Enhancing testability of VLSI arrays for fast Fourier transform

S.-K. Lu
C.-W. Wu
S.-Y. Kuo

Indexing terms: Digital signal processing, Fast Fourier transforms, Very large-scale integration, Butterfly networks, Shuffle networks

Abstract: Fast-Fourier-transform (FFT) algorithms are used in various digital signal-processing applications, such as linear filtering, correlation analysis and spectrum analysis. With the advent of very large-scale-integration (VLSI) technology, a large collection of processing elements can be gathered to achieve high-speed computation economically. However, owing to the low pin-count/component-count ratio, the controllability and observability of such circuits decrease significantly. As a result, testing of such highly complex and dense circuits becomes very difficult and expensive. M-testability conditions for butterfly-connected and shuffle-connected FFT arrays are proposed. Based on them, a novel design-for-testability approach is presented and applied to the module-level systolic FFT arrays. The M-testability conditions guarantee 100% single-module-fault testability with a minimum number of test patterns.

1 Introduction

Fast-Fourier-transform (FFT) algorithms are among the most important digital-signal-processing (DSP) algorithms. They provide a means of speeding up discrete-Fourier-transform (DFT) computations greatly. The performance improved by FFTs has made the realisation of many sophisticated signal-processing algorithms economical. Owing to the rapid advance in semiconductor-fabrication technology, a large number of processing elements can be integrated on a single chip. It is therefore possible that special purpose VLSI chips will soon be used to construct FFT systems. A straightforward implementation of the N-point FFT uses 2-input butterflies, which consist of $\log_2 N$ stages, and each stage contains $N/2$ 2-input butterflies. However, integrating a large number of processors on a single chip results in an increase in the logic-per-pin ratio, which drastically reduces the controllability and observability of the logic on the chip. Consequently, testing such highly complex and dense circuits becomes very difficult and expensive.

© IEE, 1993

Paper 9371E (C3), first received 3rd April and in revised form 26th October 1992

S.-K. Lu and S.-Y. Kuo are with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, Republic of China

C.-W. Wu is with the Department of Electrical Engineering, National Tsin Hua University, Hsinchu, Taiwan, Republic of China

Researchers have proposed fault-tolerance approaches, e.g. triple modular redundancy (TMR) with voting [1], hybrid redundancy [2], recomputing with shifted operands (RESO) [3] and triple time redundancy [4]. Some other concurrent-error-detection (CED) schemes for FFT networks can be found in References 5–9. Choi and Malek [8] propose a scheme called recomputing by alternate path, for concurrent error detection and fault diagnosis of FFT networks. Once an error is detected, a faulty butterfly can be located within $\log_2 N + 5$ additional cycles. The method has full fault detection and location capability regardless of the magnitude of the round-off errors. Reliability and availability are improved. The hardware overhead is $O(1/\log N)$ and the time redundancy is 100%. Jou and Abraham [6] have presented an algorithm-based fault-tolerant scheme for FFT networks. They have shown that 100% fault coverage and no loss of throughput can be achieved theoretically. The effect of round-off errors are also discussed. This scheme brings forth approximately $2/\log N$ hardware overhead. Lombardi and Muzio in a recent paper [9] presented a new approach for CED and fault location in homogeneous VLSI/WSI architectures for computing complex FFT. Their approach is based on the relationship between the computations of cells at a given point distance. The drawback is that the hardware overhead is too high and no efficient reconfiguration scheme is proposed. Tsunoyama and Naito [7] proposed another CED (by recomputing) scheme for FET processors and presented a fault-tolerant FFT processor using this scheme. The processor is modelled by a linear cellular automaton having the constant-weight and equidistance properties. Tao *et al.* [5] also proposed an algorithm-based CED scheme for FFT processors, which maintains the low hardware overhead and high throughput of Jou and Abraham's scheme, and at the same time increases the fault coverage significantly.

It is well known that the general logic-testing problem is NP-complete. For certain iterative logic arrays (ILAs), however, the fault-detection problem is solvable in polynomial time [10, 11]. In this paper, we show that the FFT processor can be viewed as an ILA. Our work has been grounded on the theory established in a series of papers [10, 12–15]. In these papers, testability conditions are proposed for M-testable mesh-connected arrays, hexagonally connected arrays, sequential arrays and bilateral arrays. Techniques for designing such arrays are proposed. Applications in digital signal processing also are discussed. They do not, however, cover butterfly-connected and shuffle-connected arrays, such as FFT networks and odd–even transposition sorters. In this paper, we first propose M-testability conditions for butterfly-

connected and shuffle-connected arrays, and a design-for-testability approach is then applied to the module-level systolic array for computing FFT. Our M-testability conditions guarantee 100% single-module-fault testability with a minimum number of test patterns (independent of the array size). Our design-for-testability approach is shown to require negligible hardware overhead. In addition, our testable design results in simple built-in self-test (BIST) structures [15].

2 Fast Fourier transform

The discrete Fourier transform (DFT) is defined by the equation

$$X(k) = \sum_{n=0}^{N-1} x(n) w_N^{nk} \quad k = 0, \ldots, N-1 \tag{1}$$

where

$$w_N = \exp\{-j(2\pi/N)\} \tag{2}$$

The term w_N^{nk} is known as the twiddle factor. A straightforward way of implementing an N-point FFT in hardware is by a $\log_2 N$-stage butterfly network. Each stage contains $N/2$ 2-point butterfly modules. In this paper, such a circuit is called an FFT network, which is assumed to be pipelined. An 8-point FFT network is shown in Fig. 1. The butterfly module (i.e. the 2-point

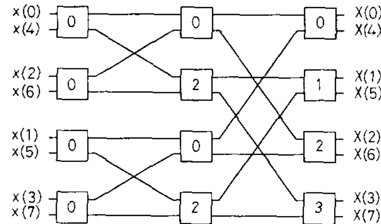

Fig. 1 *8-point FFT network*

FFT module) is shown in Fig. 2, which performs the computation

$$X_{out} = X_{in} + Y_{in} W^n \tag{3}$$

$$Y_{out} = X_{in} - Y_{in} W^n \tag{4}$$

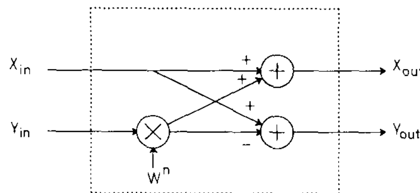

Fig. 2 *Butterfly module*

where W^n is a representative twiddle factor. All the quantities in these equations are complex-valued. For implementation purposes, it is necessary to use a functionally equivalent butterfly which employs only real quantities and real operations. Let us express X_{in}, Y_{in} and W^n in complex form as

$$X_{in} = X_R + jX_I \tag{5}$$

$$Y_{in} = Y_R + jY_I \tag{6}$$

and

$$W^n = W_R + jW_I \tag{7}$$

where $j = -1$. Combining these equations, we can recast X_{out} and Y_{out} as

$$\begin{aligned} X_{out} &= X_R^O + jX_I^O \\ &= (X_R + jX_I) + (W_R + jW_I)(Y_R + jY_I) \\ &= (X_R + W_R Y_R - W_I Y_I) \\ &\quad + j(X_I + W_I Y_R + W_R Y_I) \end{aligned} \tag{8}$$

and

$$\begin{aligned} Y_{out} &= Y_R^O + jY_I^O \\ &= (X_R + jX_I) - (W_R + jW_I)(Y_R + jY_I) \\ &= (X_R - W_R Y_R + W_I Y_I) \\ &\quad + j(X_I - W_I Y_R - W_R Y_I) \end{aligned} \tag{9}$$

The butterfly module can be constructed with four identical multiply–subtract–add (MSA) modules, as shown in Fig. 3.

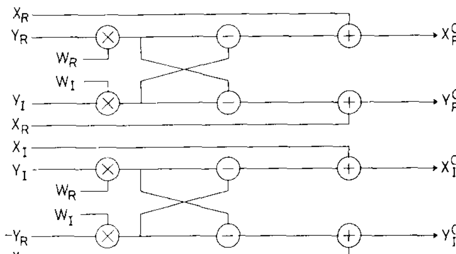

Fig. 3 *Butterfly structured as four MSA modules*

3 Testability of butterfly and shuffle networks

A cell is a combinational machine with function $f: \{0, 1\}^w \to \{0, 1\}^w$, where w denotes the input or output word length. An ILA is an array of cells. We use the terms array and ILA interchangeably. A complete or exhaustive input sequence σ for a cell is an input sequence consisting of all possible input combinations for the cell, i.e. $\sigma = \sigma_1 \sigma_2 \ldots \sigma_k$, where $\forall \sigma_i \in \Sigma$ (the input set), $i \in 1, 2, \ldots, k$. A complete output sequence $\delta = \delta_1 \delta_2 \ldots \delta_k$ is defined analogously. A minimal complete sequence is a shortest such sequence (which has a length of 2^w). We say that an array is M-testable if it is 2^w-testable, i.e. testable with only 2^w patterns. We assume that the cell's behaviour is invariant over time, even if it is faulty. A faulty cell's function may deviate from the correct one in any manner, as long as it remains combinational. That is, we are testing only for permanent combinational faults. The model adopted is the single-cell model [10], which has been used as the cell-level fault model by most researchers dealing with ILA testing.

In Reference 10, it is concluded that a k-dimensional ILA is M-testable if it has a bijective cell function, where k is an arbitrary positive integer. A 2-dimensional example is shown in Fig. 4. M-testability stands for 2^w-testability, i.e. the whole array can be pseudoexhaustively verified with only 2^w test patterns regardless of how many cells there are in the array. We may also denote that $M = 2^w$. Let $\sigma = \sigma_1 \sigma_2 \ldots \sigma_M$ be a minimal complete input sequence. Then any permutation of σ also is a

minimal complete input sequence. In Fig. 4, we see that all cells whose indices sum to the same number, i.e. those which lie in the same $+45°$ diagonal line, receive the

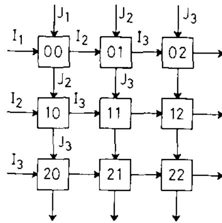

Fig. 4 *2-dimensional +45° tessellation*

same complete input sequence. This pattern is called a $+45°$ tessellation. Thanks to the bijective cell function, any fault is propagated to some observable primary output. A fault results in an input change to the cell's horizontal and/or vertical neighbouring cells. This input change is propagated to an output change. Bijectivity ensures that the change continues to ripple to some external output; the propagating paths cannot mask each other.

We now turn to the FFT arrays. A straightforward implementation of an N-point FFT network is to use 2-point butterflies, which consists of $\log_2 N$ stages where each stage contains $N/2$ 2-input butterflies. Let the inputs X_{in} and Y_{in} of a butterfly module be assigned the values a and b, respectively, and let c and d be the values of their corresponding outputs (see Fig. 2). Since

$$X_{out} = a + bw = c \tag{10}$$

and

$$Y_{out} = a - bw = d \tag{11}$$

where $w \equiv W^n$, the bijectivity of the module function can easily be verified.

Theorem 1: The butterfly module of an FFT processor has a bijective cell function.

Proof: Denote the module function as f. For an input $I_1 = (a, b)$, the resulting output is $O_1 = (a + bw, a - bw)$. Assume that $I_2 = (a + \varepsilon, b + \delta) \neq I_1$, and its corresponding output is

$$\begin{aligned} O_2 &= f(a + \varepsilon, b + \delta) \\ &= (a + bw + \varepsilon + \delta w, a - bw + \varepsilon - \delta w) \end{aligned} \tag{12}$$

If $O_1 = O_2$, then $\varepsilon + \delta w = 0$ and $\varepsilon - \delta w = 0$. This holds only if $\varepsilon = \delta = 0$, which implies that $I_1 = I_2$, a contradiction to the assumption. Hence, by definition the cell function f is bijective.

Theorem 2: An N-point FFT butterfly network can be made M-testable by swapping the outputs of the lower left cells of each 4-point module, where $N = 2^n$, $n \in N$.

Proof: The case where $N = 2$ is trivial according to the previous theorem. For $N = 4$, i.e. a 4-point FFT module as shown in Fig. 5, we apply a minimal complete sequence $\sigma_1 = (I_1, J_1)$ to both cell_{00} and cell_{10}. Since the cell function is a bijection, the output sequences $\sigma_2 = (I_2, J_2)$ of both cells are also minimal complete. We swap the outputs of cell_{10}, then cell_{01} and cell_{11} receive sequences (I_2, J_2) and (J_2, I_2), respectively. Since (I_2, J_2) is minimal complete, so is (J_2, I_2). The resulting outputs of cell_{01} and cell_{11} are (I_3, J_3) and (I_4, J_4), respectively, which are also minimal complete. Using this tessellation, each cell receives a minimal-complete input sequence. Thanks to

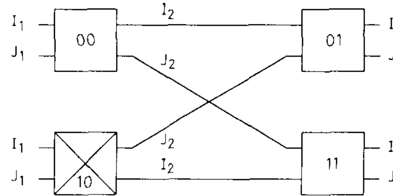

Fig. 5 *4-point FFT processor*

the bijectivity of the cell function, any fault is propagated to some observable primary outputs concurrently. The 4-point FFT module is therefore M-testable after a mechanism is implemented on cell_{10} to facilitate the capability of swapping its outputs during the test phase of the network (which will be discussed in Section 4).

Let us number the stages of the N-point FFT network from 0 to $n - 1$, and assume that all 2^p point FFT arrays at stage $p - 1$ are M-testable, where $p < n$. We need to show that the 2^{p+1}-point FFT array at stage p is M-testable. We use $m(r, p)$ to denote the rth module of the pth stage, where $0 \leqslant r \leqslant N/2 - 1$ and $0 \leqslant p \leqslant n - 1$. Now, a 2^{p+1}-point FFT array can be constructed by using two 2^p-point FFT arrays and routing their outputs to an additional (final) stage. According to our pattern tessellation, the output sequences of $m(r, p - 1)$ and $m\{(r + 2^{p-1}) \bmod N/2, p - 1\}$, $0 \leqslant r \leqslant N/2 - 1$, are the same. Moreover, $m(r, p)$ and $m\{(r + 2^{p-1}) \bmod N/2, p\}$, together with $m(r, p - 1)$ and $m\{(r + 2^{p-1}) \bmod N/2, p - 1\}$, form a 4-point FFT module (see Fig. 6 for an

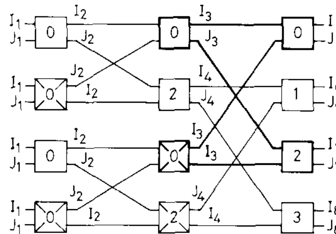

Fig. 6 *Test pattern of 8-point FFT processor*

example). To forward a complete sequence to $m(r, p)$ and $m\{(r + 2^{p-1}) \bmod N/2, p\}$, we swap the outputs of the lower left-hand module of this 4-point butterfly. Each module at stage p then receives a minimal-complete input sequence. An 8-point FFT network is shown in Fig. 6 to illustrate the approach. By induction, the butterfly-connected FFT array is M-testable after a mechanism is implemented on the lower left-hand cells of each 4-point FFT module to facilitate the capability of swapping their outputs during the testing phase of the network.

Theorem 3: An N-point FFT array with shuffle interconnections can be made M-testable by swapping the two outputs of each of the cells (2-point modules) $m(r, p)$. $N/4 \leqslant r \leqslant N/2 - 1, 0 \leqslant p \leqslant \log_2 N - 2$.

Proof: We apply a minimal-complete sequence $\sigma_1 = (I_1, J_1)$ to each cell at stage 0. Since the cell function is a

bijection, the corresponding output sequence $\sigma_2 = (I_2, J_2)$ for each module $m(i, 0)$, $0 \leqslant i \leqslant N/2 - 1$, is also a minimal-complete sequence. As in Theorem 2, we then identify all 4-point butterfly modules, and swap the outputs of their lower left-hand cells during the testing phase of the network. Let $O^x(r, p)$ and $O^y(r, p)$ denote the values of X_{out} and Y_{out} of $m(r, p)$, respectively, where $0 \leqslant r \leqslant N/2 - 1$ and $0 \leqslant p \leqslant \log_2 N - 1$. By the definition of the shuffle network, $\{O^x(r, p), O^x(r + N/4, p)\}$ is the input of $m(2r, p + 1)$, and $\{O^y(r, p), O^y(r + N/4, p)\}$ is the input of $m(2r + 1, p + 1)$, where $0 \leqslant r \leqslant N/4 - 1$ (see Fig. 7 for an example). The lower left-hand cells of the

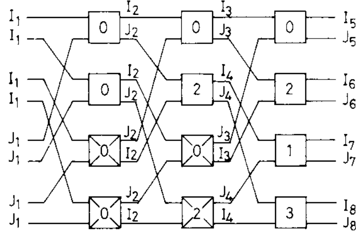

Fig. 7 *Shuffle-connected FFT processor*

4-point butterfly modules are therefore $m(r, p)$, $N/4 \leqslant r \leqslant N/2 - 1, 0 \leqslant p \leqslant \log_2 N - 2$.

According to the tessellation, $\forall 0 \leqslant p \leqslant \log_2 N - 1$, $\forall 0 \leqslant r \leqslant N/4 - 1$, $m(r, p)$ and $m(r + 2^p, p)$ receive the identical input sequence. By induction, each module receives a minimal-complete input sequence. An 8-point FFT shuffle network is shown in Fig. 7 to illustrate the approach. Following a similar argument to that given in the proof of theorem 2, we conclude that the FFT shuffle network is M-testable.

4 Designing testable FFT networks

We know from Theorem 2 that the function of a 2-point FFT module is bijective. This leaves us the task of swapping the outputs of these cells (as suggested in the previous theorems) to make them M-testable. The swapping mechanism can be implemented with negligible cost, since its property is inherent in the computation of the FFT modules.

It is assumed that the signals and spectra are complex. We rewrite the FFT equation as

$$X_{out} = X_{in} + Y_{in} W^n \tag{13}$$

$$Y_{out} = X_{in} - Y_{in} W^n \tag{14}$$

In Section 2, we showed that a butterfly module can be constructed with four identical MSA modules (see Fig. 3). Our goal now is to swap the outputs X_{out} and Y_{out} of the specified modules during the testing phase. Let X_{out} and Y_{out} be the outputs of a module after swapping, then the module performs the function

$$X_{out} = X_{in} - Y_{in} W^n = Y_R^O + jY_I^O \tag{15}$$

$$Y_{out} = X_{in} + Y_{in} W^n = X_R^O + jX_I^O \tag{16}$$

From eqns. 8 and 9,

$$X_{out} = (X_R - W_R Y_R + W_I Y_I) + j(X_I - W_I Y_R - W_R Y_I) \tag{17}$$

$$Y_{out} = (X_R + W_R Y_R - W_I Y_I) + j(X_I + W_I Y_R + W_R Y_I) \tag{18}$$

Comparing eqns. 17 and 18 with eqns. 8 and 9, respectively, we see that swapping the outputs is tantamount to changing the sign of W, or replacing the adders by subtractors and vice versa. This can be done by using four multiply–add–subtract (MAS) modules as shown in Fig. 8. When in normal mode, each 2-point FFT module is

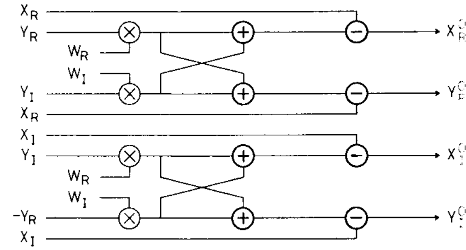

Fig. 8 *Swapped butterfly structured as four MAS modules*

configured as four MSA modules, and their outputs are not swapped. In test mode, each of the specified FFT modules is configured as four MAS modules, and their outputs are swapped. A control circuit is designed for the FFT modules to switch between these two configurations. The original butterfly consists of four MSA modules. To make the new butterfly switchable between these two modes of operation, we design the MSA module in the form of an ILA as shown in Fig. 9, where

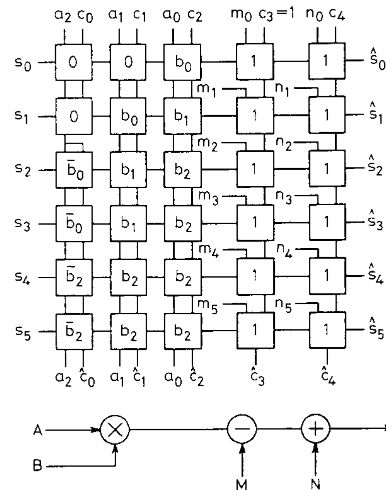

Fig. 9 *MSA in the form of an ILA*

the word length w is 3. In this design, three types of basic cells are used: the multiplier cell, the adder cell and the subtractor cell. These cells have about the same complexity: a full adder and an AND gate (plus some 1-bit latches if pipelined at the bit level). A 1-bit subtractor has an additional inverter for negating one of the operands. The detailed implementations of these three types of cells are shown in Fig. 10. To implement our swapping mechanism, we design a cell which has two modes of operation: addition and subtraction, as depicted in Fig. 11. When T is 0, the cell is a subtractor cell. Setting T to 1, instead, the cell becomes an adder cell. The CMOS XNOR gate can be implemented with four transistors. By controlling T, we can switch the function of the cell,

which makes our purpose of swapping the outputs of the specified FFT modules during the testing phase possible, with very small cost.

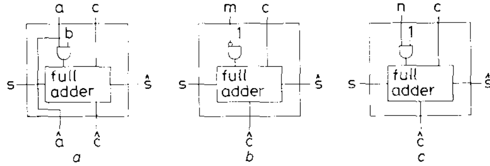

Fig. 10 *Cells*
a Multiplier cell
b Adder cell
c Subtractor cell

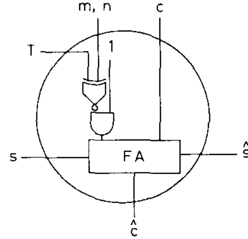

Fig. 11 *Modified adder/subtractor cell*

In the rest of this Section, we discuss the hardware overhead of our technique for realising M-testable FFT networks. Since the number of modified modules in both butterfly and shuffle-connected FFT processors is the same, the following hardware-overhead estimation is applicable to both configurations. We define C_{module} as the transistor count of a single butterfly module in the original FFT network, and C_T as the total transistor count of the network. The term C_E is defined as the number of additional transistors (compared with C_T) in our M-testable design of the network. The hardware overhead ratio is defined as

$$O_H = \frac{C_E}{C_T} \tag{19}$$

The straight implementation of an N-point FFT network using 2-point modules consists of $\log_2 N$ stages and each stage contains $N/2$ modules, i.e. the network has a total of $N/2 \log_2 N$ modules. Let w denote the word length and N denote the point size. Referring to Fig. 9, we see that different types of cells are used in the MSA module. In the 3-bit multiplier, the cells with complemented coefficient are subtractor cells (for 2s-complement multipliers). In the subtractor, we use subtractor cells, and in the adder, we use adder cells. The number of multiplier cells, adder cells and subtractor cells in an MSA module are denoted as C_M, C_A and C_S, respectively. From Fig. 9 we see that

$$C_M = w \times 2w - (w + 1) = 2w^2 - w - 1 \tag{20}$$

$$C_A = 2w \tag{21}$$

and

$$C_S = 2w + w + 1 = 3w + 1 \tag{22}$$

The transistor count of a module of the original design is

$$\begin{aligned} C_{module} &= (C_M + C_A)64 + C_S 66 \\ &= 128w^2 + 262w + 2 \end{aligned} \tag{23}$$

The total number of transistors in the network is therefore

$$C_T = (N/2 \log_2 N)(128w^2 + 262w + 2) \tag{24}$$

The number of modules with outputs swapped is $N/4(\log_2 N - 1)$, each of which requires $(4 + 2)2w$ extra transistors. A 4-transistor XNOR gate is used for turning an adder cell into an adder/subtractor cell as shown in Fig. 11. Also, two transistors are required to modify a subtractor cell. The number of extra transistors required for M-testability is hence

$$\begin{aligned} C_{TE} &= (4 + 2) \times 2w \times N/4 \times (\log_2 N - 1) \\ &= 3wN(\log_2 N - 1) \end{aligned} \tag{25}$$

The hardware overhead ratio can therefore be calculated as

$$\begin{aligned} O_H &= C_{TE}/C_T \\ &= \frac{3wN(\log_2 N - 1)}{(N/2 \log_2 N)(128w^2 + 262w + 2)} \\ &\simeq \frac{6}{128w + 262} \end{aligned} \tag{26}$$

which is very small even for a short word length. For example, if $w = 8$, then $O_H \simeq 0.47\%$.

5 Conclusion

In this paper, M-testability conditions for FFT butterfly and shuffle networks are presented, and a design-for-testability approach is developed and applied to the networks. Our M-testability conditions guarantee 100% single-module-fault testability with a constant number of test patterns, which results in a design-for-testability approach requiring very little hardware overhead. The number of test patterns needed for M-testing the FFT processors at the module level is however large for wordlengths of more than eight bits. In this case, M-testing the FFT processor at the bit level is more appropriate. Although the fault model adopted here is the single-module fault, our M-testability condition can be applied to lower-level fault models, such as delay faults and sequential faults [16]. Moreover, built-in self-test structures can easily be designed and applied to the module-level arrays [15], which can be tested at the system clock rate.

6 References

1 JOHNSON, B.W.: 'Design and analysis of fault tolerant digital systems' (Addison Wesley, 1989)
2 LARA, P.K.: 'Fault tolerant & fault testable hardware design' (Prentice-Hall, New York, 1987)
3 LAHA, S., and PATEL, J.H.: 'Error correction in arithmetic operations using time redundancy'. Proceedings of 13th Annual International Symposium on Fault-Tolerant Computing, June 1983, pp. 298–305
4 SWARTZLANDER, E.E. Jr., *et al.*: 'Sign/logarithm arithmetic for FFT implementation', *IEEE Trans.*, 1983, **C-32**, pp. 526–534
5 TAO, D.L., HARTMANN, C.R.P., and CHEN, Y.S.: 'A novel concurrent error detection scheme for FFT networks'. Proceedings of International Symposium on Fault-tolerant computing, June 1990, pp. 114–121
6 JOU, J.Y., and ABRAHAM, J.A.: 'Fault-tolerant FFT networks', *IEEE Trans.*, 1988, **C-37**, (5), pp. 548–561
7 TSUNOYAMA, M., and NAITO, S.: 'A fault-tolerant FFT processor'. Proceedings of International Symposium on Fault-tolerant computing, June 1991, pp. 128–135
8 CHOI, Y.H., and MALEK, M.: 'A fault-tolerant FFT processor', *IEEE Trans.*, 1988, **C-37**, (5), pp. 617–621

9 LOMBARDI, F., and MUZIO, J.: 'Concurrent error detection in reconfigurable WSI structures in FFT computation'. Proceedings of International Conference on Wafer scale integration, 1991, pp. 46–53
10 WU, C.-W., and CAPPELLO, P.R.: 'Easily testable iterative logic arrays', *IEEE Trans.*, 1990, **C-31,** (6), pp. 640–652
11 FUJIWARA, H., and TOIDA, S.: 'The complexity of fault detection problems for combinatorial logic circuits', *IEEE Trans.*, 1982, **C-31,** (6), pp. 555–560
12 WANG, J.-C., and WU, C.-W.: 'A bit-level systolic block FIR filter chip with built-in self-test structure'. Proceedings of International Symposium on Communications, Tainan, Taiwan, December 1991, pp. 660–663
13 WU, C.-W., and LU, S.-K.: 'Designing self-testable cellular arrays'. Proceedings of IEEE International Conference on Computer design (ICCD), Boston, USA, October 1991, pp. 110–113
14 WU, C.-W.: 'Test generation for combinatorial iterative logic arrays'. Proceedings of 3rd International Symposium on IC design and manufacturing, Singapore, September 1989, pp. 223–230
15 WU, C.-W., LU, S.-K., and WANG, J.-C.: 'Built-in self-test iterative logic arrays'. Proceedings of International Symposium on Electronics devices and material (EDMS), Hsinchu, Taiwan, November 1990, pp. 485–488
16 SU, C.-Y., and WU, C.-W.: 'Testing iterative logic arrays for sequential faults with a constant number of patterns', *IEEE Trans. Computers* (to be published)