A geometric approach to register transfer level satisfiability

Conference Paper · March 2009

DOI: 10.1109/ISQED.2009.4810306 · Source: DBLP

6 authors, including:

Juan A Montiel-Nelson
Universidad de Las Palmas de Gran Canaria
178 PUBLICATIONS **463** CITATIONS
SEE PROFILE

Víctor Navarro-Botello
Universidad de Las Palmas de Gran Canaria
19 PUBLICATIONS **93** CITATIONS
SEE PROFILE

Javier Sosa
Universidad de Las Palmas de Gran Canaria
63 PUBLICATIONS **103** CITATIONS
SEE PROFILE

J.C. Garcia
Universidad de Las Palmas de Gran Canaria
55 PUBLICATIONS **193** CITATIONS
SEE PROFILE

A geometric approach to register transfer level satisfiability

Héctor Navarro, Saeid Nooshabadi[2], Juan A. Montiel–Nelson, V. Navarro, J. Sosa and José C. García
Inst. for Applied Microelectronics, Dept. of Electronic Engineering and Automation, ULPGC, Spain
[2]Dept. of Information and Communications, Gwangju Institute of Science and Technology, Korea
E-mail: hnavarro@iuma.ulpgc.es

Abstract— In this paper, inequalities of integer hull polyhedrons are used in mixed integer linear programming (MILP) to model the behavior of combinational subsystems, introducing a new solution for the satisfiability (SAT) problem at register transfer level (RTL). Since in these models the vertexes are located at integer positions, they can be used with linear programming (LP) to solve SAT problems. Unfortunately, when combining together several models to make up a compound RTL description, internal vertexes may appear forcing to declare one or more variables as integer. However, the use of these models inside an RTL SAT problem reduces the number of branches needed to solve the whole integer problem. Results show a CPU solution time reduction greater than one order of magnitude, growing with the size of the problem.

Keywords— Register transfer level (RTL), satisfiability (SAT), design verification, linear programming, cutting planes

I. Introduction

The satisfiability (SAT) problem is the central problem of inference in propositional logic, and it has many direct applications in the electronic design automation (EDA) arena. In particular, its applications include formal verification, placement and routing, automated reasoning, computer–aided design, computer–aided manufacturing, machine vision, databases, robotics, computer architecture design, and computer network design, among others.

Boolean SAT solvers are widely studied from several perspectives. There have been many efficient SAT solvers in the last decade. The classical approach (the DPLL algorithm) was established by Davis and Putnam [1], [2], and even recent procedures are different variations over this basis. Such algorithms [3], [4], [5] include advanced features such as learning techniques, chronological backtrack–based search, and conflict analysis procedures to prune the search space.

The main bottleneck of the design cycle of VLSI systems is functional verification. Register transfer level (RTL) hardware description languages (HDLs) are widely used for the hardware specification; however, the majority of industrial hardware verification tools operate at gate level, and are based on bit–level decision procedures, such as Boolean satisfiability or binary decision diagram (BDD)–based techniques.

Recent works are addressed to extend formal verification [6] and test generation [7], [8] to RTL specifications. The formal verification of a RTL description can be formulated as a hybrid SAT problem that combines Boolean logic with word–level arithmetic operators. Although the traditional way to face this problem is to apply a logic synthesis and then a Boolean SAT solver, there have been many contributions to solve hybrid SAT problems in a more efficient way.

Fallah et at. proposed HSAT [9], this is an approach for functional vector generation from HDL models. From the RTL description, it translates Boolean logic and arithmetic blocks into conjunctive normal form (CNF) clauses and linear arithmetic constraints (LACs), respectively. Then, independent solvers are applied for each part. Obviously, backtracks between two solvers are inevitable.

To avoid this drawback, Zeng et al. proposed LP-SAT [10], where arithmetic operators and Boolean logic are combined together into a unified mixed integer linear programming (MILP) problem with Boolean and integer variables.

The rapid progress in the capacity and speed of modern DPLL–based Boolean satisfiability (SAT) solvers has lead to satisfiability modulo theories (SMT) [11]. SMT solvers integrate a combined decision procedure for both, the Boolean and the integer domain. Ario [12] and Yices [13], [14] are different examples of SMT solvers.

In this paper, we use of a special kind of polyhedrons for RTL SAT problems in MILP. In these polyhedrons, vertexes are located only at integer positions. Therefore, integer solutions can be obtained just solving the linear programming (LP) relaxation of the integer problem. This feature is extremely important, due to the fact that LP problems can be solved in polynomial time, while integer linear programming (ILP) problems require an exponential time for the worst case.

This paper is organized as follows. The adopted methodology to obtain the models is explained in Section II. Models for different subsystems and conditions for the insertion of these models into RTL satisfiability problems, are presented in Section III. Section IV shows experimental results and compares them against the main alternatives. The conclusions and the future work on the use integer hull polyhedrons for RTL satisfiability problems are presented in Sections V and VI, respectively.

II. Methodology

Given a polyhedron $P \in \mathbb{R}_+^n$ and a vector $c \in \mathbb{R}_+^n$, the LP problem can be seen as the problem of finding the extreme point $x \in P$ that maximizes the linear function $c \cdot x$.

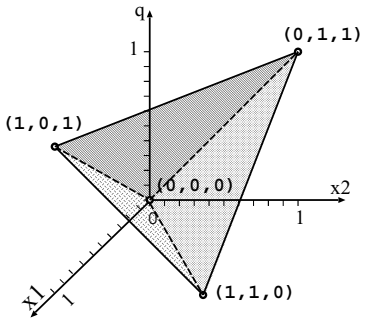

Fig. 1. Integer hull polyhedron that models an XOR2 logic gate.

Note that if all extreme points of the polyhedron P are located at integer coordinates, then the solution vector of the LP problem, x is also integer.

Given a Boolean network with I inputs and O outputs, it can be represented as a bounded polyhedron $P \in \mathbb{R}^n_+$, where the dimension of the space is $n = I + O$. As the domain for each Boolean variable is $\{0, 1\}$, the bounded polyhedron (also called polytope) belongs to the domain $[0, 1]^n$, so $P \in [0, 1]^n$.

The coordinates of the extreme points of P are determined by the truth table of the Boolean network. Therefore, the polyhedron P will have exactly 2^I extreme points. As an example, Figure 1 shows the integer hull polyhedron, $P \in \mathbb{R}^3_+$ for an XOR2 logic gate.

Fukuda's tool `cdd+` [15] was used to obtain a linear inequality representation of a polyhedron described by its extreme points. Models are expanded to an arbitrary number of inputs by obtaining, grouping and classifying linear inequalities for 1 bit, 2 bits, 3 bits, and so on.

In many cases, a combinatorial set of linear inequalities is obtained. In these cases, the number of faces of the integer hull polyhedron grows exponentially with n. Fortunately, the same Boolean network can be also modelled using polyhedrons of a higher dimension space. The set of inequalities needed to represent one of these polyhedrons may be smaller. This is, for example, the case of the n–input XOR gate.

III. **Integer Hull Polyhedra Models**

For logic gates, the set of inequalities proposed in LP-SAT [10] is valid for the AND, OR and NOT gates. These models come from the Tseitin transformation [16]. Assuming that all variables are bounded to $[0, 1]$. Equations 1, 2 and 3 show the inequalities for the inverter gate, the n–input `and` gate, and the n–input `or` gate, respectively.

$q =$`inv` (a)

$$q = 1 - a \tag{1}$$

$q =$`and` $(a_0, a_1, a_2, \ldots, a_{n-1})$

$$\begin{aligned} q &\leq a_i \quad \forall i : \ 0 \leq i \leq n-1 \\ q &\geq \sum_{i=0}^{n-1} a_i - (n-1) \end{aligned} \tag{2}$$

$q =$`or` $(a_0, a_1, a_2, \ldots, a_{n-1})$

$$\begin{aligned} q &\geq a_i \quad \forall i : \ 0 \leq i \leq n-1 \\ q &\leq \sum_{i=0}^{n-1} a_i \end{aligned} \tag{3}$$

Equations 4 and 5 show inequalities for the XOR2 and XOR3 logic gates, respectively. The integer hull polyhedron for an n–input XOR gate has 2^n inequalities. However, XOR2 and XOR3 primitives can be connected in cascade to create an n–input XOR gate. The result is an integer hull polyhedron with the same functionality of the n–input XOR gate, but with additional variables. In this case, the total number of inequalities is $4(n-1)$, and the number of additional variables is given by $\lceil \frac{n-3}{2} \rceil$.

$q =$`xor` (a_0, a_1)

$$\begin{aligned} -a_0 + a_1 + q &\geq 0 \\ a_0 - a_1 + q &\geq 0 \\ a_0 + a_1 - q &\geq 0 \\ a_0 + a_1 + q &\leq 2 \end{aligned} \tag{4}$$

$q =$`xor` (a_0, a_1, a_2)

$$\begin{aligned} 0 \leq \ -a_0 + a_1 + a_2 + q &\leq 2 \\ 0 \leq \ a_0 - a_1 + a_2 + q &\leq 2 \\ 0 \leq \ a_0 + a_1 - a_2 + q &\leq 2 \\ 0 \leq \ a_0 + a_1 + a_2 - q &\leq 2 \end{aligned} \tag{5}$$

Equation 6 shows the inequalities set for the 2–input multiplexer of 1–bit, where *sel* is the control signal (when $sel = 0$ the output $q = a_0$, otherwise $q = a_1$).

$q =$`mux` (sel, a_0, a_1)

$$\begin{aligned} a_0 + a_1 - 1 &\leq q \leq a_0 + b_0 \\ a_0 - sel &\leq q \leq a_0 + sel \\ a_1 - (1 - sel) &\leq q \leq a_1 + (1 - sel) \end{aligned} \tag{6}$$

Finally, the proposed model for an n–bit adder is presented in Equations 7, 8 and 9. In this model, a and b are n–bit inputs, and q is an output of $(n+1)$ bits. The model consists of one equality, four inequalities for the least significant bit (XOR2), and four pairs of inequalities that must be repeated for the remaining bits. This model may include a carry input bit by replacing Equations 8 by the set of an XOR3 gate, and updating Equations 7 and 9 accordingly.

Main equation:

$$\sum_{i=0}^{n-1} 2^i a_i + \sum_{i=0}^{n-1} 2^i b_i - \sum_{i=0}^{n} 2^i q_i = 0 \tag{7}$$

First bit:

$$\begin{aligned} -a_0 + b_0 + q_0 &\geq 0 \\ a_0 - b_0 + q_0 &\geq 0 \\ a_0 + b_0 - q_0 &\geq 0 \\ a_0 + b_0 + q_0 &\leq 2 \end{aligned} \tag{8}$$

Other bits: $(\forall j:\ 1 \leq j \leq n-1)$

$$
\begin{aligned}
0 \leq & \sum_{i=0}^{j} 2^i a_i + \sum_{i=0}^{j} 2^i b_i - \sum_{i=0}^{j} 2^i q_i && \leq 2^{j+1} \\
0 \leq & -2^j a_j + \sum_{i=0}^{j-1} 2^i a_i + \sum_{i=0}^{j} 2^i b_i + 2^j q_j - \sum_{i=0}^{j-1} 2^i q_i && \leq 2^{j+1} \\
0 \leq & \sum_{i=0}^{j} 2^i a_i - 2^j b_j + \sum_{i=0}^{j-1} 2^i b_i + 2^j q_j - \sum_{i=0}^{j-1} 2^i q_i && \leq 2^{j+1} \\
0 \leq & 2^j a_j - \sum_{i=0}^{j-1} 2^i a_i + 2^j b_j - \sum_{i=0}^{j-1} 2^i b_i + \sum_{i=0}^{j} 2^i q_i && \leq 2^{j+1}
\end{aligned}
\tag{9}
$$

When these models are combined together in an RTL description, polyhedrons are intersected each other and internal vertexes may arise. These vertexes appear when there are loops over the data dependency graph at bit–level. Integer variables are removed from this graph, so the study of these loops indicates the minimum set of variables that must be declared as integers.

As an example, consider the multiplexer model shown in Equation 6. It can be connected in cascade or in parallel to build an n–input multiplexer, or a 2–input multiplexer of n–bit wide. But, when connected in cascade and in parallel simultaneously to build an m–input, n–bit multiplexer, non–integer vertexes arise because of loops in the data dependency graph. To avoid this loops, at least control variables must be declared as integers.

IV. **Experimental Results**

In order to compare the proposed methodology against existing solutions, two sets of tests have been made. All tests were executed on an Intel Xeon X7350 CPU running at 3, 00 GHz, with 24 GB. of DRAM, and executing a 32–bit Linux operating system, version 2.6.22. CPLEX [17] was used as MILP solver for both, integer hull polyhedra model and LPSAT [10] MILP model, (labeled as `ihp` and `lpsat` respectively, in the graphs). MINISAT [5] was used to solve Boolean SAT problems in standard 3SAT/CNF format (`minisat`), and YICES [13] for solving SMT descriptions (`yices`). CPU runtime was limited to 3600 seconds.

The first set of experiments checks the efficiency of the model for the adder, in an ideal case, as the complexity of the problem grows. It is shown in Figure 2, and consists of 16–bit adders with some input and output ports assigned to random 0–1 values (bit level constraints). The parameter α determines the relationship between the number of assignments and the total number of ports. All problems have at least one solution.

Figure 3 shows the solution runtime versus the SAT constraint probability, α for problems with $n = 1000$ adders. This figure shows that the proposed model in LP (`ihp`) is less sensible to the problem complexity than the Boolean solver (`minisat`). The basic operations of the Boolean SAT

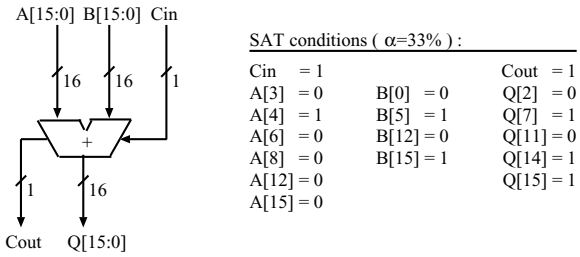

Fig. 2. Adder circuit used for the first set of experiments

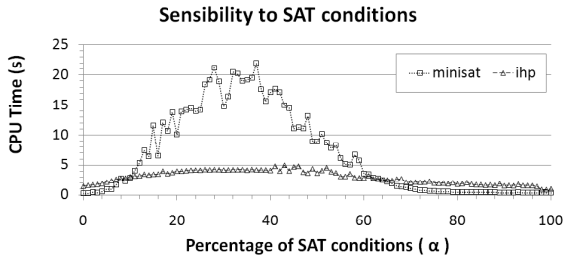

Fig. 3. Solution runtime versus percentage of SAT conditions

solver are `decisions`, `propagations` and `conflicts`. The SAT solver make a decision about the state of a variable (an hypothesis). The Boolean equations are simplified (propagations) to a reduced set of equations (constraints); otherwise, one conflict occurs. In the event of a conflict, the analysis techniques help in determining which of the decisions taken were wrong. The SAT solver (`minisat`) specifies how many propagations are done, and how many conflicts arises when the decisions are wrong. Boolean solver performs faster when `decisions` and `propagations` are in majority, but slows down when `conflicts` appear. The other solutions (`lpsat` and `yices`) were too slow, as can be observed in Figure 4.

Figure 4 plots the solution runtime versus the number of adders n for a fixed value of the constraint probability of $\alpha = 33\%$. This is an ideal case where adders are independent of each other, so all signals can be declared as reals instead of integers. CPU time reduction is about two orders of magnitude for large problems.

The second set of experiments is focused on obtaining a realistic measurement of the speedup for a particular RTL SAT problem. To this end, the multiplier circuit shown in Figure 5 was considered. This circuit consists of several

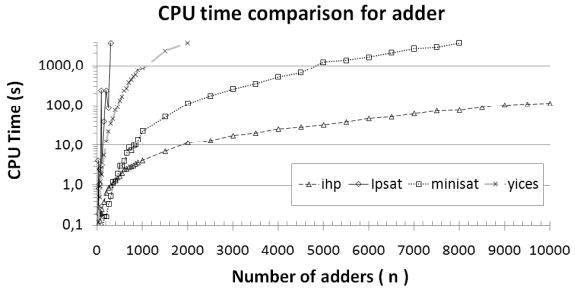

Fig. 4. Solution runtime time versus problem size for the adder

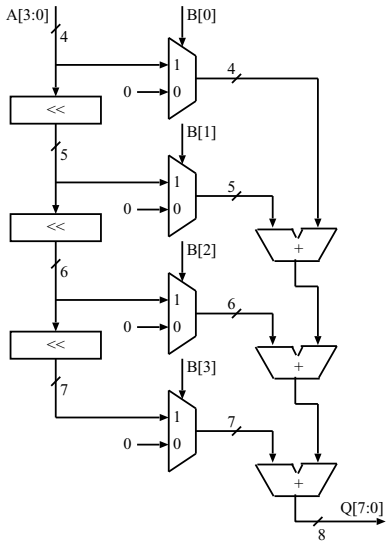

Fig. 5. Multiplier circuit used for the second set of experiments

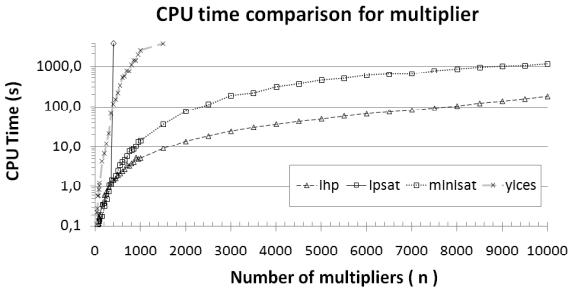

Fig. 6. Solution runtime versus problem size for the multiplier

adders and multiplexors. The output of the multiplier is fixed at a value such that the system will only have one feasible solution. Only the input bus `A[3:0]` was declared as integer on each multiplexer.

Figure 6 shows the CPU runtime comparison for solving the SAT problem versus the number of multipliers. In this case, the CPU time reduction is near one order of magnitude for large problems. Similar results are obtained for the proposed model if internal variables are also declared as integers.

V. Conclusions

In this paper, we have proposed a new solution for RTL SAT problems, based on modeling RTL subsystems as integer hull polyhedrons. We have analyzed and discussed about its usefulness to RTL descriptions. Finally, we have presented experimental results that show the growth of solution runtime versus the problem complexity.

Although integer hull models for RTL subsystems are larger in size than integer ones, they provide a representation of the SAT problem with a lesser degree of complexity. Redundant inequalities are constraints in the model that help to reduce the search by increasing the number of discarded non–integer solutions that would otherwise be solutions of the continuous relaxation. Therefore, the explicit enumeration of these cuts should reduce the number of branches needed to solve the integer problem.

VI. Future Work

Work is in progress to develop an algorithm that generates automatically the integer hull for a given Boolean network, or more generally, any Boolean SAT problem expressed in 3SAT. This algorithm will analyze the matrix A of the polyhedron $P = \{x \in \mathbb{R}^n_+ \ : \ Ax \leq b \ \}$ directly, and will identify those intersections that need to be cancelled with cutting planes.

References

[1] M. Davis and H. Putnam, "A computing procedure for quantification theory," *Journal of the ACM*, vol. 7, no. 3, pp. 201–215, Jul. 1960. [Online]. Available: http://www.acm.org/pubs/articles/journals/jacm/1960-7-3/p201-davis/p201-davis.pdf

[2] M. Davis, G. Logemann, and D. Loveland, "A machine program for theorem–proving," *Communications of the ACM*, vol. 5, no. 7, pp. 394–397, Jul. 1962. [Online]. Available: http://www.acm.org/pubs/articles/journals/cacm/1962-5-7/p394-davis/p394-davis.pdf

[3] J. P. M. Silva and K. A. Sakallah, "GRASP: A search algorithm for propositional satisfiability," *IEEE Trans. Computers*, vol. 48, no. 5, pp. 506–521, 1999.

[4] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient sat solver," in *DAC*. ACM, 2001, pp. 530–535.

[5] N. Een and N. Sorensson, "An extensible sat-solver [extended version 1.2]," in *In Theory and Applications of Satisfiability Testing (SAT'03)*. Springer, 2003.

[6] H. Jain, D. Kroening, N. Sharygina, and E. M. Clarke, "Word-level predicate-abstraction and refinement techniques for verifying RTL Verilog," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 2, pp. 366–379, Feb. 2008.

[7] L. Lingappan and N. Jha, "Satisfiability-based automatic test program generation and design for testability for microprocessors," *IEEE Trans. VLSI Syst.*, vol. 15, no. 5, pp. 518–530, May 2007.

[8] L. Lingappan, S. Ravi, and N. Jha, "Satisfiability-based test generation for nonseparable RTL controller-datapath circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 3, pp. 544–557, March 2006.

[9] F. Fallah, S. Devadas, and K. Keutzer, "Functional vector generation for HDL models using linear programming and Boolean satisfiability," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 8, pp. 994–1002, 2001.

[10] Z. Zeng, P. Kalla, and M. Ciesielski, "LPSAT: A unified approach to RTL satisfiability," in *Proc. of the 2001 Design, Automation and Test in Europe Conference and Exhibition (DATE'01)*, Mar. 2001, pp. 398–402. [Online]. Available: citeseer.ifi.unizh.ch/zeng01lpsat.html

[11] H. M. Sheini and K. A. Sakallah, "From propositional satisfiability to satisfiability modulo theories," in *In Theory and Applications of Satisfiability Testing (SAT'06)*. Springer, 2006, pp. 1–9.

[12] ——, "Ario: A linear integer arithmetic logic solver," *Formal Methods in Computer Aided Design, 2006. FMCAD '06*, pp. 47–48, Nov. 2006.

[13] B. Dutertre and L. D. Moura, "The YICES SMT solver," SRI International, Tech. Rep., 2006.

[14] B. Dutertre and L. de Moura, "A fast linear-arithmetic solver for DPLL(T)," in *Proc. of the 18th Computer-Aided Verification conference*, ser. LNCS, vol. 4144. Springer-Verlag, 2006, pp. 81–94.

[15] K. Fukuda and A. Prodon, "Double Description method revisited," in *Selected papers from the 8th Franco-Japanese and 4th Franco-Chinese Conference on Combinatorics and Computer Science*. London, UK: Springer-Verlag, 1996, pp. 91–111.

[16] G. S. Tseitin, *On the complexity of derivations in propositional calculus*, ser. Structures in Constructive Mathematics and Mathematical logic. New York: Consultants Bureau, 1968, vol. Part II, pp. 115–125.

[17] CPLEX Optimization Inc., "Using the CPLEX linear optimizer and mixed integer optimizer," Incline Village, Nevada, 1992.