Introspection: A Low Overhead Binding Technique During Self-Diagnosing Microarchitecture Synthesis

Balakrishnan Iyer Ramesh Karri
Department of Electrical & Computer Engineering
University of Massachusetts at Amherst
Amherst, MA 01003.

Abstract: **Introspection**, *a zero-overhead binding technique during self-diagnosing microarchitecture synthesis is presented. Given a scheduled control data flow graph (CDFG) introspective binding exploits the spare computation and data transfer capacity in a synergistic fashion to achieve low latency fault diagnostics with* **near zero area overheads** *without compromising the performance. The resulting on-chip fault latencies are one ten-thousandth* (10^{-4}) *of previously reported system level diagnostic techniques. A novel feature of the proposed technique is the use of spare data transfer capacity in the interconnect network for diagnostics.*

1 Introduction

High performance VLSICs typically use a large number of hardware units. Without some form of fault tolerance, such VLSI systems would be rendered inoperable by even a single fault. Notwithstanding the sophisticated scheduling strategies used in high-level synthesis, all functional units are not fully utilized in all clock cycles (control steps). This spare computation capacity can be exploited to detect and diagnose faulty functional units within the IC with negligible increase in the hardware. The available spare capacity for a wide variety of algorithms is illustrated in Table 1.

Algorithm	# CLK	# Nodes +	# Nodes *	# Nodes -	# FU +	# FU *	# FU -	% Spare Cap. +	% Spare Cap. *	% Spare Cap. -
Arai8	7	16	13	14	7	4	4	67	54	50
Dif10	9	17	20	12	4	4	4	53	44	67
Dit9	8	24	14	12	8	4	3	63	56	50
Lee10	9	17	20	12	4	4	4	53	44	67
Mcm14	13	30	30	34	5	4	7	54	42	63
Wang8	7	13	22	13	3	9	3	38	65	38
Cheb19	18	34	18	0	5	5	0	62	80	0
LMS12	12	23	32	1	5	8	1	62	67	92

Table 1: Spare Capacity for Typical Algorithms

We present **Introspection**[1] a zero-overhead binding technique for synthesising self-diagnosing microarchitectures with low diagnostic latency. During introspective binding the microarchitecture (i) looks within itself, (ii) identifies spare computation and data transfer cycles and (iii) judiciously uses these spare computation and data transfer capacities to detect and diagnose any faulty functional units and interconnect. The fault detection and diagnostic latency of **introspective binding** is of the order of a few machine clock cycles. This is 3 to 4 orders of magnitude lower as compared to previously proposed system level approaches to fault diagnosis. This feature is particularly important in real-time applications where faults must be detected as early as possible to contain/localize the faults within the system.

Interconnect usually dominates the chip area. A novel feature of **introspective binding** is that it harnesses the spare data transfer capacity of the interconnect to carry out the additional data transfers required for fault detection and diagnosis. Consequently, the area overhead of introspective binding is negligible.

We incorporate the fault detection and diagnosability constraints at the register transfer (RT) level of a top-down VLSI design methodology. This is because (i) there is a tight interdependence between the binding of the scheduled CDFG and the fault detection and diagnosability of the synthesized microarchitecture, (ii) the underlying fault model is at the RT level of functional units and (iii) the design decisions made at the architectural level have a profound influence on the cost of the synthesized microarchitecture.

Introspective binding is then formulated as a weighted bipartite matching problem and use the *Hungarian algorithm* [1] to solve it. In the process, the diagnosability of the microarchitecture is maximized The approach is flexible enough to be incorporated into any existing high-level synthesis system.

[1] Webster defines Introspection as "from introspectus, pp. of introspicere to look inside, to look within oneself"

33rd Design Automation Conference ®

DAC 96 - 06/96 Las Vegas, NV, USA

1.1 Related Research

The idea of utilizing the spare capacity in multiprocessor/multicomputer systems to execute redundant tasks was suggested in [2], where the detection and diagnosis was done in software at the system level. Hardware based detection and diagnostic schemes for VLIW (Very Long Instruction Word) and Super-scalar processors have been recently proposed by Blough *et. al.* [3]. However, this scheme requires specialized compilation of the source code to take advantage of the detection and diagnosability afforded by the modified architecture. Moreover, the error detection is at the software instruction level which typically consumes several hundred machine clock cycles which results in a detection and diagnostic latency in the order of several thousands of clock cycles as opposed to the proposed scheme which has a latency of few clock cycles.

Traditionally the problems of area and performance optimization have been addressed during microarchitectural synthesis [4]. More recently, other important goals, such as power [5], testability [6], and fault tolerance [7, 8, 9, 10] have also been addressed. In [7] an area efficient fault detection mechanism has been incorporated during scheduling. In [8], one of the first fault-tolerant binding schemes was reported. Guerra et. al. [9] present a high level synthesis technique for efficient built-in-self-repair to enhance the yield of VLSICs. Iyer et. al. [10] propose the use of programmable interconnect to overcome permanent fabrication-time faults.

2 Introspective Binding

Using an illustrative example, we will show how introspective binding achieves diagnosis with low fault-latency with almost zero hardware overhead.

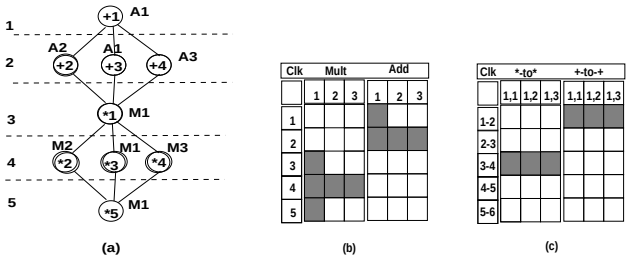

Figure 1: (a) An Example CDFG with scheduling and binding information (b) checkerboard visualization of the spare computation capacity. (c) checkerboard visualization of the spare data transfer capacity.

Consider the scheduled CDFG consisting of four add operations $(+1, +2, ..., +4)$ and five multiply operations $(\star 1, \star 2, ..., \star 5)$ as shown in Fig. 1(a). Assuming that each operation can be done in one cycle, the fastest schedule is shown in Fig. 1(a) (with the horizontal dotted lines delineating the clock cycles). Furthermore, Fig. 1(a) shows a possible operation-to-operator binding by annotating each node with the operator on which it is being executed. For example, operation $+1$ is performed on adder $A1$ while operation $\star 3$ is performed on operator $M1$. From the figure it can be seen that three adders $(A1, A2, A3)$ and three multipliers $(M1, M2, M3)$ are required to implement this schedule. It is important to notice that not all operators are being used during every clock cycle. This can be visualized using a checkerboard representation shown in 1(b). In this checkerboard, the functional units are shown along the x-axis while the clock cycles are shown along the y-axis. A shaded cell (i, j) denotes that the i^{th} functional unit is being used in the jth clock cycle. The checkerboard in Fig. 1(b) corresponds to the binding shown in Fig. 1(a). Notice that out of the fifteen multiply computation cycles available only five are being used. Similarly, out of the fifteen add computation cycles available only four are used. The unshaded cells constitute the **spare computation capacity**.

Although this **spare computation capacity** can be used to diagnose (i.e. identify) any faulty functional unit it should be realized that the ensuing interconnect overhead is critical in VLSI implementations. Consequently, the self-diagnosis technique should not entail additional interconnect overhead. The obvious question is, "is there any spare capacity in the interconnect and if so can it be exploited ?" Returning to Fig. 1(a), observe that towards implementing this design, several functional unit-to-functional unit interconnections are necessary. The checkerboard in Fig. 1(c) shows some selected point-to-point interconnects[2] and their spare capacity. The interconnects are shown along the x-axis while the clock boundaries are shown along the y-axis. For example, the dedicated interconnect between the output of adder $A1$ and the input to adder $A2$ is used only at the boundary between clock cycles 1 and 2.

The spare data transfer capacity and the spare computation capacity can be harnessed in a synergistic fashion to implement on-chip diagnosis with almost zero hardware and interconnect overhead.

For example, consider replicating operation $+3$ and executing each of these replicated copies on a distinct adder as shown in Fig. 2(a). The adders performing

[2] a point-to-point interconnection style is assumed only for simplifying the explanation.

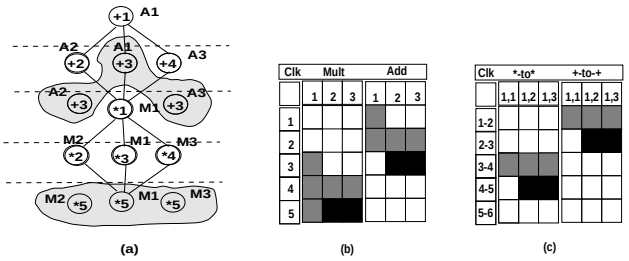

Figure 2: (a) CDFG with self-diagnosis. (b) checkerboard visualization of the spare computation capacity. (c) checkerboard visualization of the spare data transfer capacity.

these computations are $A1$ in clock cycle two and $A2$ and $A3$ in clock cycle three. The outputs of the three adders (shown enclosed in a shaded region for clarity) can be compared pairwise using voters. Since we assume that a single functional unit is faulty, the pairwise comparisons $(A1, A2)$, $(A2, A3)$, and $(A1, A3)$ yield a wealth of useful diagnostic information. For example, if $(A1, A2)$ and $(A2, A3)$ disagree, then $A2$ can be diagnosed to be the faulty functional unit. Similarly, if $(A1, A2)$ and $(A1, A3)$ disagree, then $A1$ can be diagnosed to be faulty. Notice that we need the participation of at least three distinct hardware units for successfully diagnosing the faulty function unit from among them. The checkerboards in Fig. s 2(b)-(c) are darkly shaded to show the spare computation capacity and the spare data transfer capacity that have been used for diagnosis. Since only the spare capacities have been used, the **diagnosis function did not entail any hardware and interconnect overhead**.

Fault-Latency and Fault-Coverage: In Fig. 2(a), a faulty adder can be diagnosed only at the end of clock cycle 3 although the computations required for diagnosis have started in clock cycle 2. Consequently, the time for diagnosing a faulty adder is one clock cycle. The latency of diagnosis can be reduced if enough spare capacity is available. For example, as shown in Fig. 2(a), faults in all multipliers can be diagnosed with zero latency in clock cycle five. These **on-chip latencies are quite small when compared with the tens of thousands of clock cycles required for system level diagnosis.** As far as fault coverage is concerned, the number of functional units that can be diagnosed is dependent on the available spare capacity. It being abundant, the proposed technique has a high fault-coverage.

Area and Time Overhead: We now provide a brief evaluation of the additional hardware overhead of the proposed approach to self-diagnosis. The extra hardware requirements include voter units, the associated registers and the additional control circuitry. By implementing the voter units using two levels of XOR logic gates and glue logic very little silicon area is consumed. Similarly, by ensuring that the fault diagnosis latency is minimal, the number of additional registers can be minimized. In Fig. 2(a), the only overhead apart from one voter unit is the additional register required to hold the result of operation $+1$ till clock cycle 3 when it is consumed by adders $A2$ and $A3$. No extra registers are required for diagnosing the multipliers since the result of operation $\star 3$ is consumed immediately in clock cycle 5 by the three multipliers $M1$, $M2$, and $M3$. The proposed diagnosis scheme does not entail any performance degradation. This is because the values required by the voter can be written to the appropriate registers in parallel with writing it to the appropriate functional unit register file(s). Also, the voter can operate asynchronously with a propagation delay of the order of two gate delays.

3 Fault and Architectural Model

3.1 Fault Model

In this paper we assume a single fault model, wherein at most one functional unit can fail. Furthermore, we assume that there are no failures in the interconnect and the register files. Since fault diagnosis requires that the operation be carried out on three disjoint hardware units, there need to be at least three copies of each of the functional unit that we wish to diagnose. However, these three copies need not be exact replicas of each other and they may differ in terms of silicon area and number of clock cycles required to carry out an operation. It is also important to understand that a functional unit is deemed to be faulty if and only if one or more valid combination of the input operands presented to the unit produces a faulty result. This implies that a faulty functional unit may not produce a faulty result for a given set of inputs, i.e., the error may be **masked**. This implies that detection and diagnosis require that all the operations in the CDFG be validated. Since the spare capacity is limited by the hardware availability, validation of all the operations in the CDFG may not be possible.

3.2 Architectural Model

The architectural model that we use in this paper is very similar to the Hyper [11] architectural model and is shown in Fig. 3. The architecture consists of two main parts – one consisting of the functional units (FU) and the Input/Output buses and the other part consists of the voter circuitry with its associated register files for fault detection and self diagnostics. Each

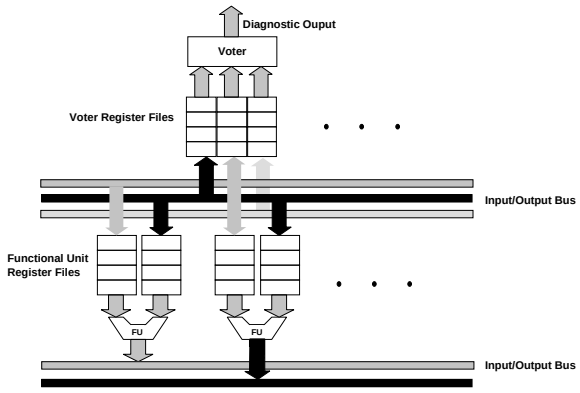

Figure 3: Hardware Model for Introspective Binding.

functional unit consists of two register files which supply the inputs to the functional unit and also receive the output from other functional units. The FU then operates on the input operands and writes the result to a bus which transports the result to the register file(s) of functional units which use the value in subsequent clock cycles. The results are also fed to the appropriate register files of the voter units using the idle buses. The voter reads the three results for the operation executed on three disjoint hardware units and compares them to detect and diagnose faults (if any) in any one of the three units. Since the results are transported to the voters over the same interconnect network, the voter can detect any faults (shorts or opens) in the bus and interconnect.

4 Problem Formulation

Formally, the problem can be stated as follows :
Given a scheduled CDFG, derive a binding which ensures fault detection and self-diagnostics in the resulting microarchitecture with minimum detection and diagnostic latency, and minimizing the hardware overheads subject to no degradation in performance.

4.1 Introspective Binding as Weighted Bipartite Matching

As explained in the previous section, introspective binding accepts a scheduled CDFG and performs the binding that ensures fault detection and diagnostics with minimal latency while minimizing the hardware overheads due to the voter circuitry. Several observations that capture the effect on the hardware overheads due to fault detection and self-diagnostics forms the basis of our algorithm :

1 The number of registers required at the voter(s) can be minimized by minimizing the fault diagnostic latency, i.e., the number of control steps between the execution of the primary copy and the last of the two diagnostic copies of the operation. This also reduces the size of the lifetime of the input variables to the functional units performing the secondary copies of the operations in the CDFG. Therefore, minimizing the fault diagnostic latency reduces the register file size at the voter and the functional units.

2 The number of voters can be minimized by distributing the number of voting operations uniformly over the control steps. The schedule also has a very large impact on the number of voters required. Furthermore, there is a trade-off between the number of voters and the register file size at the voter; minimizing the number of voter operations across the control steps requires additional registers to store the inputs to the voter.

3 Introspective binding maximizes the number of CDFG operations that are diagnosed (executed on three **disjoint** hardware units and voted). The binding algorithm must enforce the hardware disjointness constraints.

4 The algorithm must ensure that all functional units in the design participate in at least one diagnosis operation. This is done by giving a higher weight to operations involving functional units that have not yet participated in diagnosis.

Initially, a two-dimensional array $C_{i_t}[1..n][1..n]$, (where n is the number of functional units of type i in a given control step t) is constructed. The value $C_{i_t}[j][k]$ is the cost function of binding operation j to functional unit k. The cost function is computed for the operations in the primary copy (PC), the replicated operations (RC) and for violation of hardware disjointness or bus availability constraints (CV) as follows:

$$C_{i_t}[j][k] = \begin{cases} \alpha \times (\beta_1 - R_c(op_{jk})), & \text{PC} \\ \gamma(k) \times (\beta_2 - H_c(op_{jk})), & \text{RC} \\ 0, & \text{CV} \end{cases} \quad (1)$$

where, α, β_1, and β_2 are parameters. Parameters α is a user defined weighting parameter. Parameters β_1 and β_2 are suitably large positive value and are also user defined.
The function $\gamma()$ gives higher weight to the functional units that have not yet participated in diagnosis activity. Thus, the function $\gamma()$ encourages the formation of new diagnostic triples and also ensures that all functional units are diagnosed at least once. $R_c(op_{jk})$ denotes the cost of register and interconnect incurred

when performing operation j on unit k. $H_c(op_{jk})$ denotes the register, interconnect and voter unit overheads incurred from executing a replicated copy of operation j on unit k. By definition, the cost function favors operations with low hardware overhead. The bus availability and hardware disjointness constraints for the replicated operations are imposed by setting the cost function to 0 for such bindings. Let N_{i_t} be the set of primary operations of type i scheduled in control step t and D_{i_t} be the set of operations of type i that have not been diagnosed in or prior to control step t. The algorithm is outlined in Algorithm 4.1.

Algorithm 4.1
$D_{i_0} \leftarrow \emptyset,\ N_{i_0} \leftarrow \emptyset\ \forall i$
while $(t < C)\{$
  /* C = # *of control steps in the schedule* */
  $D_{i_t} \leftarrow N_{i_t} \cup D_{i_{t-1}}$
  $\forall\ j \in N_{i_t} \cup D_{i_t}$ *compute* $C_{i_t}[j][k] \forall\ k \in [1, n]$
  Sort D_{i_t} *descending on* $\max_k\ C_{i_t}[j][k] \forall\ j \in D_{i_t}$
  $S_{i_t} \leftarrow$ *top* $(n - |N_{i_t}|)$ *non-zero entries of* D_{i_t}
  $N \leftarrow N_{i_t} \cup S_{i_t},\ D_{i_t} \leftarrow D_{i_t} - S_{i_t}$
  $Hungarian(n, N, C_{i_t})$
  $t \leftarrow t + 1$
$\}$ /* *end while* */

The problem of binding is thus posed as a maximal weighted bipartite matching and solved using the Hungarian Method [1]. The vertices of the weighted bipartite graph are the n functional units, the N_{i_t} primary operations and the $n - N_{i_t}$ replicated operations chosen to be implemented in control step t. If functional unit FU_j is capable of performing operation i, then there is an edge e_{ij} between the corresponding edges in the graph. The weight w_{ij} associated with edge e_{ij} is defined by the value of the cost function.

5 Results

The binding algorithm has been implemented within the Hyper synthesis environment. We compare the area of the chip with introspective binding with that synthesized by Hyper using the low-power library.

The results for introspective binding for a number of different examples are presented in Table 2. The second column indicates the number of clock cycles in the schedule. The third column indicates the number of nodes in the CDFG. The next three columns indicate the number of operations of different types in the CDFG. The number of functional units available is indicated in the following three columns. The next three columns indicate the number of functional units that have been involved at least once in a diagnostic triple to perform a self diagnostics. The following three columns indicate the number of operations in the CDFG that have been diagnosed as a result of introspective binding. The next column indicates the number of voters that were required for introspective binding. The silicon area (in mm^2) of the original chip (without introspective binding) and the chip with introspective binding are presented in the next two columns. The last column indicates the percentage area overhead due to introspective binding.

From the results in Table 2, we note that the area overheads for introspective binding are extremely low with most of the values clustered around 1% and all values being under 5%. In all of the examples, all the functional units have been involved in at least one diagnostic check, i.e., each of the functional units has been included in at least one diagnostic triple. The only exception was in the case of FFT5, where out of the five adder units in the design only four could be used for diagnostic purposes. This is because of two reasons (i) the schedule has only four control steps and does not afford much scope for full diagnostics, (ii) the addition operations are clustered in the last few control steps which again limits the possibility of full diagnostics. In some of the examples there are less than three functional units of a given type. As stated earlier, in this case no diagnostics are possible on the functional units.

In the course of diagnosing all the functional units in the design, introspective binding tries to satisfy a secondary objective, i.e. to diagnose as many nodes in the CDFG consistent with the goal of near zero overhead in silicon area. The number of nodes in the CDFG that are diagnosed by introspective binding is also indicated in Table 2. From the table it can be observed that there is a wide variation in the number of operations in the CDFG that could be diagnosed. For example, in the case of the LMS algorithm, all 32 multiplication operations could be diagnosed whereas only 13 of the 23 addition operations could be diagnosed. This is because of the following reasons (i) the available spare capacity of the buses does not coincide with the time when functional units are available for diagnostic purposes, (ii) the structure of the algorithm may also result in the clustering of the operations in the last few control steps in the schedule, (iii) a constant equitable distribution of operations of a single type reduces the spare capacity available for self-diagnostics. This can be seen from the distribution of operations in the schedule for the LMS algorithm which is shown in Fig. 4(a). From the figure we can see that the multiplication operations in the schedule are clustered towards the top of the schedule and are available for diagnostic purposes later in

Ex. Name	# Clk. Cyc.	$\|V\|$	# Oper.			# FU			# FU Diag.			# Op. Diag.			# Vtr.	Chip Area		% Overhead
			+	*	-	+	*	-	+	*	-	+	*	-		Orig.	Intros.	
Cheb19	18	52	34	18	0	5	5	0	5	5	0	29	18	0	4	60.15	61.05	1.5
FFT6	5	22	10	2	10	3	1	3	3	0	3	2	0	2	2	8.09	8.20	1.46
FFT5	4	22	10	2	10	5	2	3	4	0	3	2	0	2	2	14.3	14.42	0.82
Wang12	11	48	13	22	13	2	5	2	0	5	0	0	6	0	3	14.48	14.66	1.19
Wang8	7	48	13	22	13	3	9	3	3	9	3	2	11	2	4	24.71	25	1.19
Lee14	13	49	17	20	12	3	3	2	3	3	0	8	6	0	2	8.82	9.22	4.54
Lee10	9	49	17	20	12	4	4	4	4	4	4	9	6	12	5	11.58	11.75	1.50
Dit12	11	50	24	14	12	6	3	2	6	3	2	21	1	0	2	10.39	10.6	2.0
Dif10	9	49	17	20	12	4	4	4	4	4	4	9	6	12	4	12.76	13.17	3.24
Arai8	7	43	16	13	14	7	4	4	7	4	4	16	7	7	4	13.32	13.65	2.51
LMS	12	56	23	32	1	5	8	1	5	8	0	13	32	0	7	50.41	51.14	1.44

Table 2: Experimental Results with Introspective Binding: Cheb19 – Chebyshev Windowed Bandstop Filter; FFT5, FFT6 – Discrete Fourier Transform for N=8; Wang8, Wang12 – Wang's DCT-II algorithm, N=8; Lee10, Lee14 – Lee's DCT-II fast algorithm, N=8; Dit10 – Decimation in Time Fast Algorithm; Arai8 – Arai's Fast JPEG-DCT-II Algorithm; LMS algorithm for adaptive noise cancellation.

the schedule. On the other hand, the earliest step in which an addition operation can be performed is control step 3 and the addition operations are more evenly distributed across the remaining control steps. The darker squares in Fig. 4(b) denote the spare computation capacity used for diagnosis.

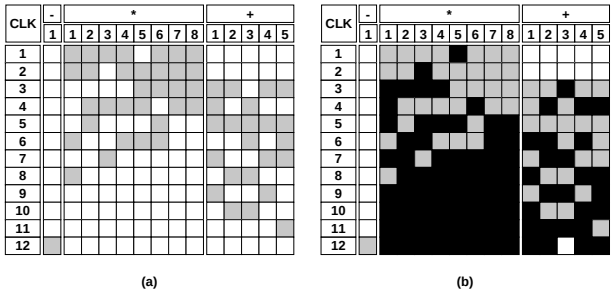

Figure 4: (a) Spare computation capacity of the LMS algorithm and (b) its utilization for diagnosis.

In the case of Lee14, the area overhead reaches the maximum value of 4.54%. In this case, the diagnostic latency of most of the operations in the CDFG was greater than two with an attendant increase in the number of registers in the functional units as well as the voter units which resulted in large area overheads.

6 Conclusions

In this paper we have presented a novel binding technique for the synthesis of low latency self-diagnosing microarchitectures with negligible hardware overhead. From the results, we conclude that all functional units in a design can be diagnosed with near zero overheads in terms of area and with absolutely no performance degradation. We also conclude that the number of nodes in the CDFG that can be diagnosed is a function of (i) the schedule of operations for the CDFG (ii) the structure of the algorithm being implemented, and (iii) the distribution of operations in the CDFG.

References

[1] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization*, Prentice-Hall, 1982.

[2] A.T. Dahbura, K.K. Sabnani, and W.J. Hery, "Spare Capacity as a Means of Fault Detection and Diagnosis in Multiprocessor Systems", *IEEE Trans. Computers*, pp. 881–891, June 1989.

[3] D.M. Blough and A. Nicolau, "Fault Tolerance in Super-Scalar and VLIW Processors", *IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems*, pp. 193–200, July 1992.

[4] M.C. McFarland, A.C. Parker, and R. Camposano, "The High-Level Synthesis of Digital Systems", *Proc. IEEE*, vol. 78, pp. 301–318, Feb. 1990.

[5] A. Chandrakasan *et. al.*, "HYPER-LP: A System for Power Minimization Using Architectural Transformations", *IEEE ICCAD-92*, pp. 300–303, 1992.

[6] I.G. Harris and A. Orailoglu, "Microarchitectural synthesis of VLSI designs with high test concurrency", *Proc. DAC*, 1994.

[7] R. Karri and A. Orailoglu, "High-level synthesis of fault-secure microarchitectures", *DAC*, pp. 429–433, 1993.

[8] S. Sokolov and R. Karri, "Allocation and Binding during Self Recovering Microarchitecture Synthesis", *ICCD*, 1994.

[9] L. Guerra, M. Potkonjak, and J. Rabaey, "High Level Synthesis Techniques for Efficient Built-In-Self-Repair", *Intl. Workshop on DFT in VLSI Syst.*, pp. 41–48. IEEE, 1993.

[10] B. Iyer, R. Karri, and I. Koren, "Phantom Redundancy: A High Level Synthesis Technique for Manufacturability", *ICCAD*, 1995.

[11] J. Rabaey, C. Chu, P. Hoang, and M. Potkonjak, "Fast Prototyping of Data Path Intensve Architectures", *IEEE Design & Test*, pp. 40–51, 1991.