Illustration of the SFG-Tracing multi-level behavioral verification methodology, by the correctness proof of a high to low level synthesis application in CATHEDRAL-II. *

Mark Genoe, Luc Claesen † Eric Verlind, Frank Proesmans, Hugo De Man †
IMEC, Kapeldreef 75, B-3001 Leuven Belgium

Abstract.

The SFG-Tracing methodology [5] addresses the automatic verification of digital synchronous circuit implementations as specified at the algorithmic level as signal (SFG) or data flow graphs. The SFG-Tracing methodology is a multi-level design verification paradigm that aims at bridging the gap between higher level specifications down to lower level implementations up to the transistor switch level. In this paper the concepts of the SFG-Tracing methodology are illustrated by the automatic verification of a transistor level implementation of a small chip generated from its high level specification by the CATHEDRAL-II silicon compiler. This application, although simple, includes a datapath, register files, a multi-branch micro coded controller, and additional circuitry as necessary for Design for Testability measurements. This application illustrates the SFG-Tracing verification methodology as applied to one member of a partitioned SFG behavioral specification. Experimental results on more complex, completely verified designs of 32000 transistors demonstrate the feasibility of the approach.

1 Introduction.

The goal of formal behavioral correctness verification in digital circuit design is to be able to guarantee design correctness as good as possible and trying to avoid costly design and processing iterations that can result in late introduction into the market of new products. Although simulation is the major CAD approach that is currently industrial available for design verification, it still leaves open the possibilities of undetected design errors. This is caused by the limited amount of simulation stimuli that can be used for checking the correctness.

The most promising approaches currently available are based on the concept of ordered binary decision diagrams (OBDD's) [1] for Boolean function representation. These can be applied very well for both practical combinatorial logic [2] and finite state machine verification [3, 4]. In the combinatorial verification, it is essential that the same variables are available at the input and the output of the combinatorial functions. Although impressive results have been reported on, the verification of FSM's is still limited in complexity to some 60 state variables [4], which is much less than available in actual hardware systems.

In [10, 12] Floyd's [11] method of software verification by inductive assertions has been explored for hardware verification. More recently [13], a similar method based on the formulation of pre- and post-conditions for pipelined hardware has been presented. The problem is however to formulate the appropriate pre- and post conditions necessary for these methods.

In [5] the SFG-Tracing methodology is presented. This methodology makes use of the available information in the high level specification of a digital system in order to perform the verification. In this paper the feasibility of the SFG-Tracing verification methodology is illustrated by the formal verification of an example called aplusb-n from its high level specification down to the transistor circuits which has been automatically synthesized by the CATHEDRAL-II silicon compiler. In [14] an alternative method for verification of high level synthesis has been presented, which is based on backannotating the specification with clock statements according to the schedule. This methodology still needs to be extended by the verification of the correctness of the schedule.

In section 2, the SFG-Tracing methodology is summarized. Hereafter a short overview of the CATHEDRAL-II system will be given in section 3. In section 4 the verification of the aplusb-n system will be described. Section 5 presents the verification results on several more complicated circuits.

2 SFG-Tracing Verification Methodology

The underlying principles and details of the SFG-Tracing verification methodology are described in [5]. A fully worked out and illustrative small example of a BCD-recognizer is explained in [15]. Here we only summarize the key aspects of the methodology. The SFG-Tracing methodology aims at bridging the gap from transistor switch level circuits, as obtained from circuit extraction, up to high level specifications. It inherits its power from the exploitation of the inherent algorithmic information in the high level (signal flow graph (SFG) level) specifications. Usually the high level specifications are much less detailed in terms of their functional and time domain behavior in comparison to the possible implementations (bit-serial, microprogrammed, bit-parallel ...).

At an algorithmic (SFG) level only the relationships among
algorithmic variables is expressed. The specific implementations of detailed behaviors in hardware and in time is left open. This reduced information content in the specification is used as a point of reference for checking if this specified behavior is realized in a correct way in the implementation.

The verification is done by a systematic symbolic tracing of all partitions of an algorithmic specification. Given the fact that the circuit designer (or the automatic synthesis tools) provides the appropriate reference signals and mapping functions, the methodology is intended to operate automatically on VLSI circuits of 100,000 transistors and more.

Although the SFG-Tracking verification methodology is generally applicable, the first application target in IMEC is the verification of high level synthesis results as obtained by the CATHEDRAL silicon compilers [6]. The underlying assumption is that the flow graph specification is synthesized while keeping track of mapping functions of a set of well chosen reference signals of the specifying flow graph and of the implementation (in space and in time). The global verification problem is reduced to a manageable size by partitioning the information in the global specification flow graph into appropriate acyclic subgraphs and providing correspondence (mapping) functions between the interface values (reference signals) in the partitioned graph and the signal values at specific cycle and clock phase times in the implementation. The correctness of each individual subgraph is proven by making use of (switch-level) symbolic simulation using COSMOS [9], to extract the semantics of the implementation. Efficient OBDDs [2] are used to check the individual verification conditions [5] for each of the individual subgraphs of the specification.

The SFG-Tracking methodology relates the algorithmic signals in the specification to their occurrence in space and in time in the implementation. Other signals and encodings due to the design implementation decisions (micro-code ROM, controller, sequencer etc.) are abstracted away, and implicitly verified by performing a symbolic simulation.

3 CATHEDRAL-II Synthesis System.

The CATHEDRAL-II silicon compiler [6, 7] addresses complex DSP-applications with sample rates from a few kHz to 1 MHz.

CATHEDRAL-II synthesizes multi-processor architectures, starting from a behavioral description in SILAGE. Customized data paths are generated, composed of predefined parameterized modules that are stored in a library. These modules communicate via dedicated bus interconnections. The data path is controlled by a powerful micro-coded controller, allowing for fast decision making and branching (e.g. loop constructs).

The CATHEDRAL-II environment allows to generate and compare several alternative designs in a short time [7], and finally produces layouts.

4 The aplusb-m Application Example.

This aplusb-m example is mainly used as an illustration of the SFG-Tracking methodology. It is an application that adds 2 incoming 8-bit numbers. (It is not the intention that someone would use CATHEDRAL-II to synthesize a micro coded architecture for such a trivial application). Although this application example is rather small - it consists of only 1935 transistors - and the specification consists of only one partition, the example has several ingredients of the more complicated VLSI design applications such as data

![Figure 1: Allocation of datapath and bus-interconnection structure according to requirements in SFG operations and dependencies.](image1)

![Figure 2: Architecture of the aplusb control logic, as synthesized by CATHEDRAL-II.](image2)
The scan clock signal is set high, these scan paths will send serially all the information which are stored into the registers to the outside. In this way the values in these registers can be controlled and observed.

4.3 Applying the SFG-Tracing methodology.

Reference signals, mapping functions and subcircuit partitioning: The aplusb-algorithm is a very small one, and contains only one single operation (addition on ALU). So it is not necessary to further partition this SFG in subgraphs. The reference signals are in this case respectively the data input- and output signals: \( a[0..7], b[0..7], \) and \( out[0..7] \).

The mapping functions, which give the relations between the specifications and the implementation for each reference signal, have all to be defined separately for a specific cycle in time and space. As discussed before, we know already that the first seven cycles are spent for initialization, so we start at cycle eight:

\[
\begin{align*}
&b[0..7] = \text{inpad}_{\text{reset}}[0..7] \quad \text{(cycle 8)}, \\
&a[0..7] = \text{inpad}_{\text{start}}[0..7] \quad \text{(cycle 9)}, \\
&\text{out}[0..7] = \text{outpad}_{\text{reset}}[0..7] \quad \text{(cycle 10)}.
\end{align*}
\]

With defining these reference signals and mapping functions, it is possible to start the verification of this partition by symbolic simulation. In the symbolic simulation of the aplusb-circuit, the reference signals such as \( a[0..7], b[0..7] \) and \( \text{out}[0..7] \) are represented symbolically. Other signals such as external clock signals, scan clock signals, reset and initialization signals will be set to their specific Boolean values in each cycle and clock phase. By doing this, the symbolic simulation will result in Boolean values \( (1.0) \) for the control logic, and symbolic variables \( (a_0, a_1, \ldots, b_0, b_1, \ldots) \) or expressions thereof for the rest of the circuit.

Concerning the initialization, it is necessary to check if the circuit will attain its steady state, and that it will stay in this state without changing the status input signals \( (\text{inpad}_{\text{reset}}, \text{inpad}_{\text{start}}, \text{inpad}_{\text{init}}) \). This can be verified by logic simulation of the controller circuitry after the initialization sequence. Before starting this logic simulation, all nodes of the circuit will be set to an undefined 'X' state.

In steady state verification, the COSMOS program \( [9] \) will derive Boolean expressions for the output signals as functions of the Boolean variables of the input signals. These expressions have to be equivalent to the desired behavior.

The verification of the aplusb-circuit is achieved by command scripts for the COSMOS \( [9] \) symbolic switch-level simulator that are generated from the high level specifications given the reference signals and mapping functions. A manual example of this is illustrated in \( [15] \).

5 Practical Results.

The SFG-Tracing verification methodology has been applied successfully to the verification from the layout extracted transistor netlist in comparison with the high level specifications for a number of designs as synthesized by CATHEDRAL-II as summarized in table 2.

Design aplusb-m is the small application described in section 4 of which the generated layout is shown in fig. 3. count-s is an application that has while-loops and conditional statements in its specification. rec3-s is a modem chip \( [7] \) implemented with 3 ALU's in the datapaths. Its layout is shown in fig. 4. The cpu times (DEC 3100) include

[Table 1: Timing diagram for aplusb-circuit: initialization state \((1..7)\) and steady state \((8..10) \) etc.]

<table>
<thead>
<tr>
<th>cycle nr.</th>
<th>prog</th>
<th>cntn</th>
<th>inpad reset</th>
<th>inpad start</th>
<th>outpad</th>
<th>[0..7]</th>
<th>\ [+B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>xxx</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>000</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>001</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>010</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>010</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>010</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>011</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>b</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>101</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>011</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>A+B</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3: Chip layout of the aplusb-circuit, as synthesized in Cathedral-II.
ing the COSMOS system available, as well as Veerle Derdu-

tomatically proving the correctness of the synthesis results
for manual designs, in the assumption that an appropriate
of synthesis results from a high level specification down to
representative circuits as available in micro coded architec-
tures: data path, register files, controllers, etc. They illus-
transistor implementations are indicated in
The applications contain

Tracing


directly with the corresponding number
of states as
counted e.g. in [4].

6 Conclusions.

In this paper the SFG- Tracing verification methodology has
been illustrated by concrete design examples as generated by
CATHEDRAL-II that have been behaviorally verified from
the high level specification down to the layout extracted
transistor level implementations. The applications contain
representative circuits as available in micro coded architectures:
data path, register files, controllers, etc. They illustrate
the feasibility of the SFG- Tracing methodology. They also illustrate the feasibility for the automatic verification of
synthesis results from a high level specification down to
the implementations. The methodology can also be used
for manual designs, in the assumption that an appropriate
high level behavioral specification is available. The SFG-
Tracing methodology is currently being worked out for
automatically proving the correctness of the synthesis results
in CATHEDRAL-I and CATHEDRAL-II [8].

The authors thank Randy Bryant and his team, for making the
COSMOS system available, as well as Veerle Derud-
der and all other colleagues in IMEC for the provision of the
CATHEDRAL designs which have enabled to perform the
research mentioned in this paper.

References

[1] R.E. Bryant, "Graph Based Algorithms for Boolean
Function Manipulation", IEEE Trans. on Computers,
Implementation of a BDD Package", Proc. 27th
of Sequential Machines Using Functional Vectors",
in "Formal VLSI Correctness Verification", North-
quential Circuit Verification Using Symbolic Model
"SFG- Tracing: a Methodology for the Automatic Ver-
fication of MOS Transistor Level Implementations
from High Level Behavioral Specifications", Proc.
ACM/SIGDA Int. Workshop on Formal Methods in
VLSI Design, ed. P.A. Subrahmanyam, Miami, Jan-
[6] H.De Man, J.Rabaey, P.Six, L.Claesen, "Cathedral-II:
A silicon compiler for digital signal processing", IEEE
Man, "Evaluation of high-level design decisions us-
ing the Cathedral-II silicon compiler to prototype a
DSP ASIC", Proc. IFIP Workshop on High Level
Behavior for Hardware Description", IEEE Tr. Comp.,
using Symbolic Logic Simulation", ICCD-89, pp. 217-
221.
Payer, "Verification of Synchronous Sequential Cir-
cuits Obtained from Algorithmic Specifications", Proc.
ACM/SIGDA Int. Workshop on Formal Meth-
De Man, "Application Example of multi-level digital
design verification by the SFG- Tracing Methodology",
384.

<table>
<thead>
<tr>
<th>design</th>
<th>aplusb -m</th>
<th>aplusb -s</th>
<th>count -s</th>
<th>rec3 -s</th>
</tr>
</thead>
<tbody>
<tr>
<td># MOS trans</td>
<td>1935</td>
<td>3592</td>
<td>7108</td>
<td>31614</td>
</tr>
<tr>
<td># latches</td>
<td>112</td>
<td>112</td>
<td>230</td>
<td>852</td>
</tr>
<tr>
<td># states</td>
<td>10^14</td>
<td>10^14</td>
<td>10^15</td>
<td>10^15</td>
</tr>
<tr>
<td># subnetworks</td>
<td>362</td>
<td>312</td>
<td>1719</td>
<td>7698</td>
</tr>
<tr>
<td># uniq. subn.</td>
<td>38</td>
<td>21</td>
<td>32</td>
<td>36</td>
</tr>
<tr>
<td># memory</td>
<td>513K</td>
<td>430K</td>
<td>700K</td>
<td>2M</td>
</tr>
<tr>
<td># mach. cycles</td>
<td>3</td>
<td>3</td>
<td>503</td>
<td>19</td>
</tr>
<tr>
<td># sim. cycles</td>
<td>10</td>
<td>10</td>
<td>20</td>
<td>29</td>
</tr>
<tr>
<td># cpu time</td>
<td>11.5 s</td>
<td>12 s</td>
<td>22 s</td>
<td>567 s</td>
</tr>
<tr>
<td># script lines</td>
<td>105</td>
<td>105</td>
<td>640</td>
<td>4782</td>
</tr>
<tr>
<td># partitions</td>
<td>1</td>
<td>1</td>
<td>14</td>
<td>68</td>
</tr>
</tbody>
</table>

Table 2: Results of verification of transistor implementation
with respect to high level specification for a number of de-
signs synthesized by CATHEDRAL-II on a DEC 3100

Figure 4: Modern chip containing 3 ALU's of 14
bits, multi-branch microprogrammed controller, program
counter, I/O circuitry, scanable register for testability.

symbolic simulation and the checking by means of OBDD's
[2] of the proof obligations per member of the partition.

To compare the verification complexities with these published
for the verification of FSM's [3, 4], the number of
latches in the transistor implementations are indicated in
table 2 together with the corresponding number of states as
counted e.g. in [4].