VLSI IMPLEMENTATION OF HIGH THROUGHPUT DSP USING FINITE RING ARITHMETIC

Graham A. Jullien
Director, VLSI Research Group

University of Windsor, Windsor, Ontario, Canada, N9B 3P4

ABSTRACT

This paper reviews recent strategies in implementing DSP systems using residue computations; in particular we highlight current work underway in the VLSI Research Group, at the University of Windsor, in the area of high throughput DSP systems on silicon The paper reviews a series of issues including recently disclosed mapping techniques, fault tolerant architectures, and area and power efficient VLSI implementation procedures. CAD tools, developed for automating the design and layout of residue DSP systems on silicon, are also discussed. An extensive bibliography is provided for further reading.

1. INTRODUCTION

Number Theory is often treated as the last vestige of pure mathematical pursuit. Some results, of course, are usefully applied [1]; signal processing examples are in the areas of cryptography, error correction/fault tolerance and in DSP arithmetic implementation. This paper is concerned with the latter application area. Residue Arithmetic for computer systems has been studied for at least 4 decades, but has never been widely applied to commercial systems. There are good reasons for this, mostly based on the awkwardness of operations that do not exist under closure. Some limited success has been reported in the area of integer arithmetic DSP systems, where the algebraic tricks associated with computing over finite rings and fields, can be used to great effect. Examples are the use of index calculus [2] (finite field 'exact' logarithms) and quadratic residue rings for reduced and separable complex arithmetic [3].

In this paper we emphasize the importance of the structure that residue computation imparts to DSP architectures. One-dimensional algorithms, for example, stay one-dimensional at the bit level, thus providing a more benign data path and clock routing environment. In approaching the automated design of such structures, we consider the choice of DSP algorithm, the mapping and resulting architecture, and an implementation strategy that is tuned to the special needs of pipelined residue computation.

2. COMPUTING OVER FINITE RINGS

In RNS systems [4] we deal with rings, or fields, that are used for the actual implementation and rings that are isomorphic to direct products of implementation rings or extensions of them. A given digital signal processing algorithm is mapped from real or complex integer arithmetic to the implementation rings, the computation is carried out there, and the result is then mapped back to obtain the final answer.

We denote by $R(m)$ the ring of integers modulo m:

$$R(m)=\{S:\oplus_m,\otimes_m\};\ S=\{0,1,..,m-1\} \tag{1}$$

where we use the notation $a\oplus_m b$ and $a\otimes_m b$ to imply the residue reduction of a and b modulo m within addition and multiplication. We can extend the notion of addition and multiplication from the elements of S to all of the integers. If R_1 and R_2 are any two rings then we can define the cross-product ring $R_1\times R_2$ as the set of pairs $(s_1,s_2)\in S_1\times S_2$, with addition and multiplication defined component wise, i.e. by eqn. (2).

$$\begin{aligned}(a_1,a_2)\oplus_{R_1\times R_2}(b_1,b_2)&=(a_1\oplus_{R_1}b_1,a_2\oplus_{R_2}b_2)\\(a_1,a_2)\otimes_{R_1\times R_2}(b_1,b_2)&=(a_1\otimes_{R_1}b_1,a_2\otimes_{R_2}b_2)\end{aligned} \tag{2}$$

The isomorphism between $R(M)$ and the direct product of $\{R(m_k)\}$ means that calculations over $R(M)$ can be effectively carried out over each $R(m_k)$, independently and in parallel. A final mapping to $R(M)$ is performed at the end of a chain of calculations. We have therefore broken down a calculation set in a large dynamic range, M, to a set of L calculations set in small dynamic ranges given by the $\{m_k\}$. This is the main advantage of using the RNS over a conventional weighted value numbering system (e.g. binary). The advantages of independent (not just parallel) computation should not be underestimated, particularly when very large special purpose high throughput processing arrays are to be built. The final mapping is found from the CRT:

$$X=\sum_{k=1}^{L}{}_{M}\left\{\hat{m}_k\otimes_M\left[x_k\otimes_{m_k}(\hat{m}_k)^{-1}\right]\right\} \tag{3}$$

with $\hat{m}_k=M/m_k$, $X\in R(M)$, $x_k\in R(m_k)$ and $(\bullet)^{-1}$ the multiplicative inverse operator. We have also used the notation Σ_M to indicate summation over the ring $R(M)$.

2.1 Polynomial rings and quotient rings

We let $R[X]$ denote the ring of polynomials in the indeterminate:

$$X:R[X]=\left\{\sum_{k=0}^{n}a_kX^k:a_k\in R,n\le 0\right\} \tag{4}$$

We define the ring $R[X_1,X_2,\ldots,X_S]$ to be the ring of multivariate polynomials in the indeterminates. We use polynomial rings, where the base ring R, is a modular ring, $R(M)$, and we write $R_M[X_1,X_2,...,X_S]$ in place of

0-7803-2431-5/95 $4.00

$R(M)[X_1, X_2, ..., X_S]$. For a given polynomial $g(X) \in R\{X\}$ we consider the set $(g(X))$ of all (polynomial) multiples of $g(X)$. This set is called the 'ideal' generated by the polynomial $g(X)$ in the ring $R[X]$. The quotient ring $R[X]/g(X)$ is then defined to consist of all elements of the form $f(X)+(g(X))$, with $f(X) \in R[X]$. We now let indeterminates represent various powers of 2 in the binary representation of the data samples [5, 6]. This allows the data to be expressed as polynomials with small coefficients. These coefficients are then mapped to a direct product ring consisting of many copies of Z_M (the ring of integers modulo m) as factors. This has been referred to as a Modulus Replication RNS (MRRNS) [5].

In order to be able to perform useful computations, the modulus, M, has to be able to contain the coefficients of result polynomials. Multiplication will be the major problem in coefficient growth, and we assume that the algorithm is arranged so that only single cascades of multipliers are used prior to the application of mapping circuitry. We can further decompose M, to allow the use of very small rings, by the application of a RNS. The mathematical derivations are somewhat tedious, and the reader is referred to a more complete description in [6].

The same features associated with computation over direct product rings are present with this technique; we may even embed a standard RNS within the mapping structure [6].

3. APPLICATIONS TO DSP SYSTEMS

Purely feedforward algorithms provide the niche area for efficient residue implementations. In terms of applications, this covers a large subset of algorithms commonly used in DSP systems. Obvious examples are DFT, DCT, FIR filters and general matrix operations. There are also non-obvious uses, such as digital waveform synthesizers [7]. Systems built either entirely, or partially, with feedforward building blocks, are potential candidates for residue implementation approaches. Recent work on minimizing operations that do not exist under closure has revealed strategies that also preserve the arithmetic efficiency of fast algorithms [8]. An example of a 15 sample Discrete Cosine Transform that supports such a single multiplication cascade is shown in Fig. 1. The strip in the centre contains the multiplications, all other operations being additions or subtractions.

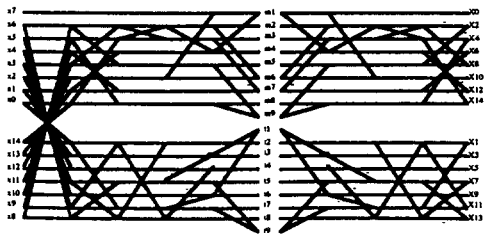

Figure 1. 15-point 'Fast' DCT with no scaling requirements

Figure 2 shows how this algorithm will be mapped to a set of independent computations over cross product rings. The algorithm is simply replicated with each computation being performed over different rings (as in the case of an RNS map), or replications of the same ring (as in the case of a MRRNS map).

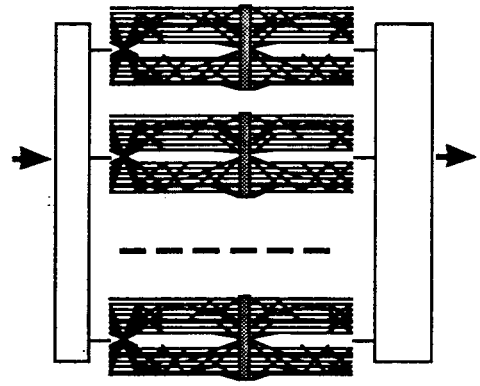

Figure 2. Computing the DCT over independent rings

In the latter case we may embed an RNS system in order to compute over large rings via the use of direct products of smaller rings. Using this approach we have been able to show the computation of the equivalent of over 40-bits of dynamic range with the moduli: {3,5,7} [6].

A major stumbling block at the circuit level is the hardware required to implement residue computations. The logic gate approach is perfectly matched to highly decomposable binary arithmetic functions, but a bane to the implementation of general residue computations. Traditional techniques have concentrated on either the use of stored tables, or the use of ring moduli that are very close to a power of 2. We will now look at the VLSI implications of computing over finite rings.

4. VLSI ARITHMETIC ARCHITECTURES

If we consider only feedforward algorithms, then we need not be concerned with pipelining issues with regard to dynamic range scaling around recursive feedback loops. Our algorithms can usually be implemented with some form of a finite ring ALU. This can take on a variety of forms depending on the algorithm. A very important building block is the inner product step processor (IPSP), and this has been discussed many times over the past decade or so. It does, however, provide a nucleation centre for discussing the computational issues.

4.1 Computing Over General Ring Moduli

If we consider computing over general ring moduli, then we normally rely heavily on the storage of tables in ROMs. It is clear that we can perform any binary (2-variable) operation over a ring modulo an n-bit modulus using tables that have 2n-bit address inputs, and n-bit word width outputs. A complete IPSP can therefore be built with two such ROMs. A much better solution is to decompose the computation so that only n-bit address lines are required (this reduces the area of the ROM from $O(2^{2n})$ to $O(2^n)$).

4.1.1 Bit-Steered Cells

An approach introduced by the VLSI Research Group uses bit-steered cells [9] to reduce inner product step

computations over finite rings to operations at the individual bit level. The cells use n-bit address width ROMs and steering logic. The steering logic forms a natural 'scan path' and so provides a free testing path (free in the sense that the path is required for the arithmetic architecture). Using an extra ROM output bit, we can implement a parity check daisy chain and so obtain circuit level concurrent error detection [9]. We have also used this circuit level detection in conjunction with MRRNS and embedded RNS systems to propose 3 architectures for general fault tolerance. The reader is referred to [10] for details. Using the isomorphic mapping property of the finite ring IPSP we can also test arbitrarily large chains with only 2^n+2 vectors [11].

4.1.2 Prime Moduli

If we restrict the general modulus to be prime (still 'fairly' general!) then the use of index calculus is particularly interesting, and is the trick used in a commercial FIR filter chip from INMOS [2]. There is also evidence from the academic research community that this represents a viable decomposition strategy [3]. Some early work from our group details the construction of prime moduli multipliers using only n-bit ROMs [12]. In the case of [3], all of the computations are performed using index calculus; multiplication is clearly easy, and we use the old 'slide-rule' trick to perform addition with a log calculator [13].

We wish to compute over the Galois Field $GF(p)=\{S:\oplus_p,\otimes_p\}$ where p is prime. We invoke the mapping, $a\otimes_p b = g^{(\alpha\oplus_{p-1}\beta)}$ between the additive group, $G_\oplus=\{\{S\}-(p-1):\oplus_{p-1}\}$, and the multiplicative group, $G_\otimes=\{\{S\}-(0):\otimes_p\}$, where g is a generator of $G_\otimes$. It is apparent that in order to perform multiplication over the field (except with the element 0), all one has to do is to map to the additive group, perform addition Mod p-1 and then map back. To perform addition, we simply factor out the addend and store all combinations of the remaining factor in a ROM.

If we consider the IPSP of eqn. (6):

$$c_{out}=\left(a_{in}\otimes_p a_{in}\right)\oplus_p c_{in}$$
$$a_{out}=a_{in} \qquad (6)$$

then we can use index calculus, as in eqn. (7).

$$\gamma_{out}=\gamma_{in}\oplus_{p-1}\Im\left(g^{(\alpha_{in}\oplus_{p-1}\beta_{in}\oplus_{p-1}[-\gamma_{in}])}\oplus_p 1\right) \qquad (7)$$

A ROM stores the forward mapping function, $\Im$, and a mod p-1 adder/subtractor is used to form the n-bit address input. Addition over general moduli has been explored by several authors (a compendium can be found in [4]), more recent work from our group can be found in [14].

4.2 Computing Over Special Ring Moduli

Much work has been reported on circuitry that can be used for special moduli [4]. The selection of moduli close to a power of 2 is of particular interest, and the modulus set $\{2^n-1,2^n,2^n+1\}$ has attracted much interest over the past 2 decades. With the introduction of the MRRNS we now have the ideal strategy of using a single, specially selected, ring modulus, and replicating computations over this ring to obtain the required dynamic range. Based on the Galois Field IPSP reported in 4.1 we can restrict our prime modulus to the form $2^{2^l}+1$ (a Fermat number which is prime, for at least the first 5!). The advantage of using a prime of this form is that the index calculus can be performed using 2^{2^l} bit binary adders. Probably the most perfect numbers for this approach are 257 and 17. We can restrict the calculations to Mod 257, but the ease with which we can embed a {17,257} RNS within a MRRNS mapping is too good to pass up. Figure 3 shows the structure of a mod 257 index calculus IPSP processor where the adder/subtractors are 8-bit (with no carry out). The block at the bottom computes the NAN (not a number) bits that are required to handle the missing group elements.

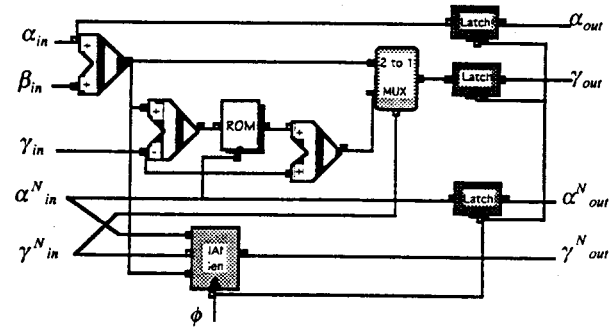

Figure 3. An Index Calculus IPSP

5. VLSI CAD ISSUES

Most of our efforts have been devoted to the lower level issues of special module generators for finite ring computational cells, though we have reported a formal mapping tool for bit-steered ROM cell DSP systems [15].

It is clear that a major component of residue computations is an n-bit ROM. Much effort has been expended on minimizing the ROM size by decomposition of the table contents (see appropriate papers in [4]). Our approach is different. We build the ROMs n-dimensionally as binary decision transistor trees, and minimize the tree structure using graph theoretic approaches [16]. This removes the excess bandwidth problem associated with 2-D ROMs (the bandwidth reduction occurs when the column decoder selects one word from a prior selected row). The trees are embedded in TSPC dynamic latches [17]; this is a restriction on the use of the technique...all output bits have to be pipelined. With current sub-micron technologies it is possible to minimize tables with 14-bit address words, allowing the implementation of residue computations with general 7-bit moduli. An example of a 10-input block is the Modulo 17 multiplier of Figure 4. Our experimental evidence suggests that we will obtain a minimum reduction of about 80% of the transistors in the full tree and we have measured a

reduction of about the same (80%) in the power dissipation compared to PLA implementations using TSPC latches.

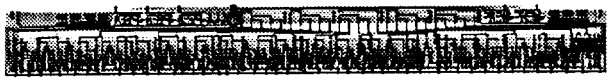

Figure 4 10-bit Minimized Look-Up Table

Recently measured results on 14-bit input blocks, fabricated in a 0.8micron BiCMOS process using a special current sense latch circuit [18], demonstrate solid operation at 50MHz, as shown in Fig. 2, with greater than 100MHz rates predicted from simulation.

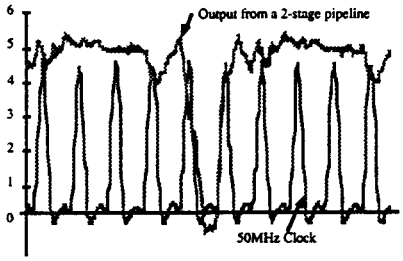

Figure 5 Pipelined 14-bit Minimized ROM at 50MHz

Both the structure of Figure 3 and the bit-steered cell reported in 4.1.1 are perfect matches for FPGA implementation; this brings the flexibility of independent computations to rapid prototyping of DSP systems. The only critical issue is the construction of the ROM, and this is normally a component that has already been automated in commercial FPGA software. We have recently disclosed a BDD design technique that reduces ROM resource size by up to 30% (based on Xilinx technology) using the fact that there are many *don't care* states in a residue table [19].

6. CONCLUSIONS

In this paper we have very briefly discussed the issues associated with computing over finite rings and bringing the results to VLSI technology for the special purpose of implementing high throughput DSP systems. We have highlighted a recent polynomial mapping technique as having considerable potential for implementing special field computations using minimal computation. We have also discussed recent results in building ROMs for RNS applications using both full-custom BiCMOS and FPGA technology.

REFERENCES

1. Schroeder, M.R., 1986. "Number Theory in Science and Communication." Springer-Verlag. Berlin.

2. Barraclough, S.R., Sotheran, M., Burgin, K., Wise, A.P., Vadher, A., Robbins, W.P. and Forsythe, R.M., 1989."The Design and Implementation of the IMS A110 Image and Signal Processor." *IEEE Custom Integrated Circuits Conf.*, pp. 24.5.1-24.5.4

3. Mellott, J.D., Smith, J.C. and Taylor, F.J., 1993."The Gauss Machine: A Galois-Enhanced Quadratic Residue Number System Systolic Array." *Proceedings of the 11th IEEE Symposium on Computer Arithmetic.*, Windsor, Canada. pp. 156-162.

4. Soderstrand, M.A., Jenkins, W.K., Jullien, G.A. and Taylor, F.J., 1986. "Residue Number System Arithmetic: Modern Applications in Digital Signal Processing." IEEE Press. New York, NY.

5. Wigley, N.M. and Jullien, G.A., 1990. "On Modulus Replication for Residue Arithmetic Computations of Complex Inner Products." *IEEE Trans. Comp.* 39, August, pp. 1065-1076

6. Wigley, N.M. and Jullien, G.A., 1994. "Large Dynamic Range Computations Over Small Finite Rings." *IEEE Trans. Comp.* Vol. 43, No. 1, pp. 76-86

7. Chren, W.A.J., 1994."Area and Latency Improvements for DDS Using the Residue Number System." *Proceedings of the 37th Mid-West Symp. on Circuits and Systems.*, Lafayette, LA. Paper 22.5 (in print).

8. Wang, Z., Jullien, G.A. and Miller, W.C., 1991."Algorithms for Length 15 and 30 Discrete Cosine Transforms."*1991 Asilomar Conference on Circuits Systems and Computers.*, Pacific Grove, CA. pp. 111-115.

9. Taheri, M., Jullien, G.A. and Miller, W.C., 1988. "High Speed Signal Processing Using Systolic Arrays Over Finite Rings." *IEEE Trans. Selected Areas in Comm.* 6, 3,

10. Jullien, G.A., Bizzan, S. and Wigley, N.M., 1994."Using Redundant Finite Rings for Fault Tolerant Signal Processors." *SPIE 1994.*, San Diego, in print

11. Jullien, G.A., Taheri, M., Bandyopadhyay, S. and Miller, W.C., 1990. "A Low-Overhead Scheme for Testing a Bit Level Finite Ring Systolic Array." *Journal of VLSI Signal Processing.* , Vol 2.3 pp. 131-138.

12. Jullien, G.A., 1980. "Implementation of Multiplication, Modulo a Prime Number, with Applications to Number Theoretic Transforms." *IEEE Trans on Computers.* Vol. C-29, No.10, pp. 899-905

13. Zelniker, G. and Taylor, F.J., 1991. "A Reduced-Complexity Finite Field ALU." *IEEE Trans. on CAS.*, Vol. 38 , No. 12 pp.1571-1573.

14. Bayoumi, M.A., Jullien, G.A. and Miller, W.C., 1987. "A VLSI Implementation of Residue Adders." *IEEE Trans. on CAS.* , Vol. CAS-34 , No.3

15. Jullien, G.A., Bandyopadhyay, S., Miller, W.C. and Frost, R., 1989."A Modulo Bit-Level Systolic Compiler." *Proceedings of the 1989 ISCAS.*, Portland, pp. 1388-1391.

16. Jullien, G.A., Miller, W.C., Grondin, R., Del Pup, L. and Zhang, D., 1992. "Dynamic Pipelined Computational Blocks for Bit-Level Systolic Arrays." *IEEE Jour. Solid-State Circuits.* submitted

17. Afghahi, M. and Svensson, C., 1990. "A Unified Single-Phase Clocking Scheme for VLSI Systems." *IEEE J. Solid-State Circuits.* 25, Feb., pp. 225-233

18. Czilli, J.C., Zhou, P., Jullien, G.A. and Miller, W.C., 1994. "BiCMOS Current Steering Pipeline Circuit Technique." *IEE Electronics Letters.* , Vol. 30 , No. 12 pp. 943-945.

19. Venkatesan, R., Phoukas, D. and Jullien, G.A., 1994."A New Algorithm for Minimizing the BDD Size of Incompletely Specified Functions." *Proceedings of the 1994 CMC Workshop on FPGAs.*, Kingston, Ont. pp. 2.6.1-2.6.5