \wedge s_{15} = f(15, l_1, l_2, m_{15}^1, m_{15}^2)\wedge \\
F\$(out(\overrightarrow{x}), l_1, gates(m_1^1,\ldots,m_{15}^1)) = tt\wedge \\
F\$(out(\overrightarrow{y}), l_2, gates(m_1^2,\ldots,m_{15}^2)) = tt \\
\Rightarrow F\$(out(nor(\overrightarrow{x},\overrightarrow{y})), l_s, gates(s_1,\ldots,s_{15})) = tt
\end{array}$$

NOR3

$$\begin{array}{l}
l_s = g(l_1,l_2,l_3) \wedge s_1 = f(1, l_1, l_2, l_3, m_1^1, m_1^2, m_1^3) \wedge \cdots \\
\cdots \wedge s_{13} = f(13, l_1, l_2, l_3, m_5^1, m_5^2, m_5^3) + 1 \wedge \cdots \\
\cdots \wedge s_{15} = f(15, l_1, l_2, l_3, m_{15}^1, m_{15}^2, m_{15}^3)\wedge \\
F\$(out(\overrightarrow{x}), l_1, gates(m_1^1,\ldots,m_{15}^1)) = tt\wedge \\
F\$(out(\overrightarrow{y}), l_2, gates(m_1^2,\ldots,m_{15}^2)) = tt\wedge \\
F\$(out(\overrightarrow{z}), l_3, gates(m_1^3,\ldots,m_{15}^3)) = tt \\
\Rightarrow F\$(out(or(\overrightarrow{x},\overrightarrow{y},\overrightarrow{z})), l_s, gates(s_1,\ldots,s_{15})) = tt
\end{array}$$

NOR4

$$\begin{array}{l}
l_s = g(l_1,l_2,l_3,l_4) \wedge s_1 = f(1, l_1, l_2, l_3, l_4, m_1^1, m_1^2, m_1^3, m_1^4)\wedge \\
\cdots \wedge s_{14} = f(14, l_1, l_2, l_3, l_4, m_{14}^1, m_{14}^2, m_{14}^3, m_{14}^4) + 1\wedge \\
s_{15} = f(15, l_1, l_2, l_3, l_4, m_{15}^1, m_{15}^2, m_{15}^3, m_{15}^4)\wedge \\
F\$(out(\overrightarrow{x}), l_1, gates(m_1^1,\ldots,m_{15}^1)) = tt\wedge \\
F\$(out(\overrightarrow{y}), l_2, gates(m_1^2,\ldots,m_{15}^2)) = tt\wedge \\
F\$(out(\overrightarrow{z}), l_3, gates(m_1^3,\ldots,m_{15}^3)) = tt\wedge \\
F\$(out(\overrightarrow{t}), l_4, gates(m_1^4,\ldots,m_{15}^4)) = tt \\
\Rightarrow F\$(out(nand(\overrightarrow{x},\overrightarrow{y},\overrightarrow{z},\overrightarrow{t})), l_s, gates(s_1,\ldots,s_{15})) = tt
\end{array}$$

NOR8

$$\begin{array}{l}
l_s = g(l_1,\ldots,l_8) \wedge s_1 = f(1, l_1,\ldots,l_8, m_1^1,\ldots,m_1^8)\wedge \\
\cdots \wedge s_{15} = f(15, l_1,\ldots,l_8, m_{15}^1,\ldots,m_{15}^8)\wedge \\
F\$(out(\overrightarrow{x_1}), l_1, gates(m_1^1,\ldots,m_{15}^1)) = tt\wedge \\
\vdots \\
F\$(out(\overrightarrow{x_8}), l_8, gates(m_1^8,\ldots,m_{15}^8)) = tt \\
\Rightarrow F\$(out(nor(\overrightarrow{x_1},\ldots,\overrightarrow{x_8})), l_s, gates(s_1,\ldots,s_{15})) = tt
\end{array}$$

6 APPENDIX B

6.1 The algorithm of f

We assume readers are familiar with basic notions and operations on *lists*. The structure of a list is defined recursively as follows:

$$list = [integer]$$

$$or$$

$$list = [integer, sublist_1, \ldots, sublist_k]$$

where $sublist_i$ is a sub-list of the same structure of $list$.

For readability's sake, we only express the algorithm of f in the case where it has only two arguments of *list* type. The algorithm of f is the following:

$$
\boxed{
\begin{array}{l}
f : int \times list \times list \times int \times int \longrightarrow int \\
\\
int\ f(arg, l_1, l_2, m_1, m_2) \\
begin\ function \\
\quad sum := sum\_redund\_val(arg, l_1, l_2) \\
return(m_1 + m_2 - sum) \\
end\ function
\end{array}
}
$$

The auxiliary function *sum_redund_val* is defined as follows :

$$
\boxed{
\begin{array}{l}
sum\_redund\_val : int \times list \times list \longrightarrow int \\
\\
int\ sum\_redund\_val(arg, l_1, l_2) \\
begin\ function \\
\quad \underline{\text{if}}\ (empty(l_1)) \\
\quad \underline{\text{then}}\ return(0) \\
\quad \underline{\text{else}} \\
\quad\quad sum := 0 \\
\quad\quad \underline{\text{foreach sublist}}\ sl\ \underline{\text{of}}\ l_1\ \underline{\text{do}} \\
\quad\quad\quad cur\_id := id\_of(sl) \\
\quad\quad\quad \underline{\text{if}}\ (cur\_id \in l_2) \\
\quad\quad\quad \underline{\text{then}}\ sum := sum + num\_gates(cur\_id, arg) \\
\quad\quad\quad \underline{\text{else}}\ sum := sum + sum\_redund\_val(arg, sl, l_2) \\
\quad\quad \underline{\text{od}} \\
\quad return(sum) \\
end\ function
\end{array}
}
$$

Let us make clear all the functions used above.

- $empty : list \longrightarrow bool$, returns *true* if the list is empty, false otherwise.

- id_of : $list \longrightarrow int$, returns the identity of a list. Here, the *identity* of a list is its first element.
- num_gates : $int \times int \longrightarrow int$, this function takes in its first argument the identity of a given conditionnal rule, and by *using* its second argument, it returns the number of gates associated to this rule.

6.2 The algorithm of g

For readability's sake, we only express the algorithm of g in the case where it has only two arguments of *list* type. Assuming that the structure of a list is always the same as the one defined in thes previous section, the algorithm of g is as follows :

```
g : list × list ⟶ list

list g(first, second)
begin function
   l := new_list()
   l := add_id(l, CURRENT_ID)
   l1 := copy_list(first)
   l2 := copy_list(second)
   l1 := del_redun_id(l1, l2)
   l := add_tail_list(l, l1, l2)
   return(l)
end function
```

The auxiliary function del_redun_id is the following:

```
del_redun_id : list × list ⟶ list

list del_redun_id(l1, l2)
begin function
   if(empty(l1))
   then return(l1)
   else
      foreach sublist sl of l1 do
         if (id_of(sl) ∈ l2)
         then l1 := del_sub_list(l1, sl)
         else l1 := del_redun_id(sl, l2)
       od
   return(l1)
end function
```

where:

- $CURRENT\_ID$ is the current identity given by the system for the new generated equation.

- $new\_list : \longrightarrow list$, returns a new empty list.
- $add\_tail\_list : list \times list \times \ldots \times list \longrightarrow list$, adds all the other lists given in arguments to the tail of the first list.
- $del\_sub\_list : list \times list \longrightarrow list$, deletes a sub-list from a list.
- $id\_of : list \longrightarrow int$, returns the identity of a list, defined by its first element.

Publications récentes du LIFO
LIFO's recent publications

97-7 E. MELIN, B. RAFFIN, X. REBEUF, B. VIROT
A Unifying Approach of Data and Task Parallelism Based on Structural Clock Communication Paradigm (extended version)

97-8 S. ANANTHARAMAN, Editor
Proceedings of UNIF'97, International Workshop on Unification

97-9 F. BENTAYEB, D. LAURENT
Inversion de l'algrèbre relationnelle et mises à jour

97-10 I. PARFENOFF
An efficient parallel algorithm for maximum matching in P_4-tidy graphs

97-11 F. ROUSSEL, I. RUSU
An O(m2+mn) algorithm to recognize Meyniel graphs

97-12 M. BAHMA, G. HAINS
Prevision des performances de la θ-jointure et de la multi-jointure pour des machines SN

97-13 F. LOULERGUE, G. HAINS
Design of an event-oriented functionnal parallel language : the finite case

97-14 S. LIMET, F. SAUBION
A proof method for logic programming

97-15 E. MELIN, B. RAFFIN, X. REBEUF, B. VIROT
An Automatic Distribution of Sequential Code Fitting Irregular Data Accesses

Pour récupérer les rapports de recherche du LIFO / To get LIFO's research reports :

- Par WWW : http://www.univ-orleans.fr/SCIENCES/LIFO/Recherche/Publis/RR/index.html
- Par FTP : ftp://ftp-lifo.univ-orleans.fr/pub/RR
- Par courrier papier adressé à / By surface mail adressed to :
  Arnaud Lallouet, LIFO, Université d'Orléans
  4, rue Léonard de Vinci, BP 6759
  F-45067 Orléans cedex 2, FRANCE
  en mentionnant votre adresse.
- Par courrier électronique à / By e-mail to : Arnaud.Lallouet@lifo.univ-orleans.fr