Preprint from Proceedings of 9. Workshop GI/ITG/GME ''Testmethoden und Zuverlaessigkeit von Schaltungen und Systemen'', Bremen, Germany, March 1997

Functional Decomposition with Integrated Test Generation

Christoph Scholl

Institute of Computer Science
Albert-Ludwigs-University
79110 Freiburg im Breisgau, Germany
email: scholl@informatik.uni-freiburg.de

Abstract

Functional decomposition is an important technique in logic synthesis, especially for the design of lookup table based FPGA *architectures. We present an approach how to compute test information, e.g. a complete test set (according to the stuck-at-fault model or the cellular fault model), for functionally decomposed circuits. We are able to compute this test information in parallel with logic synthesis by (recursive) functional decomposition.*

1 Introduction

Functional decomposition was introduced by Ashenhurst [1], Curtis [4], Roth and Karp [11, 6]. During last years functional decomposition attracted a lot of interest especially in connection with the design of lookup table based FPGA architectures [10, 7]. There were considerable improvements on the basic decomposition techniques, e.g. with respect to the extraction of common sublogic in the decomposition of multi-output boolean functions [9, 13, 12].

In [8] a simple test generation algorithm was given for a restricted class of decomposed circuits, the so–called cascaded Reed Muller circuits. However, to the best of our knowledge there has not been any general approach so far which generates test sets in parallel with functional decomposition. In this paper we give basis for the computation of test information in the general case.

A boolean function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ with input variables $x_1, \dots, x_p, y_1, \dots, y_q$ $(p + q = n)$ is decomposed with respect to a subset $\{x_1, \dots, x_p\}$ of the input variables according to Figure 1. The function $\alpha : \{0,1\}^p \rightarrow \{0,1\}^r$ is called *decomposition function* and the function $g : \{0,1\}^{r+q} \rightarrow \{0,1\}^m$ is called *composition function.*

If the boolean function is to be realized by an FPGA with n_{LUT}-input lookup tables and if $p \leq n_{LUT}$, α can be realized by r lookup tables (similarly for g). If the number of inputs of α or g is still too large, decomposition has to be applied recursively to α and g.

In this paper we propose a method to compute test information for a functionally decomposed circuit. The underlying fault model can be the (single) stuck-at fault model or the cellular fault model. We show how to extract test information in parallel with the recursive decomposition using a bottom-up computation.

2 Boolean relations

To compute test sets for functionally decomposed circuits we need the concept of boolean relations which was introduced by Cerny [3].

Definition 1 *A relation $F \subseteq \{0,1\}^n \times \{0,1\}^m$ is called* boolean relation *with n inputs and m outputs. Each boolean function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ can be viewed as a boolean relation $R(f)$ with*

$$(\epsilon, \delta) \in R(f) \iff f(\epsilon) = \delta \qquad (\forall\, \epsilon \in \{0,1\}^n, \delta \in \{0,1\}^m).$$

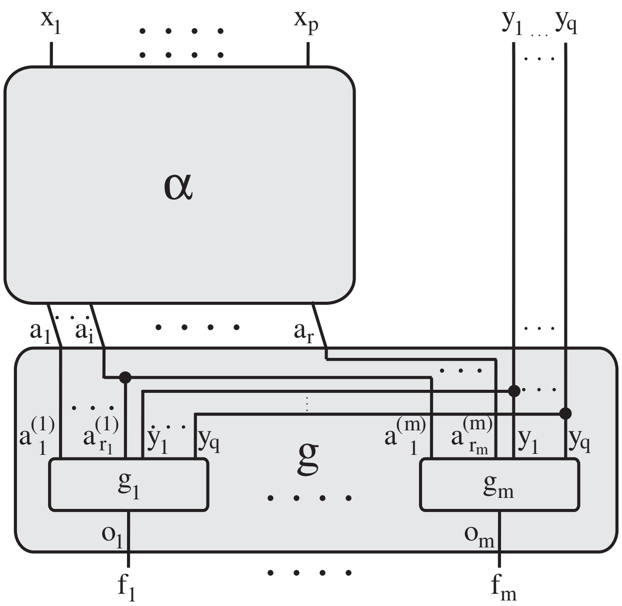

Figure 1: Decomposition of $f : \{0,1\}^n \to \{0,1\}^m$ with respect to $\{x_1, \ldots, x_p\}$.

If $R(f) \subseteq F$ for some boolean function $f : \{0,1\}^n \to \{0,1\}^m$ and some boolean relation F, we say: 'f implements the boolean relation F'.

A boolean relation F can be represented by its characteristic function, i.e. a boolean function χ_F with $\chi_F(\epsilon, \delta) = 1$ iff $(\epsilon, \delta) \in F$.

3 Recursive test pattern generation

In order to compute a complete test set for a functionally decomposed circuit S we make use of so–called freedom relations for subcircuits T of S.

Definition 2 *Let S be a circuit and T a subcircuit of S with n_T inputs and m_T outputs. Let FR_T be the* maximal *relation with n_T inputs and m_T outputs, such that the following holds for all boolean functions $f : \{0,1\}^{n_T} \to \{0,1\}^{m_T}$ which implement FR_T:*

> *If we replace T in S by a realization T' of f, then the function computed by S does not change.*

Under these conditions FR_T is called freedom relation of T with respect to S.

Now we have a close connection between freedom relations and redundant faults: Again let S be a circuit and T a subcircuit of S. Let F be a fault in T and T_F the circuit T with fault F.

> F is redundant (i.e. there is no input vector for S which makes the fault observable at the outputs) iff the function realized by T_F implements the freedom relation of T with respect to S.

To compute test sets in parallel with the recursive decomposition of boolean functions we have to meet the following conditions:

1. For each function that has to be realized by a recursive call of the decomposition procedure we need not only the specification of the function, but also the corresponding freedom relation with respect to the overall circuit.

2. For a bottom-up test generation we must be able to compute tests for function f (see Figure 1) from tests for functions α and g.

The freedom relation of a subcircuit T is the amount of information we need about the behavior of the 'environment' of T in order to generate tests for faults in T. Note that the freedom relation is valid for all faults in T, i.e. it does not depend on the choice of a fault F.

We have derived boolean equations to compute freedom relations of subcircuits (condition 1) and to achieve a bottom-up test generation (condition 2). These equations are applied in parallel with logic synthesis by decomposition.

We use *Binary Decision Diagrams* (BDDs) [2] to represent the characteristic functions of freedom relations. BDDs are directed acyclic graphs where a Shannon decomposition is carried out in each node. They have the advantage that not only manipulations on functions represented by BDDs can be performed efficiently but also BDD representations for many boolean functions occurring in practical examples are of moderate size.

3.1 Details on boolean relations for testing

In the following we will just give a brief impression of how the equations to meet conditions 1. and 2. will look like:

We assume that we decompose a boolean function $f : \{0,1\}^n \to \{0,1\}^m$ into $\alpha : \{0,1\}^p \to \{0,1\}^r$ and $g : \{0,1\}^{r+q} \to \{0,1\}^m$ (see Figure 1).

ad 1. In the first step the freedom relation of the completely specified boolean function f is given by $R(f)$:

$$\chi_{R(f)}(x_1, \ldots, x_n, o_1, \ldots, o_m) = \bigwedge_{i=1}^{m} \overline{o_i \oplus f_i(x_1, \ldots, x_n)}. \tag{1}$$

Later on we derive FR_α from FR_f and g and FR_g from FR_f and α:

$$\chi_{FR_\alpha}(\mathbf{x}, \mathbf{a}) = \forall_{\mathbf{y}}(\exists_{\mathbf{o}}(\chi_{FR_f}(\mathbf{x}, \mathbf{y}, \mathbf{o}) \cdot \chi_{R(g)}(\mathbf{a}, \mathbf{y}, \mathbf{o}))), \tag{2}$$

$$\chi_{FR_g}(\mathbf{a}, \mathbf{y}, \mathbf{o}) = \forall_{\mathbf{x}}(\chi_{R(\alpha)}(\mathbf{x}, \mathbf{a}) \Longrightarrow \chi_{FR_f}(\mathbf{x}, \mathbf{y}, \mathbf{o})) = \forall_{\mathbf{x}}(\overline{\chi_{R(\alpha)}(\mathbf{x}, \mathbf{a})} \vee \chi_{FR_f}(\mathbf{x}, \mathbf{y}, \mathbf{o})). \tag{3}$$

ad 2. If the number of inputs of the function f we have to realize is small enough (e.g. when we stop the recursion), we can compute for each non–redundant fault in the realization of f a test by complete simulation and comparison with the freedom relation of f (with respect to the overall circuit).
Otherwise we have to derive tests vectors for f from test vectors for α and g (see Figure 1):

Non–redundant fault F in α:
Suppose we have a test vector $(\tilde{x_1}, \ldots, \tilde{x_p}) \in \{0,1\}^p$ with faulty output $(\tilde{a_1}, \ldots, \tilde{a_r}) \in \{0,1\}^r$, $(\tilde{x_1}, \ldots, \tilde{x_p}, \tilde{a_1}, \ldots, \tilde{a_r}) \notin FR_\alpha$.
Compute

$$T = \overline{(\chi_{FR_f})_{x_1^{\tilde{x_1}} \ldots x_p^{\tilde{x_p}}}} \cdot (\chi_{R(g)})_{a_1^{\tilde{a_1}} \ldots a_r^{\tilde{a_r}}} \tag{4}$$

and choose $(\tilde{y_1}, \ldots, \tilde{y_q}, \tilde{o_1}, \ldots, \tilde{o_m}) \in ON(T)$.
Then $(\tilde{x_1}, \ldots, \tilde{x_p}, \tilde{y_1}, \ldots, \tilde{y_q}, \tilde{o_1}, \ldots, \tilde{o_m}) \notin FR_f$ and $(\tilde{x_1}, \ldots, \tilde{x_p}, \tilde{y_1}, \ldots, \tilde{y_q})$ is a test for F.

Non–redundant fault F in g: Suppose we have a test vector $(\tilde{a_1}, \ldots, \tilde{a_r}, \tilde{y_1}, \ldots, \tilde{y_q}) \in \{0,1\}^{r+q}$ with faulty output $(\tilde{o_1}, \ldots, \tilde{o_m}) \in \{0,1\}^m$, $(\tilde{a_1}, \ldots, \tilde{a_r}, \tilde{y_1}, \ldots, \tilde{y_q}, \tilde{o_1}, \ldots, \tilde{o_m}) \notin FR_g$.
Compute

$$T = \overline{(\chi_{FR_f})_{y_1^{\tilde{y_1}} \ldots y_p^{\tilde{y_p}} o_1^{\tilde{o_1}} \ldots o_m^{\tilde{o_m}}}} \cdot (\chi_{R(\alpha)})_{a_1^{\tilde{a_1}} \ldots a_r^{\tilde{a_r}}}. \tag{5}$$

and choose $(\tilde{x_1}, \ldots, \tilde{x_p}) \in ON(T)$.
Then $(\tilde{x_1}, \ldots, \tilde{x_p}, \tilde{y_1}, \ldots, \tilde{y_q}, \tilde{o_1}, \ldots, \tilde{o_m}) \notin FR_f$ and $(\tilde{x_1}, \ldots, \tilde{x_p}, \tilde{y_1}, \ldots, \tilde{y_q})$ is a test for F.

Using the formulas above we can use ROBDD operations to compute representations for the characteristic functions of the freedom relations for each recursive call of the decomposition procedure and we can eventually compute tests (with respect to the overall circuit) for each single stuck-at (or cellular) fault in the circuit. This can be done in parallel with logic synthesis by decomposition.

4 Extensions

The method presented so far provides the basis for several interesting extensions.

In the previous section we computed a complete test set for functionally decomposed circuits without giving attention to the size of the generated test set. To *minimize* the test size we can also recursively compute *symbolic representations* of the set of *all test vectors* for each fault by slight modifications of equations 4 and 5. To compute a minimal test set we have to solve a covering problem. This problem can be solved heuristically by repeated use of the *Lmax* algorithm recently suggested by Kam [5].

Application of the presented methods to random pattern testability are focus of current work. An advantage of the methods is the possibility to do an 'incremental' estimation of random testability at various levels of the recursion. If random testability appears to be too bad at some level we can use degrees of freedom in the decomposition process to improve testability: We can choose another 'information encoding' provided by the decomposition function α (i.e. we change α and g in Figure 1 to α' and g' with better testability properties) and we can exploit don't cares to improve testability. Another possibility is the insertion of 'test points' at levels where random testability is not good enough.

References

[1] R.L. Ashenhurst. The decomposition of switching functions. In *Int'l Symp. on Theory Switching Funct.*, pages 74–116, 1959.

[2] R.E. Bryant. Graph - based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, 8:677–691, 1986.

[3] E. Cerny and M.A. Marin. An approach to unified methodology of combinational switching circuits. *IEEE Trans. on Comp.*, 26:745–756, 1977.

[4] H.A. Curtis. A generalized tree circuit. *Journal of the ACM*, 8:484–496, 1961.

[5] T. Kam, T. Villa, R. Brayton, and A. Sangiovanni-Vincentelli. A fully implicit algorithm for exact state minimization. In *Design Automation Conf.*, pages 684–690, 1994.

[6] R.M. Karp. Functional decomposition and switching circuit design. *J. Soc. Indust. Appl. Math.*, 11(2):291–335, 1963.

[7] Y.-T. Lai, M. Pedram, and S.B.K. Vrudhula. BDD based decomposition of logic functions with application to FPGA synthesis. In *Design Automation Conf.*, pages 642–647, 1993.

[8] G. Lee, M.J. Irwin, and R.M. Owens. Test generation in circuits constructed by input decomposition. In *Int'l Conf. on Comp. Design*, pages 107–111, 1990.

[9] P. Molitor and C. Scholl. Communication Based Multilevel Synthesis for Multi-Output Boolean Functions. In *Great Lakes Symp. VLSI*, pages 101–104, 1994.

[10] R. Murgai, N. Shenoy, R.K. Brayton, and A. Sangiovanni-Vincentelli. Improved logic synthesis algorithms for table look up architectures. In *Int'l Conf. on CAD*, pages 564–567, 1991.

[11] J.P. Roth and R.M. Karp. Minimization over boolean graphs. *IBM J. Res. and Develop.*, 6(2):227–238, 1962.

[12] C. Scholl and P. Molitor. Communication Based FPGA Synthesis for Multi-Output Boolean Functions. In *ASP Design Automation Conf.*, pages 279–287, 1995.

[13] B. Wurth, K. Eckl, and K. Antreich. Functional multiple-output decomposition: Theory and implicit algorithm. In *Design Automation Conf.*, pages 54–59, 1995.