Design of Easily Testable VLSI Arrays for Discrete Cosine Transform

Shyue-Kung Lu[‡], *Cheng-Wen Wu*[†], and *Sy-Yen Kuo*[‡]

[†]Department of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan

[‡]Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan

ABSTRACT

A *design-for-testability* approach based on the *M-testability* conditions is applied to the bit-level VLSI systolic arrays for discrete cosine transform (DCT). Our M-testability conditions guarantee 100% single-cell-fault testability with a *minimum* number of test patterns. A hardware overhead of no more than 6% is sufficient to make the DCT arrays M-testable. The resulting number of test patterns is only 16, regardless of the size of the DCT array and the internal word length. Apart from the cell-fault model, we also discuss the DCT array testing using the module-fault model. This method detects all possible combinational module faults pseudoexhaustively. Since practical DCT arrays can be quite large, diagnosis for the array is considered important. We propose an off-line fault diagnosis scheme which detects and locates any faulty module by a self-comparison approach.

Keywords: testing, design for testability, discrete cosine transform, systolic array, iterative logic array.

I. INTRODUCTION

Discrete cosine transform (DCT) has been used in a wide variety of digital signal processing applications for many years. To perform DCT efficiently, high-speed computation algorithms and/or architectures have been investigated by many researchers in the past (see, e.g., [1,2,3,4]). DCT computation requires massive and complicated data manipulations that lead to consideration and implementations employing parallelism and pipelining. To perform DCT in real time, such high-speed architecture is often required. The rapid advance in VLSI technology makes high-speed, large-scale parallel architectures possible, with systolic arrays among the most widely accepted because of their pipelinability, regularity, locality, and scalability [3,5]. However, the side effect for integrating a very large number of cells onto a single chip is that the complexity, coupled with an increase in the ratio of logic to pins, drastically reduces the controllability and observability of the logic on the chip. As a result, testing of such high-complexity and high-density circuits becomes very difficult and expensive.

In this paper, we propose a novel approach for designing M-testable [6] bit-level systolic DCT arrays. An array is M-testable if it can be tested with a test set equivalent in size to a *minimal* exhaustive test set of a single cell. This approach guarantees 100% single-cell-fault testability. Although the conditions are stronger than those of C-testability (testability with a *constant* number of patterns), its corresponding built-in-self-test (BIST) structures require smaller hardware overhead as compared to those based on pI-testability [7]: it has a much simpler and more regular test generator, and an equally compact response verifier with simpler routing. The M-testability conditions are briefly reviewed. Design procedure for M-testable DCT arrays is then presented. We apply the technique to the whole bit-level systolic DCT array, and the resulting number of test patterns for the M-testable array is only 16 (regardless of size of the DCT array and the internal word length). We show that a hardware overhead of no more than 6% is sufficient to make them M-testable.

Apart from the cell-fault model, we also discuss the DCT array testing using the module-fault model. Our approach easily applies to the DCT array at the module level due to the fact that its module function is an *x-bijection*. This method detects all possible combinational module faults pseudoexhaustively.

Since practical DCT arrays can be quite large, diagnosis for the array is considered important. We propose an off-line fault diagnosis scheme which detects and locates any faulty module by a self-comparison approach.

II. DESIGN OF M-TESTABLE DCT ARRAYS

A *cell* is a combinational machine (Σ, Δ, f), where $f: \Sigma \to \Delta$. In this paper, $\Sigma = \{0,1\}^I$ and $\Delta = \{0,1\}^O$ for $I, O \in \mathbf{N}$. An *iterative logic array* (ILA) is an array of cells. We use the terms array and ILA interchangeably. An ILA is *unilateral* when signals propagate in only one sense with respect to each axis. It is *bilateral* when signals propagate in both directions with respect to some axis. An ILA is *homogeneous* when it consists of functionally identical cells, otherwise it is *heterogeneous*. Unless otherwise specified, arrays are assumed to be unilateral and homo-

geneous. For homogeneous arrays, $\Sigma = \Delta$, so $I = O = w$, which denotes the input or output word length. We therefore may say that an array is 2^w-testable when it is M-testable. A *complete* or input sequence σ for a cell is an (exhaustive) input sequence consisting of all possible input combinations for the cell, i.e., $\sigma = \sigma_1\sigma_2 \cdots \sigma_k$, where $\forall\, \sigma_i \in \Sigma$, $i \in \{1, 2, \cdots, k\}$. A complete output sequence $\delta = \delta_1\delta_2 \cdots \delta_k$ is defined analogously. A *minimal* complete sequence is a shortest such sequence. The *x-projection* of f, $f^x: \Sigma^x \times \Sigma^y \rightarrow \Delta^x$, is defined to be $f^x(i, j) = \hat{i}$, where $f(i, j) = (\hat{i}, \hat{j})$. The *y-projection* of f is defined analogously. An *x-fault* exists when $\exists (i, j) \in \Sigma$ such that $f^x(i, j) = \hat{i} / i$ where i is the intended horizontal output but $\hat{i} \neq i$ is the actual horizontal output. A *y-fault* is defined analogously. A cell is *faulty* when it has some x-fault or y-fault. We say that f is *x-injective* when $\forall\, j$, $\forall\, i_1 \neq i_2$, $f(i_1, j) \neq f(i_2, j)$; we say f^x is *x-bijective* if it is *x-injective* and $\Sigma^x = \Delta^x$ (note that if $\Delta^y = \emptyset$ then we also may say f is *x-bijective*). *Y-injective* and *y-bijective* are defined analogously.

We assume that the cell's behavior is invariant over time. That is, we are testing for permanent combinational faults only. Sequential fault testing is discussed in our other paper [8]. The fault model adopted is the *single cell fault model* [9], which has been used as the cell-level fault model by most researchers dealing with ILA testing.

THEOREM 1 [9]: A k-dimensional ILA is M-testable if it has a bijective cell function, where k is an arbitrary positive integer.

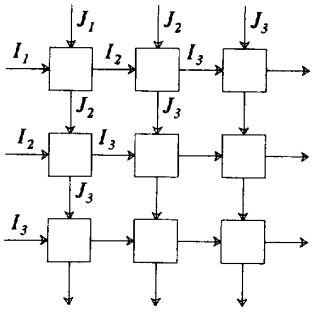


Fig. II-1: A 2-dimensional +45° tessellation.

A 2-D example is shown in Fig. II-1. Let w denote the input word length of a cell. Then M-testability stands for 2^w-testability, i.e., the whole array can be pseudoexhaustively verified with only 2^w tests regardless of how many cells there are in the array. We may also denote that $M = 2^w$. Let $\sigma = \sigma_1\, \sigma_2 \cdots \sigma_M$ be a minimal complete input sequence. Then any permutation of σ also is a minimal complete input sequence. In Fig. II-1, we see that all cells whose indices sum to the same number, i.e., those lie in the same 45° line, receive the same complete input sequence (I_i, J_i). Since the cell function is a bijection, any fault is propagated to some observable primary output concurrently. A fault results in an input change to the cell's horizontal and/or vertical neighboring cell. Bijectivity ensures that the change continues to ripple to some external output; the propagating paths cannot mask each other.

We now turn to the DCT array. Let the input data sequence be $\{X(n), n = 0, 1, \cdots, N-1\}$, and the N-point DCT of $X(n)$ be $\{Y(k), k = 0, 1, \cdots, N-1\}$. Then the DCT is given by the following relation:

$$Y(k) = c(k) \sum_{n=0}^{N-1} X(n) \cos[(2n+1)k\pi/2N],$$

where

$$c(k) = \begin{cases} 1/\sqrt{2} & k = 0, \\ 1, & \textit{otherwise}. \end{cases}$$

For ease of presentation, we neglect the scale factor $1/N$. A 5-point DCT architecture is shown in Fig. II-2 [3], where $U_N = e^{j\pi/N}$. It is evident that the number of processing elements for N-point DCT is $N + 1$, and the data sequence can be entered in its natural order.

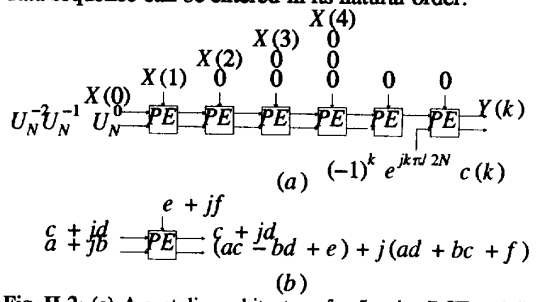


Fig. II-2: (a) A systolic architecture for 5-point DCT and (b) the cell function.

We can realize each processing element with four inner product processors (IPPs) as shown in Fig. II-3. Fig. II-3(a) illustrates the detailed implementation of a processing element, in which four inner product processors are contained. Each IPP performs the function as shown in Fig. II-3(b).

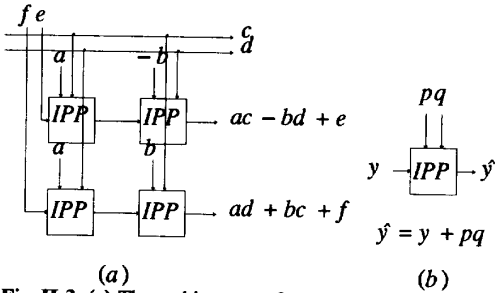


Fig. II-3: (a) The architecture of a processing element and (b) the function of an IPP module.

Rather than using word-level cells, we may extend the benefit of the systolic paradigm to the bit level, using bit level cells. The IPP is actually a 2's-complement array multiplier [10, 11]. The DCT array therefore can be viewed as the combination of several array multipliers. For the purpose of testing, we consider our DCT array as a large array of bit-level cells. A

(3-bit)×(3-bit) multiplier is shown in Fig. II-4(a), where the cells marked D are dummy cells (containing only pipeline latches). Each cell performs the 1-bit logic function

$$\hat{a} = a$$
$$\hat{b} = b$$
$$\hat{s} = s \oplus c \oplus a \cdot b$$
$$\hat{c} = s \cdot c + c \cdot a \cdot b + s \cdot a \cdot b$$

where a and b represent multiplier and multiplicand bits, s is the summand bit, and c is the carry bit. Note that the latches are not shown. From the truth table we see that the cell function is not bijective.

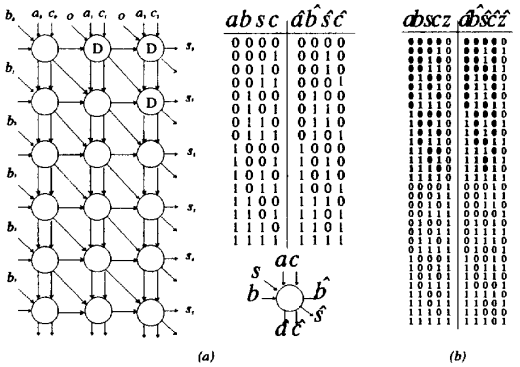


Fig. II-4: (a) Bit-parallel pipelined array multiplier and (b) the truth table after function modification.

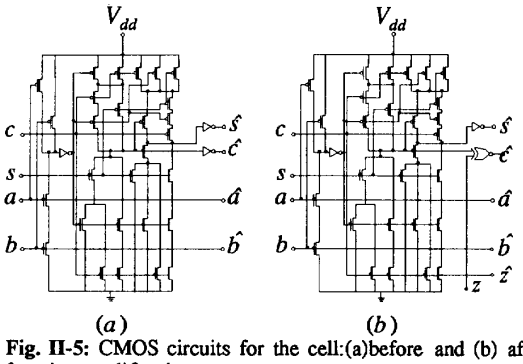


Fig. II-5: CMOS circuits for the cell:(a)before and (b) after function modification.

For example, $(a, b, s, c) = (0, 0, 0, 1)$ and $(0, 0, 1, 0)$ map to the same output $(0, 0, 1, 0)$. In order to meet the condition of Thm. 1, we need to modify the cell function to make it bijective. Our modification results in a circuit that has two modes: *normal operation mode* and *test mode*. Since no 3 different inputs map to the same output in the original function, we add an extra input and an extra output to each cell, and assign output values such that the new cell function is bijective. In test mode, the added wires are treated as ordinary I/O wires, and the augmented truth-table is used. In normal operation mode, the added input wires are grounded and the added output wires are neglected, i.e., the original truth-table is used.

A novel algorithm for the assignment of the augmented truth-table is proposed in our previous work [6]. This method makes the augmented truth-table bijective with minimal overhead. The heuristic algorithm for our output assignment is based on a simple goal—*reflection* or *complementary reflection* (defined below). If α is a feasible assignment of values for the entries of the augmented truth table, then α_v ($\alpha_{\hat{v}}$ resp.) denotes α confined to the input (output resp.) variable v ($\hat{v}$ resp.), i.e., α_v ($\alpha_{\hat{v}}$ resp.) assigns only the values for v ($\hat{v}$ resp.). The assignment α confined to the upper half of the table is denoted α_0; the assignment confined to the lower half is α_1. An assignment α_v for an input or output variable v is said to be *reflexive* if in the augmented truth table, the values assigned by $_1\alpha_v$ are one-to-one corresponding to those assigned by $_0\alpha_v$. It is *complementarily reflexive* or simply *complementary* if the assigned values by $_0\alpha_v$ are one-to-one corresponding to the complement of those assigned by $_1\alpha_v$. We apply our algorithm to the original truth table, and generate the bijective augmented truth table as shown in Fig. II-4(b). The new cell functions are:

$$\hat{a} = a$$
$$\hat{b} = b$$
$$\hat{s} = s \oplus c \oplus a \cdot b$$
$$\hat{c} = (s \cdot c + c \cdot a \cdot b + s \cdot a \cdot b) \cdot \bar{z} + \overline{(s \cdot c + c \cdot a \cdot b + s \cdot a \cdot b)} \cdot z$$
$$\hat{z} = c \quad (\text{or } \hat{z} = s)$$

The function modification is based on our truth-table augmentation algorithm, where $\alpha_{\hat{a}}$, $\alpha_{\hat{b}}$, $\alpha_{\hat{s}}$ are reflexive assignments, and $\alpha_{\hat{c}}$ is complementary. The extra output $\hat{z}$ can be assigned as that for s or c.

We may use, for example, the CMOS implementation as shown in Fig. II-5. Fig. II-5(a) shows the original cell with 34 transistors. Fig. II-5(b) is the modified cell which has 36 transistors—an overhead of 6%. This low hardware overhead is based on our reflexive and complementary assignment of the augmented truth table. Another important factor is the design of the simple XNOR circuit as shown in Fig. II-6, where only 4 transistors are required.

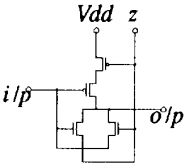


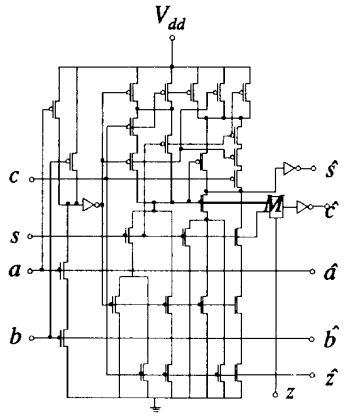
Fig. II-6: CMOS circuit for the XNOR gate.

If we consider our DCT array as a large array of bit-level cells, and modify the cell function as presented, then the whole array is M-testable according to Thm. 1. The resulting number of test patterns is only 32, regardless of the size of the array.

There is yet another approach to make the cell function of the array multiplier bijective, which requires no extra input and output bits to be added to the basic cell. The approach is illustrated in Fig. II-7. The modified cell function is shown in Fig. II-7(a). The

resulting number of test patterns is only 16, i.e., half that of the previous method.

$absc$	$\hat{a}\hat{b}\hat{s}\hat{c}$
0 0 0 0	0 0 0 0
0 0 0 1	0 0 1 0
0 0 1 0	0 0 1 1
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 1
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 1
1 0 1 1	1 0 0 1
1 1 0 0	1 1 1 0
1 1 0 1	1 1 0 0
1 1 1 0	1 1 0 1
1 1 1 1	1 1 1 1



(a) (b)

Fig. II-7: Modified cell function and the CMOS circuit.

We may use the CMOS implementation as shown in Fig. II-7(b), where a multiplexer (marked M) is introduced. This multiplexer is controlled by the mode selection signal that indicates whether the ILA is in test or normal mode. If it is in test mode, the multiplexer takes input from s, and thus $\hat{c} = s$. The cell function in test mode clearly is bijective, so it is M-testable according to Thm. 1. When it is in normal operation, the multiplexer takes input from its normal path.

The highlighted line segment in Fig. II-7(b) can not be tested in test mode. In order to increase our fault coverage, we generate 1 and 0, respectively, to this line and observe its output response. From the truth table in Fig. II-7(a), this can be done by applying (0, 0, 0, 0) and (1, 1, 1, 1), respectively, to the cell's primary inputs. Fig. II-8 shows the bit-level array of one processing element for computing DCT, with word length 3. It is assume that the coefficients can be preloaded into the cells. The cells used in Fig. II-8(b) are M-testable cells modified by the approach described above. In test mode, we consider the DCT array as a large bit-level systolic array, which is also M-testable.

III. Testing at the Module Level

Theorem 2: The ILA as shown in Fig. III-1 is M-testable if its cell function f is x-bijective.

Proof: Let $\sigma_1 = (I_1, J_1, J_2)$ be a minimal complete input sequence for a single cell. We apply σ_1 to $cell_{00}$ and $cell_{10}$ simultaneously. Let the resulting horizontal output sequences of the fault free cells $cell_{00}$ and $cell_{10}$ be both I_2. Since f is *x–bijective* and $\Sigma^* = \Delta^*$, the sequence $\sigma_2 = (I_2, J_1, J_2)$ is also a minimal complete input sequence for both $cell_{01}$ and $cell_{11}$. Reiterating this process, each cell in this array receives a minimal complete input sequence. If, say, $cell_{00}$ ($cell_{10}$ resp.) is faulty, then since f is *x–bijective*, the fault will be propagated horizontally through $cell_{01}$ ($cell_{11}$ resp.). Inductively, the fault will be observable at the primary output terminals. Since σ_1 is a minimal complete input sequence, $|\sigma_1| = 2^w$. The entire array therefore is M-testable. ❑

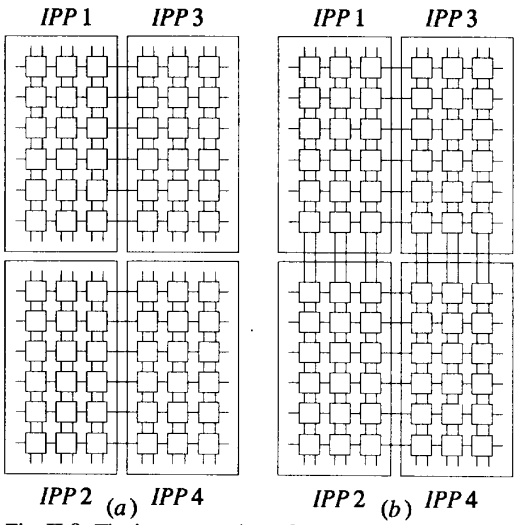


Fig. II-8: The interconnection of a processing element in (a) normal mode and (b) test mode.

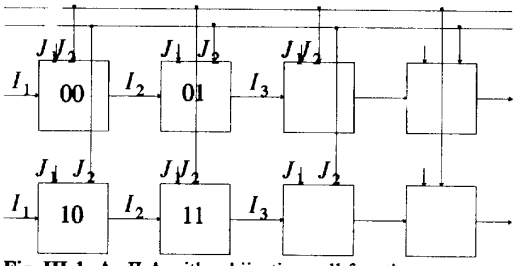


Fig. III-1: An ILA with x-bijective cell function.

A 2-point DCT array is shown in Fig. III-2, where each square box stands for an IPP. We will show that this architecture can be made M-testable by including a few multiplexers to reroute the interconnection wires during test mode.

Corollary: An N-point DCT array can be made M-testable by including $2N$ multiplexers, where M is the length of a minimal complete input sequence for an IPP.

Proof: Since the IPP has an x-bijective cell function (see Fig. II-3(b)), and the architecture of the DCT circuit (Fig. III-2) can be transformed (during test mode) into the array of the form as shown in Fig. III-1 by using multiplexers to redirect the signal wires (see Fig. III-3), the conclusion follows directly from Thm. III-1. ❑

A modified 2-point DCT array as shown in Fig. III-3 requires 12 IPPs and 4 multiplexers. Let w denote the word length, then the entire array consists of $12 \cdot 2w^2$ bit-level cells. A bit-level cell consists of 64 transistors (including pipelined latches), so the total number of transistors used in the DCT array is $64 \cdot 12 \cdot 2w^2 = 1636w^2$. Using a 6-transistor multiplexer, the number of added transistors is $6 \cdot 4 \cdot w = 24w$. The

overhead is about $24w/(1636w^2) = 1/64w$. For a reasonable word length (e.g., $w = 12$), the overhead is negligible. Built-in self-test (BIST) techniques can easily be applied to the module-level array in which the DCT array can be tested at the system clock rate.

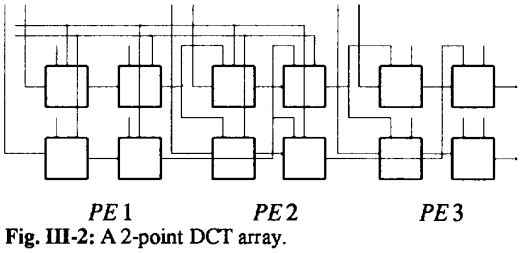


Fig. III-2: A 2-point DCT array.

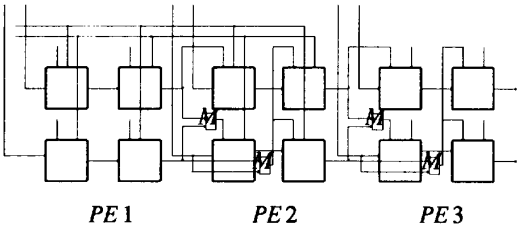


Fig. III-3: The modified 2-point DCT array.

Since practical DCT arrays can be quite large, requiring a multi-chip implementation, diagnosis for the array is considered important. We propose an off-line fault diagnosis scheme which detects and locates any faulty module by a self-comparison approach. In Fig. III-3, we divide the entire DCT array into two independent liner array in test mode. Both of the modules at the same column receive the same input sequence. If they are fault free, their corresponding output sequences are the same. Under the single module fault assumption, if a fault exists, the output sequences of the two modules are different. A comparator can be used to detect the fault (see Fig. III-4). Using this method, each processing element consists of 4 IPPs and 2 comparator. It is assumed that the XOR gates in the comparators are self-checking, thus for practical purposes they can be considered always fault free. They also account for negligible area in the chip layout.

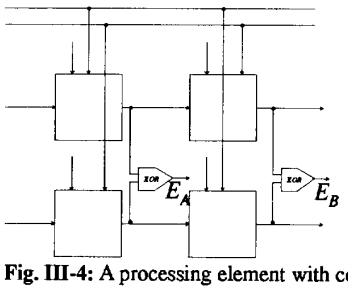


Fig. III-4: A processing element with comparators.

IV. CONCLUSIONS

A *design-for-testability* approach based on the *M-testability* conditions is applied to the bit-level VLSI systolic arrays for discrete cosine transform (DCT), which guarantee 100% single-cell-fault testability with a *minimum* number of test patterns. A hardware overhead of no more than 6% is sufficient to make the DCT arrays M-testable. The resulting number of test patterns is only 16, regardless of the size of the DCT array and the internal word length. DCT array testing using the module-fault model also is discussed. M-testable arrays are proposed. Finally, an off-line fault diagnosis scheme which detects and locates any faulty module in the DCT array by self-comparison is presented.

References

1. N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Commun*, vol. COM-23, pp. 90-93, Jan. 1974.
2. W.Kou and Jon W. Mark, "A New Look at DCT-Type Transforms," *IEEE Trans. Acoustic, Apeech, and Signal Processing* , vol. 37, no. 12, pp. 1899-1908, Dec. 1989.
3. N. I. Cho and S. U. Lee, "DCT Algorithms for VLSI Parallel Implementations," *IEEE Trans. ASSP*, vol. 38, pp. 121-127, Jan. 1990.
4. W. Li, "A New Algorithm to Compute the DCT and its Inverse," *IEEE Trans. Signal Processing*, vol. 39, no. 6, pp. 1305-1313, Jun. 1991.
5. L. W. Chang and M. C. Wu, "A Unified Systolic Array for Discrete Cosine and Sine Transforms," *IEEE Trans. Signal Processing*, vol. 39, no. 1, pp. 192-194, Jan. 1991.
6. C. W. Wu and S. K. Lu, "Designing Self-Testable Cellular Arrays," *Proc. ICCD 91*, pp. 110-113, Oct. 1991.
7. E. M. Aboulhamid and E. Cerny, "Built-in testing of one-dimensional unilateral iterative arrays," *IEEE Trans. Computers*, vol. C-33, no. 6, pp. 560-564, June 1984.
8. Su, C.-Y and C.-W. Wu, "Testing iterative logic arrays for sequential faults with a constant number of patterns," *Submitted IEEE Trans. Computers*, June. 1991.
9. C.-W. Wu and P. R. Cappello, "Easily testable iterative logic arrays," *IEEE Trans. Computers*, vol. C-39, no. 5, pp. 640-652, May 1990.
10. J. V. McCanny and J. G. McWhirter, "Completely iterative, pipelined multiplier array suitable for VLSI," *IEE Proc.*, vol. 129, no. 2, pp. 40-46, Apr. 1982.
11. P. R. Cappello and C.-W. Wu, "Computer-aided design of VLSI FIR filters," *Proc. of the IEEE*, vol. 75, no. 9, pp. 1260-1271, Sep. 1987.