A Constraint Programming Approach to Model ATPG Related Problems

Francisco Azevedo & Pedro Barahona
{fa,pb}@di.fct.unl.pt - Universidade Nova de Lisboa

***Abstract**. Combinatorial problems in Electronics Computer Aided Design (ECAD), namely in the field of Automatic Test Pattern Generation (ATPG), have usually been handled either by specific tools or by mapping them into a general problem solver, requiring the consideration of (at least partially) a duplication of the circuits. Constraint Logic Programming (CLP) with special domains was proposed as an useful alternative. In this paper we define an 8-valued logic for modelling the problem of diagnosis, and show that set constraints can generalise it to handle ATPG optimisation problems.*

1. Introduction

The usefulness of CLP in the ECAD area was already exemplified and discussed [1]. In addition to the basic problem of finding an input test pattern for a specific faulty line, there are some related and more complex problems that have usually been handled either by specific tools or by modelling them in some appropriate form to be subsequently dealt with by a general problem solver (e.g. a Boolean SAT-based solver [2]). Current techniques to deal with the problem of diagnosis use some sort of circuits duplication [3,4], that significantly increases complexity.

In this paper we show how to associate a set of fault dependencies to each digital signal in a combinational circuit. This association is used to model three ATPG related problems concerning a given set of diagnoses *D*, (where a diagnosis *d* is a set of faults, and each fault is a stuck line): **1) Diagnosis**. Find an input test pattern *i* that differentiates two diagnoses $\{d_1,d_2\} = D$, i.e. find an input that yields a different value, in at least one output bit, for diagnoses d_1 and d_2; **2) Maximisation**. Find an input *i* which is a test pattern for a maximum number of diagnoses in *D*; **3) Minimisation**. Find a minimal set *S* of input test patterns that cover all diagnoses in *D*.

Our proposed models require the specification of specialised logics, as well as constraint solvers that can handle non-conventional domains. In [5], an 8-valued logic to differentiate two alternative theories was presented. Applied to circuits diagnosis, there are 3 circuit models involved: *N*, a model with no faults; and T_1 and T_2 two alternative models, whose changes with respect to theory *N* consist of some (faulty) lines. A "bit" *Z* has thus 2^3 possible values:

N	0	0	0	0	1	1	1	1
T_1	0	0	1	1	0	0	1	1
T_2	0	1	0	1	0	1	0	1
Z	0	d_2-0	d_1-0	m-0	m-1	d_1-1	d_2-1	1

We then have constant values (0/1), *d*-values (dependent on one diagnostic theory, i.e. on its faults) and *m*-values (dependent on both theories). For instance, *m-0* means a normal 0 value, but an actual value of 1 if any of T_1 or T_2 is correct, whereas d_1*-0* only depends on T_1.

Boolean operations were defined in this logic, together with possible faulty lines referred to below as S-buffers.

2. Generic Modelling

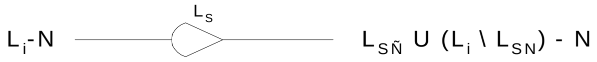
In this section we generalise the modelling technique for more than two alternative diagnostic models. A signal in the circuit can thus be represented as a pair *L-N*, where *L* is a set of diagnoses and *N* is a Boolean value, with the meaning that normally it has value *N* (0 or 1) but it depends on any of the diagnoses in *L*, i.e. if one such diagnosis in *L* is correct (i.e. lines are really faulty), the value is the opposite of *N*. For instance, *{{f/0,g/0}, {f/1,h/0}} - 0* is normally 0, but it becomes 1 if either *f* is stuck-at-0 and *g* stuck-at-0, or if *f* is stuck-at-1 and *h* stuck-at-0. Hence, ∅*-N* represents a constant value *N*, independent of any fault.

Normal Gates. We exemplify the gates' behaviour with the basic 'not' and 'and' gates. The negation of *L-N* is simply *L*. As for the and-gate, three distinct situations may arise:

	L_1-1 ∧ L_2-1	**L_1-0 ∧ L_2-0**	**L_1-0 ∧ L_2-1**
Result:	$L_1 \cup L_2$ -1	$L_1 \cap L_2$ -0	$L_1 \setminus L_2$ -0

S-Buffers. An S-buffer for line *g* has associated a set of the possible diagnoses where *g* appears as stuck. We call this set L_S. Since *g* can appear either as stuck-at-0 or stuck-at-1, we split this set in two ($L_S = L_{S0} \cup L_{S1}$), one for each type of diagnoses (e.g. $L_{S0} = \{diag \in D\text{: } g/0 \in diag\ \}$).

We then have our S-buffer model:



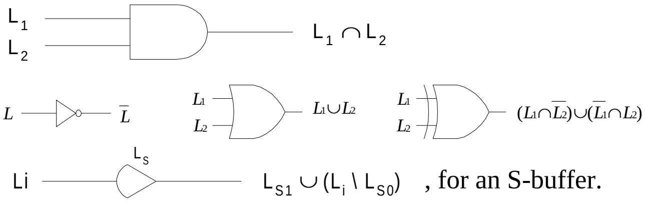
Solving. To solve the diagnosis problem and differentiate diagnoses $D=\{d_1,d_2\}$, either $\{d_1\}$-N or $\{d_2\}$-N must be present in a circuit output bit (or, equivalently, L-N, where #L=1). For problem number 2 (maximisation), the goal is *maximise*(#$\cup_b L_b$) where b ranges over each circuit output bit b with signal L_b-N_b. The third problem (minimisation) is a typical set covering problem.

2.1 Improved Representation with Sets

With the previous representation, both the set and the Boolean value can be variables, which makes the representation hard to handle. Instead, we can represent L-0 simply as L, and L-1 as $\overline{L}$ (the complement of L, with respect to D), by means of transformation *transf*:

$$transf\,(S)=\begin{cases}L, & S=L-0\\ \overline{L}, & S=L-1\end{cases}$$

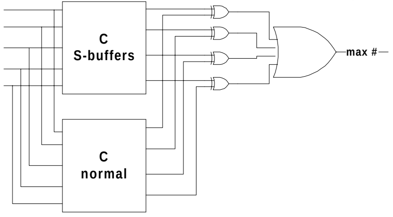
As such, gates are modelled with the usual set operations:



, for an S-buffer.

Solving. To solve the diagnosis problem using just sets, we can still simply ensure that a set L with cardinality 1 is present at a circuit output bit. With the set $D=\{d_1,d_2\}$ of diagnoses to differentiate, it is equivalent to have an L (#L=1) as an output bit in the sets representation, or to have an L-N (#L=1) in the previous representation.

For the maximisation problem, we must know exactly if an output signal depends on its set or on its complement. This is done by recovering the lost information (the normal output values) as shown in the next figure.



Circuit c with S-buffers is kept, and we add the same circuit but with all lines normal (i.e. with no S-buffers). Circuits share the inputs. The xor-gates in the output bits receive a set L from the faulty circuit and either ∅ or D (the universe) from the normal one. Therefore, L is kept as L if the normal value was 0, and recovered to $\overline{L}$ if the normal value was 1. A maximisation on the union of these real dependencies can now be performed to reach our goal.

Since we now only have set constraints, these can be actively used by a set constraint solver and choice-points are avoided. Also, labelling (the exponential component of search) is only performed at the circuit with S-buffers, in contrast with Boolean SAT approaches, which consider one extra circuit for each diagnosis, which is unacceptable, in practice, for a large set of diagnoses [2].

The minimisation problem is a meta-problem: it involves sets of solutions to set problems and is still an open problem.

3. Conclusions and Further Research

We have shown how a constraint programming approach is able to model a number of ATPG related problems. Clearly, the practical interest of this work depends on the ability to develop adequate constraint solvers to deal with the domains that have been used (8-valued logic and sets [6]). In both cases we have been developing constraint solvers over existing CLP languages (SICStus and ECLiPSe) and checking our systems with the ISCAS standard circuits. We expect to be able to prove that the results are competitive with SAT based approaches (as shown in [7] in the pure ATPG problem), at least in some types of problems. Meanwhile, we reckon that the expressive power and flexibility of the constraint programming approach makes it very attractive and deserves further research work.

Acknowledgement. The first author was financially supported by "Sub-Programa Ciência e Tecnologia do 2º Quadro Comunitário de Apoio".

References

1 H. Simonis. *Constraint Logic Programming Language as a Digital Circuit Design Tool*, Thesis, 1992.

2 L. G. Silva, L. M. Silveira and J. P. Marques-Silva, *Algorithms for Solving Boolean Satisfiability in Combinational Circuits*, IEEE/ACM Design and Test in Europe Conf. (DATE), 1999.

3 T. Gruning, U. Mahlstedt, H. Koopmeiners, *DIATEST: A Fast Diagnostic Test Pattern Generator for Combinational Circuits*, Procs of ICCAD91, 194-197, 1991

4 I. Pomeranz, S.M. Reddy, *A Diagnostic Test Generation Procedure for Synchronous Sequential Circuits based on Test Elimination*, Procs of ITC98, 1074-1083, 1998.

5 F. Azevedo and P. Barahona. *Generation of Test Patterns for Differential Diagnosis of Digital Circuits (Extended Abstract)*, in CP'98, M. Maher and J.-F. Puget (Eds.), Springer, p. 462, 1998. Long version: ERCIM/COMPULOG Workshop on Constraints, K. Apt, P. Codognet, E. Monfroy (Eds.), 1998.

6 C. Gervet, *Interval Propagation to Reason about Sets: Definition and Implementation of a Practical Language*, Constraints, vol. 1 (3), Kluwer Academic Pub, 191-244, 1997.

7 H. Simonis. *Test Generation using the Constraint Logic Programming Language CHIP*, 6th Int. Conf. on Logic Programming, MIT Press, 101-112, 1989.