Novel Design and Verification of a 16 x 16-b Self-Repairable Reconfigurable Inner Product Processor

Rong Lin
Department of Computer Science
SUNY–Geneseo
Geneseo, NY 14454
lin@cs.geneseo.edu

Martin Margala
Electrical and Computer Engineering
University of Rochester
Rochester, NY 14627
margala@ece.rochester.edu

ABSTRACT
A novel self-repairable and reconfigurable inner-product processor with low-power, fast CMOS circuits and DFT techniques is presented. It takes the advantage of recently proposed decomposition based arithmetic circuit design approach for simple implementation of the reconfigurations, component replacements, and high-quality tests.
The processor can be dynamically reconfigured for two types operations: 4 x 8 x 8-b inner product computation and 16 x 16-b multiplication. The self-repair is provided by choosing a fault-free one from 17 possible architectures during the test, which covers more than 52% transistors for the specified faults. Only one extra bit is needed for all reconfigurations, repairs, and tests. The proposed exhaustive DFT technique greatly reduces the test vector length, from $17*2^{32}$ to $1.5*2^{13}$, which is as short as that required by the pseudo-exhaustive DFT method recently reported in literature.

Keywords
Reconfigurable, Decomposition Algorithms, Self-Repair, VLSI, Arithmetic Circuits, Image Processing, Fault Tolerance

1. INTRODUCTION
Fast, low-power, low-cost, high-yield processors for multiplication and inner-product computation have become increasingly important to the rapidly growing computing industry, particularly for SoC designs [1, 2, 11-14] . The main hurdles preventing an efficient design of such processors include irregularity of the architectures, large VLSI area, high complexity of testing, and difficulty of fault recovering.
In this paper we present the design and tests of a highly regular, 16 x 16-b repairable and reconfigurable inner product processor/multiplier architecture and design using low-power, fast CMOS circuits and new DFT techniques.
The work utilizes a recently proposed recursive partial product matrix decomposition method [3-8] for simple implementation of the reconfigurations, component replacements, and high-quality tests. The processor can be dynamically reconfigured for two types operations: 4 x 8 x 8-b inner product computation and 16 x 16-b multiplication. The self-repair is done by static reconfiguration of up to 17 architectures during the test, which covers a large percentage of transistors for the specified faults. The repairing and testing take the advantage of the "clean" partitioning of the circuit, which leads to high controllability and observability (refer to Section 6). The proposed exhaustive DFT technique greatly reduces the test vector length, from $17*2^{32}$ to $1.5*2^{13}$, which is as short as that required by the pseudo-exhaustive DFT method recently reported in [10].
The proposed processor basically works as follows (refer to Fig. 6): First, as a 16x16 multiplier, the reconfiguration switches are set to state 0 (by M/I-sele bit), and all latches are set to open. Two 16-b numbers are received from X(0-15) and Y(0-15) (or X1, X2, Y1 and Y2). The 24 input bits X2, X3, Y2(0-3) and Y3(0-3), which will be used for inner product inputs, are now used for setting the repair controls. The operation consists of: (1) distributing the input bits to 17 locations including an extra one using a two-level full 4-branch tree structured network; then at each location generating a 4x4-b partial product matrix; (2) reducing each of the 4x4 partial product matrix into a single number to result in16 8-bit products, P1,.., P16 and an extra one, P0, in parallel; (3) distributing the extra product P0 to the 16 locations, and muxing each Pi ($i \neq 0$) and P0 to produce an actual 4x4 product, thus with one or zero of the 16 4x4 multipliers being repaired by the extra one; (4) adding the outputs of every four 4x4 multipliers into a 16-b number, the product of an 8 x 8-b multiplier, using a carry-look-ahead adder (called a mid final adder); (5) adding three 16-b numbers plus a 16-b zero, using another carry-look-ahead adder to generate a 32-b number, S0 to S31, which is the final product of the 16 x 16 multiplier.
Second, as an inner product processor, all reconfiguration switches are set to state 1, all 64 input bits from X0-X3 and Y0-Y3 are now used to receive and distribute data from two input arrays, and all 24 latches are set to close, keeping the repair-code values, which were set up during multiplication mode, unchanged. The operation consists of: (1) to (4) which are the same as for the multiplication above, except that data are routed along slightly different paths; (5) adding four 16-b numbers using the same adders I and II, and taking 18 bits, S8 to S25, as the output of the inner product.
Conventionally, to replace a fault component of a circuit, one would have to take its input bits, generate the output bits using an extra component, then send the output back to replace the output of the fault component. Our regularly decomposed architecture allows us to generate the inputs for the extra unit

GLSVLSI'02, April 18-19, 2002, New York, New York, USA.

directly from original inputs. This significantly reduces the number of connection lines and VLSI area needed for the 16-option component repairing. Also the regular decomposition structure enables us to have a maximized component sharing, including for the most of routing wires and the 24 inner product array input lines (sharing for the repair control setups). This minimizes the total VLSI area significantly. As shown in Table 1, The addition of inner product computation adds only 4% of the total transistors, while the addition of both the repairing and reconfiguration adds only 23% of the total transistors. The total delay increase, about 20%, is also not significant in practice.

It is sufficient to detect the fault 4x4 multiplier (if exists) with the reconfiguration switch being set to multiplication state only. The repair procedure generates a 24-bit repair-code that will be attached to each individual processor as a fixed constant. The constant is the part of the inputs, when the processor is used as a 16 x 16-b multiplier, i.e. bits X2, X3, Y2(0-3) and Y3(0-3), which are spare lines at multiplication mode (ref. to Fig 6).

Another advantage of the approach is that the exhaustive DFT method can be applied effectively to screen out a fault-free one over 17 possible different architectures. It greatly reduces the test vector length from $17*2^{32}$ to $1.5*2^{13}$, which is as short as that required by the pseudo-exhaustive DFT method reported in literature [10].

Our SPICE circuit simulations, using a 0.25-micron process with a 2.5-V supply, have demonstrated that the proposed processor has a worst-case total delay of 4.85 ns, which is equivalent to 6.8 times a (4, 2) counter delay with the same simulation, and has a relative smaller power dissipation, compared with the designs traditionally used for such circuits.

2. THE PARTIAL PRODUCT MATRIX DECOMPOSITION-BASED MULTIPLIER ARCHITECTURE

To be self-contained, we present briefly the main approach and the overall design in this and the following four sections. Figure 1 illustrates how an 8 x 8-b multiplier is constructed with four 4 x 4-b multipliers and two special adders using the decomposition based approach. Two 8-b input numbers are first partitioned and distributed to four 4 x 4-b multipliers (Figure 1a), where the 4 x 4 partial product matrices are generated and the four 8-b products are produced. The weighted bits of the four products are then added by two adders. Adder-I adds three 8-bit numbers to result in the final sum bits 4 to 11, while Adder-II adds the 4 MSB bits of the product of multiplier D and the two carry-in bits from Adder-I to result in the final sum bits 12 to 15. Since no addition is needed for the output bits 1 to 3, all 16 bits of the product have been correctly produced. The block diagram of the 8 x 8 multiplier schematic is shown in Figure 2.

3. THE 16x16-b MULTIPLIER ARCHITECTURE

The above circuit construction method can be applied recursively to result in larger multipliers [3-8]. In this section we present, without involving component repair, the overall 16 x 16 multiplier architecture, which requires two levels of recursive decomposition of the partial product matrices.

As shown in Figure 3, the 16 x 16-b multiplier is composed of the followings: (1) The full 4-branch tree-structured input net. It distributes two 16-bit inputs X and Y to the 16 4 x 4 multipliers in 2-levels (level-one is shown in Figure 2). (2) The 16 identical 4 x 4-b multipliers. Each is constructed by a 4 x 4-b partial product matrix generator and a few closely connected "tiny" pass-transistor parallel counters, featuring low-power, fast, and minimized VLSI area as described in Section 5. (3) The 4 identical 8 x 8 multipliers. Each consists of four 4 x 4 multipliers and an Adder-I plus an Adder-II. (4) The final adder. It consists of a carry-look-ahead Adder-I which adds three 16 bit numbers and an Adder-II which adds two carry bits from Adder-I and the 8 MSBs of the product of 8x8-b multiplier D.

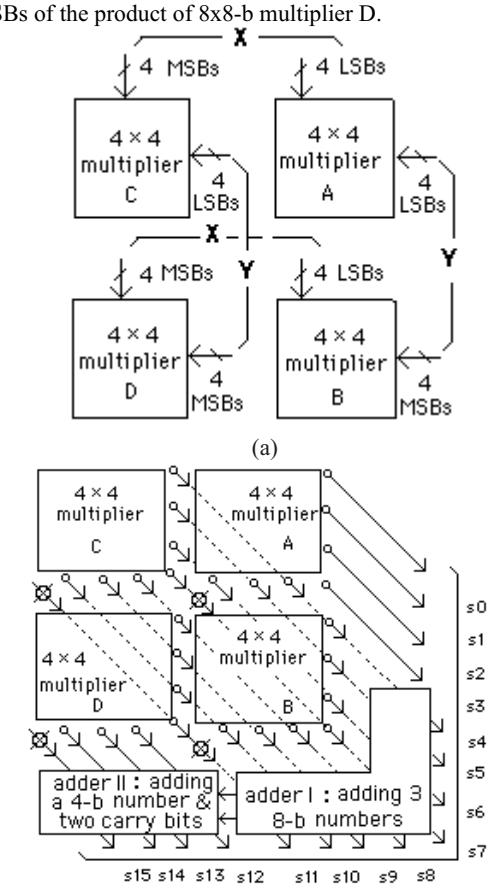

Figure 1. The 8 x 8-b decomposition based multiplier. (a) The input partitioning and distribution; (b) 8 x 8-b multiplier constructed by four 4 x 4-b multipliers and two special adders.

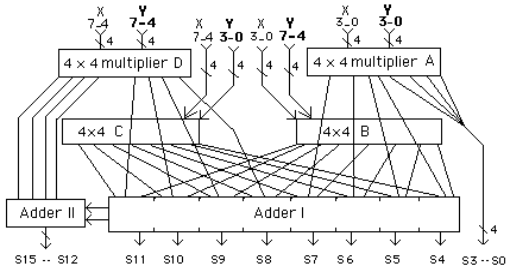

Figure 2. The 8x8-b decomposition-based multiplier.

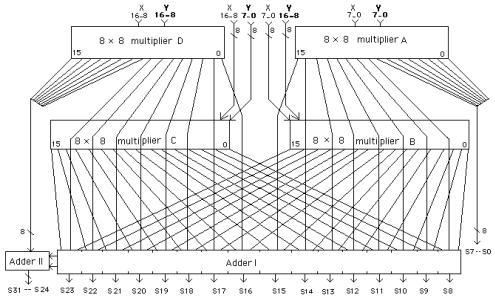

Figure 3. The 16 x 16-b multiplier.

4. THE SELF-REPAIRABLE MULTIPLIER ARCHITECTURE

Figure 4 illustrates the self-repairable multiplier architecture modified from Figure 3. Four repair-select bits (Xa, Xb, Ya, Yb) plus one repair-enable bit are used to generate repair controls for all 4x4 multipliers including the extra one as shown in the top of the Figure 4. The repair-input muxing-unit takes the original inputs (i.e. two 16 bit numbers) and produces the two desired input segments for the extra multiplier. The product P0 of the extra multiplier is then distributed to all 4x4 multipliers. Then the 4x4 multiplier to be repaired, which is specified by the given 4 bits, Xa, Xb, Ya, Yb plus E, abandons its own output and replaces it by the one from the extra multiplier. It should be noticed that the power supply of the disabled unit (one of the 17 4x4 multiplier) will be turned off through a power enable control to reduce power dissipation.

The self-repairability possessed by the multiplier allows us to recover the specified faults over 4686 transistors, 54% of all 8704 transistors. The approach can be extended to recover almost all the transistors in the circuit for more faults, however, the trade-off between the yield gain and the extra cost needs to be carefully considered in practice.

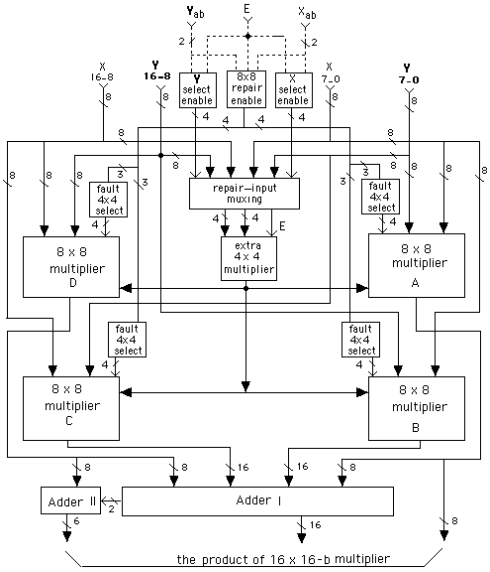

Figure 4. The self-repairable 16x16 multiplier.

5. THE RECONFIGURABLE INNER PRODUCT PROCESSOR (IPP) ARCHITECTURE

Next, we consider performing two types of computations using a single (modified) network of four 4 x 4 multipliers modified from Figure 1. First, it should be able to multiply two 8-bit numbers, XY, in a way similar to that described in Section 3, and then able to compute the inner product of two arrays of four 4-bit items.

The modification is simple and is conceptually shown in Figure 5. The 4 MSBs (most significant bits) of the product of multiplier A and the 4 LSBs of multiplier D in Figure 2 are moved to the new positions, i.e. the top and the left of multiplier B. The connections from multipliers to the adders are shown by four lines for each column, and the simple reconfiguration switches, marked as switch sets 1, 2 and 3, are added in three indicated areas. Each switch has two states 0 and 1, defined as follows. When in state 1, switches in set 1 and 2 are connected to ground, diagonal switches in set 3 are on, while horizontal and vertical switches in set 3 are off. The architecture is clearly for

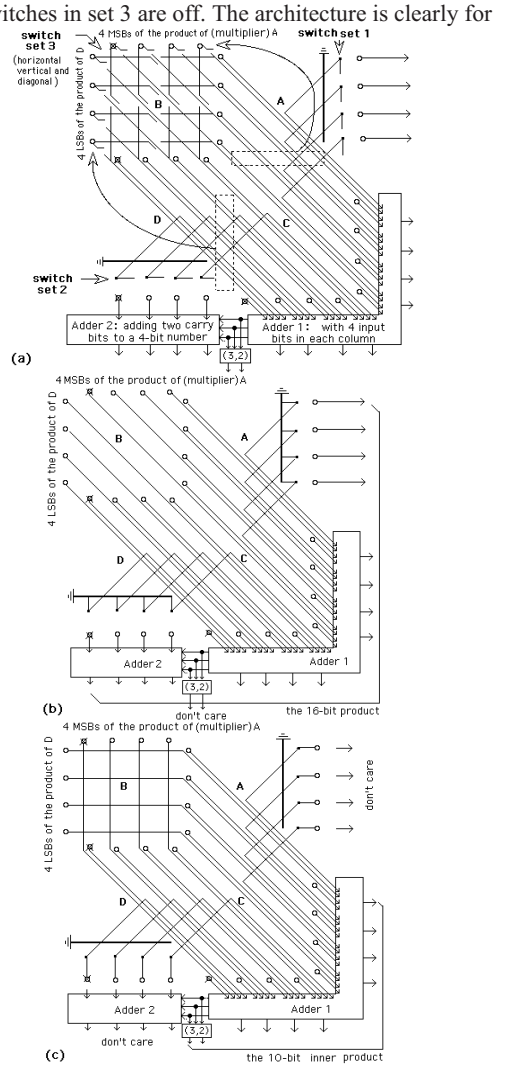

Figure 5. The Conceptual illustration of reconfigurable processor: (a) The modification of Figure 1 with the MSBs of A and LSBs of D re-positioned and three sets of reconfiguration switches added; (b) switch state 1: for the product of two 8-bit numbers; (c) state 0: for inner product of two arrays, each with four 4-bit numbers.

multiplication. When in state 0, switches in sets 1 and 2 are connected to the small multiplier outputs i.e. small circles, the diagonal switches in set 3 are off while horizontal and vertical switches are on. It is now for inner product computation.

6. THE SELF-REPAIRABLE RECONFIGURABLE PROCESSOR ARCHITECTURE

Figure 6 illustrates the self-repairable and reconfigurable multiplier-inner-product processor architecture modified from Figure 4. We first describe the part of the new architecture supporting self-repair, then the other part supporting the reconfiguration of two operations.

Five repair-control/select bits (E, Xa, Xb, Ya, Yb) and three enable units are now removed. The eight bits, which were generated by these units, used to select a pair of 4-bit input segments from X(0-15) and Y(0-15) sending to the extra 4x4 multiplier, are now provided directly from the spare input lines Y2(0-3) and Y3(0-3) to two arrays of latches during multiplication mode. The bits for selecting the fault 4x4 multiplier, i.e. for receiving the product generated by the extra 4x4 multiplier, are initially set by the spare input lines X2(0-7) and X3(0-7), and sampled by the other two arrays of latches when operation mode is turned to inner-product. All 16 bits are set zero, except that the one (if exists, determined during the tests, see Section 8) for repairing is set to 1. If no repair is needed, the 24-bit repair-code consisting of X2, X3, Y2(0-3), and Y3(0-3) is set to 0. The level-sensitive latches are open (closed) when M/I-sele is set to 1(0), i.e. for multiplication (inner product). The input lines for sending X (or X0, X1) and Y (or Y0, Y1) to the repair-input muxing now are shared by X2 Y2, X3 and Y3, through the use of reconfiguration switches (mul switches), controlled by the M/I-sele bit.

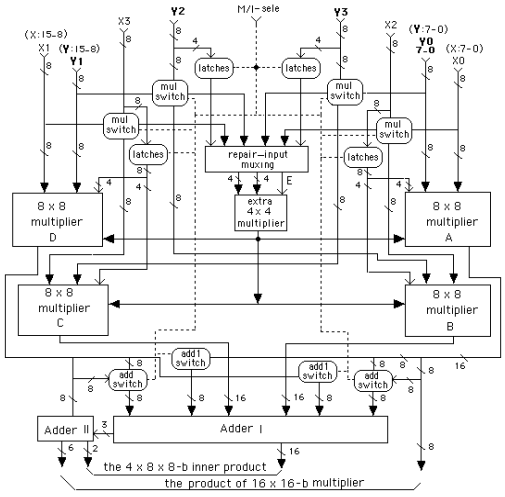

Figure 6. The self-repairable reconfigurable 16x16 multiplier/inner-product processor.

The mul switches are also used to control the sharing of routing lines, which either copy X and Y or send x2, x3, y2 y3 to the 8 x 8 multipliers B and C according to one of the desired operations. Furthermore, the routing lines sending the outputs of the four 8 x 8 multipliers to the adders I and II are now shared for two operations, as shown in Figures 5b and 5c, through the use of four reconfiguration switches (add and add1 switches). All the reconfiguration switches and latches are simple and directly controlled by a single extra input bit, i.e. M/I-sele. The performance of the processor shows negligible degrading compared with either a pure multiplier or a pure inner product processor. Also the power supply control for the extra 4x4 multiplier can be provided by Y2(0-3), which contains a repair-generate bi, i.e. disable if it is 0, enable if non-zero. For each of other 16 4x4 multipliers, it is provided by its own repair control bit (Section 9), i.e. disable if the bit is1, enable if 0.

The self-repairability and reconfiguration possessed by the processor allow us to recover the specified faults over 4686 transistors, 52% of all 9052 transistors for two operations. The approach can be extended to recover almost all the transistors in the circuit for more faults, however, the trade-off between the yield gain and the extra cost (see Section 9) needs to be further studied.

7. THE COMPONENT CIRCUITS

Though any existing 4x4 multiplier and parallel counters may be used to implement the cost efficient processor, in this design we adopt only three small complementary pass-transistor parallel counters, (2, 2), (3, 2) and (4, 2), recently proposed in [4, 8]. As shown in Figure 7. The parallel counters are "tiny" and robust with transistor counts of 11, 20 and 38 respectively. Since the components to be repaired, such as the 4x4 multiplier and the 4-bit group adder used in Adder-I and II, are all relatively small enough (significantly smaller than a traditional Wallace-adder-tree [12] for the implementation), almost all counter connections within each component can be made without a buffer. This significantly reduces the VLSI area and power dissipation, while increases circuit speed. Our preliminary layout of the components has verified the superiority of the design. The cost for the addition of the self-repairability is shown in Table 1.

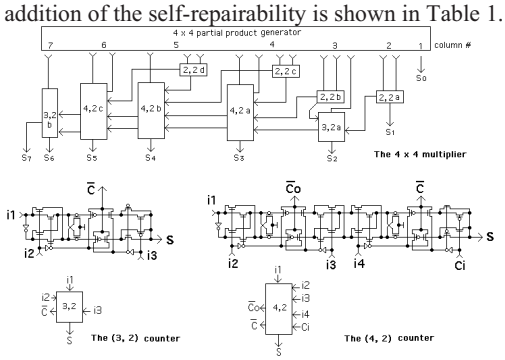

Figure 7. The 4 x 4 multiplier with tiny parallel pass-transistor counters (2, 2), (3, 2) and (4, 2).

Table 1. Comparisons of the proposed non-repairable and self-repairable multipliers.

	transistor count	delay	test vector length: exhaustive (this work)	test vector length: pseudo exhaustive (reported in [8])	exhaustive / pseudo exhaustive
non-repair multiplier A	7352	4.0 ns	$1.5 * 2^{12}$	$1.7 * 2^{11}$	1.76
repairable multiplier B	8704	4.65ns	$1.3 * 2^{13}$	$1.7 * 2^{12}$	1.5
repairable inn/mul C	9052	4.85ns	$1.5 * 2^{13}$	NA	NA
B / A	1.18	1.16	1.7	2	
C / A	1.23	1.21	2	NA	
C / B	1.04	1.04	1.15	NA	

8. THE EXHAUSTIVE DFT TECHNIQUE

In addition to the superiority in construction of self-repairing and reconfiguring circuits, the decomposition-based design approach has another important advantage over the traditional designs (with a single large-partial product matrix): significantly higher controllability and observabitlity for tests. An exhaustive test procedure can be practically developed to reduce the test cost and improve the quality of products.

We show the DFT technique which is used to screen out a fault-free one over 17 possible different architectures when the processor is set to multiplication mode. The length of an exhaustive test vector is shorter than $1.5*2^{13}$.

The test procedure is obtained based on the following observations: (1) The processor can be partitioned into the following 22 components (referred to a clean partition): 16 identical 4x4 multipliers, one extra 4x4 multiplier with the repair-control units, four mid-final adders each for an 8x8 multiplier, and one final adder. (2) If there is a fault in any of the five adders or there exist more than one fault 4x4 multipliers, the circuit should be rejected for un-repairable. (3) To exhaustively test a specified 4x4 multiplier the corresponding inputs of X(0 .. 15) and Y(0 .. 15) can be generated as follows: generate all combinations of X(4i .. 4i+3), Y(4j .. 4j+3) pairs for the given $0 \le i, j \le 3$, then for each pair of them add 0s into all remaining bit positions to result in a pair of the test inputs; (note that this will guarantee that all 4x4 multipliers, except the specified, always generate a product of 0 (if one does not, the fault will be detected by our procedure below). (4) A 4x4 multiplier test output received from the final adder is always the sum of the actual output of the multiplier and four 4-bit 0s, two added to the sum in a mid-final adder and another two added to the sum in the final adder (plus a number 0 provided by add1 switches). (5) If for all input combinations the test results of a 4x4 multiplier are correct, we say the 4x4 multiplier is fault free (the proof is omitted). (6) If all 16 4x4 multipliers (including the case of one being repaired) are all fault-free then any inputs (three numbers) to any mid-final adder are all fault-free, thus mid-final adder can be tested exhaustively column by column. Note that each column may have up to three carry-bits, and only the sums of the carries can be tested correctly, however, that is good enough for the functionality test. All needed is that for each column we provide pre-generated all possible inputs to it and compare each test result with the pre-generated correct result. (7) If all mid-final adders are fault-free then the final adder can be tested in the same way.

The repairing procedure (finding a 4x4 multiplier to be replaced by the extra one)

Assume that the 16 4x4 multipliers are denoted by M1, M2, .., M16, and the extra one by M0. We set a temporary repair-code for Mn as: set the repair-control bit, X2(n)=1, if n <8, or X3(n-8)=1 if n>8; also set two repair-generate bits, Y2(n DIV 4)=1; Y3(n MOD)=1, and finally set all other 21 bits in X2X3Y2(0-3)Y3(0-3) to 0.

Step 1. Set fault-list empty and n=1.

Step 2. Exhaustively test 4x4 multiplier Mn as described in (3) above. If a fault is found, add number Mn to the fault-list and then replace Mn by M0 (note that no re-test for the new Mn at this time).

Step 3. Let n++. If n < 17 go to Step 2, if n =17 then exhaustively re-test all multipliers in the fault-list, if a fault is found, reject the circuit as un-repairable immediately, otherwise declare the current architecture being fault-free.

Step 4. If it is fault-free, set the 24-bit repair-code as follows: if fault-list is empty set all 24 bits 0, otherwise assume that Mj is the one finally replaced by M0, set the Mj's temporary repair-code as the final re-pair-code.

The proof of the correctness of the procedure is straightforward: once the only fault multiplier has been replaced by the good one then all 4x4s will be tested as fault-free, i.e. all Mi in the fault-list, except the last one, are not a candidate for repairing. Since the two operations use the same set of hardware, with the multiplication involving a larger final adder, the test for multiplication will be sufficient if we also include a few tests for the reconfiguration state changes. Now we have

The complete-test procedure

Step 1. Call the above repairing procedure. If reject, exit (claim the circuit un-repairable).

Step 2. Column by column test all mid-final adders as described in (6) above. If there is a fault, exit.

Step 3. Column by column test the final adder column by column as described in (7) above , if there is a fault exit, otherwise accept the circuit as fault free and return the 24-bit final repair-code.

The total length of the complete-test vector is shorter than $1.5*2^{13}$. For all 4x4 tests the vector length is $2*16*256=2^{13}$, and for all mid-final and final adders tests, the length is $3*25*(13+20+9) < 2^{12}$ (note that each column has a maximum of 26 possible inputs).

The test vector length is as short as that required by the pseudo-exhaustive DFT method recently reported in [10], which requires a vector length 256+x (including a few dozens of random tests) for an 8x8 multiplier, and a total test length about $4*(256+x)*4 + 2^{11} = 1.7* 2^{12}$(for x=44) for the 16x16 repairable multiplier.

9. CONCLUDING REMARKS

A highly regular self-repairable and reconfigurable 16 x 16-b parallel multiplier/inner product processor along with a low-power fast CMOS circuit implementation and an exhaustive DFT method has been presented. The circuit can be efficiently reconfigured into 17 different architectures, recovering the specified faults over 52% of transistors. Both the repairing and testing take the advantage of the "clean" partitioning of the circuit, which results in high controllability and observability, inherent in the decomposition approach.

The processor can be directly extended for operations in two's complement form, refer to [3,15], with a negligible amount of VLSI area increase. We have also extended the repairing coverage to allow one fault mid-final adder and one fault 4-bit group adder in the final adder to be recovered, which provides 17*5*5=425 different architectures for repairing. This would recover the specified faults for almost all transistors in the circuit. However, the additional VLSI area (transistors and, particularly, lines), the delay (5 ns) and test vector length (1.5*217) needed are non-proportionally larger than the one proposed in this paper, mainly due to that all component inputs must be collected, instead of generated. The trade-off between the yield gain and the extra costs paid are currently under study.

Acknowledgment

The work was supported, in part, by National Science Foundation grant CCR-0073469 and by New York State Office of Science, Academic & Research (MDC) grant NYSTAR C000063.

REFERENCES

[1] Y. Hagihara, S. Inui, A. Yoshikawa, S. Nakazato, S. Iriki, R. Ikeda, Y. Shibue, T. Inaba, M. Yamashina, A 2.7ns 0.25um CMOS 54 x 54b multiplier, in ISSCC Dig. Tech. papers, vol. 41, Feb. 1998, pp. 296-297.

[2] G. Goto, A. Inoue, R. Ohe, S. Kashwakura, S. Mitarai, T. Tsuru, and T. Izawa, A 4.1-ns compact 5454-b multiplier utilizing sign-select Booth encoders, IEEE JSSC,Vol. 32; No 11, pp. 1676-1682, Nov,1997.

[3] R. Lin, Reconfigurable Parallel Inner Product Processor Architectures, IEEE Transactions on Very Large Scale Integration Systems(TVLSI), Vol. 9, No 2. pp. 261-272, April 2001.

[4] R. Lin, A Regularly Structured Parallel Multiplier With Non-Binary-Logic Counter Circuits, in intl. J. of VLSI Design, ol. 12, No 3, pp. 377-390, March, 2001.

[5] R. Lin, A Run-Time Reconfigurable Array of Multipliers Architecture, in Proc. of 8th Reconfigurable Architectures Workshop (RAW 2001), San Francisco, April , 2001.

[6] R. Lin, Trading Bitwidth For Array Size: A Unified Reconfigurable Arithmetic Processor Design, in Proc. of IEEE 2001 Intl. Symp. on Quality of Electronic Design, San Jose, California, pp. 325-330, March 2001.

[7] R. Lin, Parallel VLSI Shift Switch Logic Devices (US Patent 96125379).1999; Reconfigurable inner product processor architecture (US Patent pending, No. 09/512,380), 2000 and A Family of High Performance Multipliers and Matrix Multipliers (US Patent pending, No. R-1265-125), December, 2000.

[8] M. Margala, "Low-Voltage Adders for Power-Efficient Arithmetic Circuits", Microelectronics Journal, vol.30, no.12, pp.1241-1247, December 1999.

[9] M. Margala, X. Chen, J. Xu, and H.Wang, Design verification and DFT for an embedded reconfigurable low-power multiplier in system-on-chip applications, in proceeding of 14th Annual IEEE International ASIC/SOC Conf., Washington, D.C. September, 2001.

[10] N. Kazakova, R. Sung, N. G. Durdle, M. Margala, and J. Lamoureux, "Fast and Low-Power Inner Product Processor", in Proc. of the IEEE Intl. Symp. on Circuits and Systems, Sydney, Australia, Vol. 4, pp. 646-649, May 2001.

[11] C. S. Wallace, A suggestion for a fast multiplier, IEEE Trans. Electronic Computers, Vol. Ec-13, 1964.

[12] K. Yano, T.Yamanaka, T. Nishida, M. Saito,K. Shimohigashi, and A. Shimizu, A 3.8-ns CMOS 16 x 16 multiplier using complementary pass-transistor logic, IEEE J. of SSC,Vol. 25; No 2, April 1990.

[13] L. Breveglieri and L. Dadda, A VLSI inner product macrocell, IEEE Transactions on VLSI Systems, vol. 6, No. 2. June, 1998.

[14] C. R. Baugh and B. A. Wooley, A Two's complement parallel array multiplication algorithm, IEEE Tran. on Computers, Vol. C-22, pp. 1045-1047, 1973.