Easily Testable and Fault-Tolerant FFT Butterfly Networks

Jin-Fu Li, Shyue-Kung Lu, Shih-Arn Hwang, *Member, IEEE*, and Cheng-Wen Wu, *Senior Member, IEEE*

***Abstract*—With the advent of deep submicron very large scale integration technology, the integration of a large fast-Fourier-transform (FFT) network into a single chip is becoming possible. However, a practical FFT chip is normally very big, so effective testing and fault-tolerance techniques usually are required. In this paper, we first propose a C-testable FFT network design. Only 20 test patterns are required to cover all combinational single-cell faults and interconnect stuck-at and break faults for the FFT network, regardless of its size. A spare-row based fault-tolerant FFT network design is subsequently proposed. Compared with previous works, our approach shows higher reliability and lower hardware overhead, and only three bit-level cell types are needed for repairing a faulty row in the multiply–subtract–add module. Also, special cell design is not required to implement the reconfiguration scheme. The hardware overhead for the testable design is low—about 4% for 16-bit numbers, regardless of the FFT network size.**

***Index Terms*—Butterfly network, C-testable, concurrent error detection, design-for-testability, fault tolerance, FFT, logic testing.**

I. Introduction

THE DISCRETE Fourier transform (DFT) is widely used in analysis and design of linear time-invariant systems in the digital domain, and is more and more important as digital signal processing (DSP) is finding more and more applications. To increase its speed, many efficient algorithms have been developed in the past. The most notable one is the fast Fourier transform (FFT) algorithm. The FFT algorithm is currently used in various DSP applications, such as digital filtering, correlation, image processing, speech and audio coding, and spectrum analysis [1]–[4]. The algorithm is also an excellent candidate for parallelization. A typical N-point FFT network is constructed using two-input butterflies. The network has $\log_2 N$ stages, and each stage consists of $N/2$ butterflies as shown in Fig. 1, where $N = 8$. Although in practice N can be very large, with the advent of deep-submicron very large scale integration (VLSI) technology and system-on-chip design methodology, we have seen the possibility of integrating a large FFT network on a single chip [2], [5]–[9]. Limited accessibility of the circuit components in such a chip, however, poses considerable challenges in testing and diagnosis. Particularly, the cost of diagnosis is rapidly increasing along with the complexity of the chip. It is indispensable to control these costs and provide a cost-effective solution. Therefore, it is important to develop a fault detection and fault location approach. Continuous advances in VLSI processing technologies also have increased the density and reduced the feature size of the components, resulting in a higher degree of difficulty for guaranteeing the reliability and quality of today's VLSI chips. Therefore, fault-tolerance techniques are required.

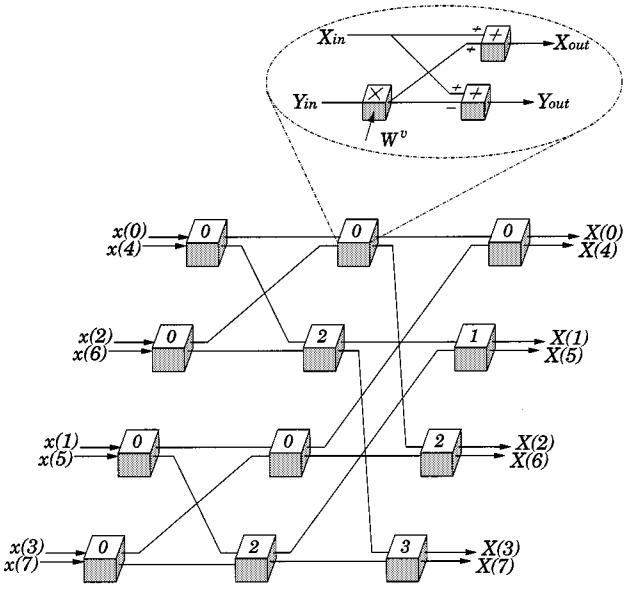


Fig. 1. An 8-point FFT butterfly network.

To improve the chip testability and reliability of FFT networks, several testable structures and fault-tolerant designs have been proposed [10]–[23]. Jou and Abraham [10] introduced a fault-secure FFT network design, whose hardware overhead ratio is $O(2/\log_2 N)$. A time-redundancy method was also used to detect and locate the faulty modules. In [11], a *recomputing by alternate path* approach was proposed to detect errors during normal operation. Once an error is found, the faulty butterfly module is located within $\log_2 N + 5$ extra cycles. A wafer-scale 170 000-gate FFT processor with built-in self-test (BIST) circuits was also reported in [12]. Then, in [13], a novel partition scheme and self-testing of arithmetic cells to obtain higher fault coverage was reported. The one-pass procedure derived from the algorithmic flow proposed in [14] allows detection and location of specified functional faults. This procedure needs $O(N)$ operations for an N-point FFT network. In [15], a fault-detection and -location approach for

Manuscript received August 1999; revised April 2000. This work was supported in part by the National Science Council, R.O.C., under Contract NSC87-2218-E262-005 and Contract NSC89-2215-E007-001. This paper was recommended by Associate Editor S. Sriram.

J.-F. Li, S.-A. Hwang, and C.-W. Wu are with the Department of Electrical Engineering, National Tsing Hua University, Hsinchu, Taiwan 30013, R.O.C.

S.-K. Lu is with the Department of Electronics Engineering, Fu Jen University, Taipei, Taiwan.

Publisher Item Identifier S 1057-7130(00)07758-2.

homogeneous VLSI/WSI FFT architectures was reported. Tao and Hartmann [17] then developed a fault detection approach with 5% hardware penalty and less incurred time delays than the scheme in [10], and fairly increased the fault coverage. Feng *et al.* [16] put emphasis on the design of a C-testable FFT network based on the single butterfly-module fault model and a linear testability scheme without modification of the circuit. In their design, one can locate a faulty module with time complexity $O(\log_2 N)$. In [18], the butterfly networks were shown to be testable with only 32 test patterns by using a novel design-for-testability technique. The proposed test scheme guarantees 100% coverage of combinational single cell faults. An algorithm-based fault-tolerance approach for FFT networks with lower hardware overhead and higher fault coverage than that in [10] was later proposed [20]. Also, Oh and Youn [21] presented two concurrent error location and correction methods for FFT networks. With an extra try in addition to $\log_2 m$ comparisons of m corrected outputs, a faulty component can be located. They later presented an algorithm-based concurrent error detection scheme using the checksum approach [22]. This design allows high error coverage with low false-alarm rate by applying linear weight factors to the checksums. In one of our previous works [23], we proposed a bit-level fault-tolerant design for FFT networks. The reliability was increased significantly, and the hardware overhead was about 23% for an FFT network with 16-bit input words.

In this paper, an improved design-for-testability scheme based on C-testability conditions is presented. We show that only 18 test patterns are sufficient to achieve 100% coverage of single cell faults in the *iterative logic array* (ILA) test mode. All the undetected stuck-at and break faults of interconnects can be covered with two additional patterns, i.e., only 20 test patterns are required for the FFT network, regardless of its size. The proposed scheme also allows us to use row redundancy instead of column redundancy for reconfiguration, resulting in a lower hardware overhead. Simulation results show that the reliability of the proposed row-replacement scheme is higher than that of the column-replacement scheme with lower hardware overhead. For example, the overhead of the testable FFT network is only 4% for 16-bit words regardless of the network size. Compared with previous works, our approach shows higher reliability and lower hardware overhead, and only three bit-level cell types are needed for repairing a faulty row in the multiply–subtract–add (MSA) module. Also, special cell design is not required to implement the reconfiguration scheme.

II. C-Testable FFT Network

A. FFT

The discrete Fourier transform of a data sequence $\{x[r], r = 0, 1, \ldots, N-1\}$ is defined as

$$X(k) = \sum_{r=0}^{N-1} x[r] w_N^{kr}, \qquad k = 0, 1, \ldots, N-1 \tag{1}$$

where $w_N = e^{(-j(2\pi/N))}$. The FFT network takes complex-valued quantities in real applications. The butterfly module performs the radix-two butterfly computation, in which the complex numbers X and Y are computed according to the following equations:

$$X_{\text{out}} = X_{\text{in}} + Y_{\text{in}} W^{\nu}$$
$$Y_{\text{out}} = X_{\text{in}} - Y_{\text{in}} W^{\nu} \tag{2}$$

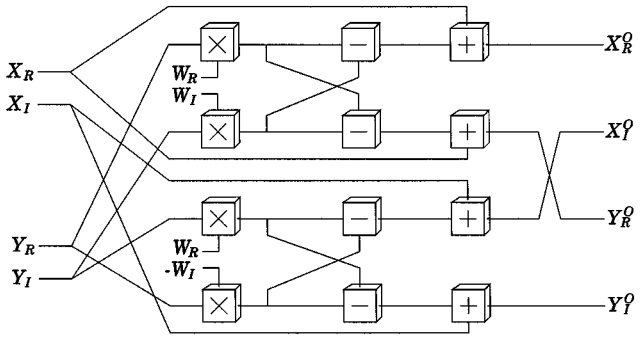


Fig. 2. A butterfly constructed with four MSA modules.

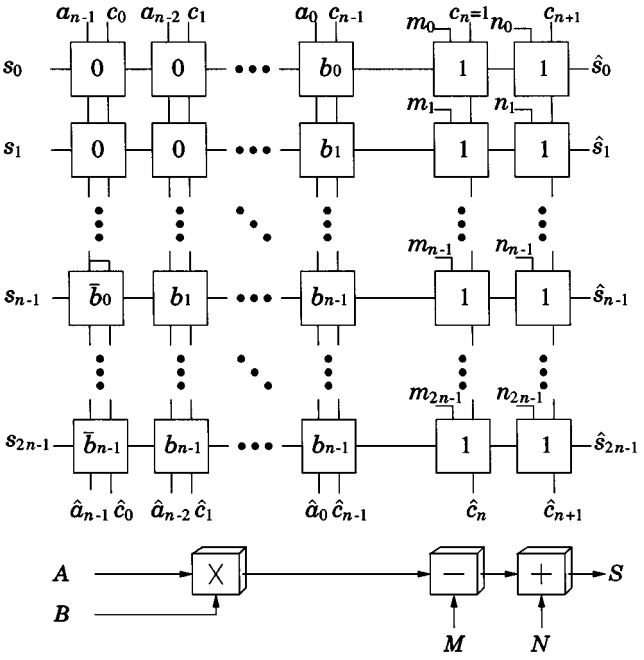


Fig. 3. Bit-level array design for an MSA module.

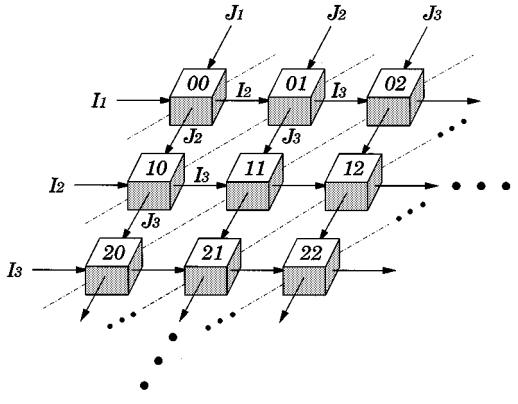


Fig. 4. 2-D ILA.

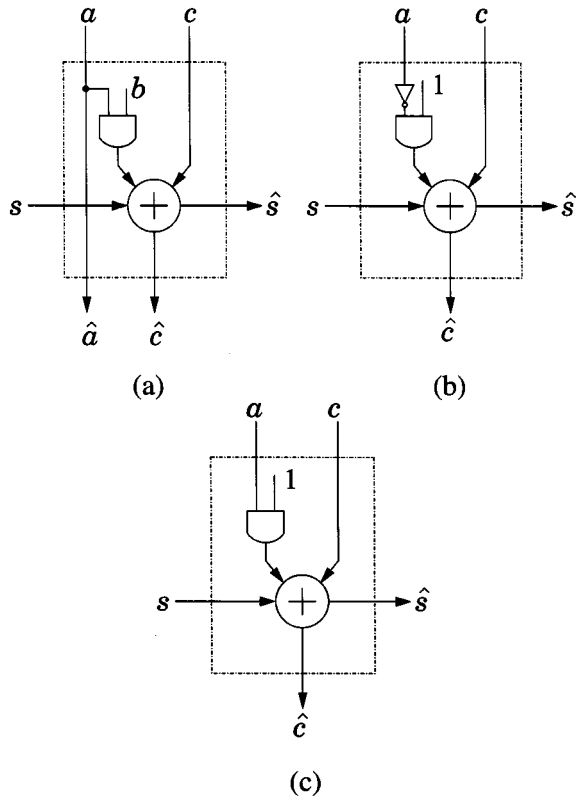


Fig. 5. (a) Multiplier cell. (b) Subtractor cell. (c) Adder cell.

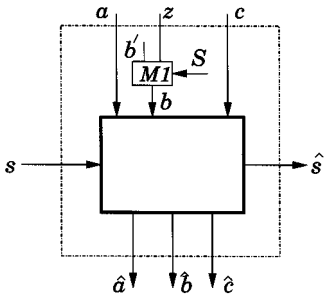


Fig. 6. Modified multiplier cell.

where W^{ν} represents the twiddle factor, i.e., $W^{\nu} = e^{-j2\pi\nu/N}$. To implement these equations, it is common to use a functionally equivalent butterfly which employs only real number operations.

Let us express X_{in}, Y_{in}, and W^{ν} in complex form as follows:

$$X_{\text{in}} = X_R + jX_I$$
$$Y_{\text{in}} = Y_R + jY_I$$
$$W^{\nu} = W_R + jW_I \tag{3}$$

where j is the square root of -1. Combining these equations, X_{out} and Y_{out} can be recast as follows:

$$X_{\text{out}} = (X_R + W_R Y_R - W_I Y_I) + j(X_I + W_I Y_R + W_R Y_I),$$
$$= X_R^O + jX_I^O$$
$$Y_{\text{out}} = (X_R - W_R Y_R + W_I Y_I) + j(X_I - W_I Y_R - W_R Y_I)$$
$$= Y_R^O + jY_I^O. \tag{4}$$

The computation of (4) can be done by four identical MSA modules [12], [13], as shown in Fig. 2. Each MSA module can be implemented in the form of an ILA, as shown in Fig. 3 [23], where M and N represent the subtrahend and addend of the subtractor and adder, respectively. The array size is $2n \times (n+2)$, where n is the word length of the operands.

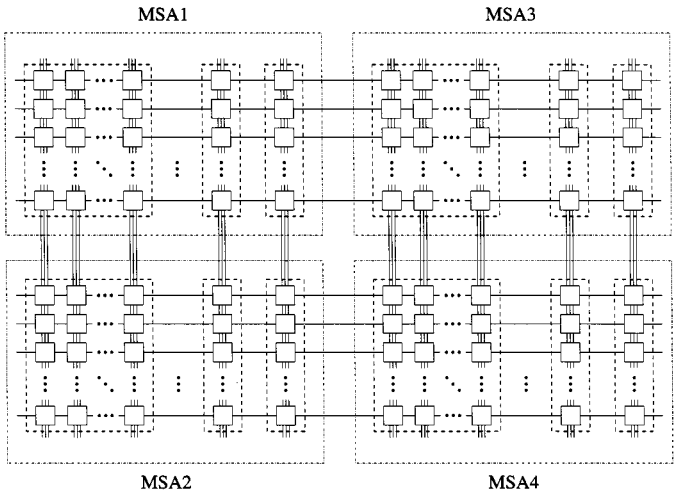


Fig. 7. Configuration of a butterfly in test mode.

It has been known that the general logic testing problem is NP-complete, but for certain ILA's the testing problem is solvable in polynomial time [24], [25]. We will show next that the FFT network can be treated as an ILA which can be tested easily after a simple design-for-testability technique is applied.

B. C-Testable ILAs

The definitions given in [25] are followed in this paper. A *cell* is a combinational machine (Σ, Δ, f), where $f : \Sigma \rightarrow \Delta$ is the cell function and $\Sigma = \{0, 1\}^I$ and $\Delta = \{0, 1\}^O$ for $I, O \in \boldsymbol{N}$. An ILA is an array of cells with identical function. We assume that the cell function is invariant over time and there is at most one faulty cell in the array. That is, sequential faults are not considered, and the *single-cell fault model* is adopted. A cell function is *injective* if $\forall(i_1, j_1) \neq (i_2, j_2)$, $f(i_1, j_1) \neq f(i_2, j_2)$. If a function is injective and $\Sigma = \Delta$, then the function is *bijective*. An input sequence consisting of all possible input combinations for a cell is a *minimal complete input sequence*. A *minimal complete output sequence* is defined analogously. A *C-testable* array is an array testable with a constant number of test patterns independent of the size of the array. A k-dimensional ILA with a bijective cell function has been shown to be C-testable, where k is an arbitrary positive integer [25]. Fig. 4 depicts a 2-D example. If a minimal complete input sequence (I_1, J_1) is fed to $cell_{00}$, then all cells lie in the same $+45°$ diagonal receive the same minimal complete input sequence. This input sequence forms the $+45°$ tessellation [25], [26]. When this sequence is applied, any fault can automatically be propagated to some observable primary outputs. If the cell function is not bijective, then a design-for-testability technique can be used to modify the cell function to make it bijective [27].

C. Design for C-Testability

The MSA module shown above can be implemented with a 2-D bit-level ILA, in which three types of basic cells are used: 1) the multiplier ($\boldsymbol{M}$) cell; 2) the subtractor ($\boldsymbol{S}$) cell; and 3) the

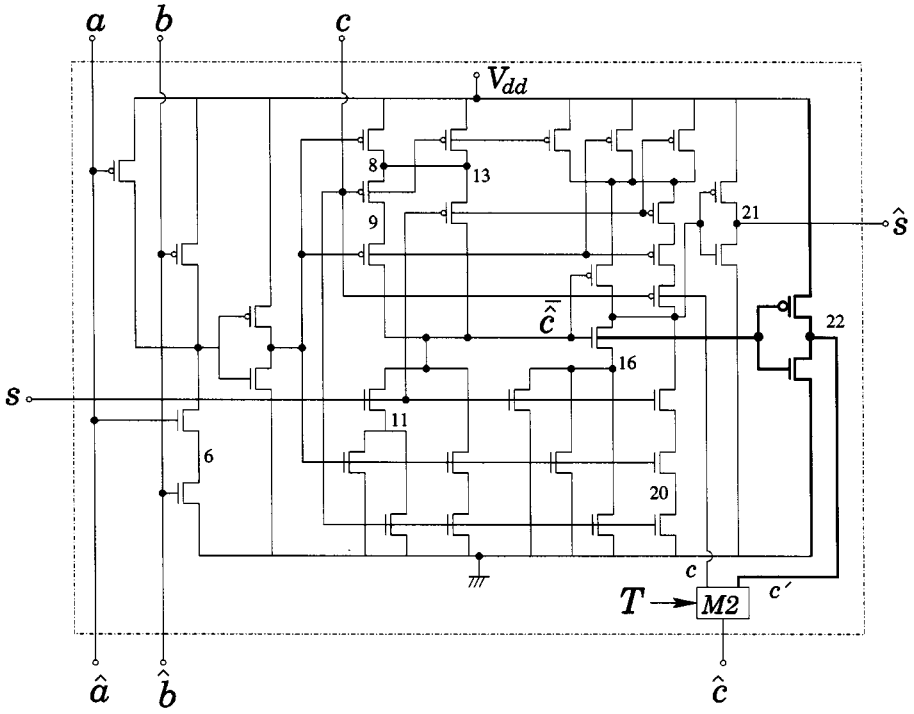


Fig. 8. Modified CMOS core circuit for the multiplier cell.

TABLE I
TRUTH TABLE OF THE MULTIPLIER CELL

a	z	s	c	$\hat{a}$	$\hat{b}$	$\hat{s}$	$\hat{c}$
0	0	0	0	0	0	0	0
0	0	0	1	**0**	**0**	**1**	**0**
0	0	1	0	**0**	**0**	**1**	**0**
0	0	1	1	0	0	0	1
0	1	0	0	0	1	0	0
0	1	0	1	**0**	**1**	**1**	**0**
0	1	1	0	**0**	**1**	**1**	**0**
0	1	1	1	0	1	0	1
1	0	0	0	1	0	0	0
1	0	0	1	**1**	**0**	**1**	**0**
1	0	1	0	**1**	**0**	**1**	**0**
1	0	1	1	1	0	0	1
1	1	0	0	1	1	1	0
1	1	0	1	**1**	**1**	**0**	**1**
1	1	1	0	**1**	**1**	**0**	**1**
1	1	1	1	1	1	1	1

TABLE II
MODIFIED TRUTH TABLE OF THE MULTIPLIER CELL

a	z	s	c	$\hat{a}$	$\hat{b}$	$\hat{s}$	$\hat{c}$
0	0	0	0	0	0	0	**0**
0	0	0	1	0	0	1	**1**
0	0	1	0	0	0	1	**0**
0	0	1	1	0	0	0	**1**
0	1	0	0	0	1	0	**0**
0	1	0	1	0	1	1	**1**
0	1	1	0	0	1	1	**0**
0	1	1	1	0	1	0	**1**
1	0	0	0	1	0	0	**0**
1	0	0	1	1	0	1	**1**
1	0	1	0	1	0	1	**0**
1	0	1	1	1	0	0	**1**
1	1	0	0	1	1	1	**0**
1	1	0	1	1	1	0	**1**
1	1	1	0	1	1	0	**0**
1	1	1	1	1	1	1	**1**

TABLE III
CONFIGURATIONS OF S AND T

(S,T)	Operation mode
(0,0)	Normal operation
(1,1)	Phase-1 test
(1,0)	Phase-2 test

adder ($\boldsymbol{A}$) cell, as shown in Fig. 5. They differ slightly in complexity. Note that none of the three cell functions are bijective, so we have to modify them to make them bijective. Without loss of generality, we show the modification procedure only for the multiplier cell. The adder and subtractor cells can be modified in a similar way.

As Fig. 3 shows, the twiddle factor B is normally preloaded, i.e., it cannot be controlled in test mode. Since in the ILA all the cell inputs have to be controllable, we modify the $\boldsymbol{M}$ cell as shown in Fig. 6. The coefficient bit b' is preloaded, and z is the output $\hat{b}$ from the previous cell. The selection is made by S: $b = b'$ when $S = 0$, and $b = z$ when $S = 1$. All cell inputs can be controlled from the primary inputs when $S = 1$, and the cell input set is equal to the cell output set, i.e., $\Sigma = \Delta$. The truth table of the $\boldsymbol{M}$ cell is shown as Table I, where there are four pairs of identical output rows which are highlighted. The cell function apparently is not bijective, so it is modified to become bijective during test mode [27], as shown in Table II. From the table, the output carry bit $\hat{c}$ is assigned the same values as the input carry bit c, such that all the output entries become distinct, and the modified cell function is bijective. The inner box of the schematic shown in Fig. 6 represents the circuit which implements the core logic for both Tables I and II. A CMOS circuit

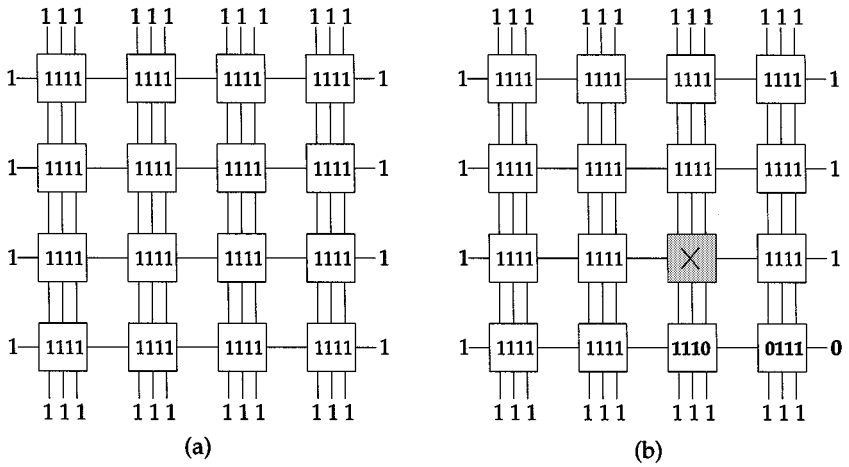


Fig. 9. (a) Applying the all-1 pattern to the array multiplier. (b) Detection of a faulty cell.

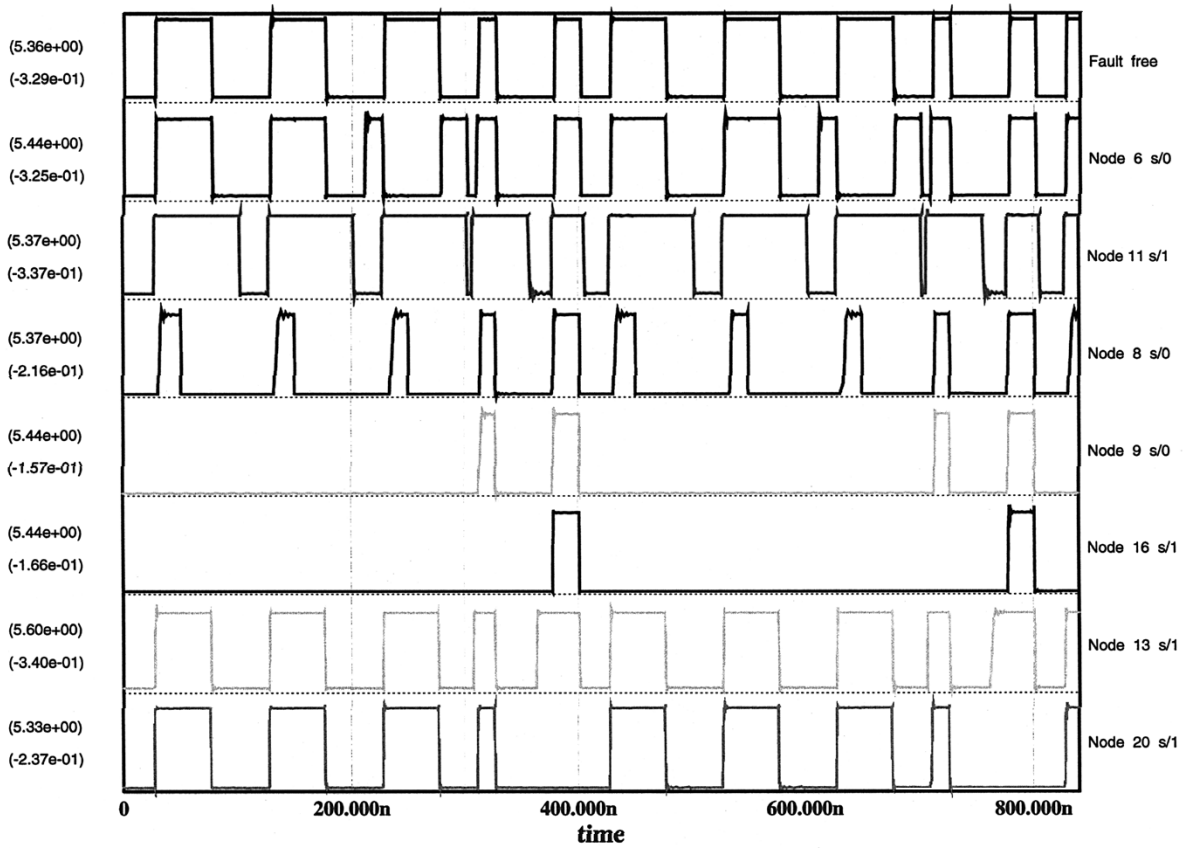


Fig. 10. Sample of the SPICE simulation for the Phase-1 test. Fault simulation of s/0 and s/1 at all modes.

for the core is shown in Fig. 8, where a multiplexer is added to the original circuit. The multiplexer $M2$ is controlled by the mode selection signal T that selects either c' (when $T = 0$) or c (when $T = 1$). From the above discussion, the modified multiplier is controlled by two signals: S and T, and their configurations are shown in Table III. The MSA modules are connected as a butterfly network during the *normal operation mode*, and as a bit-level C-testable ILA during the *test mode*, as shown in Fig. 7. In the Phase-1 test, the cell function is bijective, and only 16 test patterns are required to test the array pseudoexhaustively since there are four inputs to each cell.

In Fig. 8, the node $\overline{\hat{c}}$ is connected to $\hat{s}$ through two inverters. Since the inverter automatically propagates fault effect, observing $\overline{\hat{c}}$ from $\hat{s}$ is guaranteed. However, the highlighted line segments are not tested in this phase. In order to cover possible defects there, we apply the all-0 and all-1 patterns to the cell in the Phase-2 test. These two patterns can detect all the stuck-at faults of the inverters and the highlighted interconnects during normal operation mode. Also, parallel testing of all the cells and automatic fault propagation using these two tests is guaranteed. For example, a 4×4 array multiplier receiving the all-1 pattern is shown in Fig. 9(a). Each and every cell receives

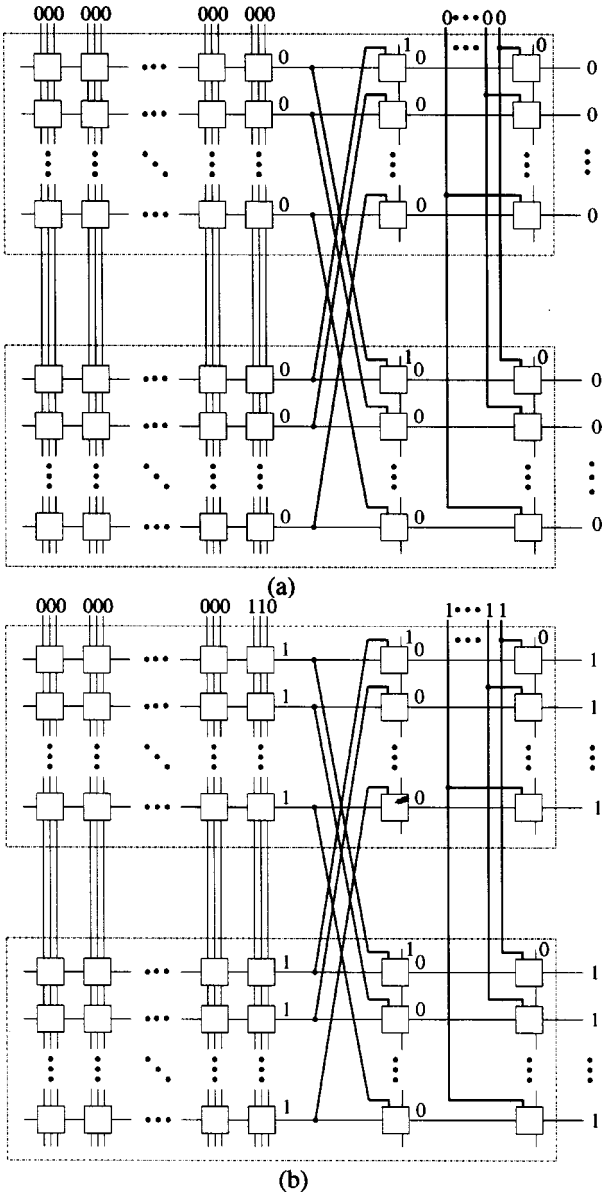


Fig. 11. (a) The test pattern for stuck-at-0 faults. (b) The test pattern for stuck-at-1 faults.

the input sequence $(s, a, z, c) = (1, 1, 1, 1)$. If a faulty cell exists, such as the shaded one in Fig. 9(b), then the fault effect can be propagated to the primary sum output of the neighboring row below. The all-0 test pattern can detect stuck-at faults in a similar way. Therefore, only 18 patterns are required to achieve 100% fault coverage for the MSA module. No extra data pins are required since the truth table is not augmented. This also results in a design of lower area overhead.

We have simulated the modified multiplier cell by SPICE. The simulation results show that 100% fault coverage is achieved after Phase-1 and -2 tests. A sample of the Phase-1 test is shown in Fig. 10, where the first row is the fault-free waveform of the sum output ($\hat{s}$), and each of the other rows represent the waveform of a particular fault. For example, the second row represents the sum output when there is a stuck-at-0 fault at node 6 in Fig. 8. Since it is different from the fault-free output, this fault is detected.

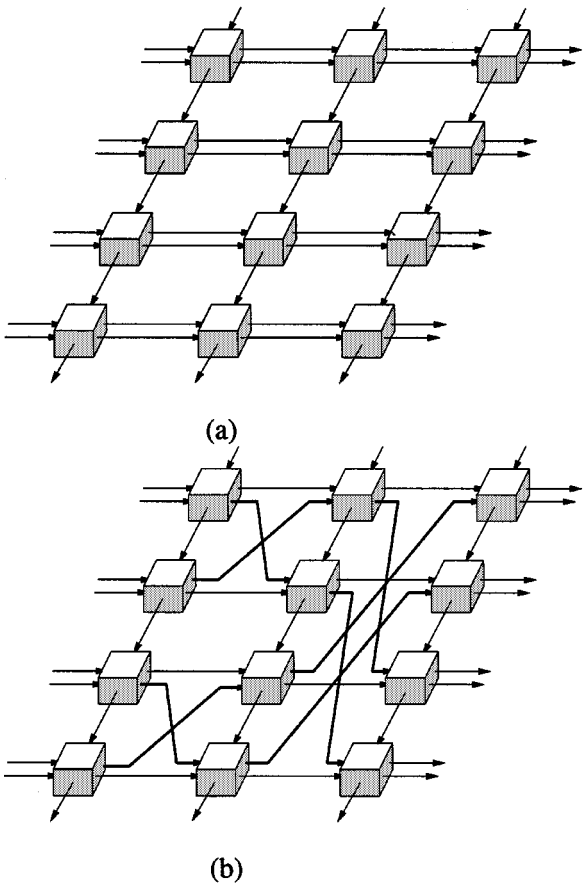


Fig. 12. (a) Module-level ILA. (b) Interconnect testing in normal mode.

As discussed above, the butterfly module configured as an ILA can be tested with only 18 test patterns, regardless of the array size. However, some interconnects which are used only in normal mode are not covered. In order to test those faults, two additional patterns are applied in normal mode, wherein the adder and subtractor cells of the two neighboring MSA modules are connected as in the original network (see Fig. 11) instead of the ILA form. We test the stuck-at and break faults of the interconnects (highlighted in the figure) by two test patterns. Let $A = (a_{n-1}, \ldots, a_0)$, $Z = (z_{n-1}, \ldots, z_0)$, $C = (c_0, \ldots, c_{n-1})$, and $N = (n_0, \ldots, n_{2n-1})$. The first pattern is the all-0 pattern, i.e., $\{A, Z, C, N\} = \{(0, \ldots, 00), (0, \ldots, 00), (0, \ldots, 00), (0, \ldots, 00)\}$, as shown in Fig. 11(a). It is easy to see that the fault-free output of the MSA module is also all-0. The second pattern is $\{A, Z, C, N\} = \{(0, \ldots, 01), (0, \ldots, 01), (0, \ldots, 00), (1, \ldots, 11)\}$, as shown in Fig. 11(b). The $(1, 1, 0)$ input pattern at the periphery indeed propagates to the input (a_0, z_0, c_{n-1}) of each multiplier cell in the last column of the array multiplier. The fault-free output of the multiplier cell is 1 according to Table I, so the fault-free output of the MSA module is the all-1 pattern. Apparently, any stuck-at or break faults of the highlighted interconnects in the MSA module can be detected by these two patterns. Therefore, as a whole, only 20 test patterns are required: 18 are used to detect all combinational single cell faults of the MSA module, and two

are used to detect the internal interconnect stuck-at and break faults.

Now we consider the entire FFT network. It can be configured in the form of a mesh-connected ILA based on the butterfly module discussed above, as shown in Fig. 12(a). Again, some original interconnects which are not part of the ILA, and thus not covered in test mode, have to be tested in normal mode [see the highlighted interconnects in Fig. 12(b)]. Fortunately, these interconnects can be tested by the Phase-2 test for all stuck-at and break faults, and the entire FFT network can be tested with only 20 test patterns.

III. Fault-Tolerant FFT Network

A. Reconfiguration Scheme

Fault-tolerant design by adding spare columns and/or spare rows is a well-known technique [28]. A fault-tolerant FFT network design using a spare column was proposed in [23]. In their scheme, the faulty column is replaced by the neighboring column to its right, which is in turn replaced by the next column to the right, and so on. Since the cells in different columns have different functions, special cells are required to implement the array. The functions of these cells are selected by control signals after reconfiguration.

We have noted that in the MSA module, all cells in the same column are the same. Furthermore, there are fewer cells in a row than in a column. Therefore, it is better to use a spare row instead of a spare column if possible. A fault-tolerant MSA module with a spare row (called the MMSA module) is shown in Fig. 13, where the spare row is located on the top of the array. Since two's-complement numbers are assumed, the array multiplier only needs to calculate a partial product term in the first row of the array [29], i.e., only one physical multiplier cell is required in the first row of the array, and the rest of the multiplier cells are dummy, as shown in the figure. Therefore, only three cell types (i.e., ***M***, ***S***, and ***A***) are required for the spare row. Note that this number is independent of the word length, and the cells need not be modified. Since the primary sum inputs of the multiplier should be all-0 in normal operation, the primary sum input of the spare row is grounded. Therefore, only reconfiguration switches are required to be added to the ***S*** cells, ***A*** cells, and primary outputs. The number of switches required is smaller than that in the spare-column scheme, which requires switches to be added to the multiplier, in addition to those mentioned above.

The reconfiguration of our fault-tolerant design is simple. If the lth row is faulty, then it is replaced by the $(l-1)$st row, which is in turn replaced by the $(l-2)$nd row, and so on. Finally, the first row is replaced by the spare row. Every cell in the faulty row must be bypassed. An example is shown in Fig. 14. Suppose the second row is faulty, it is replaced by the first row, and the first row by the spare row. The cells in the second row are bypassed after reconfiguration. The horizontal inputs of the ***S*** and ***A*** cells (i.e., m_1 and n_1, respectively) of the second row are routed to the first row, and those of the first row to the spare row, respectively. The outputs are rerouted likewise. The reconfiguration mechanism is simple, and only one control signal is needed to reconfigure the array. In contrast, to reconfigure columns with different functions, eight control signals are required in addition to more complicated cells [23].

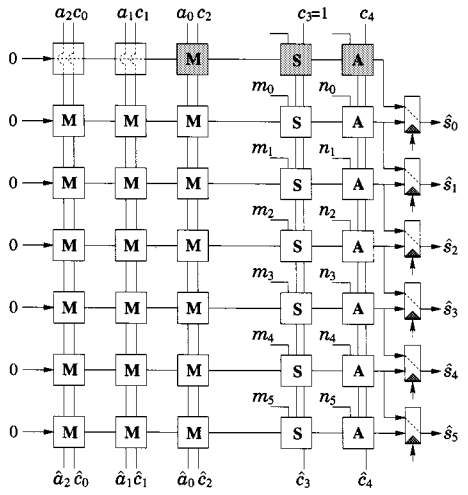


Fig. 13. MMSA module.

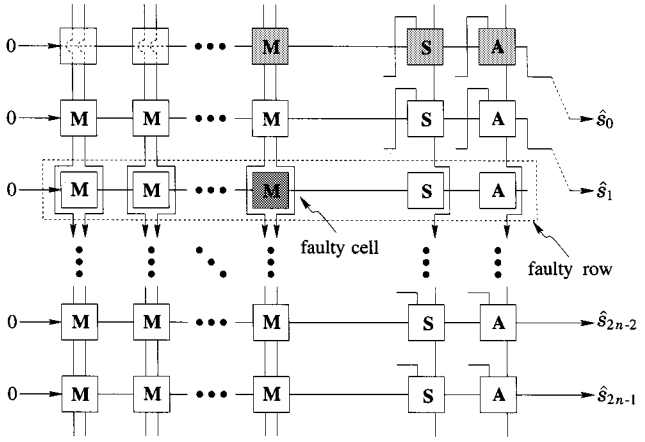


Fig. 14. A reconfiguration example.

B. Fault-Location Scheme

We will show that using the proposed approach, a faulty row can be located in $O(n)$ time, where n is the word length. As shown above, the bit-level ILA can be tested by 18 test patterns. The fault location and reconfiguration procedure is simple.

1) The 18 test patterns are applied to the array. If a fault is detected, the first row is replaced by the spare row.
2) The same test patterns are applied to the reconfigured array. If the output is correct, then the faulty cell is in the first row, else the first row is in turn replaced by the spare row, and the second row by the first row, and so on.
3) Repeat the previous step until the output is correct.

The worst case is that the faulty cell is in the last row and the reconfiguration step must be repeated $2n$ times, i.e., a faulty row can be located by at most $36n$ test patterns. Note that the reconfiguration is completed concurrently.

IV. Reliability and Hardware Overhead Analysis

A. Reliability

Let the reliability of the fault-tolerant schemes using a spare row and a spare column be R_r and R_c, respectively, and the size of 2-D array be $x \times y$. Assume that each cell becomes faulty

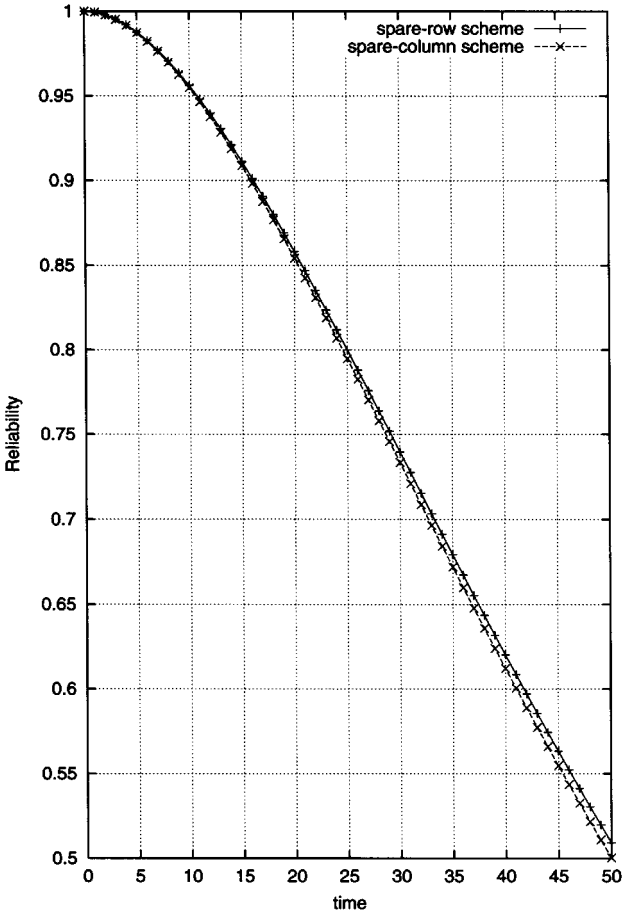


Fig. 15. Comparison of reliability between the spare-row and spare-column schemes for $\lambda = 0.0002$.

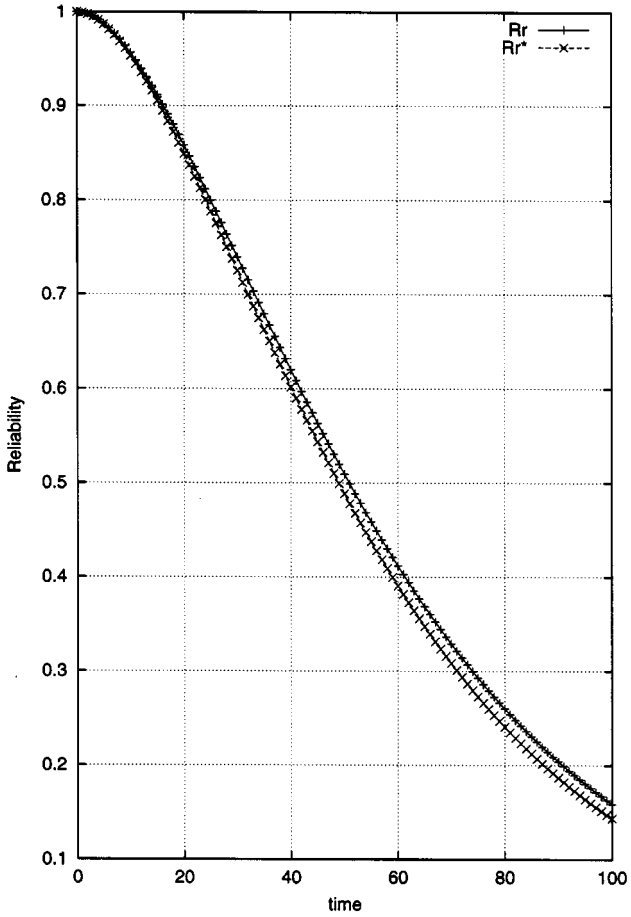


Fig. 16. Reliability of the spare-row scheme with and without considering the switch reliability.

randomly and independently, with a constant failure rate λ. The cell reliability is represented by $e^{-\lambda t}$, and the reliability of a row and a column are $e^{-y\lambda t}$ and $e^{-x\lambda t}$, respectively. Then

$$R_r = \sum_{i=0}^{1} \binom{x+1}{i} \cdot (e^{-y\lambda t})^{x+1-i} \cdot (1-e^{-y\lambda t})^i = (x+1-xe^{-y\lambda t})e^{-xy\lambda t} \quad (5)$$

$$R_c = \sum_{i=0}^{1} \binom{y+1}{i} \cdot (e^{-x\lambda t})^{y+1-i} \cdot (1-e^{-x\lambda t})^i = (y+1-ye^{-x\lambda t})e^{-xy\lambda t}. \quad (6)$$

The MSA module is implemented by an $x \times y$ bit-level array. As shown in Fig. 3, $x = 2n$, and $y = n+2$. Therefore

$$R_r = [2n+1-2ne^{-(n+2)\lambda t}]e^{-(2n^2+4n)\lambda t} \quad (7)$$

$$R_c = [n+3-(n+2)e^{-2n\lambda t}]e^{-(2n^2+4n)\lambda t}. \quad (8)$$

The spare-row approach has higher reliability as compared with the spare-column approach. The comparison of reliability over time for $\lambda = 0.0002$ is plotted in Fig. 15, where the reliability is normalized by Rr.

In the above reliability calculation we assumed that the extra switches are fault free. If the switch reliability (r_s) is not neglected, then the cell reliability should be $r_s \times e^{-\lambda t}$, and the MMSA module reliability becomes

$$R_r^* = [(2n+1) - 2n(r_s e^{-(n+2)\lambda t})]r_s e^{(2n^2+4n)\lambda t}. \quad (9)$$

The switches reduce the reliability of the MMSA module since $r_s < 1$. To estimate the effect, let the failure rate be proportional to the hardware complexity, i.e., the cell reliability is $e^{-(1+HO)\lambda t}$, wherein $r_s = e^{-(HO)\lambda t}$. For example, if the hardware overhead ratios (discussed later) are 4% and 23% for spare-row and spare-column schemes (typical for 16-bit words), respectively, then the reliability differences for both schemes are as shown in Figs. 16 and 17, respectively, where $\lambda = 0.0002$. Note that R_r^* and R_c^* represent the reliability with the switches considered, for the spare-row and spare-column schemes, respectively. From Fig. 16, we see that the switch effect on the reliability is small. On the contrary, Fig. 17 shows that the reliability of the spare-column scheme is significantly affected by the switches.

B. Hardware Overhead

Let TC_{msa} and TC_{mmsa} be the transistor counts of the MSA and MMSA modules, respectively. The hardware overhead ratio is defined as

$$HO = \frac{TC_{\mathrm{mmsa}} - TC_{\mathrm{msa}}}{TC_{\mathrm{msa}}}. \quad (10)$$

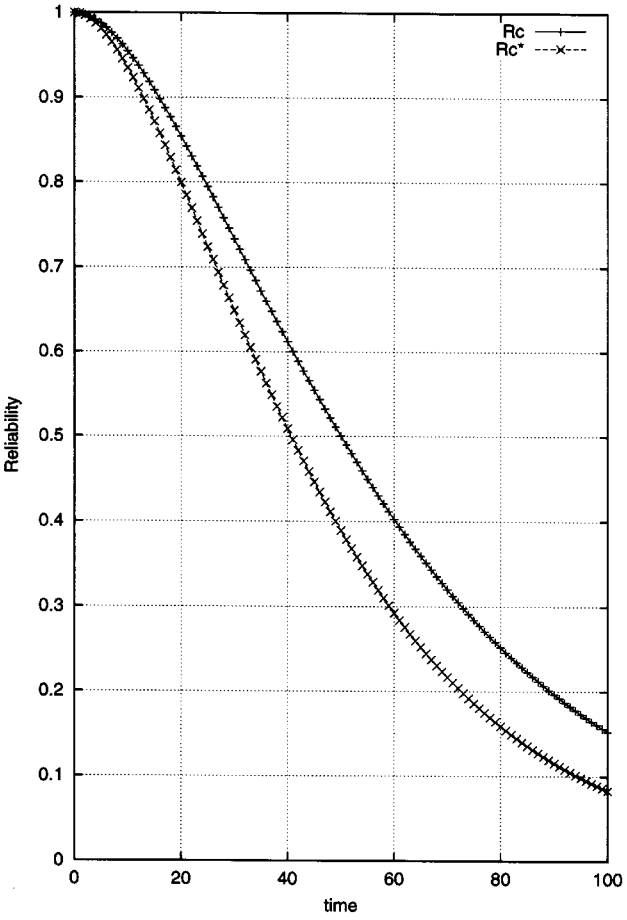


Fig. 17. Reliability of the spare-column scheme with and without considering the switch reliability.

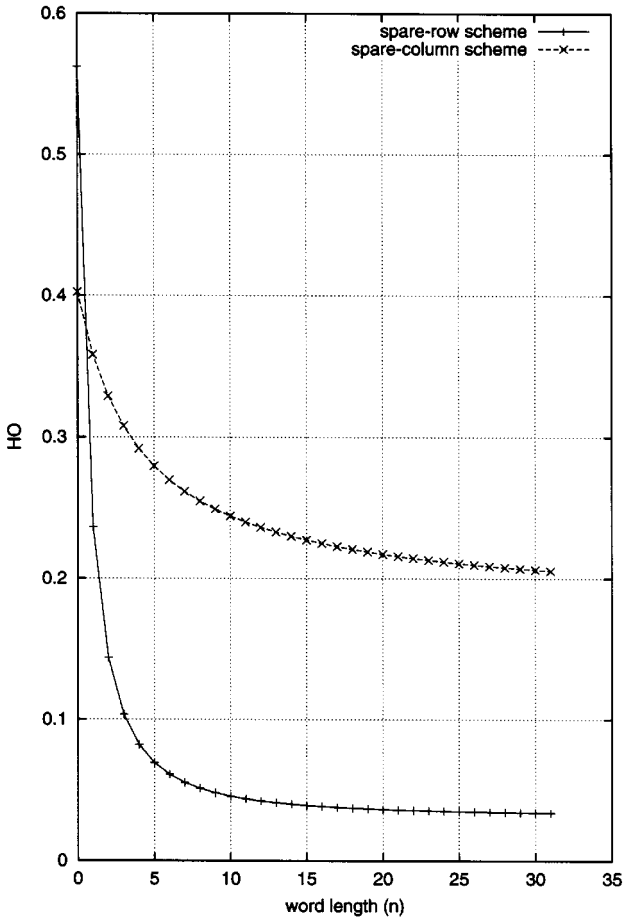


Fig. 18. Comparison of HO between the spare-row and spare-column schemes.

The switches in the MMSA module are implemented by transmission gates, so $TC_{\rm mmsa} = 264n^2+544n+398$ and $TC_{\rm msa} = 256n^2 + 516n$ [23]. We obtain

$$HO = \frac{8n^2 + 28n + 398}{256n^2 + 516n}. \tag{11}$$

Note that HO is only about 4% for n =16. Since the reconfiguration mechanism is simple and the cell function needs not be modified, the area for additional interconnects in our design is small. In contrast, the spare-column scheme needs complex reconfiguration mechanism and more control signals. In addition, the cell function must be modified, and the amount of additional interconnects is very large. The interconnects in the FFT network is the major contributor to silicon area, so the proposed scheme is much more cost-effective. The HO comparison between the spare-row (this work) and spare-column [23] schemes is shown in Fig. 18. From the figure, we see that the HO of the spare-row scheme is about one-sixth of that of the spare-column scheme.

Similarly, the hardware overhead introduced by the multiplexers $M1$ and $M2$ of the design-for-testability scheme is

$$\frac{8n^2 + 16n}{256n^2 + 516n} \tag{12}$$

and the value is 3.1% for $n = 16$. Note however that $M2$ can be shared with the fault-tolerant design, since a multiplexer must be added to the carry output of each multiplier cell to prevent the fault effect from propagating to the next row. Therefore, the overall hardware overhead for the testable and fault-tolerant design is about 4%+1.56% =5.56%.

V. Comparison and Example

Several off-line testing schemes for FFT networks were proposed in the past [12]–[14], [16]. The comparison of our C-testable design with these schemes is shown in Table IV. The scheme in [12] uses a BIST circuitry in each 8-point FFT network with 16-bit words. Since the pseudo-random test patterns are applied, 100% fault coverage (FC) can not be guaranteed with a test length (TL) of 4096. With a two-level test scheme, the interconnect faults (IFs) can be covered. Similarly, the scheme in [13] also is equipped with a BIST circuitry using 10 752 deterministic test patterns, achieving 99.1% FC. To test the IF's and the functionality of the system, a 1149.1 test-bus is required. In [14], the test procedure derived from algorithm flow graphs (AFGs) allows detection and location of all single faults. In addition, the IFs can be covered in $O(N)$ operations for an N-point FFT network. In [16], a C-testability scheme based on component-level faults was proposed. In this scheme, $16 + 3|T_m|$ test patterns are needed to test an FFT network,

TABLE IV
TEST FEATURE COMPARISON WITH PREVIOUS SCHEMES

Feature	[12]	[13]	[14]	[16]	Proposed
TL	4096	10752	$O(N)$	16+3$\|T_m\|$	20
FC	< 100%	99.1%	100%	100%	100%
IFs	Yes	No	Yes	No	Yes
Method	BIST	BIST	AFG	C-testability	C-testability
FL	No	No	Yes	No	Yes

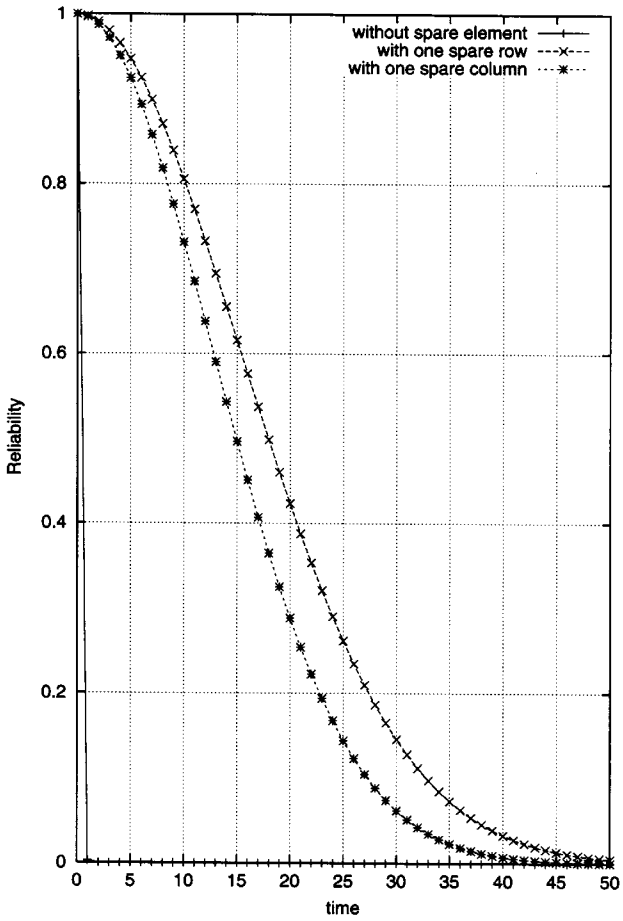


Fig. 19. Reliability of the entire FFT butterfly network with $\lambda = 4 \times 10^{-6}$.

where T_m is the number of test vectors required for testing a component. However, the IF's are not covered. Our proposed scheme, an improvement over our previous work [23], is superior to these as can be seen from the table. Moreover, our design has the fault location (FL) capability as in [14].

Example 1: Assume that there is a 512-points FFT butterfly network with 8-bit words. A 2-point butterfly module without design-for-testability circuit requires 2^{48} test patterns to guarantee 100% combinational fault coverage, since there are $6n$ inputs to the module. Moreover, $2^{48\times 256}$ test patterns are needed to exhaustively test the entire network since the primary inputs are composed of the inputs of 256 2-point butterfly modules. With the proposed C-testable scheme, however, only 20 test patterns are required.

Since each 2-point butterfly module has 4 MSA modules, the FFT circuit has $4 \times (N/2) \log_2 N$ MSA modules. Assume that the reliability of an MSA module without spare is R, then the reliability of the entire FFT butterfly network is

$$R_T = \prod_{i=0}^{2N \log_2 N - 1} R_i. \tag{13}$$

For $N = 512$, $R_T = R^{9216}$. If each MSA module has one spare row (column), then the reliability of the entire FFT network is $(R_r^*)^{9216}$ ($(R_c^*)^{9216}$). The comparative results are shown in Fig. 19, where we assume $\lambda = 4 \times 10^{-6}$. From the figure, we see that the reliability can be considerably improved by using just one spare row or one spare column. Also, the spare-row scheme is superior to the spare-column scheme. ■

VI. CONCLUSION

We have proposed a design-for-C-testability technique for the FFT butterfly network. The entire network can be completely tested with only 20 test patterns, regardless of the network size. A fault-tolerant extension to the scheme based on row redundancy also has been presented. We showed that only three bit-level cell types are required, and the cell function needs not be modified for reconfiguration. The reliability of the proposed design is higher than that of the previous column-redundancy design. For 16-bit words, e.g., the hardware overhead for the testable FFT network is only about 4%, regardless of the network size, which is much lower than that of the previous work, and is within a practical limit.

The proposed design-for-C-testability scheme based on the assumption that the FFT array is no sharing of hardware. If it is applied to the time-multiplexing architecture, then the portion of faults for the multiplexers can not be detected. But they can easily be covered with specified test patterns. The hardware overhead due to the interconnections for configuring the network between test and normal modes is not estimated. However, the interconnections are local and located at the same metal layer. Thus, they cannot result in larger hardware overhead than that of the previous approach [16] (additional global interconnects are required).

REFERENCES

[1] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
[2] C. C. W. Hui, T. J. Ding, J. V. McCanny, and R. F. Woods, "A 64-point Fourier transform chip for video motion compensation using phase correlation," *IEEE J. Solid-State Circuits*, vol. 31, pp. 1751–1761, Nov. 1996.
[3] A. E. Cetin, O. N. Gerek, and Y. Yardimci, "Equiripple FIR filter design by FFT algorithm," *IEEE Signal Processing Mag.*, pp. 60–64, Mar. 1997.
[4] P. Noll, "MPEG digital audio coding," *IEEE Signal Processing Mag.*, pp. 59–81, Sept. 1997.
[5] E. Bidet, D. Castelain, C. Joanblanq, and P. Senn, "A fast single-chip implementation of 8192 complex point FFT," *IEEE J. Solid-State Circuits*, pp. 300–305, Mar. 1995.
[6] J. O'Brien, J. Mather, and B. Holland, "A 200 MIPS single-chip 1k FFT processor," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Feb. 1989, pp. 166–167 and 327.
[7] M. Wosnitza, M. Cavadini, M. Thaler, and G. Troster, "A high precision 1024-point FFT processor for 2D convolution," in *Proc. IEEE Int. Solid-States Circuits Conf. (ISSCC)*, Feb. 1998, pp. 118–119 and 424.
[8] B. M. Baas, "A low-power, high-performance, 1024-point FFT processor," *IEEE J. Solid-State Circuits*, vol. 34, pp. 380–387, Mar. 1999.

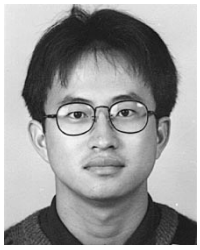
[9] T. Chen, G. Sunada, and J. Jin, "COBRA: a 100-MOPS single-chip programmable and expandable FFT," *IEEE Trans. VLSI Syst.*, vol. 7, pp. 174–182, June 1999.
[10] J.-Y. Jou and J. A. Abraham, "Fault-tolerant FFT networks," *IEEE Trans. Comput.*, vol. 37, pp. 548–461, May 1988.
[11] Y.-H. Choi and M. Malek, "A fault-tolerant FFT processor," *IEEE Trans. Comput.*, vol. 37, pp. 617–621, May 1988.
[12] K. Yamashita, A. Kanasugi, S. Hijiya, G. Goto, N. Matsumura, and T. Shirato, "A wafer-scale 170 000-gate FFT processor with built-in test circuits," *IEEE J. Solid-State Circuits*, vol. 23, pp. 336–342, Apr. 1988.
[13] V. K. Jain, S. A. Al-Arian, D. L. Landis, and H. A. Nienhaus, "Fully parallel and testable WSI architecture for an FFT processor," *Int. J. Computer-Aided VLSI Design*, vol. 3, pp. 113–13, 1991.
[14] A. Antola and M. G. Sami, "Testing and diagnosis of FFT arrays," *J. VLSI Signal Processing*, no. 3, pp. 225–236, 1991.
[15] F. Lombardi and J. Muzio, "Concurrent error detection and fault location in an FFT architecture," *IEEE J. Solid-State Circuits*, vol. 27, pp. 728–736, May 1992.
[16] C. Feng, J. C. Muzio, and F. Lombardi, "On the testability of the array structures for FFT computation," *J. Electron. Testing: Theory and Applic.*, vol. 4, pp. 215–224, Aug. 1993.
[17] D. L. Tao and C. R. P. Hartmann, "A novel concurrent error detection scheme for FFT networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, pp. 198–221, Feb. 1993.
[18] C.-W. Wu and C.-T. Chang, "FFT butterfly network design for easy testing," *IEEE Trans. Circuits Syst. II*, vol. 40, pp. 110–115, Feb. 1993.
[19] S.-K. Lu, C.-W. Wu, and S.-Y. Kuo, "Enhancing testability of VLSI arrays for fast Fourier transform," *Proc Inst. Elect. Eng.—E*, vol. 140, no. 3, pp. 161–166, May 1993.
[20] S. J. Wang and N. K. Jha, "Algorithm-based fault tolerance for FFT networks," *IEEE Trans. Comput.*, vol. 43, pp. 849–854, July 1994.
[21] C. G. Oh and H. Y. Youn, "On concurrent error location and correction of FFT networks," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 257–260, June 1994.
[22] C. G. Oh, H. Y. Youn, and V. K. Raj, "An efficient algorithm-based concurrent error detection for FFT network," *IEEE Trans. Comput.*, vol. 44, pp. 1157–1162, Sept. 1995.
[23] S.-K. Lu, C.-W. Wu, and S.-Y. Kuo, "On fault-tolerant FFT butterfly network design," *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, pp. 69–72, May 1996.
[24] H. Fujiwara and S. Toida, "The complexity of fault detection problems for combinational logic circuits," *IEEE Trans. Comput.*, vol. C-31, pp. 555–560, June 1982.
[25] C.-W. Wu and P. R. Cappello, "Easily testable iterative logic arrays," *IEEE Trans. Comput.*, vol. 39, pp. 640–652, May 1990.
[26] P. R. Menon and A. D. Friedman, "Fault detection in iterative arrays," *IEEE Trans. Comput.*, vol. C-20, pp. 524–535, May 1971.
[27] S.-K. Lu, J.-C. Wang, and C.-W. Wu, "C-testable design techniques for iterative logic arrays," *IEEE Trans. VLSI Systems*, vol. 3, pp. 146–152, Mar. 1995.
[28] B. W. Johnson, *Design and Analysis of Fault Tolerant Digital Systems*. Reading, MA: Addison-Wesley, 1989.
[29] P. R. Cappello and C.-W. Wu, "Computer-aided design of VLSI FIR filters" (in also translated into Russian), *Proc. IEEE*, vol. 75, pp. 1260–1271, Sept. 1987.

Jin-Fu Li received the B.S.E.E. degree in 1995 from National Taiwan University of Science and Technology, Taipei, Taiwan, and the M.S.E.E. degree in 1999 from National Tsing Hua University, Hsinchu, Taiwan, where he is currently working toward the Ph.D. degree.

His research interests include VLSI design and testing, fault-tolerant design, memory testing, diagnosis, and self-repair.

Shyue-Kung Lu received the Ph.D. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1995.

From 1995 to 1998, he was an Associate Professor in the Department of Electrical Engineering, Lunghwa Junior College of Technology and Commerce. Since 1998, he has been with the Department of Electronics Engineering, Fu Jen Catholic University, Taipei, Taiwan, where he is an Associate Professor. His research interests include the areas of VLSI testing and fault-tolerant computing.



Shih-Arn Hwang (S'96–M'98) received the B.S. and Ph.D. degrees in electrical engineering from National Tsing Hua University, Hsinchu, Taiwan, in 1994 and 1998, respectively.

His research interests include VLSI testing and design of high performance application-specific VLSI circuits and systems. He is currently with Realtek Semiconductor Corp., Hsinchu, Taiwan.

Cheng-Wen Wu (S'86–M'87–SM'95) received the B.S.E.E. degree in 1981 from National Taiwan University, Taipei, Taiwan, and the M.S. and Ph.D. degrees, both in electrical and computer engineerin from the University of California at Santa Barbara (UCSB) in 1985 and 1987, respectively.

From 1981 to 983, he was an Ensign Instructor at the Chinese Naval Petty Officers' School of Communications and Electronics, Tsoying, Taiwan. From 1983 to 1984, he was with the Information Processing Center of the Bureau of Environmental Protection, Executive Yuan, Taipei, Taiwan. From 1985 to 1987, he was a Post Graduate Researcher at the Center for Computational Sciences and Engineering, UCSB. Since 1988, he has been with the Department of Electrical Engineering, National Tsing Hua University, Hsinchu, Taiwan, where he is currently a Professor. He also has served as the Director of the University's Computer and Communications Center during 1996–1998, and the Director of the University's Technology Service Center during 1998–1999. From August 1999 to February 2000, he was a Visiting Faculty of the ECE Department, UCSB. His interests lie in design and testing of high-performance VLSI circuits and systems.

Dr. Wu was the Technical Program Chair of the IEEE Fifth Asian Test Symposium (ATS'96), and is the General Chair of ATS'00. He also is the Guest Editor of the *Journal of Information Science and Engineering, Special Issue on VLSI Testing*. He received the Distinguished Teaching Award from NTHU in 1996 and the Outstanding Electrical Engineering Professor Award from the Chinese Institute of Electrical Engineers (CIEE) in 1997.He is a Member of CIEE.