Data Structure Manipulation for NNIG and PTNNIG: Towards a Unified Power and Timing Analysis

Rashmi Mehrotra*, Ka Lok Man†, Emanuel Popovici*†, Michel Schellekens†
*Dept of Microelectronic Engineering / †Centre for Efficiency-Oriented Languages
University College Cork
Cork, Ireland
r.mehrotra@student.ucc.ie, klm1@cs.ucc.ie
e.popovici@ucc.ie, m.schellekens@cs.ucc.ie

***Abstract*—Structural representation and technology mapping of a Boolean function is an important issue in the design of digital circuits. Various data structures such as *Binary Decision Diagrams* (BDDs), *And Inverter Graphs* (AIGs) and *Nand Nor Inverter Graphs* (NNIGs) have been widely used for structural representation, logic synthesis and technology mapping of digital circuits. The tool ABC has been developed for the logic synthesis and manipulation of AIGs. This paper presents a tool which builds and manipulates NNIGs as a sub-package in ABC. To the best of our understanding, it is the first tool developed for the manipulation of NNIGs. Experimental results illustrate the applicability and efficiency of the tool.**

The paper also presents a new data structure *Probabilistic Timed Nand-Nor-Inverter Graph* (PTNNIG) that can be used for accurate power estimation and timing analysis of digital circuits. Such data structures are incorporated with probability and timing as parameters and will lead to a better and combined power and timing analysis.

***Index Terms*—SOP, Algebraic factorisation, AIGs, NNIGs, ABC**

I. Introduction

In the literature, data structures such as *Reduced Ordered Binary Decision Diagrams* (ROBDDs) [1], *And-Inverter Graphs* (AIGs) [2] and *Nand-Nor-Inverter Graphs* (NNIGs) [3] are commonly used for the structural representations of binary logic networks at the technology-independent stage. They are widely used in the EDA community and their main application domains are logic synthesis, testing and formal verification. For details, we refer to [4] and [5]. Not only structural representation but also optimisation of multi-level logic networks plays an important role in EDA. During multi-level network synthesis, each node of a Boolean network can be represented by minimal ON- and/or OFF- covers and in some cases together with factored expressions derived from the minimal sum-of-products (SOPs). ESPRESSO [6] is an academic standard two-level logic minimiser which is being widely and most commonly used to obtain minimised SOPs. Factoring Boolean functions is also a basic operation in algorithmic logic synthesis. Factoring is the translation of a function in the SOP form (also called disjunctive form) to a form with parentheses and having a minimum number of literals [7]. For instance a, ab'c' and a(b+c+d)+e are all factored forms. A factored form can be represented as a tree structure, where each internal node is an AND or OR operator and each leaf is a literal. There are mainly two methods to obtain the factored form of a two-level representation of the function. One is Algebraic division [8] and [7], also known as weak division which is quite fast. The other one is Boolean division [9], also known as strong division which is slower but capable of giving better results in many cases. In general, the algebraic methods are fast because the logic function is treated as a polynomial, and hence fast methods of manipulation are available.

According to the statistics presented in [3], NNIGs are found to exhibit 3.97% lesser node count compared to AIGs and consume 23.74% lesser library cells than AIGs for the representation of the same circuits. To automate the implementation of NNIGs, in this paper, we present a new tool *Data Structure Manipulation* (DSM) which has been developed as a sub-package in ABC [2]. The implementation of the tool is based on algebraic factorisation of minimised SOP form of the combinational logic, similar to the techniques mentioned in [3]. For experimentation, we did power study on NNIGs against AIGs of the same circuits using SIS [10]. We also did comparisons of DSM against ABC tool.

After having built the tool DSM, we realised that NNIGs are still not ideal for power and timing analysis as they do not incorporate power and timing as parameters. Hence we propose a new data structure *Probabilistic Timed Nand-Nor-Inverter graph* (PTNNIG) which can be used for more accurate power estimation as well as combined timing and power analysis. Switching activity and delay from RTL simulation are represented in PTNNIGs as probability and timing parameters respectively. To the best of our understanding, PTNNIGs are the first data structures incorporating timing and switching activity for the graphical representations of digital circuits. We presented a way of one to one mapping between a Verilog netlist and NNIG structure to map switching activity and timing values using an integrated design flow. Experimental results in the end show the implementation of PTNNIGs.

A. Organisation

The remainder of the paper is organised as follows. Section 2 gives an overview about different graph structures. Section 3 presents the tool DSM. Section 4 shows some experimental results which includes comparisons made on NNIGs against

978-1-4244-4398-7/09/$25.00 ©2009 IEEE

AIGs on the basis of power estimation. It also includes comparisons of DSM against ABC tool. Section 5 introduces the new data structure PTNNIG along with an example and explains how switching activity and delay from RTL simulation can be obtained and mapped on NNIGs for combined power and timing analysis. Section 6 again presents some experimental results on BLIF against optimised BLIF using realistic switching activity information. Finally concluding remarks and proposals for future work are made in Section 7.

II. Background of data structures

Compact multi-level binary networks can be efficiently and effectively represented using AIGs and NNIGs. AIG is a *Directed Acyclic Graph* (DAG) that represents a structural implementation of the logic functionality of any random Boolean network. As the name implies, it consists of only two-input AND gates and inverters. Hence it consists of two-input nodes, representing logical conjunction, and edges optionally containing markers indicating logical negation. For more details we refer to [2].

Similar to AIG, we find that a logic circuit can be conveniently represented using NNIG. A NNIG also corresponds to a DAG representation for a logic function. In this case, the network only comprises of two-input NAND gates, two-input NOR gates and inverters. For details we refer to [3]. Both AIG and NNIG can be built recursively using De-Morgan's laws. NNIGs and AIGs are non-canonical structures. A Boolean function can have many functionally equivalent NNIG representations corresponding to different expressions at the two-level logic, typically two structures would be compact representations; one obtained from a factored minimum sum of products (MSOP) of the function and other based on factored minimum products of sum (MPOS) of the function.

III. DSM tool

This section presents the tool DSM implemented as a sub-package in ABC. The theoretical foundation behind the tool is based on the papers [3] and [11]. The tool DSM allows us to build a NNIG for a given circuit. The input of DSM is given in *Berkeley Logic Interchange Format* (BLIF) [10] which is the most popular format used in academic tools for describing logic-level hierarchical circuits in a textual form. By means of some optimisation steps, we obtain the output as the NNIG of the digital circuit (given as input in BLIF). Then NNIG is converted back to a BLIF file format. The flow involved in the making of NNIGs via DSM is shown in Fig. 1 more precisely. First, the BLIF file is read as a network in ABC whose functionality is converted to SOP. Using ESPRESSO, the SOP undergoes a two-level logic minimisation. The minimised SOP obtained is then factorised using the algebraic factorisation technique, such factorised minimised SOP is firstly converted to AIG in the environment of ABC and then further transformed to a NNIG network using a sub-command AIG2NNIG. The functionality of the NNIG network is converted back to SOP using a sub-command SOP2NNIG used in the DSM tool and then to BLIF file format using the translator in ABC. This tool is implemented in C as a sub- package in ABC. It can handle both sequential and combinational digital circuits as inputs.

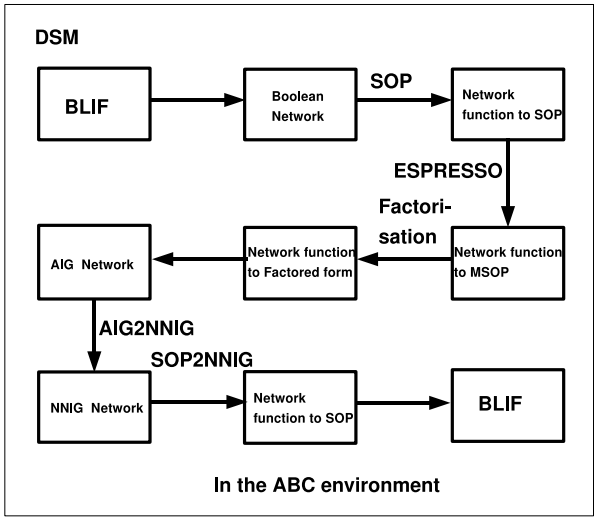

Fig. 1. DSM Tool Flow.

IV. Experimental Results 1

Following the same DSM tool flow till ESPRESSO, the corresponding AIG structure is derived from minimum SOP form of the function in the environment of ABC and then converted to BLIF file format again using ABC.

The BLIF files corresponding to AIG and NNIG network are given as inputs in SIS (in the environment of ABC) for power estimation. Table I presents a list of randomly taken 10 combinational and 5 sequential circuits and their power estimation in uW. Some BLIF examples used here are from the MCNC benchmark circuits and some of the BLIF files used are those mentioned in [3]. Here, for power estimation, the clock frequency is 20 MHz and Vdd is 5 Volts. The table clearly indicates that NNIGs of many combinational and sequential digital circuits consume lesser power than their corresponding AIGs. On an average power reduction in power consumption is 22.06%.

We performed another set of experiments on the BLIF files obtained from the DSM tool against the BLIF files obtained from ABC. The BLIF files from ABC are those obtained after AIG network creation using one of the standard scripts of ABC namely alias syn. In this script, the series of synthesis commands are balancing, rewriting, rewriting with zero cost replacement, balancing, rewriting with zero cost replacement and balancing in the end. These synthesis commands are used to reduce the number of nodes, reduce the number of logic levels and to have a minimum delay. We did power estimation on the two BLIF files using SIS. Table II presents a list of combinational circuits and their power estimation in uW. The BLIF files used here are those mentioned in [3]. Here, for power estimation, the clock frequency is 20 MHz and Vdd is 5 Volts. The table indicates that for some of the combinational

TABLE I
POWER ESTIMATION ON AIG AND NNIG

Circuit	AIG network	NNIG network
lf3	29.5	22.8
lf6	36.3	26.7
lf11	41.0	37.8
lf8	56.0	32.5
lf5	60.1	44.1
cm85a	205.7	179.6
cm162a	249.8	162.5
cc	367.2	289.2
b9	639.9	540.1
adder	1242.5	998.8
bbara	162	94.6
tav	245.5	241.7
ex7	378.9	325.9
ex3	418.7	267.9
ex1	878.1	805.9

circuits, the tool technique used in DSM can prove better than some of the synthesis techniques used in ABC, resulting in BLIF files with lower power consumption. On an average power reduction is 17.69%.

TABLE II
POWER COMPARISONS AGAINST ABC AND DSM

Circuit	ABC	DSM
lf2	24.2	18.1
lf3	24	22.8
lf5	56.6	44.1
lf6	28.3	26.7
lf7	71.4	61.0
lf10	49.4	37.7
lf12	64.2	51.2
lf13	24.2	18.1

V. PROBABILISTIC TIMED NNIGS

A. Background

In recent years, power dissipation is being given increased weightage in design metrics. $P_{dynamic}$ is the dynamic power dissipation, also called the switching power which is the dominant source of power consumption in most of the digital circuits. It is generally represented by the following approximation:

$$P_{dynamic} = \alpha.C_l.V_{dd}^2.f_{clk} \quad (1)$$

where α is the switching activity factor (also called transition probability), C_l is the overall capacitance to be charged and discharged in a reference clock cycle, V_{dd} is the supply voltage and f_{clk} is the clock frequency.

Various data structures have been used for the manipulation of switching activity and delay such as those referred in [12] and [13]. However they were not ideal for a combined power estimation and timing analysis. This section hence proposes the new data structure PTNNIG. Here switching activity and timing both represented as probability and timing parameters respectively are specifically defined in the syntax of PTNNIGs for power and timing analysis. Optimising such graphs will imply power and timing minimisation. Our ultimate goal is to provide a data structure which enables efficient power driven timing optimization and timing driven power optimisation.

A PTNNIG is defined as a tuple (V, E, P, T). This tuple is a quadruple where V is a set of vertices, E is a set of edges, P is a pair set of probabilities and T is a pair set of timing values. Probability and timing are two parameters added to existing NNIG data structures, these parameters are associated to each edge from the set E.

B. Mapping of Probability and Timing Issues on PTNNIGs

For illustration purpose, this section presents a way to map probability and timing issues on NNIG to build PTNNIG. First, the BLIF file obtained as a final output of DSM is translated to a Verilog file using the translator available in ABC. Such Verilog file is then synthesised in Synopsys Design Compiler DC [14] to obtain a netlist. While synthesising the design, we set to use only NAND, NOR and inverter cells for technology mapping. Using such constrains we can obtain a netlist equivalent to a NNIG since it contains only NANDs, NORs and inverters. Now we have a NNIG obtained from the tool DSM and a netlist obtained from DC. A one to one mapping is possible between the information associating the Verilog netlist to the nodes of the NNIG. This mapping is explained in Fig. 2.

According to the Fig. 2, the probabilities in the set P of PTNNIG can be derived from the SAIF files [15]. The SAIF files are obtained after RTL simulation of a Verilog design using various tools. For example they can be obtained using VCS [14], a commercial tool from EDA vendor Synopsys. SAIF files contain information of the switching activities for all the nodes of the digital circuit. See also [16] for some details. Switching activity of a node in a digital circuit is highly dependent on the circuit inputs of the current clock cycle and the previous clock cycle. It actually indicates us the number of times a particular circuit node switches from 0-1 and from 1-0 in a given clock cycle. SAIF files begin with a header and contain a set of activity metrics such as T0, T1, TX, TC, TS and DUR for each circuit node.

- T0, T1 and TX represent the total time spent at logic 0, 1 and X respectively.
- TC represents the total number of logic transitions (0 to 1 and 1 to 0).
- TS represents the SAIF header 'TIMESCALE' field.
- DUR represents the SAIF header 'DURATION' field.

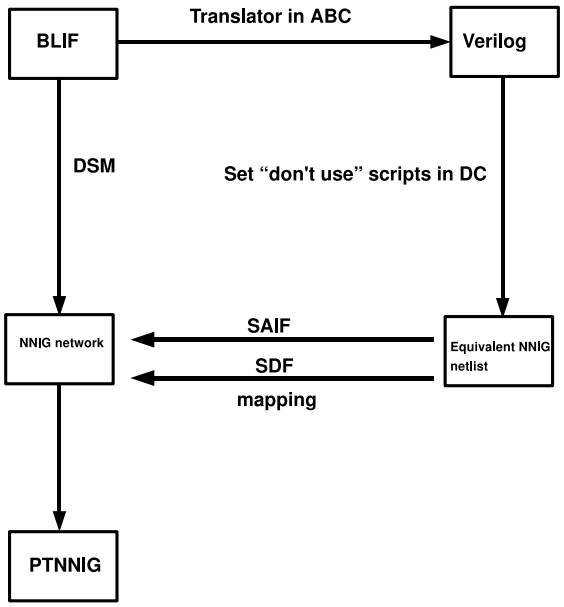

Fig. 2. Mapping of Probability and Timing Issues on PTNNIGs.

Hence the P set from the quadruple is actually a pair (p_1, p_0) $\in$ P where we define p_1 and p_0 parameters as

$$p_1 = \frac{T1}{T} \tag{2}$$

$$p_0 = \frac{T0}{T} \tag{3}$$

where the simulation duration T is defined as

$$T = TS * DUR \tag{4}$$

Using the one to one mapping concept, the pair (p_1, p_0) $\in$ P obtained from the SAIF files can be mapped to the NNIG structure obtained from DSM.

Similarly, according to the Fig. 2, the timing values in the set T can be obtained from the *Standard Delay Format* (SDF) files [17]. The SDF files are generated when the digital circuit design is synthesised to a gate level netlist in Synopsys Design Compiler DC [14] and delays as specified in the SDF files are annotated onto the netlist using PrimeTime [14]. See also [16] for some details.

SDF files are used for the representation and interpretation of timing data used at any stage of the logic synthesis. It includes path delays, timing constraint values and interconnect delays for each of the cells. The files contain the cell names used in the mapping along with the individual delays from each of the input to the output of the corresponding cells. The term IOPATH represents the path referenced to, followed by the worst case and the best case values at rising edge, which is then followed by the worst case and the best case values at falling edge. Here the rising edge delay is defined as t_r and falling edge delay is defined as t_f, using such values we define the T set from the quadruple as a pair (t_f, t_r) $\in$ T. Using the same one to one mapping concept, the pair (t_f, t_r) $\in$ T obtained from the SDF files can be mapped to the NNIG obtained from DSM.

C. Example

As an example, we build the PTNNIG of the logic circuit representing the logic function a*b*(c+d). The description of such a circuit in BLIF is shown in Fig. 3. This BLIF file is translated to Verilog using various translators such as the translator in ABC. The Verilog design is then read and synthesised in DC with constraints to use only NANDs, NORs and inverters. This Verilog file is shown in Fig. 4. Using the procedure discussed in the above sub-section, SAIF and SDF files of such Verilog design are generated and shown in Fig. 5 and Fig. 6 respectively. Only a part of the SAIF and SDF files is shown. Hence according to the procedure shown in Fig. 2, switching activity and timing issues from these SAIF and SDF files are mapped on the NNIG representing the logic function a*b*(c+d). On mapping such switching activity and timing issues on NNIG, gives PTNNIG. The PTNNIG is shown in Fig. 7. The notation $\bullet$ in the Fig. 7 indicates negation. The symbols $\bigotimes$ and $\bigoplus$ in the Fig. 7 represent AND and OR respectively. The symbol $\bigotimes$ with the notation $\bullet$ linked to it indicates NAND in the Fig. 7, similarly the symbol $\bigoplus$ with the notation $\bullet$ linked to it indicates NOR in the Fig. 7. On the edges we show the probability and timing parameters. For readability we show only the p_0 and t_r values in the PTNNIG. p_0 is a probability value hence less than 1 and t_r is measured in nanoseconds.

```
.model ptnnig
.inputs a b c d
.outputs e
.names a b c d e
1101 1
1110 1
1111 1
.end
```

Fig. 3. BLIF file of the circuit representing the logic function a*b*(c+d)

```
module ptnnig ( a, b, c, d, e );
input a, b, c, d;
output e;
wire   n5, n6, n7, n8;
NR2D0HVT U6 ( .A1(c), .A2(d), .ZN(n8) );
NR2D0HVT U7 ( .A1(n5), .A2(n6), .ZN(e) );
INVD0HVT U8 ( .I(b), .ZN(n5) );
ND2D0HVT U9 ( .A1(a), .A2(n7), .ZN(n6) );
INVD0HVT U10 ( .I(n8), .ZN(n7) );
endmodules
```

Fig. 4. Verilog file of the circuit representing the logic function a*b*(c+d)

VI. Experimental Results 2

In SIS, while power estimation, we can input a file which allows the specification of input probabilities, node capaci-

```
          ........
(e
     (T0 18081) (T1 1919) (TX 0)
     (TC 127) (IG 0)
     )
(n5
     (T0 16820) (T1 3180) (TX 0)
     (TC 126) (IG 0)
     )
(n6
     (T0 1920) (T1 18080) (TX 0)
     (TC 191) (IG 0)
     )
          .........
```

Fig. 5. SAIF

```
              .........
(CELL
(CELLTYPE "ND2D0HVT")
(INSTANCE U9)
(DELAY
(ABSOLUTE
(IOPATH A1 ZN (0.010::0.023)
(0.011::0.031))
(IOPATH A2 ZN (0.013::0.032)
(0.014::0.041))
)
)
)
(CELL
(CELLTYPE "INVD0HVT")
(INSTANCE U8)
(DELAY
(ABSOLUTE
(IOPATH I ZN (0.009::0.022)
(0.006::0.015))
)
)
)
          ...........
```

Fig. 6. SDF

tances and node delays of the circuit, namely SA file. Here in our experiments, we plugged in only the probability values of primary inputs and primary outputs. For this, we built a tool SAIF2SIS in C which was used in [16]. This tool reads the relevant switching activity information from a SAIF file, calculates the pair $(p_1, p_0) \in P$ using equations 2 and 3 and allows us to specify these probabilities of the primary inputs and primary output in the SA file. Here we have taken BLIF files of 10 combinational MCNC benchmark circuits. Power estimation results in uW against a BLIF file and an optimised BLIF file obtained from the tool DSM are shown in Table III. Here we have used the same SA file for both the categories of BLIF files. The clock frequency is 20 MHz and Vdd is 5

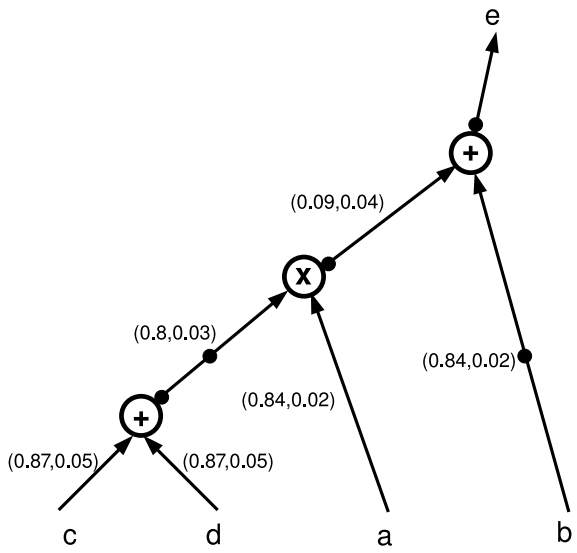

Fig. 7. PTNNIG of the logic circuit representing the logic function a*b*(c+d).

Volts for power estimation. The NNIG network corresponding to each BLIF file obtained from the DSM tool used here can be referred as the incomplete PTNNIG, with probability parameter plugged in through SA file. From the results we can see that the power consumption is lesser in the case of BLIF files obtained from the tool DSM. The average reduction in the power estimation is 16.7%.

TABLE III
POWER ESTIMATION AGAINST BLIF AND OPTIMISED BLIF

Circuit	BLIF	optimised BLIF
b1	29.3	23.4
cm85a	126.4	119.3
cm162a	125.2	116.6
cmb	126.7	117.5
pm1	143.7	108.1
x2	143.4	121.7
mux	195.3	115.7
cc	217	206.7
b9	212.5	148.4
c8	497.4	414.4

VII. CONCLUSION

The paper presents a tool to build optimised NNIG networks of digital circuits within ABC. From the results, we obtained an average power reduction of 22.06% in NNIG networks against AIG networks. We obtained an average power reduction of 17.69% in BLIF files from DSM tool against BLIF files from ABC.

In the context of a unified framework for power and timing

analysis and optimisation, we also presented the new data structure PTNNIG which represents the functionality of any digital design along with switching activity and delay as parameters on the edges of the PTNNIG. We also obtained an average power reduction of 16.7% in optimised BLIF files against BLIF files without any optimisation. Here in these power comparisons, we plugged in node probabilities for power estimation using an integrated design flow as used in [16]. This implies that NNIG logic data structures can be used for enabling low power logic designs.

Such data structures will also help us to apply a statistical approach to the power analysis with constraints on timing parameter, similarly a statistical approach to timing analysis with constrains on switching activity parameter. These statistical approaches can be done using probability density functions, e.g. in the form of Gaussian functions and Monte Carlo method. Such constraints will allow us to obtain timing aware power optimisation and power aware timing optimisation. As future work, we aim to extend the DSM tool for the implementation of PTNNIGs of the digital circuits.

ACKNOWLEDGMENT

We gratefully acknowledge the support of IDA Ireland and Synopsys in funding our research as indicated in this paper. Our sincere thanks also go to our colleague Tom English who has contributed significantly to this paper.

REFERENCES

[1] R.Bryant. "Graph-based algorithms for boolean function manipulation". *IEEE Transactions on Computers*, vol. 35, no. 8, pp. 677–691, 1986.

[2] A. Mishchenko and S. Chatterjee and R. Brayton. "Dag-aware aig rewriting a fresh look at combinational logic synthesis". *Proceedings of the 43rd annual conference on Design automation*, pp. 532–535, 2006.

[3] Padmanabhan Balasubramanian and Karthik Anantha. "Power and delay optimized graph representation for combinational logic circuits". *International Journal of Computer Science*, vol. 2, pp. 47–53, 2007.

[4] A. Mishchenko and R.K. Brayton. "Scalable logic synthesis using a simple circuit structure". *Proceedings of International Workshop on Logic Synthesis*, pp. 15–22, 2006.

[5] A. Mishchenko, S. Chatterjee, R. Brayton and P. Pan. "Integrating logic synthesis, technology mapping and retiming". *ERL Technical Report, EECS Dept.*, April 2006.

[6] P. C. McGeer, J. V. Sanghavi, R. K. Brayton and A. L. Sangiovannivincentelli. "ESPRESSO-SIGNATURE: A new exact minimiser for logic functions". *IEEE Transactions on VLSI*, vol. 1, no. 4, pp. 432-440, 1996.

[7] R. Brayton and C. McMullen. "The decomposition and factorisation of Boolean expressions". *Proc. IEEE International Symposium on Circuits and Systems*, pp. 49-54, 1982.

[8] R. Brayton. "Factoring logic functions". *IBM Journal of Research and Development*, vol. 31, no. 2, pp. 187-198, 1987.

[9] T. Stanion and C. Sechen. "Boolean division and factorisation using binary decision diagrams". *IEEE Trans. on CAD of Integrated Circuits and Systems*,1994.

[10] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli. "Sis: A system for sequential circuit synthesis". *Electronics Research Laboratory*, Memorandum No. UCB/ERL/ M92/41, May 1992.

[11] P. Balasubramaniana, R. T. Naayagib, A. Karthikc and B. Raghavendrad. "Evaluation of logic network representations for Achilles' Heel boolean functions". *International Journal of Computers, Systems and Signals*, vol. 9, no. 1, 2008.

[12] M. T. P. Lindgren, M. Kerttu and R. Drechsler. "Low power optimisation technique for BDD mapped circuits". *ASP-DAC*, 2001.

[13] A. Biere. "The aiger and-inverter graph (aig) format version 20070427". *Johannes Kepler University*, Tech. Rep. 2007.

[14] Synopsys. "Synopsys design platforms". http://www.synopsys.com/products/products.html.

[15] Synopsys SAIF. "Switching Activity Interchange Format". http://www.synopsys.com/partners/tapin/saif.html.

[16] Rashmi Mehrotra, Tom English, Ka Lok Man, Emanuel Popovici and Michel Schellekens. "Digital power estimation flow combining academic and industrial tools". *IEEE proceedings of the 5th IEEE International SoC Design Conference - IEEE/ISOCC'08*, 2008.

[17] Synopsys. "Standard Delay Format (SDF) for the electronic design process (IEEE Std 1497-2004)". *IEEE Delay and power calculation standards - Part 3*, 2004.