The Gauss Machine: A Galois-Enhanced Quadratic Residue Number System Systolic Array *

Jonathon D. Mellott Jeremy C. Smith Fred J. Taylor
High Speed Digital Architecture Laboratory
University of Florida

Abstract

The Gauss machine is a SIMD systolic array architecture which takes advantage of the Galois-enhanced residue number system (GEQRNS) to form reduced complexity arithmetic elements. The Gauss machine is targeted at front-end signal and image processing applications. A discrete prototype has been constructed which achieves a peak rating of 320 million arithmetic operations per second when performing complex arithmetic while operating at 10 MHz. A VLSI implementation of the Gauss machine's processor cell has been created. The VLSI implementation is implemented in 2.0 micron CMOS and achieves greater than 20 MHz performance and uses less than 2.0 mm^2 die area. It is also shown that techniques for defect tolerance in RNS systolic arrays can result in substantial yield enhancement, thereby making larger than conventional (ULSI) systems possible.

I. INTRODUCTION

There exist a number of signal processing applications which demand high computational throughput in combination with high reliability, small size, and low power dissipation. In the past, high performance has come at the expense of reliability, size, power, and cost requirements. The prevalent arithmetic system used in digital hardware is two's complement. While two's complement is easy to use, it suffers from several impediments to achieving high performance. The speed of the adder in a two's complement system decreases at least with the log of the word width of the adder due to the propagation of the carry term across the adder. The two's complement multiplier suffers not only from the "curse of carry," but it also suffers from geometric growth of the required die area as the word width of the operands increases [1]. Multiplier structures continue to occupy large die area on modern VLSI microprocessors. Since the RNS (Residue Number System) is a carry-free arithmetic system, word widths of arbitrary size may be produced with no speed penalty in the adder. The size of the multiplier also grows linearly with respect to the word width of the multiplicands, rather than geometrically as in two's complement schemes.

At the University of Florida High Speed Digital Architecture Laboratory we have developed a high performance multiprocessor architecture based upon the RNS. Our first RNS processor, the Gauss machine [2] is described. The Gauss machine is a hybrid systolic array and vector processor of GEQRNS processing elements which can achieve the peak equivalent of 320 million operations per second when performing complex arithmetic. Besides high performance, the RNS enables a high degree of fault-tolerance at the architectural level [3, 4]. The RNS is also uniquely positioned to realize the full promise of VLSI/ULSI systolic arrays [5, 6] and the high levels of integration offered by MCM (multi-chip module) packaging which may take advantage of the small size, modularity, fault and defect tolerance which RNS features.

II. THEORY

A. *The Chinese Remainder Theorem*

There are two large penalties in performing arithmetic in the two's complement system: the carry must propagate across the entire word for addition operations, and the size of the multiplier grows as the square of the width of the word. The Chinese

*This work supported by Naval Air Warfare Center, Army Research Office, and Florida High Technology Industry Council

1063-6889/93 $03.00 © 1993 IEEE

Remainder Theorem (CRT)[7, 8] suggests a means of eliminating the carry propagation problem and of producing a multiplier that grows linearly with the width of the word. The CRT is presented below.

Theorem 1 (The Chinese Remainder Theorem) *Let* $M = \prod_{i=1}^{L} p_i$, *where for* $i, j \in \{1, 2, 3, \ldots, L\}$, $\gcd(p_i, p_j) = 1$ *for all* $i \neq j$, *and each* $p_i \in \mathbf{Z}^+$. *Then there exists an isomorphism* ϕ: $\mathbf{Z}_M \leftrightarrow \mathbf{Z}_{p_1} \times \mathbf{Z}_{p_2} \times \mathbf{Z}_{p_3} \times \cdots \times \mathbf{Z}_{p_L}$ *described by the following.*

Let $m_i = M/p_i$, *and* $m_i m_i^{-1} \equiv 1 \pmod{p_i}$ *for all* $i \in \{1, 2, 3, \ldots, L\}$. *If* $X \in \mathbf{Z}_M$, *let* $\phi(X) = (x_1, x_2, x_3, \ldots, x_L)$ *where* $x_i \equiv X \pmod{p_i}$ *for all* $i \in \{1, 2, 3, \ldots, L\}$ *then* $X = \phi^{-1}(x_1, x_2, x_3, \ldots, x_L)$ *is described by the following congruence*

$$X \equiv \left\{ \sum_{i=1}^{L} m_i \langle m_i^{-1} x_i \rangle_{p_i} \right\} \pmod{M}$$

where $\langle \bullet \rangle_p$ *indicates the unary (mod p) operation.*

The CRT forms the basis for the RNS. In the RNS, two's complement integers are converted to their L-tuple residue representation by the ring isomorphism ϕ: $\mathbf{Z}_M \leftrightarrow \mathbf{Z}_{p_1} \times \mathbf{Z}_{p_2} \times \mathbf{Z}_{p_3} \times \cdots \times \mathbf{Z}_{p_L}$ described by the CRT. The numbers which are in their L-tuple representation may be added and multiplied componentwise and reconstructed via the CRT to form the correct result in $\mathbf{Z}_M$.

Generally, the moduli are chosen to be small enough that the adders and multipliers may be implemented in a reasonably small memory-based lookup table. In a VLSI implementation advanced memory technology might be leveraged and thereby achieve greater speed and smaller die area.

B. *Complex Residue Number System (CRNS)*

The RNS may be used to perform computations with complex numbers by using RNS arithmetic elements to emulate the operations which would be performed using two's complement hardware. The use of RNS arithmetic to perform complex operations is called complex RNS or CRNS. Take the Gaussian integers $a + jb, c + jd \in \mathbf{Z}_M[j]/(j^2 + 1)$, and ψ denotes the isomorphism between the Gaussian integers and the CRNS: ψ: $\mathbf{Z}_M[j]/(j^2+1) \leftrightarrow \mathbf{Z}_{p_1} \times \mathbf{Z}_{p_2} \times \mathbf{Z}_{p_3} \times \cdots \times \mathbf{Z}_{p_L} \times \mathbf{Z}_{p_1} \times \mathbf{Z}_{p_2} \times \mathbf{Z}_{p_3} \times \cdots \times \mathbf{Z}_{p_L}$. Then

$$\begin{aligned}
(a + jb) + (c + jd) &= (a + c) + j(b + d) \\
&= \psi^{-1}\{\psi(a) + \psi(b)\} + \\
&\quad j\psi^{-1}\{\psi(b) + \psi(d)\} \\
(a + jb) \times (c + jd) &= (ac - bd) + j(ad + bc) \\
&= \psi^{-1}\{\psi(a)\psi(c) - \psi(b)\psi(d)\} + \\
&\quad j\psi^{-1}\{\psi(a)\psi(d) + \psi(b)\psi(c)\}.
\end{aligned}$$

While the complex addition takes only two additions, the complex multiplication takes four multiplications and two additions: the CRNS requires the same number of additions and multiplications as the Gaussian integers.

C. *Quadratic Residue Number System (QRNS)*

The QRNS [4] is a variation upon the RNS which allows complex additions to be performed with two RNS additions and complex multiplications to be performed with two RNS multiplications. This enhancement is accomplished by encoding the real and imaginary components into two independent components. Given a prime p of the form $p = 4k + 1$ where $k \in \mathbf{Z}$ then the congruence $x^2 \equiv -1 \pmod{p}$ has two solutions in the ring $\mathbf{Z}_p$ that are multiplicative and additive inverses of one another. Let $\hat{j}$ and $\hat{j}^{-1}$ denote the two solutions to the above congruence. Define a mapping θ: $\mathbf{Z}_p[j]/(j^2 + 1) \rightarrow \mathbf{Z}_p \times \mathbf{Z}_p$ by

$$\begin{aligned}
\theta(a + jb) &= (z, z^*) \\
z &\equiv (a + \hat{j}b) \pmod{p} \\
z^* &\equiv (a - \hat{j}b) \pmod{p}.
\end{aligned}$$

Furthermore, the inverse mapping θ^{-1}: $\mathbf{Z}_p \times \mathbf{Z}_p \rightarrow \mathbf{Z}_p[j]/(j^2 + 1)$ is given by

$$\theta^{-1}(z, z^*) = \langle 2^{-1}(z + z^*) \rangle_p + j\langle 2^{-1}\hat{j}^{-1}(z - z^*) \rangle_p.$$

Suppose $(z, z^*), (w, w^*) \in \mathbf{Z}_p \times \mathbf{Z}_p$. Then the addition and multiplication operations in the ring $(\mathbf{Z}_p \times \mathbf{Z}_p, +, \cdot)$ are given by

$$\begin{aligned}
(z, z^*) + (w, w^*) &= (z + w, z^* + w^*) \\
(z, z^*)(w, w^*) &= (zw, z^* w^*).
\end{aligned}$$

The isomorphism θ is generally implemented via arithmetic elements and table lookup. Since the z

and z^* channels are independent, parallel hardware may be constructed to perform operations on both channels at the same time without any communication between the channels. This parallelism allows a complex addition or multiplication to be performed in one cycle. While parallel hardware would allow a CRNS addition in one cycle, the multiplication in the CRNS requires two additions and four multiplications. Using the same amount of hardware as a QRNS multiplier-accumulator, a CRNS multiplier-accumulator would take twice as many cycles to complete a single multiply-accumulate operation.

D. *Galois Enhanced QRNS (GEQRNS)*

The QRNS requires a multiplier which takes N bit inputs and produces an N bit output. The multiplier could be implemented using either a direct implementation with modular correction or a lookup table. The primary disadvantage of this is that despite the small size of the RNS adder, the multiplier is still large. By taking advantage of the properties of Galois fields [9], it is possible to simplify the implementation of an RNS multiplier.

For any prime modulus p there exists some $\alpha \in \mathbf{Z}_p$ that generates all non-zero elements of the field $GF(p)$. That is to say $\{\alpha^i \mid i = 0, 1, 2, \ldots, p-2\} = GF(p) \setminus \{0\}$. Thus, all non-zero elements of $\mathbf{Z}_p$ may be uniquely represented by their exponents. These number theoretic logarithms may be added modulo $p-1$ to produce multiplication: $\alpha^{\langle i+j \rangle_{p-1}} = \langle \alpha^i \alpha^j \rangle_p$. Note that since zero is not an element of $GF(p) \setminus \{0\}$ the zero must be handled as an exception. Practically, this means that the inputs must be checked before the number theoretic logarithm to determine whether either one is a zero, and if one of the inputs is a zero, then the output of the multiplier should be set to zero.

The architecture of a GEQRNS multiplier is illustrated in Figure 1 without the zero detection and handling indicated. The multiplier requires two duplicate 2^N-entry memories to perform the number theoretic logarithm, an adder to add the logarithms, and an 2^{N+1}-entry table to perform the modulo $p-1$ correction and number theoretic exponentiation. Note that while the modulo $p-1$ correction and number theoretic exponentiation represent two separate steps, they may be integrated into a single table. Alternatively, if a modular adder is used the 2^{N+1}-entry table may be replaced with a 2^N-entry table. Typically, the multiplicands will be converted to the GEQRNS number theoretic logarithm form by the conversion engine which computes the residues of the integer inputs.

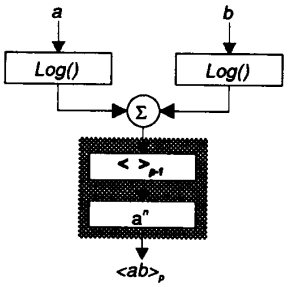

Figure 1: Block Diagram of a GEQRNS Multiplier

III. MOTIVATION

Historically, RNS has been used primarily when conventional arithmetic technology could not fulfill design requirements. Typically, this has led to the development of application specific solutions which were not portable to other problems. The benefits of RNS computing could be realized in more applications if standard RNS parts are available, either in the form of VLSI standard cells, or in the form of standard processor components.

The advantages of RNS are numerous. Since carry does not have to propagate across the full width of arithmetic elements, speed is significantly enhanced. A multiplier of arbitrary word width is built by replication of small multipliers so that the sum of the word widths of the small multipliers is the target word width: thus, the size of the RNS multiplier grows only linearly with word width, rather than geometrically as in conventional VLSI multipliers [1]. RNS multipliers are smaller than their conventional counterparts. For example, an eight by eight parallel multiplier from the University of South Florida cell library[10] is 2.956mm×0.834mm while the developed GEQRNS multiplier-accumulator is 2.0 mm × 1.0 mm (both in 2.0 micron CMOS). Since RNS is substantially smaller, it also realizes substantial power savings when compared with conventional technologies. The RNS has been shown to be capable of real time fault tolerance [3, 4]. Additionally, since the developed technology is largely memory based, well under-

stood techniques for yield enhancement[11] may be applied to the implementation thus creating the possibility for construction of larger than conventional (e.g., ULSI) systems using the RNS. Finally, the advantages of RNS may be combined to give substantial cost advantages over conventional arithmetic technologies with similar performance characteristics. In particular, the integration, power, fault and defect tolerance characteristics of the RNS can lead to substantial system cost reductions.

Systolic arrays have been shown to offer many desirable traits for signal processing systems [5, 6]. In particular, they are highly parallel and potentially may realize significant savings in control logic overhead. An additional promise of systolic arrays which, for the most part has not been realized yet, is speedup associated with the elimination of interpackage delays. In particular, elimination of interpackage delay was one of the original motivations for systolic arrays, yet placement of many processors on a single die has, for the most part[1], have not been realized [5, 12].

The combination of the developed RNS processor technology and systolic arrays promises to yield substantial advantages. First, because the RNS processor technology is substantially smaller than its conventional counterpart (e.g., an eight-bit RNS multiplier-accumulator implemented in 2.0 micron CMOS occupies less than 2 mm^2 of die area), many RNS processors may be arrayed on a conventional sized die. Secondly, because the RNS is memory based, conventional yield enhancement technologies may be applied to make larger than conventional single chip architectures manufacturable. Finally, the fault tolerant properties of the RNS may be used to further enhance the manufacturing yield of a highly integrated RNS systolic array. Therefore, large processor arrays may be realized without the speed, cost, or reliability penalties associated with large quantities of interpackage connection.

IV. DISCRETE IMPLEMENTATION

Most systolic arrays are designed to solve some particular problem optimally: there is considerable treatment of these problems in the literature. There exist many problems which could benefit from a high performance arithmetic accelerator which does not necessarily have to achieve an optimal solution. In particular, if a standard part is available to accelerate arithmetic operations, then it is not necessary to devote considerable engineering resources to developing an optimal processor. The advantages of mass production will allow the optimization of the standard part thereby mitigating some of the cost of a suboptimal solution.

A discrete prototype of the Gauss machine has been designed and constructed. This prototype is a 2×2 systolic array processor composed of three seven-bit GEQRNS channels for a total of six seven-bit RNS channels. The array of processors is arranged in a mesh-connected topology with unidirectional dataflow. Each single channel 2×2 array is implemented on a single card (see Figure 2) which is plugged into a host processor, see Figure 3. A block diagram of the implemented multiplier-accumulator processing element in depicted in Figure 4. The prototype was constructed with relatively pedestrian CMOS technology and operates at 10 MHz. This prototype serves as a testbed for algorithms and also as a proof of concept.

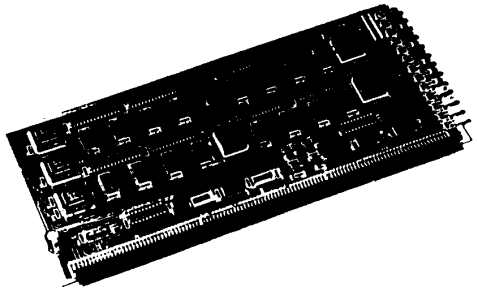

Figure 2: Photograph of Single Channel Processor Board

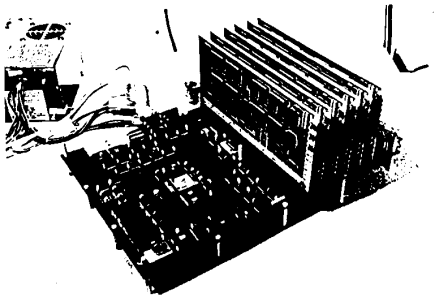

Figure 3: Photograph of Gauss Machine Prototype

The Gauss machine is designed to perform level 3 operations[13] very efficiently using the whole array,

[1]There are examples of single chip systolic arrays, such as the Martin Marietta GAPP processor.

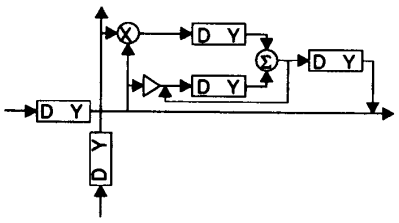

Figure 4: Block Diagram of Discrete Processor Implementation

and to perform level 2 and level 1 operations efficiently using its vector processing subarray. These types of operations are very commonplace in signal processing and thus the Gauss machine can accelerate many signal processing applications. Since the Gauss machine is designed to perform certain high level operations very well, these are generated in advance for the user who uses the array via a high level programming interface.

The array mode of operation utilizes the whole array to perform matrix multiplication. The array accumulates results in place, see Figure 5. This architecture allows the array to be used for larger matrix multiplication problems by using the array to compute 2×2 blocks of the result of a larger problem.

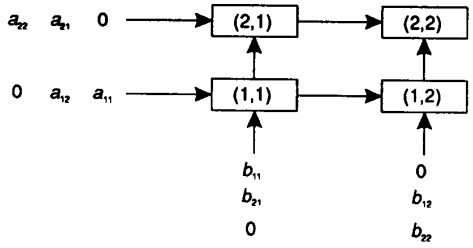

Figure 5: Matrix Multiplication Example

Since sloped data fronts are presented to the array, and results are accumulated in place necessitating shift out of the results, an analytic result for the number of clock cycles required to perform a matrix multiplication was developed. Suppose $\mathbf{A} \in \mathbf{C}^{m \times n}$, and $\mathbf{B} \in \mathbf{C}^{n \times r}$ then

$$O(\mathbf{AB}) = \lceil k/2 \rceil \lceil r/2 \rceil (n+6) \text{ cycles.}$$

The vector mode of operation utilizes two of the four processors to perform level 2 and level 1 operations, see Figure 6. The vector mode of operation may be used to efficiently perform inner products, vector accumulation, and pointwise vector addition and multiplication.

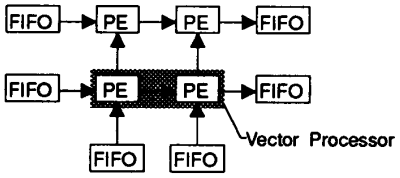

Figure 6: Block Diagram of Array with Vector Processor Indicated

Analytical results for the number of cycles required to perform vector operations have been computed. Using the vector mode of addition, K vectors of length N may be added in $KN + 3$ clock cycles. Using vector mode to pointwise multiply two vectors of length N requires $\lceil N/2 \rceil$ clock cycles. A performance summary is given in Table 1.

Operation	Rate
2×2 Cmplx. Mat. Mult.	1.25M/s
4×4 Cmplx. Mat. Mult.	250K/s
10×10 Cmplx. Mat. Mult.	26.7K/s
1000 point Cmplx. Ptws. Vect. Mult.	20K/s
1000 point Cmplx. Ptws. Vect. Add.	10K/s

Table 1: Performance Results Based on 10 MHz Operation

The array has very limited control needs. The array controller is an Am29CPL154 single-chip microsequencer. The host interface processor is a Motorola 68030 which communicates with the host workstation via SCSI bus or serial lines.

V. VLSI IMPLEMENTATION

A. *Description*

Figure 7 depicts the architecture of the GEQRNS multiplier accumulator cell. A die photograph is shown in Figure 8. The $4k + 1$ prime modulus chosen for this chip was 113 and the generator of $GF(113) \setminus \{0\}$ was chosen to be $\alpha = 3$. The basic operation of the cell is as follows: two seven bit input operands x and y which are the exponents of elements $\alpha^x, \alpha^y \in GF(p) \setminus \{0\}$, are fed to a seven

proach (case I) relies upon having a defective processor switched out of the array (see Figure 9). Clearly this only works well in certain types of systolic arrays (such as the linear array pictured). The second approach (case II) relies upon discarding all processors of one modulus if that single modulus array is not a viable processing unit (see Figure 9). In order to gauge the impact of these approaches to fault tolerance a yield estimate is developed.

Assume a simple Poisson model for fatal defect distribution. Then the probability of having n faults per cell is given by

$$P(N=n) = \frac{e^{-\lambda}\lambda^n}{n!}, \qquad (1)$$

where N is a random variable denoting the number of faults and λ is the average number of faults per processing element[11]. Note that $\lambda = AD$ where A is the processor element area and D is the average fatal defect density. Assume that each individual processing element cannot tolerate a single fault, then the case for $n = 0$ results and Equation 1 simplifies to

$$Y = P(N=0) = e^{-\lambda} = e^{-AD}. \qquad (2)$$

A conservative value of D for a typical process is one fatal defect per cm^2, and substituting this value into Equation 2 along with the processor element area ($A \approx 2.0$ mm^2) gives a survivability $S_{PE} = 0.9802$ for individual processors. The survivability of an N processor single modulus array is given by

$$S_A = S_{PE}^N + N S_{PE}^{N-1}(1 - S_{PE}),$$

when the case I scheme for defect tolerance is applied (take one processor element to be a spare), and $S_A = S_{PE}^N$ when the case I scheme is not applied. The system survivability for an M modulus system is given by

$$S_S = S_A^M + M S_A^{M-1}(1 - S_A),$$

when the case II scheme for defect tolerance is applied (discard one single modulus array), and $S_S = S_A^M$ when the case II scheme is not applied. Using the above formulas and the computed S_{PE}, yield projections for mesh connected arrays and linear arrays with four working moduli were generated, see Table 2, and Table 3, respectively.

From Table 2, it is seen that a twenty-five percent overhead in die area produces substantial yield

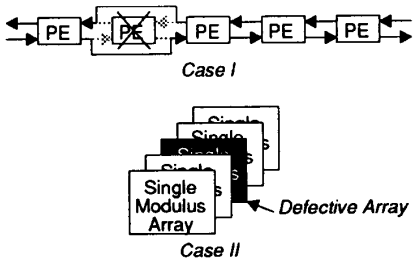

Figure 9: Case I: Linear Array with Defective Processor Bypassed, and Case II: Unused Defective Array

Size (N)	Defect Tolerance Scheme	S_A	S_S	Die Area Overhead
2×2	—	92.3 %	72.6 %	0 %
2×2	II	92.3 %	94.9 %	25 %
4×4	—	72.6 %	27.8 %	0 %
4×4	II	72.6 %	58.3 %	25 %

Table 2: Yield Results for Mesh Connected Arrays: $M = 4$, or $M = 5$ in case II

enhancement. Even more dramatic results are realized by application of the case I approach to yield enhancement, as noted in Table 3. Also note that it is possible to incorporate defect tolerance at the processor level (namely, in the GEQRNS exponentiation ROM[11]), a feat which is not generally possible with traditional arithmetic technologies.

VI. CONCLUSIONS

The Gauss machine represents a new class of processors using the RNS to perform front-end signal and image processing operations. In the past the RNS has been restricted to a limited set of applications to which an RNS system would be hardwired. The Gauss machine is more general purpose in scope; it is able to perform a variety of operations which may be expressed as level 1, level 2, or level 3 operations.

The Gauss machine has been implemented as a 2×2 array of GEQRNS multiplier-accumulators which may be operated in a systolic mode to perform level 3 operations, or may be operated in a vector mode to perform level 1 and level 2 operations. The Gauss machine is a six channel system based upon seven-bit moduli and achieves a dynamic range of 20.2 bits

Length (N)	Defect Tolerance Scheme	S_A	S_S	Die Area Overhead
4	—	92.3 %	72.6 %	0 %
4	II	92.3 %	94.9 %	25 %
5	I	99.6 %	98.5 %	25 %
5	I,II	99.6 %	99.9 %	56 %
20	—	67.0 %	20.2 %	0 %
20	II	67.0 %	46.8 %	25 %
21	I	93.6 %	76.7 %	5 %
21	I,II	93.6 %	96.3 %	31 %

Table 3: Yield Results for Linear Arrays: $M = 4$, or $M = 5$ in case II

or 122 dB in each of the real and imaginary components while achieving a peak equivalent performance of 320 million operations per second.

A custom VLSI implementation of the an eight bit GEQRNS multiplier accumulator has been fabricated using the MOSIS 2.0 micron CMOS process. The multiplier-accumulator cell occupies approximately 2.0 mm^2 and has been tested at 20 MHz.

Future directions of development for the Gauss machine include integrating a 2×2 array of four moduli GEQRNS accumulators onto a single die, and investigating MSIMD architectures utilizing "Gauss machine on a chip" as the processing engine. It is hoped that such an architecture may achieve high performance at low cost in terms of hardware and software development costs.

The Gauss machine offers high performance while minimizing power dissipation, physical form factor, and cost. It has been demonstrated that modest increases in hardware complexity can produce significant gains in fault and defect tolerance. Consequently, this technology has the potential to scale to ULSI proportions and provide low cost and high reliability in the same package. This technology has benefits to technologies as diverse as RADAR, and communications to medical diagnostic signal processing.

References

[1] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective.* Reading, Mass.: Addison-Wesley Publishing Company, 1985.

[2] F. J. Taylor, J. Mellott, J. Smith, and G. Zelniker, "The Gauss machine — a DSP processor with high RNS content," in *Proc. IEEE International Conf. on Acoustics, Speech, and Signal Processing*, 1991.

[3] W. K. Jenkins, "The design of error checker for self-checking residue number arithmetic," *IEEE Trans. Computers*, vol. 32, pp. 388–396, Apr. 1983.

[4] J. V. Krogmeier and W. K. Jenkins, "Error detection and correction in quadratic residue number systems," in *26th Midwest Symposium on Circuits and Systems*, 1983.

[5] H. T. Kung and others;, "iWarp: an integrated solution to high-speed parallel computing," *IEEE Trans. Computers*, vol. 38, pp. 330–339, Sep. 1988.

[6] S. Y. Kung, "VLSI array processors," *IEEE ASSP Magazine*, pp. 4–22, July 1985.

[7] M. Griffin, F. J. Taylor, and M. Sousa, "New scaling algorithms for the chinese remainder theorem," in *Proc. 22nd Asilomar Conf. on Signals, Syst., and Computers*, 1988.

[8] M. Griffin, M. Sousa, and F. J. Taylor, "Efficient scaling in the residue number system," in *Proc. IEEE International Conf. on Acoustics, Speech, and Signal Processing*, 1989.

[9] G. Zelniker and F. J. Taylor, "A reduced complexity finite field alu," *IEEE Trans. on Circuits and Systems*, vol. 38, pp. 1571–1573, Dec. 1991.

[10] H. A. Nienhaus and S. Alyea, "CMOS cell library development project," Tech. Rep., University of South Florida, May 1987.

[11] S. E. Schuster, "Multiple word/bit line redundancy for semiconductor memories," *IEEE Journal of Solid-State Circuits*, vol. SC-13, pp. 698–703, Oct. 1978.

[12] R. Simar, "The TMS320C40: a DSP for parallel processing," in *Proc. IEEE International Conf. on Acoustics, Speech, and Signal Processing*, pp. 1089–1092, 1991.

[13] G. H. Golub and C. F. van Loan, *Matrix Computations.* Baltimore: Johns Hopkins University Press, 2nd ed., 1989.