BEHAVIORAL SYNTHESIS OF LOW-COST PARTIAL SCAN DESIGNS FOR DSP APPLICATIONS

Sujit Dey Miodrag Potkonjak Rabindra K. Roy
C&C Research Laboratories, NEC USA
Prinecton, NJ 08540

ABSTRACT

Partial scan is a popular design for testability technique for cost-effective sequential ATPG. An efficient partial scan approach selects flip-flops (FFs) in the minimum feedback vertex set (MFVS) of the FF dependency graph, so that loops are broken. Through an analysis of the sources of loops in the data path, this paper proposes a new high-level synthesis methodology to synthesize DSP designs which have low-cardinality MFVS, thereby reducing the cost of partial scan significantly. A test efficiency of 100% could be achieved for all designs synthesized by the proposed approach, requiring a significantly less number of FFs to be scanned compared to the original implementations.

1 Introduction

Automatic test pattern generation (ATPG) of sequential circuits is a very difficult problem [1]. While full-scan design solves the testability problem, it can be very costly. That's why, partial scan design has gained wide acceptance. Cheng and Agrawal [2] used the S-graph, which displays the dependencies among the FFs of a sequential circuit, to indicate that cycles in the S-graph are primarily responsible for sequential ATPG complexity. They presented a partial scan approach which selects FFs in the minimum feedback vertex set (MFVS) of the S-graph. Recently, high level synthesis techniques have been used to generate easily testable data path circuits [3, 4, 5].

We propose a high-level synthesis methodology to synthesize circuits with low-cardinality MFVS for DSP applications. Since these circuits are datapath-intensive, the controller has only a few FFs, which can be scanned without much overhead, thus allowing us to concentrate on the data path of the designs.

A circuit synthesized from behavioral specifications has a natural tendency to contain loops, partly due to the presence of loops in the CDFG, and partly due to hardware sharing used to optimize hardware resources like execution units. A comprehensive analysis of the formation of loops, in circuits synthesized by behavioral synthesis, is performed. The analysis uses the data dependencies and the compatibilities of the operations of the Control Data Flow Graph (CDFG) specification, to develop a regular expression-based representation to identify conditions under which loops will be created.

Based on the loop analysis, simultaneous scheduling and assignment algorithms are presented to implement circuits with reduced MFVS cardinality. To make the low-cardinality MFVS implementation cost effective, the scheduling and assignment algorithms simultaneously optimize for resource utilization.

2 Formation of Loops in the Data Path

In this section, we identify and formulate the formation of different types of loops in the data path. We begin by modeling the data dependencies and the compatibility of the operations of the CDFG by a Data-Dependency and Compatibility Graph (DDCG). Each node in the graph represents an operation of the CDFG. There is a (undirected) compatibility edge between two operations if there is a non-zero probability that both the operations can be assigned to the same module. There is a (directed) data-dependency edge from operation v to operation w if w depends on data produced by v.

Figure 1(a) shows segments of two paths in a CDFG, and Figure 1(b) shows the corresponding data-dependency and compatibility graph. Each compatibility edge $c(v, w)$, shown dotted, is weighted by an estimate of the compatibility between two nodes v and w [6].

The paths of the DDCG can be represented using regular expressions. d^+ represents a path consisting of a sequence of one or more data-dependency edges d. The concatenation

0-7803-1775-0/94 $3.00

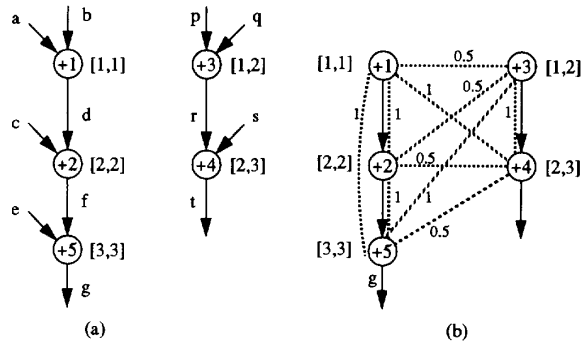

Figure 1: Illustrating the formation of loops: (a) A CDFG, (b) the corresponding DDCG.

of two paths p and q is represented by p.q, or simply, pq. For example, the regular expression cd^+ represents a path starting with a compatibility edge c, followed by one or more occurrence of data-dependency edges. In Figure 1(b), the path $(+_5, +_1), (+_1, +_2), (+_2, +_5)$ can be represented by cd^+.

2.1 Detecting formation of Loops Using DDCG

(1) Data-Dependency Loop: A Data-Dependency Loop is formed in the data path if there exists a cycle of the form d^+ in the DDCG. In other words, if all the edges of a cycle in the DDCG, or the CDFG, are data-dependency edges, then a loop is formed in the data path, irrespective of the register and module assignment.

(2) Assignment Loop: During assignment of operations to modules (EXUs), we say that a compatibility edge is *used* if the two operations associated with the compatibility edge are assigned to the same module. An assignment loop is formed in the data path if there exists a cycle of the form cd^+ in the DDCG, and the compatibility edge c is used during module assignment.

In the DDCG shown in Figure 1(b), there is a cycle $\{(+_5, +_1), (+_1, +_2), (+_2, +_5)\}$ of the form cd^+. Using the compatibility edge $(+_5, +_1)$ to assign the compatible operations $+_5$ and $+_1$ to the same module, creates an assignment loop in the data path. Let the schedule and assignment of the operations be: $\{+_1 : (1, A1), +_2 : (2, A2), +_3 : (2, A1), +_4 : (3, A2), +_5 : (3, A1)\}$. It satisfies the constraint of three control steps, and uses the minimum number of EXUs (2 adders). The resultant data path, shown in Figure 2(a), and its corresponding S-graph, has an assignment loop (RA1,LA2,RA1).

A self-loop is a special case: it can be formed in the data path either by the presence of a data-dependency loop of the form d, or an assignmnt loop of the form cd.

(3) Sequential False Loop: A sequential loop in the data path is termed false when the loop cannot be sensitized under normal operation. A false loop is a special case of a false path. Let the schedule and assignment of the operations of the CDFG in Figure 1(a) be: $\{+_1 : (1, A1), +_2 : (2, A2), +_3 : (1, A2), +_4 : (2, A1), +_5 : (3, A2)\}$. It satisfies the constraint of three control steps, and uses the minimum number of EXUs (2 adders). The resultant data path, shown in Figure 2(b), has two loops.

Consider the loop in Figure 2(b) shown in bold. To sensitize the loop, the required control signals to the multiplexers M1 and M4, c1 and c2, should be $\{c1 = 1, c2 = 0\}$ (or, $\{c2 = 0, c1 = 1\}$) in any two consecutive control steps. However, this necessitates execution of operations $+_4$ followed by $+_2$ (or $+_2$ followed by $+_4$), which is clearly not possible as a normal control sequence. Consequently, the sequential loop can never be sensitized under normal operation, and is a false loop. The S-graph corresponding to the data path has a sequential false loop (LA1,RA2,LA1). Note that the only other loop in the data path is the self-loop (RA2,RA2) which is an assignment loop. However, in our testing model, all control FFs are scanned, hence the illegal control sequences can be achieved by scanning in desired values at scan FFs. Thus all sequential false loops become real loops in our testing model, and must be taken into account.

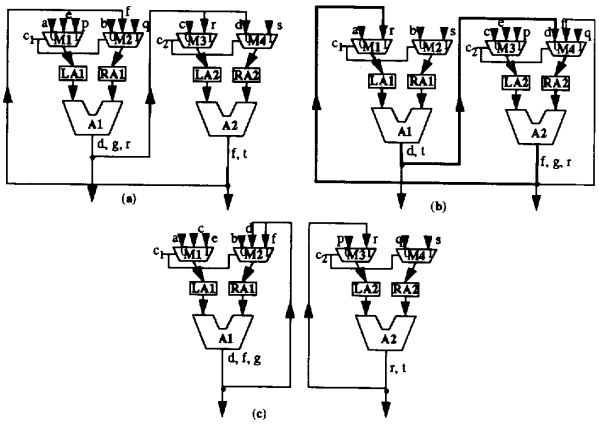

Figure 2: Data Paths formed by different assignments of CDFG in Figure 1(a): (a) Assignment Loop, (b) Sequential False Loop, (c) No Loops except Self-Loops

Let $c = (v, w)$ be a compatibility edge in the DDCG such that there does not exist any data-dependency edge from v to w. If there is a cycle of the form $(cd^+)(cd^+)^+$ in the DDCG, and the compatible edges c in the cycle are used during module assignment, a sequential false loop is formed in the data path. In the DDCG shown in Figure 1(b), there is a cycle $(+_4, +_1), (+_1, +_2), (+_2, +_3)(+_3, +_4)$ of the form $(cd^+)(cd^+)^+$. Assigning the compatible pairs $(+_4, +_1)$ to adder A1, and $(+_2, +_3)$ to adder A2, leads to a sequential false loop as illustrated in the data path in Figure 2(b).

2.2 An Example of Forming a Data Path Without Loops

Having analyzed the conditions which lead to the formation of loops in the data path, we show a sample scheduling and assignment which avoids the formation of loops in the data path. The basic idea is to avoid using any compatibility edge in the DDCG which is a part of a cycle of the form cd^+ or $(cd^+)(cd^+)^+$. Consider the following schedule and assignment which satisfies the performance constraints, and which uses the minimum number of execution units: $\{+_1 : (1, A1), +_2 : (2, A1), +_3 : (1, A2), +_4 : (2, A2), +_5 : (3, A1)\}$. The resultant data path, shown in Figure 2(c), does not contain any loop, besides two self-loops. While one register needs to be scanned to break the loops of the data paths in Figures 2(a) and (b), no register needs to be scanned for the data path in Figure 2(c).

3 Scheduling, Assignment and Allocation Algorithms

We present integrated scheduling and assignment algorithms to synthesize a design such that the implementation has minimum feedback vertex set with a low-cardinality, while satisfying the user specified throughput requirements. The new high level synthesis approach has three phases. The first phase is allocation of the set of execution units (EXUs), exclusively targetting resource utilization. For this task, we use Hyper [7]. In the second phase, we simultaneously schedule and assign each operation of the CDFG while allocating interconnects and registers, so that resulting data path has high global resource utilization and its corresponding S-graph has as small as possible MFVS. In the final phase, the FFs belonging to the minimal feedback vertex set are identified and made scan FFs, using the gate-level partial scan tool OPUS [8].

After the initial allocation of EXUs, we simultaneously schedule and assign each operation of the CDFG, using global testability and resource utilization measures [6]. The aim is to produce a testable data path, by either avoiding the formation of loops in the S-graph, or by ensuring that resulting S-graph has small minimum feedback vertex set. However, equal priority is also given to throughput (number of control steps) and resource utilization, so that the final design is not only cost effective in terms of number of FFs that need to be scanned, but also competitive in terms of hardware cost.

At each iteration of the algorithm, from the operations that have not yet been scheduled and assigned, an operation op_i with the smallest slack (ALAP - ASAP) is selected. The set of (module, control step) pairs, $\{(M_i, C_i)\}$, where the module belongs to the set of modules to which the operation can be assigned and the control step belongs to the set of control steps in which the operation can be scheduled, are identified. For each pair, the cost in terms of the size of MFVS, resource utilization and flexibility for scheduling and assignment of subsequent operations, is computed [6]. Subsequently, a pair with the smallest cost is selected. Details of the algorithm can be found in [6].

4 Experimental Results

We synthesized the following benchmarks from DSP applications: (1) 3rd order cascade IIR Filter (3rdIIR) [9] (2) Speech Filter (Speech) [9], (3) MA Lattice Filter (MAL) [9], and (4) the popular 5th order elliptical wave digital filter (EWF) [10]. In the sequel, **O** refers to the original implementation using conventional high level synthesis techniques to ensure maximal throughput and minimal number of execution units. **SFT** refers to the low-cardinality MFVS implementation, using the new synthesis approach.

Design	B	CS	A	M	Reg		Mux		Inter	
					O	SFT	O	SFT	O	SFT
3rdIIR	16	5	2	3	11	11	12	8	12	9
Speech	20	17	2	3	12	12	20	9	20	9
MAL	16	7	2	2	10	9	10	8	11	10
EWF	16	17	3	3	23	24	29	32	20	27

Table 1: Characteristics of the Designs Synthesized

Table 1 shows the characteristics of the original and the SFT implementations for each benchmark synthesized. The column **B** refers to the word length. Both the original and the SFT implementation takes the same number of control steps **CS** achieving maximal throughput, and need the minimum number of adders **A** and multipliers **M**. The number of registers **Reg**, multiplexers **Mux** and interconnects **Inter**, are shown in Table 1 for the original and SFT implemen-

tations. They reflect the area overhead needed, if any, for low-cardinality MFVS implementation.

Design	Bits	Feedback Vertex Set			
		RT-Level (Regs)		Gate-Level (FFs)	
		Orig	SFT	Orig	SFT
3rdIIR	16	2	1	48	16
Speech	20	3	1	60	20
MAL	16	3	1	48	16
EWF	16	14†	9†	240	163

†: Lower bound

Table 2: Feedback Vertex Set of S-graphs at Register-Transfer and Gate Level

Table 2 shows the size of the feedback vertex sets required to break all the loops, except self-loops, of the S-graphs of the original and SFT implementations. At the RT-level, the size of the minimum FVS is reported (in terms of registers), except for EWF, where the lower bounds of the FVS are reported. At the gate-level, OPUS [8] was used to identify the FVS of the circuits, consisting of FFs, as opposed to registers. Table 2 shows that for each design, the size of the FVS needed for the SFT implementation is significantly less than the FVS needed for the original implementation. In the case of 3rdIIR, only 16 FFs are needed to break all loops, except self-loops, for the SFT circuit, compared to 48 FFs needed for the original circuit.

Design	Type	Faults		FC%	TE%	CPU (secs)
		Total	Abort			
3rdIIR	Orig	7112	6732	1	5	26200.8
	SFT	6510	6104	1	6	22512.5
Speech	Orig	10004	9677	0	3	23883.7
	SFT	8514	8186	0	4	19614.2
MAL	Orig	6192	191	93	97	1159.6
	SFT	5928	71	95	99	656.0
EWF	Orig	9088	8879	0.3	2	25668.0
	SFT	9791	9514	0.2	3	29666.0

Table 3: Sequential ATPG without Partial Scan

A gate-level sequential ATPG tool, HITEC [1], was used to identify the testability of the circuits. Table 3 reports the ATPG results for the original and SFT implementations, without using partial scan. The total number of faults, the number of aborted faults, the percentage fault coverage (**FC%**), the percentage test efficiency (**TE%**), and the ATPG time in seconds on a SUN Sparcstation 2 are reported. The Table shows that in most cases, both the original as well as the SFT implementations are very hard to test.

Table 4 reports the ATPG results for each circuit after scanning the FFs of the feedback vertex set identified by

Design	Type	FFs		Faults		FC%	TE%	CPU (secs)
		Total	Scan	Total	Abort			
3rdIIR	Orig	176	48	7112	3	96	100	94.5
	SFT	176	16	6510	10	96	100	98.7
Speech	Orig	240	60	10004	3	97	100	163.9
	SFT	240	20	8514	22	96	100	192.8
MAL	Orig	160	48	6192	2	97	100	63.7
	SFT	144	16	5928	5	96	100	86.1
EWF	Orig	368	240	9088	0	98	100	209.1
	SFT	384	163	9791	0	97	100	183.3

Table 4: Sequential ATPG after scanning FFs in Feedback Vertex Set

OPUS. Column **FFs** shows the total number of FFs (**Total**), and the number of FFs that needed to be scanned (**Scan**). As expected, the effect of scanning the FFs of the FVS is remarkable: a test efficiency of 100% could be achieved for all the circuits. However, to achieve the high test efficiency, the SFT circuits need a significantly less number of FFs to be scanned compared to the original circuits.

5 Conclusions

We presented a methodology for behavioral synthesis of low-cost partial scan DSP ASICs using comprehensive analysis of loop formation. The experimental results clearly demonstrate the effectiveness of the proposed approach.

References

[1] T. M. Niermann and J. H. Patel . HITEC: A Test Generation Package for Sequential Circuits. In *Proc. EDAC*, pages 214–218, 1991.

[2] K.T. Cheng and V.D. Agrawal. A Partial Scan Method for Sequential Circuits with Feedback. *IEEE Transactions on Computers*, 39(4):544 – 548, April 1990.

[3] T. C. Lee, N. K. Jha, and W. H. Wolf. Behavioral Synthesis of Highly Testable Data Paths under Non-Scan and Partial Scan Environments. In *Proc. Design Automation Conf.*, pages 292–297, 1993.

[4] S. Dey, M. Potkonjak, and R. Roy. Exploiting Hardware Sharing in High Level Synthesis for Partial Scan Optimization. In *Proceedings of the International Conference on Computer-Aided Design*, pages 20 – 25, November 1993.

[5] J. Steensma and W. Geurts and F. Catthoor and H. De Man. Testability Analysis in High Level Data Path Synthesis. *Journal of Electronic Testing: Theory and Applications*, 4(1):43 – 56, Februray 1993.

[6] S. Dey, M. Potkonjak, and R. Roy. Behavioral Synthesis of Partial Scan Designs with Low-Cardinality Minimum Feedback Vertex Sets. Technical report, C&C Research Labs, NEC USA, May 1993.

[7] J. Rabaey, C. Chu, P. Hoang, and M. Potkonjak. Fast Prototyping of Data Path Intensive Architectures. *IEEE Design and Test*, pages 40 – 51, 1991.

[8] V. Chickermane and J. H. Patel. A Fault Oriented Partial Scan Design Approach. In *Proceedings of the International Conference on Computer-Aided Design*, pages 400 – 403, November 1991.

[9] R.A. Haddad and T.W. Parsons. *Digital Signal Processing: Theory, Applications and Hardware*. Computer Science Press, New York, NY, 1991.

[10] R.A. Walker and R. Camposano. *A Survey of high-level synthesis systems*. Kluwer Academic Publishers, Boston, MA, 1991.