Proceedings IEEE European Design Automation Conference, EDAC-92
March 1992, pp. 388-392.

Partial Strength Ordering applied to Symbolic Switch-Level Analysis *

E. Verlind, L. Claesen, M. Genoe, F. Proesmans, H. De Man
IMEC vzw, Kapeldreef 75, B-3001 Leuven, Belgium

Abstract

This paper presents the application of partially ordered strength sets, as introduced by Agrawal et.al., within the symbolic switch-level analysis of digital MOS circuits, where until now only a straightforward approach using a total ordering was used. The paper gives a mathematical formulation for the analysis. The method has been implemented within the existing switch-level analyzer ANAMOS and has been applied successfully to practical circuits.

1 Introduction

Much of today's switch-level analysis and simulation activity is based on the switch-level model of R.E. Bryant [2]. This model captures phenomena typically found in digital MOS circuitry: bidirectional information flow, ratio effects and charge sharing. Our goal is to carry out automatic verification of circuits using the *SFG-Tracing* approach [7, 9], which means that analysis methods that operate automatically are necessary. The method described in this paper helps in automating the symbolic switch-level analysis process and thus automatic verification.

A total ordering of transistor strengths causes transistors that differ in on-resistance to be assigned the same strength level. The same can happen in the assignment of discrete node sizes derived from node capacitances. The result is that conflicting paths that differ considerably in on-resistance do not differ in strength, which leads to an unknown 'X' result where a determinate value would be appropriate. This is observed in [1] and for this, they introduce a partial ordering scheme and apply it to logic switch-level analysis. The added value of this paper is that the partial ordering approach is now applied to *symbolic* switch-level analysis, thus improving the extraction of functionality from a transistor netlist into Boolean expressions. Because of this, a wider class of circuits becomes automatically analyzable.

In the total ordering approach, a path of strength s is dominated by paths of strength larger than s. In the case of *partial* ordering, a strength level is not implicitly dominated by the next higher one and stronger. Instead of this, explicit *dominance* relations exist between the various levels. Because of this, the symbolic formulation of the analysis as presented in [6] has to be adjusted such that it takes these explicit relations into account. For keeping the number of strength levels small for efficiency reasons, procedures from [1] are implemented in our system as a preprocessing step. Because of the increased accuracy as compared to the original approach, the application of the partial ordering increases CPU-time consumption as compared to total ordering, although the increase is rather acceptable. Practical figures will be given.

Section 2 describes the concept of partially ordered strengths in the context of switch-level analysis. Section 3 deals with the application of partially ordered strengths in the symbolic switch-level analysis software. Section 4 discusses some experimental results of analysis performed on designs generated using a silicon compiler.

2 Partially Ordered Strength Sets

In Bryant's switch-level approach, a first abstraction in modeling is the view of the circuit as a network of switched linear resistors and linear node capacitances. Starting from this view, a switch-level model consisting of transistors characterized by their strengths and storage nodes characterized by their sizes is derived. This model is capable of representing the functionality of most digital MOS circuits. The most important aspect then is the method of strength assignment and the relations between these strengths. The straightforward method as introduced by [5, 6] uses a total ordering on the ordered lists of node capacitances and transistor on-resistances. It can lead to an inaccurate assignment such that too much information about the functionality is lost and the results are of no use.

Furthermore, the analysis as proposed by Bryant [6] uses an approximation in which for instance the strength of a path of conducting transistors in series is equal to the strength of the weakest transistor and in which the strength of parallel conducting transistors is equal to the strength of the strongest one. This is called the *order of magnitude* approach, as the approximation is correct only if transistors and nodes differ significantly in size.

So there are basicly two problems with the modeling and analysis approach as used by now: the problem of too coarse strength and size partitioning leading to overly conservative results and the problem of inaccuracy resulting from the order of magnitude approach. The approach of Agrawal et.al. [1] is an attempt to surmount some of the difficulties in automatic switch-level modeling. Instead of using a total ordering of the strengths on which a 'greater than' operator is defined, they define a *dominance* relation. Since this dominance relation is transitive, antisymmetric and irreflexive, it is a *(strict) partial ordering* on the set

*Research sponsored by CHARME ESPRIT BRA 3216

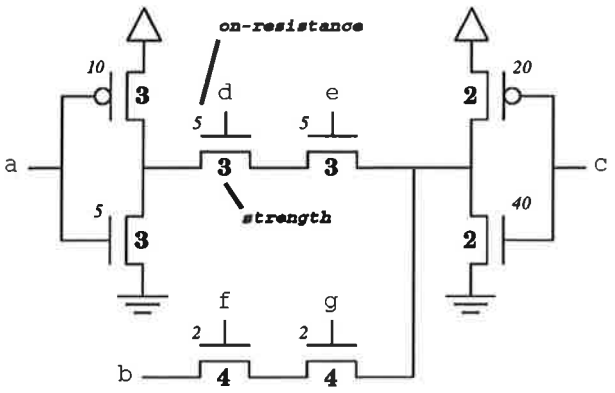

Figure 1: Example circuit

	1	2	3	4
1	X	1	1	1
2	0	X	1	1
3	0	0	X	X
4	0	0	0	X

Table 1: Dominance matrix

of modeling strengths. The structure of the circuit is explored in order to determine which paths can come into conflict with each other. The procedure followed for obtaining the set of modeling strengths is a heuristical one. This means that an upper bound is found and with this all possible conflicts are found. However, there may be less conflicts in reality, which sometimes means a somewhat larger strength set than strictly necessary. Nevertheless, it seems not to be a real problem. An example: given a resistor separation ratio of 3, figure 1 depicts a circuit with its transistors showing both their on-resistances and assigned strengths. Table 1 shows the manually derived dominance matrix. The 5 ohms transistors and the 10 ohms transistor have both been assigned strength value 3, while the 2 ohms transistors have been assigned strength value 4. However, the (4, 3) and the (3, 4) entries differ, reflecting the different result at node n in the case that $a = 0, d = e = f = g = 1, b = 0$ (result is 0) compared with the case that $a = 1, d = e = f = g = 1, b = 1$ (result is indeterminate).

3 Partial Ordering applied to Symbolic Switch-Level Analysis

The approach of [1] makes it possible to do more of the switch-level analysis in an automatic fashion. For the verification system that is currently worked out within the framework of the *SFG-Tracing* verification methodology [7], the COSMOS software [3, 4, 5, 6] is used for performing symbolic simulation at the switch-level. The program used for performing the symbolic switch-level analysis proper is part of the COSMOS software and is called ANAMOS. We have now implemented a new version of the highest level of ANAMOS, which inherits a strength assignment and a dominance relation for each CCC from a preprocessing step, which is an implementation of the algorithms proposed by [1]. A way of expressing the dominance relation is by means of the *dominance matrix* as discussed in the previous section. In our formulation of the symbolic analysis, however, this dominance information is contained in the *ding1*, *ding0*, *dted1* and *dted0* relations.

The function *ding0*($s1$) has as its argument a strength $s1$ that represents a strength of a polarity 1 signal. The result of the function is a polarity 0 signal strength such that in case of a conflict between a signal of polarity 1 having strength $s1$ and a signal of polarity 0 having strength $s0 \geq$ *ding0*($s1$) the resulting signal level is 0, while a conflict with a signal of polarity 0 having strength $s0 <$ *ding0*($s1$) gives either 1 or X (but not a definite 0) as a result. So *ding0*($s1$) is the weakest polarity 0 strength dominating polarity 1 signals of strength $s1$. If there is no transistor level nor node level that dominates the polarity 1 signal of strength $s1$, *ding0*($s1$) yields input node strength ω. Define the function *ding1*($s0$) in an analogous way.

The function *dted0*($s1$) also has as its argument a polarity 1 strength $s1$. Its result is a polarity 0 signal strength such that in case of a conflict between a signal of polarity 1 having strength $s1$ and a signal of polarity 0 having strength $s0 \leq$ *dted0*($s1$), the resulting signal level is 1, while a conflict with a signal of polarity 0 having strength $s0 >$ *dted0*($s1$) gives either 1 or X as a result. So *dted0*($s1$) is the strongest polarity 0 strength dominated by polarity 0 signals of strength $s1$. In case there is no transistor level nor node level which is dominated by the polarity 1 signal of strength $s1$, *dted0*($s1$) yields strength λ. Define the function *dted1*($s0$) in an analogous way.

In the original total ordering approach, the functions *ding1*($s0$), *ding0*($s1$), *dted1*($s0$) and *dted0*($s1$) are implicitly there and could be made explicit, with *ding1*(s) = *ding0*(s) yielding $s + 1$ for the levels under the highest level under the input node strengths, for which the functions yield ω. In the total ordering approach the function *dted1*(s) = *dted0*(s) and yields s - 1 for $s > 0$ while for $s = 0$ the result is λ.

For the dominance matrix of table 1 the functions are defined as follows: *ding1*($s0$) = $s0 + 1$ for $s0 < 3$ and ω for $s0 \geq 3$, *dted0*($s1$) = 2 for $s1 = 4$, $s1$ - 1 for $s1 = 3$ or 2 and λ for $s1 = 1$, *ding0*($s1$) = $s1 + 1$ for $s1 < 4$ and ω for $s1 = 4$, *dted1*($s0$) = $s0$ - 1 for $s0 > 1$ and λ for $s0 = 1$.

The formulation below closely follows that of [6]. The path relation $\mathcal{P}1_s$ is defined as m $\mathcal{P}1_s$ n when there exists an unblocked polarity 1 path of strength greater than or equal to s from m to n. Similarly, $\mathcal{P}0_s$ is defined as m $\mathcal{P}0_s$ n when there exists an unblocked polarity 0 path of strength greater than or equal to s from m to n. Like in [6] a discussion of the form of an unblocked polarity 1 or 0 path from node m to node n can be held. For a polarity 1 path, the following can be said: it can be an unblocked path of strength greater than s, in which case $\mathcal{P}1_{s+1}$ holds. The other possibility is that the path is of strength s. If s is a transistorstrength we are dealing with a driving path, in which case node m must be an input node of polarity 1 connected to a path consisting of a (possibly empty) sequence of transistors of strength greater than s to a node l, followed by a transistor having

strength s, followed by a (possibly empty) sequence of transistors with strength greater than or equal to s reaching node n. The part from node m to node l can not be blocked and $\mathcal{P}1_{s+1}$ holds. No node other than l in the remaining part from node l to node n can be the destination of a definite path of polarity 0 having a strength that dominates s. If s is a node size, we are considering a charging path, in which case storage node m has size s and the path from node m to node n is formed by a sequence of transistors such that no node is the destination of a definite polarity 0 path having a strength that dominates s. Using a few additional predicates and relations we will give a formal definition of $\mathcal{P}1_s$. An analog discussion is possible for $\mathcal{P}0_s$.

For each node n, define the predicate $\mathcal{C}1_s$(n). This predicate expresses the condition that the node is *clear* for polarity 1, which means that the predicate is holding when paths of polarity 1, having strength s or lower, are still able to influence the result.

Define the relation $\mathcal{Q}1_s$ as m $\mathcal{Q}1_s$ n when the following relations hold:

- There is a path p in the network with $Root(p)$ = m and $Dest(p)$ = n consisting only of transistors with state 1 or X and strength greater than or equal to s.
- $\mathcal{C}1_{ding0(s)}$(l) holds for every node l in p other than m.

1

$$\mathcal{P}1_s = \mathcal{P}1_{s+1} \cup \{(m,n) \mid \exists l,\ m\,\mathcal{P}1_{s+1}\,l \text{ and } l\,\mathcal{Q}1_s\,n\} \cup \{(m,n) \mid m\,\mathcal{Q}1_s\,n,\ \mathcal{C}1_{ding0(s)}(m) \text{ and } m \in \mathcal{N}_s\}$$

Like in [6], we define:

2

$$N.1_s = \bigvee_{m\,\mathcal{P}1_s\,n} m.1$$

Substitution for $\mathcal{P}1_s$ (1) in definition 2 gives the result of (3)

3

$$N.1_s = \bigvee_{m\,\mathcal{P}1_{s+1}\,n} \vee \bigvee_{m\,\mathcal{P}1_{s+1}\,l\,l\,\mathcal{Q}1_s\,n} m.1 \vee \bigvee_{m\,\mathcal{Q}1_s\,n} \mathcal{C}1_{ding0(s)}(m) \wedge (m \in \mathcal{N}_s) \wedge m.1$$

After a few rewritings, the relation of (4) results.

4

$$N1_s = N.1_{s+1} \vee \bigvee_{m\,\mathcal{Q}1_s\,n} \left(M.1_{s+1} \vee [\mathcal{C}1_{ding0(s)}(m) \wedge (m \in \mathcal{N}_s) \wedge m.0]\right)$$

Until now, we have made a distinction between $C1$ and $C0$ values. However, it is not always necessary to have separate values. In fact, it is only necessary to have a polarity-related *clear* value if there is a domination relation within one strength level, which is the case if for a certain strength level s, either $ding0(s) = s$ or $ding1(s) = s$. This situation can be caused by optimization procedures that reduce the number of strengths, such as those from [1]. If at a certain strength level s, there is a dominance of paths of polarity 1 over paths of polarity 0, it must not be the case that the *clear* value used at polarity 0 is set false beforehand due to polarity 0 paths of strength s. For this, only paths of polarity 1 may be taken into account. An analog case is the dominance of polarity 0 paths over polarity 1 paths within one strength level.

If, however, there does not exist a dominance relation within one strength level, there is no reason to do calculations of the *clear* value for both polarities separately. If at strength level s, the (combined) *clear* value becomes true due to polarity 1 paths, possible paths of the same polarity at lower strength levels are blocked. If the *clear* value would not have been set to false for polarity 1, these lower-strength paths would be dealt with in the calculations, which would not change the results as they are redundant with respect to the higher-strength paths that could have set the *clear* value to false. An analog situation exists when dealing with polarity 0. As these lower-strength paths play no role at all, it is conceptually more natural to block them beforehand, which is achieved by using a combined *clear* value. This means that all the $C1$ and $C0$ values in formulas (3) and (4) must be replaced by C values in case of a dependency between different strength levels. The result of this is that we can then also define $\mathcal{Q}_s$ and $\mathcal{P}_s$ based on the $\mathcal{C}_s$ predicate. The difference with [6] lays in the explicit reference to the next higher dominating value by the *ding1* and *ding0* relations.

The result of the preprocessing procedure of [1] is a dominance matrix denoting the ordering between all strength levels. From this matrix, dominance vectors are derived which are then used within symbolic analysis.

Having the above relations, we can move to a formulation in terms of Boolean equations. In this formulation, we will use the same notation as in [6].

For all strength levels s, with $\omega > s \geq 1$ the analyzer sets up and solves systems of Boolean equations. For the case without intra strength level dominance, the system of Boolean equations $[Conduct_s, init1_s]$ as well as the system $[Conduct_s, init0_s]$ is solved. If necessary (see later), after this, the system $[Indef_s, initc_s]^D$ is solved. In case of intra strength level dominance, one more calculation is carried out. In the case that $ding1(s) = s$, first the calculations for the dominat*ing* polarity 1 are done, which means that the system $[Conduct1_s, init1_s]$ is solved, followed by the solution of $[Indef_s, initc0_s]^D$. Following this, the calculations for the dominat*ed* polarity 0 are performed, which means that first the system $[Conduct_s, init0_s]$ is solved followed (possibly) by the solution of $[Indef_s, initc_s]^D$.

Define for a transistor t the predicates *potential*(t)

type	*indefinite(t)*	*potential(t)*
n-type	*n.0*	*n.1*
p-type	*n.1*	*n.0*
d-type	0	1

Table 2: Transistor-Related Formulas

and *indefinite(t)*, as depicted in table 2. The initial labelings *init1*$_s$ and *init0*$_s$ used with partial ordering are grossly the same as those used in [6]. For *init1*$_s$ we obtain the assignment of (5).

5

$$init1_s(\mathrm{n}) = \begin{cases} N.1_{s+1} \vee \left(N.c_{ding0(s)} \wedge \bigvee_{\mathrm{m}\in\mathcal{N}_\omega} \bigvee_{t\in\mathcal{T}_s(\mathrm{m},\mathrm{n})} [potential(t) \wedge m.1]\right) & s > k \\ N.1_{s+1} \vee \left(N.c_{ding0(s)} \wedge n.1\right) & n \in \mathcal{N}_s \\ N.1_{s+1} & \text{else} \end{cases}$$

For the initial vertex labeling *initc*$_s$ used in calculation of the *clear* predicate, things have changed. The $N.c_s$ value is calculated at a strength level s for which there is no intra strength level dominance, so when $ding1(s) \neq s$ and $ding0(s) \neq s$ and $\exists\, s'$: $(ding1(s') = s$ and $ding0(s') = s)$. The $N.c1_s$ value is calculated at a strength level s for which holds that $ding0(s) = s$, while the $N.c0_s$ value is calculated at level s if $ding1(s) = s$. During analysis we keep one *clear* field per strength level for each storage node, which can be referred to at lower levels or at the same level immediately after calculating it, in case of intra strength level dominance. This *clear* field contains the $N.c_s$, $N.c1_s$ or $N.c0_s$ information. The notation for the *clear* information at node n at strength s is $N.clear_s$. With calculations performed from high to low in terms of the strength level we have the assignment of (6), which calculates $N.c_s$ (and assigns $N.clear_s$) using already calculated *clear* fields, under the conditions mentioned for calculating this value.

6

$$initc_s(\mathrm{n}) = \begin{cases} N.clear_{nexthi(s)} \wedge \bigwedge_{\mathrm{m}\in\mathcal{N}_\omega} \bigwedge_{t\in\mathcal{TU}} indefinite(t) & \\ \quad \text{with } \mathcal{TU} = \bigcup_{i=s}^{nexthi1(s)-1} \mathcal{T}_i(\mathrm{m},\mathrm{n}) & s > k \\ 0 & n \in \mathcal{N}_s \\ N.clear_{nexthi(s)} & else \end{cases}$$

In case of calculating $N.c1_s$ for the case where $ding1(s) = s$ holds (the *clear* calculation for the dominating polarity in case of intra strength level dominance), the formula of (7) is used.

7

$$initc1_s(\mathrm{n}) = \begin{cases} N.clear_{nexthi(s)} \wedge \bigwedge_{\mathrm{m}\in\mathcal{N}_\omega} (m.0 \vee \bigwedge_{t\in\mathcal{TU}} indefinite(t)) & \\ \quad \text{with } \mathcal{TU} = \bigcup_{i=s}^{nexthi1(s)-1} \mathcal{T}_i(\mathrm{m},\mathrm{n}) & s > k \\ 0 & n \in \mathcal{N}_s \\ N.clear_{nexthi(s)} & else \end{cases}$$

In (7), *nexthi*(s) represents the next-higher strength level s' for which holds that $s' = \max\{ ding1(dted0(s)), ding0(dted1(s))\}$. The *Conduct*$_s$ value as found in [6] forms the initial edge labeling for the Gaussian elimination step; the partial ordering approach contains the *Conduct1*$_s$ and *Conduct0*$_s$ labelings. Equation (8) gives the formula for *Conduct1*$_s$.

8

$$Conduct1_s(\mathrm{m},\mathrm{n}) = N.clear_{ding0(s)} \wedge \left[Conduct1_{s+1}(\mathrm{m},\mathrm{n}) \vee \bigvee_{t\in\mathcal{T}_s(\mathrm{m},\mathrm{n})} potential(t)\right]$$

The formula for the edge labeling *Indef*$_s$ used with the calculation of the *clear* values is given by (9).

9

$$Indef_s(\mathrm{m},\mathrm{n}) = Indef_{s+1}(\mathrm{m},\mathrm{n}) \wedge \bigwedge_{t\in\mathcal{T}_s(\mathrm{m},\mathrm{n})} indefinite(t)$$

4 Experimental Results

As partial ordering of strength sets allows to deal with more complicated situations than total ordering does, more information is preserved and processed. For this, it is likely that the implementation of the analyzer which operates with a partially ordered strength set takes more time than the original one. We have done a few experiments, of which the results are listed in table 3. The *aplusb-mge* design is a simple processor design generated by Cathedral-II [8] using the module generator *MGE*, which contains all basic characteristics of more elaborate designs. The *rec3-sc* design is a modem chip and was generated with standard cells. Both examples were analyzed by the original analyzer as well as by the one that we adjusted to operate with partially ordered strengths. After analysis, a verification script was run on both examples by COSMOS. The results show that using partial ordering leads to a higher number of unique CCCs being distinguished as compared to using total ordering. The CPU-time used by the original program is higher than the time used by the adjusted one, although the difference is not that large for the more elaborate example.

5 Conclusions

Symbolic switch-level analysis based on the switch-level model of R.E. Bryant can play an important role

as a low-level starting point for automatic formal verification using the methodology of *SFG-Tracing*. It is well-suited for use in the *Cathedral* environment where it bridges the gap between a netlist extracted from a layout as generated by the synthesis system and a functional description of the implementation's behavior. The papers [5, 6] describe algorithms for symbolic switch-level analysis, which have been implemented at Carnegie Mellon University in the ANAMOS program which forms part of the COSMOS system. The deficiency in the described method is the actual modeling in terms of strength sets, which is based on a straightforward ordering of transistor on-resistances and node capacitances. From these ordered values, a set of strength levels is derived, with a total ordering defined on it. Because of this, potentially much information is lost because elements of which the on-resistances resp. the capacitances differ considerably are still grouped in one strength level. This leads to an indeterminate result in case of a conflict: no information.

Partial ordering of strength levels extends the class of circuits that can be analyzed automatically by a (symbolic) switch-level analyzer, as more information is extracted and kept track of. Results show that symbolic analysis using a partial ordering leads to an increase in effort in terms of CPU-time which remains within acceptable bounds as compared to the original situation.

Despite the improvement due to partial ordering, it is certainly not possible at this moment to extract the functionality of every circuit generated by the module generator *MGE* nor of manual designs. First, circuits exist which can not be abstracted to a network consisting of switchable on-resistances and capacitances, which is the starting point for the analyzer. For example, certain designs are done based on current equations, which could lead to small ratios between transistors. Such circuits can be analyzed and shown to behave correctly by a circuit simulator, but not by tools using less accurate models of circuit reality. Second, problems can arise even in the case that the analyzer's starting point is right. This is caused by the fact that the analyzer assumes that an *order of magnitude* model can be used, which may lead to overly indeterminate and even to erroneous results.

For these reasons, it will not be possible to automate the verification completely if there are no provisions for dealing with circuits that can not be modeled by the methods used now. A solution to the problem of getting too indeterminate results would be to extend the verification method with a controlling procedure which refines the used modeling in case inaccurate results are found. So, suppose first the use of an efficient, less accurate modeling procedure yielding a certain implementation signal to be at X while the specification requires a determinate value. We could then re-evaluate using more accurate models until we can determine whether the X really means that the implementation produces a indeterminate result (error in the design) or that it was just the inaccuracy of the original description level which was responsible for the X.

The validation of the application of the switch-level model as well as verification based on analysis at different levels of accuracy will be the subject of future research.

	aplusb-mge		*rec3-sc*	
	tot. o.	**part. o.**	**tot. o.**	**part. o.**
Analysis				
#un.CCCs:	42	67	36	43 Mb
mem. usage:	0.6Mb	1.1Mb	9.7Mb	15.4 Mb
CPU-time:	5s	11s	129s	133s
Simulation				
mem. usage:	0.3Mb	0.3Mb	9.2Mb	8.2Mb
CPU-time:	2.8s	4.5s	232s	261s

Table 3: Experimental Results

References

[1] P. Agrawal, "Automatic Modeling of Switch-Level Networks Using Partial Orders", *IEEE Transactions on Computer-Aided Design*, volume 9, No. 7, pp. 696-707, July 1990.

[2] R.E. Bryant, "An Algorithm for MOS Logic Simulation", *Lambda*, fourth quarter 1980, pp. 46-53, 1980.

[3] R.E. Bryant, "A Switch-Level Model and Simulator for MOS Digital Systems", *IEEE Transactions on Computers*, volume C-33, No. 2, pp. 160-177, February 1984.

[4] R.E. Bryant, "Symbolic Verification of MOS Circuits", *1985 proc. Chapel Hill Conference on VLSI*, ed. H. Fuchs, pp. 419-438, 1985.

[5] R.E. Bryant, "Algorithmic Aspects of Symbolic Switch Network Analysis", *IEEE Transactions on Computer-Aided Design*, volume CAD-6, No. 4, pp. 618-633, July 1987.

[6] R.E. Bryant, "Boolean Analysis of MOS Circuits", *IEEE Transactions on Computer-Aided Design*, volume CAD-6, No. 4, pp. 634-649, July 1987.

[7] L. Claesen, F. Proesmans, E. Verlind, H. De Man, "*SFG-Tracing*: a Methodology for the Automatic Verification of MOS Transistor Level Implementations from High Level Behavioral Specifications", *Proc. International Workshop on Formal Methods in VLSI Design*, ed. P.A. Subrahmanyam, January 1991.

[8] H. De Man, J. Rabaey, P. Six, L. Claesen, "Cathedral-II: A silicon compiler for digital signal processing", *IEEE Design & Test of Computers*, volume 3, No. 6, pp. 73-85, December 1986.

[9] M. Genoe, L. Claesen, E. Verlind, F. Proesmans, H. De Man, "Automatic Formal Verification of Cathedral-II Circuits from Transistor Switch-Level Implementation up to High Level Behavioral Specification by the SFG-Tracing Methodology", *Proc. IEEE European Design Automation Conference*, March 1992.