SAME 98

Session: Tools / Flow Back End

# CIRCUIT DISASSEMBLY FOR VERIFICATION AND FUNCTIONAL ABSTRACTION OF CMOS CIRCUITS

Anthony Lester, Pirouz Bazargan-Sabet, Alain Greiner

LIP6/ASIM Laboratory
Université Pierre et Marie Curie (Paris 6)
4, Place Jussieu, 75252 Paris Cedex 05

## Abstract:

The continuing increase in the complexity of integrated circuits, together with the resulting automation of the design process has resulted in a greater emphasis on post-layout verification. This involves checking the functional correctness of the final design, as well as verifying the timing and power consumption constraints. Here we present a fast algorithm of extracting an oriented and characterized gate-netlist from a post-layout transistor-level netlist. This netlist can be used by gate-level timing and power verification tools, in addition to being used to generate an extracted behavioral description for comparison with the initial circuit specifications.

# 1. INTRODUCTION

### 1.1. Background

The typical approach to post-layout circuit verification assumes that a design has been performed using a cell library. This implies that a gate-level netlist of the complete design is already available, which can be back annotated to take into account layout information. However, modern circuits are often designed using different design approaches for different blocks. Significant blocks of the circuit may be designed using parameterized generators or even full-custom, such that a gate netlist does not exist for these blocks.

The lack of a gate netlist renders impossible the use of rapid gate-level verification tools, whilst, for large designs, transistor-level verification with methods such as SPICE remain computationally prohibitive. To resolve this problem, a means of generating an oriented gate netlist from an unoriented transistor netlist is required; this technique is known as circuit disassembly.

Two basic methods exist; the first method uses pattern-matching techniques to identify predefined gates within the transistor netlist, the set of gates being constituted from a user-defined library [2][3]. This method, however, implies that the user must define the rules for the disassembly, a process that can be complex and error prone, particularly for full-custom blocks.

### 1.2. Objectives

Our method is based on a second approach, which can be defined as a partitioning of the transistor netlist according to a limited number of generic rules. The result of this partitioning is a set of oriented gates, which, although not necessarily representing the original netlist, nevertheless express the signal flow within the circuit. This kind of approach has the advantage of being fully automatic, however, the majority of implementations of this approach suffer from an inability to handle diverse design styles.

Our implementation, whilst being essentially targeted to CMOS designs, can efficiently handle complex design styles such as precharge logic or pass-transistor logic. This is achieved by performing functional analysis during the partitioning phase so that the behavior of extracted gates is exploited to condition the extraction of remaining gates.

# 2. METHOD

### 2.1. The Circuit Partition Model

Our algorithm, known as YAGLE, uses a partition model defined by an older disassembler tool DESB [1](also developed at the LIP6). The idea is to partition the circuit into extracted gates such that the states of all the circuit nodes connected to at least one transistor gate are defined. To achieve this, a "pseudo-gate" or "cone" is constructed for each of these nodes. A cone is defined as the set of paths for current flow between the power supplies or external connectors and the node on which the cone is built.

The result of this partitioning is a set of cones, which completely describe the state of all transistor gates and drivable external connectors. Each cone being made up of the set of branches corresponding to the current paths. The principle of the cone partitioning is illustrated in Figure 1 for a small example.

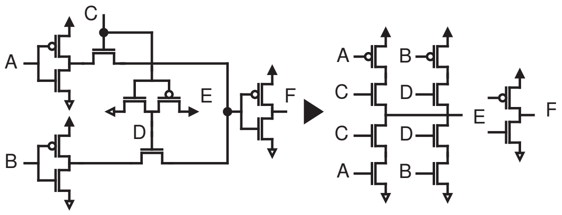


**Figure 1: Transistor Netlist to Cone Netlist**

Figure 1 shows the branch structure of two of the cones produced for the given transistor netlist. This resulting cone netlist is subsequently completed with connectivity information; each cone contains the list of cones by which it is driven and the list of cones which it drives. The cone netlist than then be considered to be a directed acyclic graph of the signal flow within the circuit.

### 2.2. Cone Analysis

The branch structure of the cone allows its functionality to be readily expressed in terms of its set and reset conditions. Thus, for each cone the following pair of Boolean equations is generated:

$$S_{up} = \sum_{VddPaths} \prod_{TN_i} x_i \prod_{TP_j} \bar{x}_j \tag{1}$$

$$S_{down} = \sum_{VssPaths} \prod_{TN_i} x_i \prod_{TP_j} \bar{x}_j \tag{2}$$

where x corresponds to the signal on the transistor gate.

Equation 1 defines the set condition and Equation 2 defines the reset condition.

If

$$F_{up} \cdot F_{down} = 0 \tag{3}$$

then the cone is locally non-conflictual since there exists no input pattern which can activate an up path and a down path simultaneously.

Conversely, if

$$F_{up} + F_{down} = 1 \tag{4}$$

then the cone is locally non-tristate since there exists no input pattern which activates neither an up path nor a down path.

The functional behavior of the cone can therefore be obtained as soon as the cone is built, hence allowing it to be exploited to condition further cone construction. However, Equations 3 and 4 (the cone tautology expressions) represent the conditions for local high impedance and conflictuality. The non-satisfaction of Equations 3 and 4 implies that there exist direct input combinations that result in conflictuality or high impedance. However, the preceding logic may render impossible these combinations.

The construction algorithm detailed in the following section allows global versions of the cone tautology expressions to be verified by allowing the set and reset conditions to be expressed in terms of primary variables. These variables constitute the smallest set of independent variables that can entirely express the cone functionality.

# 3. ALGORITHM

### 3.1. The Main Problem

The main problem, which needs to be addressed by the cone construction algorithm is the prevention of false branch construction. An illustration of what is meant by a false branch is shown in Figure 2.

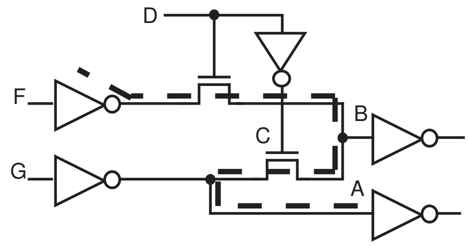


**Figure 2: False Branch Risk**

Figure 2 clearly shows that if the cone branches are built without taking into account the surrounding logic then there is a risk that functionally redundant branches will be built. In the case of Figure 2, the branch marked by the dashed line will be built if the mutual exclusion of signals C and D is ignored.

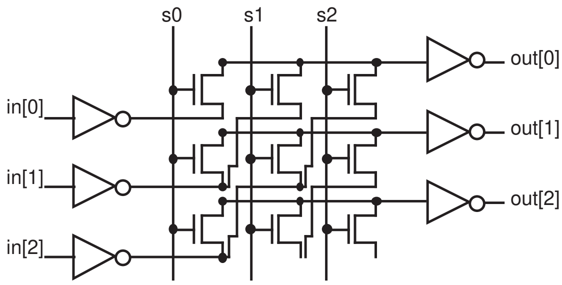


**Figure 3: Barrel Shifter Architecture**

For certain design styles, notably barrel shifters (see Figure 3) the number of possible false branches can be extremely large, and hence computationally expensive, in terms of time and memory to eliminate a posteriori. The important problem to resolve is therefore the detection of signal correlations within the circuit.

### 3.2. Overview

The objective is to exploit signal correlation during the cone construction phase, whilst minimizing the computational overhead where the cone extraction is simple enough to not require functional analysis. To achieve this, a two-phase algorithm has been developed.

The first phase is the extraction of all CMOS dual cones. Since our target is CMOS VLSI, a large proportion of any design is likely to be made up of simple CMOS gates. These do not require functional analysis of the logical context, since their structure does not allow false branches. In addition, the cone tautology conditions are verified locally, so no further analysis is required in order to exploit their behavior to extract further cones.

We define CMOS dual cones as those which satisfy the following criteria:-

a) Up branches contain only P-type transistors, and down branches contain only N-type transistors.

b) No internal node of the resulting cone attacks the gate of a transistor.

c) Equations (3) and (4) are satisfied.

In addition, all circuit nodes on which a cone must be built are identified, and the corresponding cone is initialized.

The second phase is the construction of all remaining cones, conditioned by their surrounding logic. The various stages of this phase are detailed in the subsequent sections.

### 3.3. Circuit Traversal

During the second phase of the cone construction, a cone is built as soon as its surrounding logic has been completed. Thus, in order to minimize redundant traversal of the incomplete cone list, an initial approximation of the cone netlist connectivity is generated, producing an approximation of the global dependency graph (GDG) of the circuit. The list of cones is then traversed according to a depth-first traversal of the GDG.

For each cone traversed, an attempt is made to build the local dependency graph (LDG) for the cone. The LDG is a directed single root node acyclic graph built, by recursive construction, separately for each cone and bounded by the following conditions:-

a) Globally conflictual or high impedance node.

b) Looped node.

c) External connector node.

d) Node out of depth. The LDG is limited by a user-specified depth parameter defined, for a given node, as the longest path between that node and the root node

e) Node corresponding to incomplete cone.

Criteria a) to d) correspond to valid stop criteria which are constant between iterations. If the construction of the LDG is only limited by these criteria then it is considered complete. Criterion e) implies that the graph is incomplete within the required logic depth, and hence the cone is skipped in the current iteration.

### 3.4. Primary Variables

As soon as an incomplete cone with a complete LDG is reached, it is then necessary to detect the useful correlations within the graph, thus we need to find the primary variables of the cone.

<u>Defn</u>.: The primary variables of a circuit node are the smallest set of mutually uncorrelated variables which entirely define the state of that node within the specified depth.

A typical LDG produced for a depth of 4 is shown in Figure 4. In the example of Figure 4, the set of nodes that corresponds to the primary variables is {B, D, E}. This set can be readily obtained from the LDG, since it corresponds to the set of articulation points closest to the root node.

<u>Defn</u>.: An articulation point, in a single root directed acyclic graph, is a node such that no node upon which it depends has a path between itself and the root node which does not pass through this node.

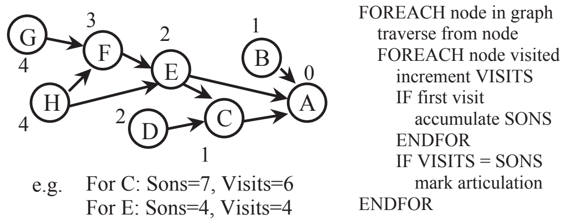


**Figure 4: LDG Primary Variable Detection**

From this definition, the articulation points of a single-root graph are identified using the algorithm of Figure 4. It is then straightforward to traverse the graph to obtain those closest to the root.

### 3.5. Cone Construction

When the set of primary variables of the cone under construction have been obtained, the branches are built by recursive construction from the cone output node. For each transistor link added to a branch, the conductance of the resulting branch is verified. If the correlations between the primary variables are such that the branch is never active, its construction is abandoned immediately.

This verification is optimized by the dynamic maintenance of the cone support graph, as shown in

Figure 5. This is a two-level graph whose outputs are the cone direct inputs and whose inputs are the primary variables.

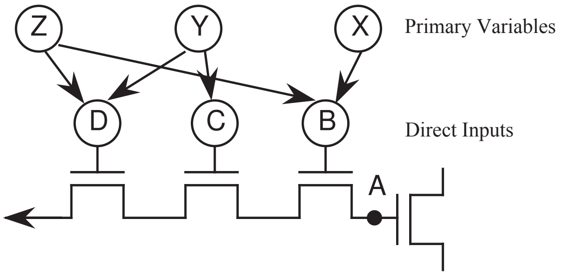


**Figure 5: The Cone Support Graph**

For each transistor added, this graph is used to obtain the set of direct inputs correlated with its own input. The product of these direct inputs is obtained in terms of the primary variables. A reduced ordered binary decision diagram representation is used for the Boolean expression manipulation. As a result, if the addition of the transistor has blocked the branch, the product of the correlated direct inputs will directly evaluate to zero.

On termination of the branch construction, the global tautology expressions are verified by expanding the cone functionality to the primary variables. This subsequently allows the immediate exploitation of the cone to condition the construction of cones dependent upon it.

### 3.6. Treatment of Loops

Loops in the cone netlist can occur for a number of reasons at different stages of the disassembly.

False loops can occur in the initial approximation of the global dependency graph, these merely render the cone list traversal sub-optimal and make subsequent iterations necessary. These disappear during the cone construction phase as false branches and inputs are eliminated.

Any genuine loops of more than two cones, remaining after the construction phase, are considered to be design errors and reported as such. Any two-cone loops are compared to a number of predefined bleeder and latch forms, and marked correspondingly in the case of a positive identification. Any unidentified two-cone loops are reported.

## 4. RESULTS

The algorithm described above has been implemented in a tool distributed for evaluation as part of the ALLIANCE CAD System [4].

A number of execution time results of the approach are shown in Table 1. The results are from execution on a Sun UltraSparc 1 workstation with 64Mb RAM, except the last result, which required a machine with 150 Mb.

The first three circuits are combinatorial blocks generated by optimized, parameterizable generators, hence no original gate netlist exists. Disassembly is the only way that gate-level verification tools can be applied to these circuits. The last two circuits are a complete 32 bit microprocessor and a PCI component respectively, both complete chips.

| Circuit | Number of Transistors | Number of Gates | Time |
|---|---|---|---|
| 1024x64 ROM | 75,048 | 1,788 | 4m22s |
| 64bit MULT | 91,144 | 23,824 | 9m46s |
| 32bit SHIFT | 2,828 | 391 | 21s |
| DLXm | 37,794 | 7,823 | 2m27s |
| PCIDDC | 202,264 | 54,217 | 29m48s |

**Table 1: Execution Times**

## 5. CONCLUSION

The results show that a generalized algorithmic approach to the problem of circuit disassembly can be developed. The cone partitioning algorithm combined with a parallel exploitation of an extracted cone functionality can efficiently handle the task of generating an oriented gate netlist for an unoriented “sea of transistors” for complex and diverse design styles.

This oriented gate netlist, being both functionally and electrically equivalent to the original circuit can be used for functional verification, by formal proof or simulation, or timing and power analysis.

## References

[1] M. Laurentin, A. Greiner, R. Marbot, “DESB, a Functional Abstractor for CMOS VLSI”, IEEE European Design Automation Conference, 1992.

[2] T. Kostelijk, B. De Loore, “Automatic Verification of Library-Based IC Designs”, IEEE Journal of Solid-State Circuits, March 1991.

[3] J.-B. Guignet, “Abstraction Fonctionnelle des Composants VLSI”, PhD Thesis, Université Pierre et Marie Curie, March 1998.

[4] A. Greiner, F. Pécheux, “Alliance: A Complete Set of CAD Tools for Teaching Digital VLSI Design”, 3rd Eurochip Workshop on VLSI Design Training, pp. 230-237, September 1992.