# Derivation of Signal Flow Direction in MOS VLSI : an Alternative Approach

*W.De Rammelaere, I.Bolsens, L.Claesen*[†], *H.De Man*[*]

IMEC Lab.[‡], Leuven, Belgium, Phone: +32-16-281201

**Abstract**

This paper presents new rules to derive the signal flow in a VLSI-design. The method is based on sound logical principles to guarantee the correctness of the results and a maximal design-style independency.

The different steps in the taggingprocess are explained and illustrated with examples. An interesting application of the signal flow results for verification purposes is the concept of Intended Unilateral Blocks. It reflects the intended behavior of a design and its use leads to an increased efficiency in electrical as well as timing verification programs.

## 1 Introduction

Several VLSI CAD tools, especially in the field of electrical and timing verification require some knowledge about the signal flow through MOS transistors (i.e. the propagation of the logic values through the switch network from external inputs to outputs) in order to understand the circuit behavior and/or increase the efficiency of the tools. This knowledge about the signal flow can be provided manually or it can be derived by expert-systems in an automated way.

This paper describes new rules, based on simple logical principles, to derive the signal flow through MOS VLSI-circuits which are applicable within electrical verifiers and which are independent of the design methodology. In section 2 the basic aspects of this theory are explained, and the differences with existing flow-derivation tools are pointed out. Sections 3 and 4 give a brief overview of the implementation and on the results for various circuit types. In a further section the application area for the dataflow is discussed. Concluding remarks are given in section 6.

## 2 Basic Approach

### 2.1 Existing Techniques vs. Electrical Verification

Many timing verifiers, such as TV [1] and Chrystal [2] require the derivation of signal flow through transistors in order to reduce the number of false paths, and to increase efficiency. Some simulation programs (CSWAN [9]) also use signal flow partitioning to reduce complexity.

It was generally experienced that the derivation of signal flow for VLSI custom design is best performed by combining automated tagging (by reliable direction-finding rules) with hand-tagging for difficult transistors. A general technique to derive this signal flow has been extensively treated in [3]. Recently, an interesting approach based on electrical properties was proposed in [13]. We think that the rules that were proposed in [3] may be erroneous in complex design styles but moreover, the rules are not applicable for our needs of signal flow :

Indeed, our aim is to examine the electrical correctness of VLSI-circuits statically within the DIALOG-environment [5] based upon a formal theory of correct electrical behavior of a circuit [6]. In order to derive the intended behavior of the circuit we need a tool to find the flow direction through transistors. It is of course vital that the rules which accomplish this task do not depend upon electrical correctness of the circuit. In a first stage we implemented the safe rules of Jouppi [3] in DIALOG, excluding the rule on K-ratio's because of the **explicit** requirement of electrical correctness (following the suggestions in [3]). We concluded that these remaining local rules are indeed effective in some cases but unsuited for electrical verification : Some of the rules **implicitly** require electrical correctness and it is therefore impossible to guarantee their safety (e.g. the Always On Invertor rule is only correct if the composing transistors are sufficiently large). Moreover, because of the local character of the rules, feedback cannot be handled properly. A small example of a register-file (figure 1) in our Cathedral-library illustrates these drawbacks : N1 and N2 can be directed by the Always On Invertor rule. If N1 is directed first by invertor I1, then this results in an incorrect tagging, and the rule is not applicable any more for invertor I2. The Complement Gate Detection rule is not applicable to direct M1 and M2 because a more complex decoder (instead of a simple invertor) controls the gates of the passtransistor-network. Many transistors will remain undirected. Similar problems occur in RAM's (e.g. with 6 transistor CMOS RAM-cell). Finally, the independency of design style of some of the safe rules may be questioned, since the Invertor K-ratio is explicitly suited for NMOS and the Always On Invertor is explicitly CMOS.

The method proposed in [13] is more straightforward because it is strictly based on solid electrical principles. However, it is our experience that static verification cannot work properly if the surrounding circuitry of local circuit parts is not considered (relations between circuit nodes, etc...). Since the method also considers transient effects, we expect many unset transistors in CMOS with pass-gate circuitry. Moreover, we may expect efficiency problems using the proposed method with large channel connected components.

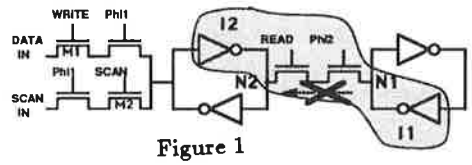


Figure 1

### 2.2 Reformulation of the dataflow-problem

When a circuit has not yet been verified electrically or when electrical verification is the major goal behind the dataflow problem, the logical principles and the electrical reality can clash. Mixing electrical and logical principles to obtain meaningful results is excluded. One possible solution is to direct transistors based on electrical principles ([13]) before switch level simulation. The other solution is dataflow determination based on logical principles. The following arguments favor this approach :

[*] Professor at K.U.Leuven

[‡] Interuniversity Microelectronics Center

- Logical principles are likely to be design style independent because of their more general character.
- Logical principles can work on a higher level of abstraction than the actual circuit components. We will show this by introducing the reduced switch graph. An increased efficiency results from it.
- Static verification requires more than an input-output behavioral description of the circuit. It needs a behavioral description of lower circuit parts in order to allow for verification of these parts. The logical dataflow determination is a way to obtain a part of this behavioral description, and the global rules provide some of the necessary general information.

When verifying the circuit electrically, the physical (electrical) dataflow will be checked against the results obtained from the logical properties, which will allow us to find electrical bugs in designs : the **intended behavior** derived from the logical principles is verified against the electrical behavior.

### 2.3 The logical principles

**Definition 1** *A node $n_i$ is a* **source** *of information for another node $n_j$ if the flow of information goes from node $n_i$ to $n_j$ via* source-drain *connections of transistors, i.e. if the value of node $n_i$ may determine the value of node $n_j$. Node $n_j$ is then a* **sink** *of node $n_i$ [3].*

**Definition 2** *A node $n_i$ is a* **1(0)-source** *for another node $n_j$ if $n_i$ may activate a* drain-source *path between $n_j$ and vdd (gnd) via the gate of a transistor.*

**Definition 3** *A node $n_i$ is a* **X-source** *of information for another node $n_j$ if it is a 0-source, 1-source or source for $n_j$. In that case $n_j$ is an X-sink from $n_i$.*

**Definition 4** *A node $n_i$ is a* **bidirectional** *with another node $n_j$ if it is a X-source for $n_j$ and a X-sink from $n_j$.*

A transistor is directed if an assignment of the opposite direction would create a design that violates the following **logical** principles :

1. Every node in the network contributes to the functionality. This implies that all nodes (except for primary inputs and outputs) should be source and sink of information. This principle leads to a set of rules that operate locally on the circuit.
2. A primary input is only a source of information.
3. Every gate node of a transistor should be able to switch the transistor on and off i.e. every gate node in the circuit (except for primary inputs) can be driven to 1 and 0. This principle results also in local rules.
4. Every node in the circuit should have a path to a primary input, in a direction opposite to the assumed signal flow (via X-sink relations) and every node should have a path to a primary output in the signal flow direction (X-source relations). This statement introduces rules operating on the global circuit.

The principles one to three form the true safe subset of the design principles defined by [3], but reformulated in a node oriented way that will enable the use of a smaller search graph. The fourth principle allows to cope with feedback circuits and it has an important influence on circuits with intensive use of passtransistors. It should be noted that the dual principle of the second principle ('A primary output is only a sink of information') is not valid in many cases and is therefore not used.

## 3 Implementation

The formulation of the logical principles is node-oriented, but the application of these rules on all nodes of the circuit is unnecessary and undesirable for efficiency reasons. Indeed, the global principle endangers the linear complexity of the problem; this danger is largely neutralized using reduction rules to decrease the search graph. Therefore, the switch graph of the circuit is reduced to a partially directed **Reduced Switch Graph** on which the logical principles will be fired in rule-form. An example is shown in figure 2.

- The vertices of this graph are the boundary and connecting nodes of the biconnected components ([11]) of the switch graph; in [4] they are called output nodes. If a node is a vertex of the graph, then it must be connected through independent paths to three or more distinct vertices or supply nodes (theorem 4.3 in [4]). The supply nodes are not vertices of the graph.
- The edges of the graph are defined as follows :

  $E = \{(v_i, v_j) | v_i$ has a source-drain path to $v_j$ without passing a $v_x$ with $v_i, v_j, v_x$ vertices of the reduced switch graph V$\}$

  $\cup \{< v_n, v_m > | v_n$ is a 1-source or a 0-source of $v_m$ without passing a $v_x$ with $v_n, v_m$ and $v_x$ vertices of $V\}$

  () indicate an unordered pair (an **unknown** relation) and <> indicate an ordered pair.

The rules direct the graph as much as possible by changing the unknown edges into source (and sink) edges. Edges which are directed in the two directions are called bidirectional. When the Reduced Switch Graph is directed maximally, expansion of the graph to the original circuit takes place. This can be done without introducing any conflicts, because of the nature of biconnected components.

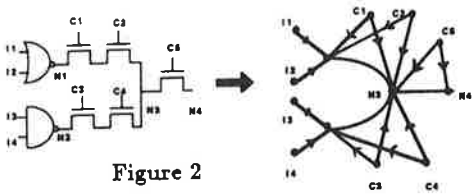

Figure 2

The four logical principles resulted in ten basic rules, described with the LEXTOC language in DIALOG [5]. Rules 3 and 4 reflect the first principle, rules 5 and 6 resulted from the third principle. The second principle is repeated in rule 10.

In order to master the complexity of the global principle (which may involve a long search towards terminal nodes) we introduced the first two rules : these rules treat nodes that already have a path to an input terminal as an input-terminal for the global rule. By firing these two rules regularly during the dataflow determination, the execution time of the global rules is kept very reasonable. The global principle itself can be found back in rules 7 and 8.

```
* For Vi a non-terminal graphnode :
1. Path to input
     Conditions  : A node Vi has a path to an in-terminal
                   via X-sink relations.
     Action      : Vi is a pseudo-in-terminal
2. Path to output : Dual case of rule 1

* For Vi a non-terminal graphnode :
3. Sink Node
     Conditions : All nodes Vj that have an Edge-relation
                  with Vi, X-source Vi except one vertex
                  Vx which is unknown with respect to Vi.
     Action     : Vx is sink of Vi
4. Source Node  : Dual case of rule 3
5. No path to VDD
     Conditions : All nodes Vj that have an Edge-relation
                  with Vi are 0-sources or X-sinks of Vi
                  except for one vertex Vx.
     Action     : Vx is source of Vi (in fact a 1-source)
6. No path to GND : Dual case of rule 5
```

```
7. No path to output
     Conditions : All nodes Vj that have an Edge-relation
                  with Vi are X-sources or X-sinks of Vi
                  except for one vertex Vx which is unknown
                  with respect to Vi.
                  None of the X-sink vertices of Vi have
                  a possible path to a pseudo-out-terminal.
     Action     : Vx is sink of Vi
8. No path to input
     Conditions : All nodes Vj that have an Edge-relation
                  with Vi are X-sources or X-sinks of Vi
                  except for one vertex Vx which is unknown
                  with respect to Vi.
                  None of the X-source vertices of Vi have
                  a possible reverse path to a
                  pseudo-in-terminal.
     Action     : Vx is source of Vi
9. Bidirectionality
     Conditions : A node Vi is X-source and X-sink of Vj
     Action     : Vi is bidirectional with respect to Vj.
* For Vi an in-terminal graphnode :
10. In-terminals
     Conditions : Graphnode Vj is unknown with respect
                  to Vi
     Action     : Vi is a source of Vj
```

# 4 Results

## 4.1 Experimental Results

The rules have been applied to several real-life examples, different in style and size, that have been designed within the Cathedral-II [8] environment : a 4 bit divider Execution Unit (Exu) of 300 transistors, an 8 and an 24 bit Comparator Exu (with slightly different building blocks in it) of 1500 and 4000 transistors, and a DES-chip ([7]) containing 11800 transistors. All these designs are CMOS designs, but they still offer a wide range of different problems : passgate-logic such as passtransistor-exors, registerfiles with conditional registers by means of passgates, pseudo-nmos decoders, tristate buffers, scan-paths, pla's, etc... .

The order of execution of these rules does not affect the result, but it affects of course the efficiency of the program. For that purpose, the local rules are executed before the global rules. Moreover, all 1-source and 0-source edges can be set in one single pass, and already drastically reduce the number of edges that have to be tagged by the rules. Local and global rules are executed alternatively until no further nodes can be directed. Normally the number of iterations is two or three. Without a conversion to a reduced graph, the number of iterations would of course increase. Table 1 shows the experimental results for the 4 circuits : The reduction to the reduced switch graph and the final dataflow determination of the transistors take maximally about 30 % of this total time. The tests were done on a Symbolics 3670 with 2Mword of memory. The table also shows the average number of transistors that were set per CPU-second and the remaining undirected transistors. The second part of the table focuses on the reduced switch graph. The original number of directed and undirected edges are indicated, and also the final resulting graph. The bidirectional edges in the 8-bit comparator are caused by the tristate-configuration of figure 3.

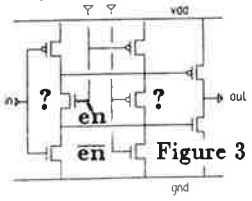

Figure 3

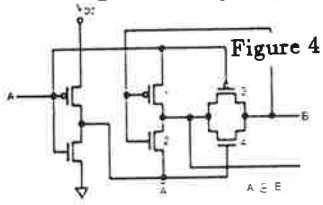

Figure 4

| Circuit | Div | Comp8 | Comp24 | DES |
|---|---|---|---|---|
| Circuit Size | 268 | 1561 | 3879 | 11884 |
| Nodes | 192 | 809 | 2034 | 4092 |
| Run time (sec) | 21 | 105 | 310 | 1180 |
| tors/sec. | 12.9 | 14.9 | 12.3 | 10.0 |
| **Start info :** | | | | |
| Graphnodes | 143 | 564 | 1303 | 3411 |
| (Vi,Vj) | 71 | 307 | 843 | 1754 |
| $< Vn, Vm >$ | 181 | 968 | 2331 | 8779 |
| **Final info :** | | | | |
| 1-source | 104 | 545 | 1362 | 3761 |
| 0-source | 77 | 423 | 969 | 5018 |
| source | 66 | 290 | 819 | 1245 |
| bidir | 0 | 17 | 0 | 1245 |
| untraceable | 5 | 0 | 48 | 536 |

Table 1: **Experimental results**

## 4.2 Comparative Results

Table 2 gives more detailed information on the rule-spectrum, and on the comparison with the Jouppi-rules for three difficult circuits : A registerfile of 4 bit with many asynchronous feedbacks, a 24-bit CMOS-version of the Mead&Conway ALU, and a 32-bit Adder with 3 passgate-exors in every bitslice. We experienced that the number of directed transistors is comparable to the number of tagged directions with the Jouppi rules, and in some designs drastically better (when omitting the doubtful rule on CMOS-inverters in [3]). In order to have a fair comparison with [3], we implemented the Jouppi-rules in DIALOG. The 'directed logic'-row indicates the number of transistors that were directed using our logical principles. The next row indicates the transistors that were set with the Jouppi-rules. The numbers between parentheses indicate the same number when the 'unsafe' Always On Invertor is used. **For the registerfile however, this resulted in 16 wrong directions !** The table also shows when the global rules are more effective than the local rules . The results are less optimal because of the complex circuitry : In add32 the passgate-exor (fig 4) is responsible for 176 (96 times 2) remaining edges. In ALU8 the difficult tagging is caused by the extensive use of passgates : almost all graphnodes are connected to four (or more) other graphnodes. In this way even the global rules do not succeed because there are too many possible ways to an output.

If electrical verification is not the major goal or if the circuit is known, there is no objection in entering an AD HOC rule in the system. By adding one specific rule we were able to direct in total more than 95 % of all transistors. This rule is a extended, higher level version of the Always On Invertor-rule. This special rule can be formulated as follows :

```
Conditions : Vertex Vi is an output of a fully static
             gate and it is unknown with
             respect to only one Vj which
             is not the output of a static gate
Action     : Vi sources Vj
```

The second number in the 'directed logic' column indicates the number of directed transistors if this rule is applied.

| Circuit | Add32 | ALU8 | Reg4 |
|---|---|---|---|
| Circuit Size | 1428 | 508 | 216 |
| Directed logic | 1111 (1426) | 344 (508) | 208 (208) |
| Directed Jouppi | 1072 (1426) | 316 (508) | 174 (216) |
| (Vi,Vj) | 292 | 96 | 23 |
| local | 113 | 0 | 3 |
| global | 0 | 60 | 16 |
| untraceable | 179 | 36 | 4 |

Table 2: **Comparison:** () **indicates use of unsafe rules**

### 4.3 Additional Heuristics

Passgate configurations often behave as a multiplexer : one or more control signals allow one path to be active, while all other passgate-paths are closed. If this is the case, local or global rules may not have the desired effect because they do not 'understand' this (knowledge of surrounding circuitry). In [3] this was solved unsatisfactory with the Complement Gate Detection Rule. We try to accommodate to this difficulty in another way : The user can pinpoint a set of control nodes (possibly input terminals), and our program will run through all possible inputcombinations of these nodes, each time adapting the graph by deleting some edges and executing the rules on the adapted graph. After each inputcombination the original graph is restored. In this way we were able to direct all transistors in shifter-like circuits.

## 5 Application Area

Partitioning the network into smaller subnetworks and analyzing each subnetwork separately is a well known technique to reduce complexity. Several verification programs divide a circuit into **DC-connected** (or channel-connected) **components** (DCN) [10]. The interaction between subnetworks is unambiguous (directed interaction graph) and complicated effects due to the bidirectionality of transistors are handled in the context of these smaller subnetworks. An example is given in figure 5.

Although many DC-connected components are small, the use of pass transistor logic can lead to very large subcircuits of a few thousand transistors. The interaction between DCN's remains unambiguous, but a DCN on itself does not reflect the designer's original gate-level building block. Wrong understanding of the circuit (e.g. in finding feedbacks) and non linear complexity result from this. Figure 5 shows a feedback between DCN's 3 and 4.
The knowledge of the dataflow allows us to split up a DCN further in Intended Unilateral Blocks (IUB's) :

**Definition 5** *For every graphnode $n_i$ :*
*Every transistor on a source-drain path from $n_i$ to any other graphnode $n_j$ that does not violate the dataflow, is in the same* **Intended Unilateral Block (IUB)**. *$n_i$ is an output of the IUB if it has a transistor of another IUB connected to it. Every output of $IUB_j$ which gates a transistor of $IUB_i$ or which is source or drain of a transistor of $IUB_i$, is an IUB-input of $IUB_i$.*

The interaction between IUB's is unambiguous for the intended (or logically meaningful) behavior (the designer intended one IUB to serve as input for another IUB). This interaction is described in the **Data Dependency Graph** : The vertices of the graph are the outputs of the IUB's or primary inputs and the directed edges indicate a relation *has-influence-on* between circuit nodes (figure 5). The new graph fully reflects the composition of the basic cells as the designer conceived them and feedbacks are intended feedbacks. The original feedback in the data interaction graph changed into two reconvergent paths between IUB Z and IUB D (fig. 5).
This suggests the use of the IUB concept for timing tools, but the study of the IUB's instead of the DCN may also be used in electrical verification tools such as [6] : it was proven in [10] that the electrical correctness approval of IUB's results in a correctness acceptance of the DCN (when using the switch model). In this way the 'granularity' of the verification problem is changed drastically, and the efficiency may improve accordingly. Table 3 compares the DCN-concept with the IUB-concept : The largest DCN (IUB) and the number of DCN's (IUB's) are shown for 3 different circuits. For our verification program, which involves path searching, generation of pullup and pulldown functions and tautolgy checking, we gained a factor of 5 in efficiency for the MCALU24-example. Other applications of our dataflow approach are the elimination of false paths in timing verifiers (e.g. shifters), and the generation of simple pullup and pulldown boolean functions ([10],[12]) which reflect the switch level behavior of output nodes.

| Circuit | Shift8 | MCALU24 | Add32 |
|---|---|---|---|
| NR DCN | 26 | 282 | 448 |
| NR IUB | 93 | 363 | 448 |
| Max DCN | 88 | 232 | 8 |
| Max IUB | 5 | 9 | 8 |

Table 3: **DCN-IUB Size Comparison**

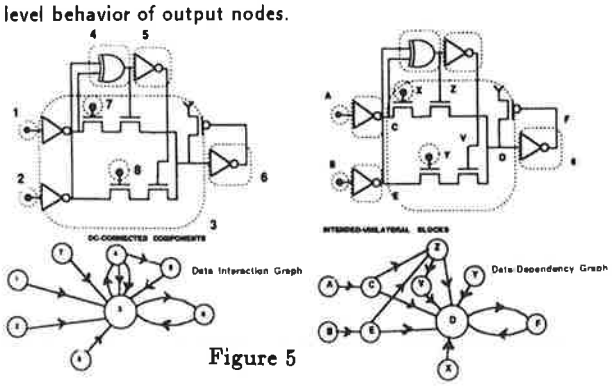


Figure 5

## 6 Conclusions

We have presented a safe method to derive the dataflow in a MOS circuit. This method is based upon design-style independent logical principles, which makes the method suitable for circuits which have not been checked on their electrical correctness. The use of two global principles has an important influence for more complicated designs but also results in an efficiency penalty which could be kept very reasonable by introducing graph reduction rules and heuristics.

The dataflow rules have been integrated in the overall DIALOG-system to perform electrical verification, and they offered an invaluable help to break up the verification problem in subproblems and to cope with feedback in designs, by introducing the concept of intended unilateral blocks.

## References


[1] N.Jouppi : "Timing Analysis and Performance Improvement for MOS VLSI Designs", IEEE Tr.on CAD, CAD-6, no 4, pp 650-665, July 1987.
[2] J.K. Oosterhout, "A Switch-level Timing Verifier for Digital MOSVLSI", IEEE Tr.on CAD, CAD-4, no 3,pp 336-349, July 1985.
[3] N.P. Jouppi : "Derivation of Signal Flow Direction in MOS VLSI" IEEE Tr.on CAD, CAD-6, no 3,pp 480-490, May 1987.
[4] M.R. Dagenais : "Timing Analysis for Mosfets : An Integrated Approach", Doctoral Thesis, VLSI Design Laboratory, Dep. of Electr. Eng, McGill Univ, June 1987.
[5] H.J De Man, I. Bolsens e.a. : "Dialog : an Expert Debugging System for MOSVLSI Design". IEEE Tr.on CAD, CAD-4, no 3, pp 303, July 1985.
[6] I.Bolsens, W. De Rammelaere e.a.: "A Formal Approach Towards Electrical Verification Of Synchronous MOS circuits" Proc. ISCAS-88, Helsinki, June 1988.
[7] I.Verbauwhede, F.Hoornaert e.a.:"Security and Performance Optimisation of a new DES data encryption chip" Special Issue of ESSCIRC'87 Conf., IEEE J.of SSCIRC, June 1988.
[8] H.De Man e.a.: "Cathedral-II: A Silicon Compiler for Digital Signal Processing", IEEE Design&Test,pp 13-25, Dec.1986.
[9] P.Odent e.a.: "Feedback loops and large subcircuits in the multiprocessor implementation of a relaxation based circuit simulation", Proc. DAC89, June 1989, pp. 25-30.
[10] I.Bolsens :"Electrical Correctness verification of MOS VLSI Digital Circuits Using Expert System and Symbolic Analysis Techniques", Doct. Thesis, Kath.Univ.Leuven, May 1989.
[11] R.Tarjan, "Depth first search and linear graph algorithms", SIAM Journ.Comput.,Vol 1, no.2,June 1972,pp.146-160.
[12] P.Johannes e.a.: "SLOCOP-II: A versatile timing system for MOSVLSI", Proc. EDAC 90, pp 518-523.
[13] D.Blaauw, D.Saab e.a.:"Derivation of Signal Flow for Switch-Level Simulation", Proc. EDAC '90, pp.301-305.