FFT Arrays with Built-in Error Correction

Yuang-Ming Hsu and Earl E. Swartzlander, Jr.

Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712, USA

Abstract

Fast Fourier Transform (FFT) arrays with built-in error correction are proposed in this paper. A time shared TMR scheme is used to achieve the error correcting capability. A quarter of the original FFT array is triplicated and voted in each stage. Therefore the hardware complexity of the error correcting FFT array is a little more than 75% of the original FFT array. This is significant since the error correcting design is smaller than the original. The price for this hardware reduction is that the delay time increases by a factor of 4. However, the throughput penalty can be minimized by pipelining. A technology-independent gate-level analysis of hardware complexity and delay time is included in this paper.

1. Introduction

The Fast Fourier Transform (FFT) is an essential tool widely used in digital signal processing [1]. The reliability of a digital system can be improved by introducing redundancy into the system [2]. The redundancy can be in the form of extra hardware, computation time, or both. In this paper, time redundancy is used to improve the reliability of FFT arrays. The error correcting FFT arrays have lower hardware complexity than the original non-redundant FFT arrays. The delay penalty is modest and decreases as the length of the transform increases.

An 8-point FFT array is depicted in Figure 1. There are three stages in the array. Each stage has 4 processing elements (PEs) which perform the radix-2 *butterfly* operation. The circuit is fully combinational. An output is produced in 3 *stage delay times*. The stage delay time is the computation time of a radix-2 butterfly operation (i.e., time for one complex multiplication and one complex addition). The hardware complexity of the array is 12 radix-2 butterfly processing elements, each containing one complex multiplication and two complex additions. For an FFT with m-bit data, both the complex multiplication and the complex additions are m-bits wide.

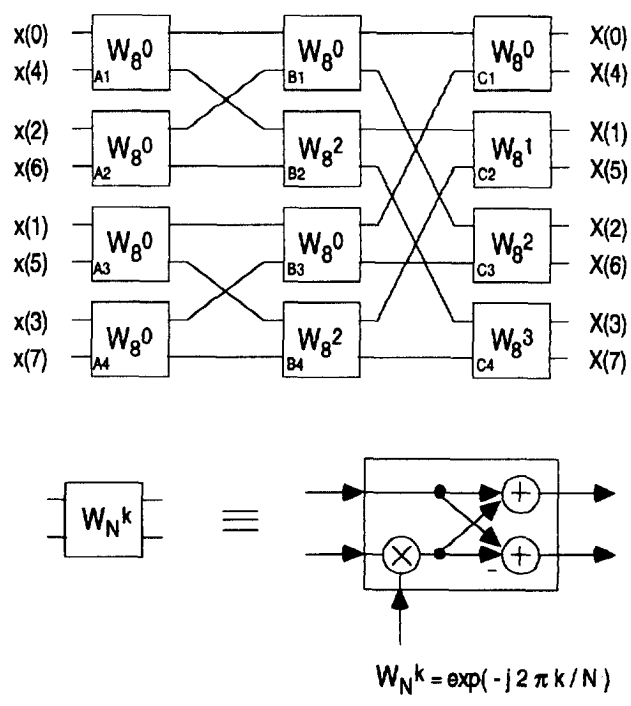


Figure 1. An 8-point FFT array.

There are FFT processors with error detecting capability [3]-[5]. Several fault-tolerant FFT processors have also been proposed [6]-[10]. They usually can tolerate only a single fault in the FFT array. Their hardware complexity is higher than the original and the delay penalty can be substantial when totally self-checking components are used. In this paper, an architecture of error correcting FFT arrays with lower hardware complexity is proposed. The design is based on time shared TMR. Any error that would produce an incorrect result in a processing element can be corrected (or masked). This covers a large class of faults, which result in errors. Multiple errors in different FFT stages can also be corrected. The delay penalty can be reduced by pipelining.

1058-6393/95 $4.00 © 1995 IEEE

Using time redundancy to achieve error correction was discovered independently almost simultaneously in [11], [12] and [13]. Apparently all were inspired by Johnson's REcomputation with Duplication With Comparison (REDWC) scheme for error detection [14]. With the time shared TMR concept, a quarter of the original FFT array is triplicated and a majority vote is taken at each PE's output. The voted result is then sent to the next stage, which is also triplicated and voted. Four computations are required to complete the whole FFT operation. However, the PEs can be pipelined to produce reasonable efficiency.

The paper is organized as follows. An 8-point error correcting FFT array is described first to demonstrate the concept. The result can be easily extended to larger transforms. Next, the hardware complexity and delay time are analyzed for the general N-point error correcting FFT arrays. Some variations of the concept are discussed in Section 4. Finally, a conclusion summarizes the results.

2. An 8-point error-correcting FFT array

In this section, an 8-point time shared TMR error correcting FFT array is presented to demonstrate the concept. The discussion can be easily extended to larger transforms.

The FFT array in Figure 1 can be partitioned into 4 horizontal slices, each having 3 processing elements (A_i, B_i, and C_i, i = 1, 2, 3, 4). With proper storage elements inserted, the 8-point FFT can be computed using one of the four slices (e.g., the slice composed of A_1, B_1, and C_1) by repeating the calculation 4 times:

Step 1: x(0) and x(4) are input to A_1. The *twiddle factor* W_N^k is set to W_8^0. Outputs are stored in registers.

Step 2: x(2) and x(6) are input to the same PE A_1. Outputs are stored in registers.

Step 3: x(1) and x(5) are input to A_1. Outputs are also stored in registers.

Step 4: x(3) and x(7) are input to A_1. Outputs are again stored in registers.

After the fourth step, the outputs of A_1 in Step 1 and Step 2 are fed to the processing element B_1. Its twiddle factor is set to W_8^0. The outputs are also stored. The operation of B_1 is also repeated 4 times. However, the twiddle factor now alternates between W_8^0 and W_8^2. Finally, the third stage operation is repeated 4 times in C_1, with twiddle factors W_8^0, W_8^1, $W8^2$, and W_8^3, respectively. This process is illustrated in the signal flow diagram of Figure 2, where the processing elements (A_1, B_1, and C_1) are denoted by A, B, and C.

Furthermore, the PEs can be pipelined as in Figure 3. Here, the initial delay of the output is 9 time units instead of the 12 in Figure 2. After the initial delay, an output is obtained every 4 time units. Note that the original non-redundant (and pure combinational) FFT array produces an output every 3 time units.

The above is the basic method of partitioning the FFT array. Next, each processing element in the slice is triplicated and a majority vote is taken at the output. In this way, any errors within a processing element can be corrected (or masked) by the voter. Since voting is used in every stage, multiple errors in different stages can also be corrected. The block diagram of the time redundant 8-point error correcting FFT array is shown in Figure 4. There are three triplicated and voted processing elements (A, B, and C). The inputs to the processing elements are selected by 4-to-1 and 2-to-1 multiplexers. The control signals of these multiplexers are also given in the figure. As can be seen, the depths of the registers needed are different at each node. The maximum depth is 5. Since the multiplexers, voters, and storage elements are relatively small compared to the processing elements, they are assumed to be fault-free.

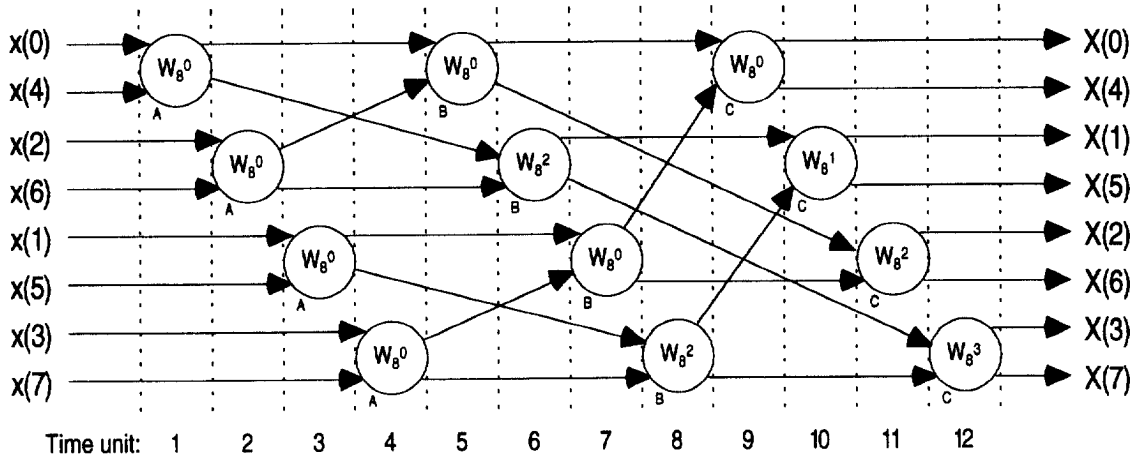


Figure 2. Signal flow diagram of the 8-point FFT array in Figure 1.

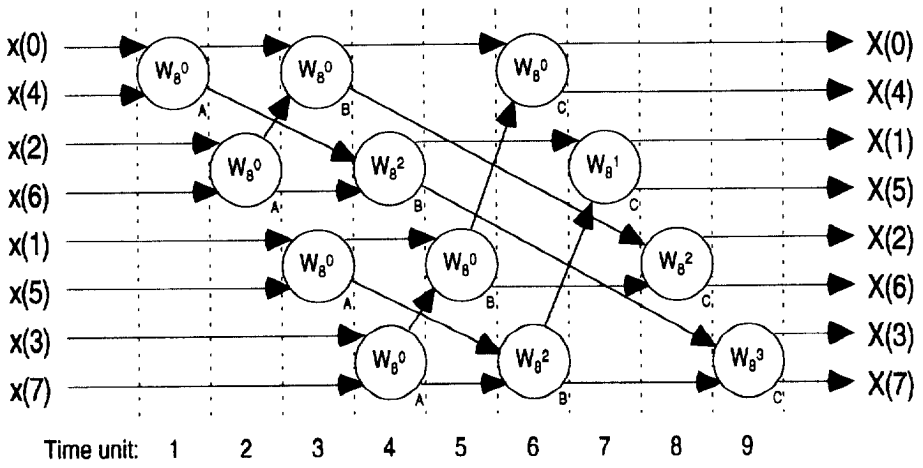
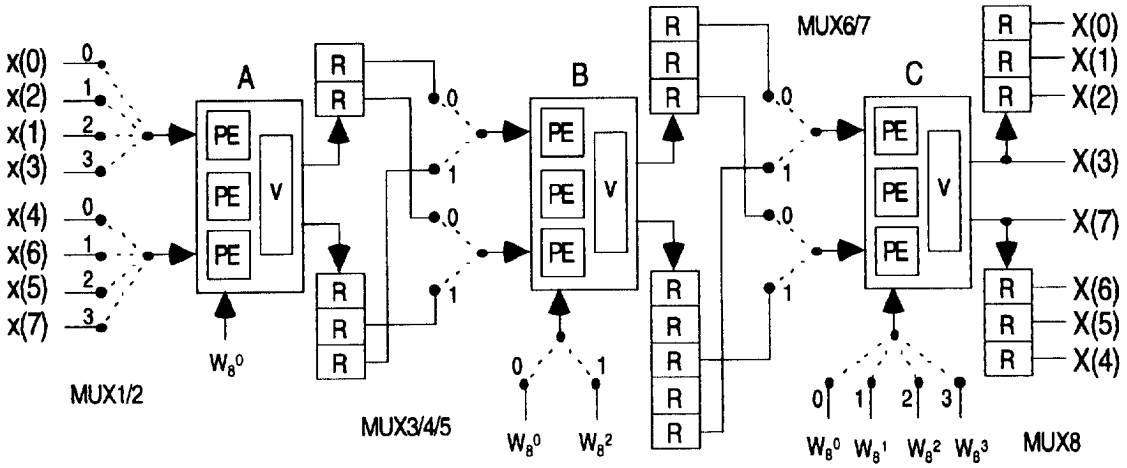


Figure 3. Signal flow diagram of the 8-point FFT array (pipelined).



Time	1	2	3	4	5	6	7	8	9
MUX1/2	0	1	2	3					
MUX3/4/5			0	1	0	1			
MUX6/7						0	0	1	1
MUX8						0	1	2	3

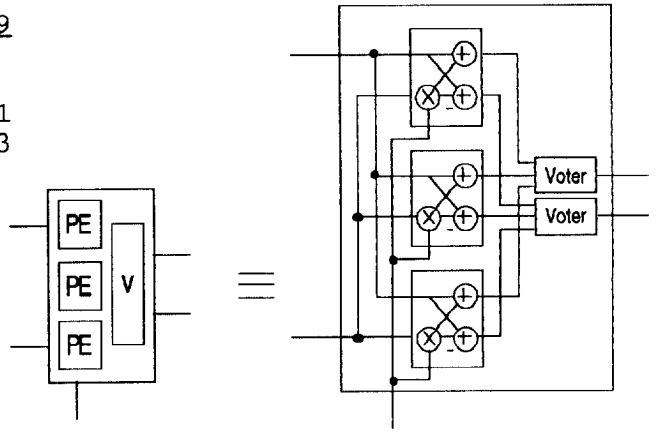


Figure 4. Block diagram of the error correcting 8-point FFT array.

3. Complexity

In this section, the hardware complexity and delay penalty of the proposed time shared TMR error correcting FFT arrays are analyzed.

3.1 Hardware complexity

The hardware complexity is analyzed based on a technology-independent approach, first at the function-level and then at the gate-level. The data of the N-point FFT are assumed to be complex pairs of m-bit wide words.

In an N-point conventional FFT array, there are $\log_2(N)$ stages, each having N/2 processing elements (thus a total of $N/2*\log_2(N)$). Each PE has one complex multiplication and two complex additions. Note that a complex addition consists of two real additions. A complex multiplication is performed by four real multiplications and two real additions. Hence a PE has 6 adders and 4 multipliers. In the proposed time redundant error correcting N-point FFT arrays, a quarter of the PEs are triplicated, hence there are $3/4*N/2*\log_2(N) = 3N/8*\log_2(N)$ processing elements. The extra circuits in the error correcting design are voters, registers, and multiplexers. This function-level analysis is shown in Table 1.

Table 1. Comparison of hardware complexity of FFT arrays (function-level).

Transform length		N =	8	16	64	256	1024
Conventional:							
m-bit adder	$6*N/2*\log_2(N)$		72	192	1152	6144	30720
mxm mult.	$4*N/2*\log_2(N)$		48	128	768	4096	20480
Error correcting:							
m-bit adder	$18*N/8*\log_2(N)$		54	144	864	4608	23040
mxm mult.	$12*N/8*\log_2(N)$		36	96	576	3072	15360
m-bit voter	$4*N/8*\log_2(N)$		12	32	192	1024	5120
m-bit register	$2*N/8*(2*\log_2(N)+13)$		38	84	400	1856	8448
m-bit 4:1 mux	$6*N/8$		6	12	48	192	768
m-bit 2:1 mux	$10*N/8$		10	20	80	320	1280

The gates used in the gate-level analysis are inverters, 2- to 3-input AND, and OR gates. Six m-bit ripple carry adders (9 gates in each full adder) and four m-bit by m-bit array multipliers are used in the processing elements. It is assumed that: each register is composed of 6 gates; a voter has 4 gates; a 4-to-1 mux has 6 gates; and a 2-to-1 mux has 4 gates. These assumptions are relatively conservative since many inverters in the multiplexers can be shared. In an m-bit by m-bit array multiplier, there are m^2 AND gates for bit product generation, m half adders (4 gates each), and m(m-2) full adders. Therefore there are a total of 2m(5m-7) gates in an array multiplier. The ratio of the error correcting FFT array gate count to the conventional FFT array gate count for different data width (m) and point number (N) is listed in Table 2. As can be seen, the complexity of the time redundant error correcting FFT arrays is lower than that of the original FFT arrays. The ratio approaches 0.75 as the transform length N and data wordsize m increase. This justifies the fact that the multiplexers, voters, and storage elements take up a relatively small portion of the circuit and can be assumed to be fault-free.

Table 2. Gate count ratio of error correcting FFT arrays to the conventional FFT arrays.

Transform length	N =	8	16	64	256	1024
Data wordsize	m = 4	0.940	0.908	0.877	0.861	0.851
Data wordsize	m = 8	0.844	0.829	0.813	0.805	0.800
Data wordsize	m = 16	0.797	0.789	0.781	0.777	0.775
Data wordsize	m = 32	0.773	0.770	0.766	0.764	0.763

3.2 Delay

The price paid for the time shared TMR error correcting capability and reduced hardware complexity is the increased delay time in the FFT computation. The delay time (or the time from when the data applied at the input to the time the output is available) in terms of the stage delay time (called the time unit) is $\log_2(N)+6$. If the PEs are fully pipelined as described in the previous section, then on the average, one output is available every 4 time units. This number is constant for all transform lengths. The results are shown in Table 3.

Table 3. Comparison of delay time of FFT arrays.

Transform length	N =	8	16	64	256	1024
Delay (conventional)	$\log_2(N)$	3	4	6	8	10
Delay (error correcting)	$\log_2(N)+6$	9	10	12	14	16
Delay penalty	$6/\log_2(N)$	200%	150%	100%	75%	60%

4. Variations

There are several variations in the design of time shared TMR error correcting FFT arrays. In addition to Figure 1, another FFT array based on perfect shuffle is shown in Figure 5. In this network, every stage has the same interconnection. The advantage is that the design can be more modular. The disadvantage is the delay time is longer than that of the proposed design. This can be verified by a signal flow diagram similar to Figure 3 (for N=8, the delay is 10; for N=16, the delay is 13.) The number of storage elements required is also higher.

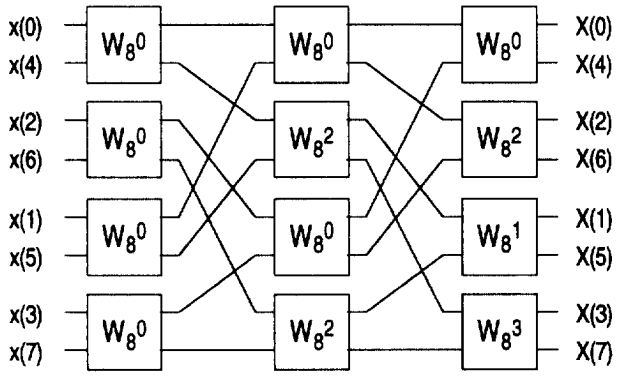


Figure 5. 8-point FFT array based on perfect shuffle.

Another variation is to triplicate half of the FFT array. This approach has the benefit of reduced delay penalty: only 67% for N=8 and 20% for N=1024. However, the hardware complexity is 50% more than the original FFT array.

5. Conclusions

In this paper, an approach to improve the reliability of FFT arrays is proposed. Time shared TMR is used to achieve error correction and reduced hardware complexity. A quarter of the original FFT array is triplicated and voted. Any errors in the processing elements can be corrected (or masked). The computation is performed in four cycles. The delay penalty can be reduced by pipelining. From the analysis, the hardware complexity is only about 80% of the original non-redundant FFT array and the delay penalty is less than 60% for pipelined versions of long transforms. The proposed approach is suitable for applications where high reliability and low hardware complexity are the primary concern.

References

[1] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice-Hall, 1975.

[2] B. W. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems*, Addison-Wesley, 1989.

[3] F. Lombardi and J. C. Muzio, "Concurrent error detection and fault location in an FFT architecture," *IEEE J. Solid-State Circuits*, Vol. 27, No. 5, pp. 728-736, 1992.

[4] D. L. Tao and C. R. P. Hartmann, "A novel concurrent error detection scheme for FFT networks," *IEEE Trans. Parallel and Distributed Systems*, Vol. 4, No. 2, pp. 198-221, 1993.

[5] T.-H. Chen and L.-G. Chen, "Concurrent error-detectable butterfly chip for real-time FFT processing through time redundancy," *IEEE J. Solid-State Circuits*, Vol. 28, No. 5, pp. 537-547, 1993.

[6] J.-Y. Jou and J. A. Abraham, "Fault-tolerant FFT networks," *IEEE Trans. Comput.* Vol. 37, No. 5, pp. 548-561, 1988.

[7] Y.-H. Choi and M. Malek, "A fault-tolerant FFT processor," *IEEE Trans. Comput.* Vol. 37, No. 5, pp. 617-621, 1988.

[8] M. Tsunoyama and S. Naito, "A fault-tolerant FFT processor," *Proc. 1991 IEEE Fault-Tolerant Computing Symp. (FTCS-21)*, pp. 128-135.

[9] A. Antola, et al., "Fault tolerance in FFT arrays: time redundancy approaches," *J. VLSI Signal Processing*, Vol. 4, No. 4, pp. 295-316, 1992.

[10] C. G. Oh and H. Y. Youn, "On concurrent error detection, location, and correction of FFT networks," *Proc. 1993 IEEE Fault-Tolerant Computing Symp. (FTCS-23)*, pp. 596-605.

[11] Y.-M. Hsu and E. E. Swartzlander, Jr., "Time redundant error correcting adders and multipliers," *Proc. IEEE Int'l Workshop on Defect and Fault Tolerance in VLSI Systems*, pp. 247-256, 1992.

[12] S. Al-Arian and M Gumussel, "HPTR: hardware partition in time redundancy technique for fault tolerance," *Proc. IEEE SOUTHEASTCON 1992*, Vol. 2, pp. 630-633.

[13] T. H. Chen, et al., "Design and analysis of VLSI-based arithmetic arrays with error correction," *Int'l J. Electronics*, Vol. 72, No. 2, pp. 253-271, 1992.

[14] B. W. Johnson, et al., "Efficient use of time and hardware redundancy for concurrent error detection in a 32-bit VLSI adder," *IEEE J. Solid-State Circuits*, Vol. 23, No. 1, pp. 208-215, 1988.