USING REDUNDANT NUMBER REPRESENTATIONS FOR EFFICIENT VLSI IMPLEMENTATION OF MODULAR ARITHMETIC

M G Parker, M Benaissa

Introduction The VLSI implementation of multiplication is usually achieved as an addition of partial products. Recent research has proposed the use of Redundant Number Representations (RNR) to limit the carry propagation inherent within these additions [1], enabling high-speed multiplier designs. Multiplication over a modulus is often accomplished by interleaving the partial product additions with a modular adjustment to prevent word growth [2]. However, for moduli other than 2^N or $2^N \pm 1$, this modular adjustment complicates the multiplier structure, requiring asymmetrical feedback or extra additions. This complication arises because the binary data is represented using consecutive powers of 2 (canonical basis). In this paper, an initial basis conversion of the data is proposed. Conventional binary logic is retained but consecutive bits now represent consecutive powers of β (a β-basis). The order of β, mod M, is N if $\langle \beta^N \rangle_M = 1$ where $\langle \beta^n \rangle_M \neq 1$ for $n < N$. If N is equal to or slightly greater than the β-basis wordlength of the data, then multiplication by a power of β can be achieved efficiently using only bit rotation. By concatenating r basis conversions (BCs), multiplications by $\beta_0^{n_0}.\beta_1^{n_1}.....\beta_{r-1}^{n_{r-1}}$, mod M, are achieved using only bit rotations. Finally, it is shown how basis conversion (BC) facilitates the design of fully symmetric and systolic general modular multipliers, requiring no modular adjustment. In general, these new bases are redundant, some or all of the integers $0, 1, \ldots, M-1$ having multiple representations in the new basis. Whereas conventional RNR schemes ease the additive task, the RNRs discussed here ease the multiplicative task, modular adjustment being eliminated by a slight 'widening' of the data wordlength.

Modular Multiplication Using Basis Conversion Consider the product,

$$c = \langle a.b \rangle_M \tag{1}$$

and the following conditions,

1. $b \in \{\beta^0, \beta^1, \ldots, \beta^{N-1}\}$.
2. β has order N, mod M, where N is not much greater than the input wordlength $\lceil \log_2(M) \rceil$.
3. The minimum number of bits required to form a canonical β-basis, mod M, is equal to or slightly less than N.

If the above conditions are met, (1) is conveniently accomplished by converting a and b to a β-basis, then computing (1) using bit rotation, (Fig 1). If b is known beforehand, then $\log_\beta(b)$, mod M can be passed directly to the rotation module (Fig 2). For example, if $M = 547$ (a prime), and $\beta = 3$, where 3 has order 14, mod 547, then a $2 \rightarrow 3$ BC enables the rotational computation of the products, $c = \langle a.3^n \rangle_{547}$, $0 \leq n < 13$. The input is conventional 10-bit, 2-basis data, the o/p 14-bit 3-basis data (RNR). If $a = 342 = 0101010110$ then,

$$a' = \begin{array}{cccccccccccccc} 3^{11} & 3^{10} & . & . & . & 3^2 & 3^1 & 3^0 & \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & = 342 \end{array}$$

(Note, a minimum of 12 bits are necessary to represent every integer from $0 \ldots 546$ in a canonical 3-basis). The product is performed by rotating a' n times within a 14-bit wordlength. If $n = 9$,

$$\langle a'.3^9 \rangle_{547} = \begin{array}{cccccccccccccc} 3^{13} & 3^{12} & . & . & . & . & . & 3^2 & 3^1 & 3^0 & \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & = 204 \end{array}$$

The Number Theoretic Transform (NTT) is given by,

$$X[k] = \left\langle \sum_{n=0}^{N-1} x[n].\beta^k \right\rangle_M \tag{2}$$

where β has order N, mod M.

School of Eng, Huddersfield Univ, Huddersfield HD1 3DH, UK, Tel 0484 472165, E-Mail M.Benaissa@eng.hud.ac.uk

The BC-based multiplier can be used to compute all products inherent within the NTT. For instance, the previous example can form part of a 14-point NTT, mod 547, (Fig 3). All 14 distinct products of $a[n]$ are achieved using one $2 \rightarrow 3$ BC.

Modular Multiplication Using Concatenated Basis Conversion Consider (1) again, and the following conditions for the concatenation of r BCs,

1. $b \in \{\prod_{i=0}^{r-1} \beta_i^{n_i}\}$ where $0 \leq n_i < N_i$.
2. β_i has order N_i, mod M, $0 \leq i < r$, where all N_i are equal to or slightly larger than $\lceil \log_2(M) \rceil$.
3. The minimum number of bits required to form a canonical β_i-basis, mod M, is equal to or slightly less than N_i, $0 \leq i < r$.

If the above conditions are met for $r = 2$, (1) is conveniently accomplished as shown in Fig 4, where $b = \beta_0^{n_0}.\beta_1^{n_1}$. As a special case, if $\gcd(N_0, N_1) = 1$, then an element g exists, such that,

$$g = (\beta_0^{n_0}.\beta_1^{n_1})_M \quad 0 \leq n_0 < N_0 \quad 0 \leq n_1 < N_1 \tag{3}$$

and Fig 4 can compute all distinct products, $a.g^0, a.g^1, \ldots, a.g^{N_0.N_1-1}$, mod M. Extending the previous example: Choose $M = 547$, $\beta_0 = 3$ and $\beta_1 = 46$ to implement products of the form, $(a.3^{n_0}.46^{n_1})_{547}$, $0 \leq n_0 < 13$, $0 \leq n_1 < 12$. (Note, β_1 has order 13, mod 547, and 12 bits are necessary to represent every integer from $0 \ldots 546$ in a canonical 3 or 46 basis). The multiplier uses $2 \rightarrow 3$ and $3 \rightarrow 46$ BCs, interleaved with rotation modules. This system can form part of a $14.13 = 182$-point NTT, mod 547.

Systolic Basis Converters Although basis conversion can be accomplished using ROMs, it is highly desirable to use systolic arrays operating at the bit-level. The authors are currently developing $\alpha \rightarrow \beta$ BC architectures of the form shown in Fig 5. These are modifications of the standard sum-carry cell, where consecutive rows and columns represent consecutive powers of α and β, respectively. The designs are fully-pipelined down to the cell level, where each cell is identical, comprises as little as three or four bits, and can be specified by a small state transition table. Instead of bits representing consecutive powers of the basis, two or three bits will be grouped to form a 'digit-set', $(\{0, d_0\}, \{0, d_1\}, \{0, d_2\}, \ldots$ etc), where each identical digit-set is weighted by consecutive powers of the basis. There are 'Modulus-Dependent' and 'Modulus-Independent' BCs. The 'Modulus-Dependent' BCs operate over specific moduli and use small cells. However they require special border cells and data offsets and are, to date, hard to find. The 'Modulus-Independent' BCs, on the other hand, exist for all bases and are easily found, but the cell size increases with basis. 'Modulus-Independent' $2 \rightarrow 7$ and $7 \rightarrow 2$ BCs form part of the general modular multiplier example, discussed next.

General Modular Multiplication Using Multiplicative and Additive RNR By combining the 'rotational', multiplicative RNR, discussed in this paper with 'partial-product', additive RNR [1], general modular multiplier designs are possible over a wide range of moduli, including prime and minimally-composite moduli. As an example, consider a general modular multiplier where $M = 329554457$ $(= 11\,23.293459)$, (Figs 6, 7, and 8). This can be implemented using basis-conversion of a and b from a 2-basis using the conventional digit-set, $\{0, 1\}$, and of wordlength 29, to a' and b', in a 7-basis, using the digit-set $\{0, 1\}, \{0, 2\}, \{0, 3\}$, and of wordlength 11 (Fig 6, Note, the digit-sets are duplicated as both a and b are being converted). This allows the product $c' = a'.b'$ to be computed by a parallel modular multiplier (Fig 7) using the additively redundant digit-set, $(\{0, 1\}, \{0, 2\}.\{0, 4\})$, before final conversion of c' back to a 2-basis of wordlength 31 bits (Fig 8). (Note, the 7-basis input data is permuted systolically to a $\sqrt{7}$-basis, prior to input into the parallel multiplier). A 29-bit, 2-basis, parallel modular multiplier has been replaced by an (11×3)-bit 7-basis parallel modular multiplier. The minimal increase in wordlength (from 29 to 33 bits) is offset by the regularity and elimination of modular adjustment hardware. Multipliers of this form have a very high throughput and become increasingly competitive for larger moduli and when used for higher arithmetic functions such as exponentiation.

Conclusion Novel, marginally redundant basis representations have been proposed to remove the need for modulus adjustment within the modular multiplier and enable rotational solutions to a subset of products. By concatenating basis conversions, this subset of possible products increases. The multipliers are well suited to signal processing tasks such as the NTT, where multiplications by powers of a kernel

are required. Highly pipelined systolic basis converters were then outlined which operate on 'digit-sets' instead of bits and, finally, the inherent symmetry exhibited by a low order basis over a given modulus is exploited to realise a general parallel modular multiplier which is regular and comprises only a few simple cell types. The redundancy due to the unusual basis representation is coupled with additive redundancy, which limits carry propagation, to achieve a very high throughput, systolic modular multiplier.

[1] S.C.Knowles,J.G.McWhirter, "The Application of Redundant Number Systems to the Design of VLSI Recursive Filters", *Proc of IEE Coll on Maths for Sig Proc*, pp 27-42, '88

[2] E.F.Brickell, "A Fast Modular Multiplication Algorithm with Application to Two Key Cryptography", *Advances in Cryptography, CRYPTO '82, PLENUM NY*, pp 120-126, '83

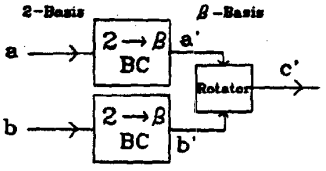

Fig 1. BC-Based Multiplier

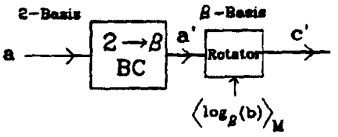

Fig 2. BC-Based Exponent Multiplier

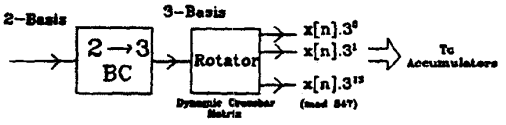

Fig 3. Product Generator for 14-Point NTT, Mod 547

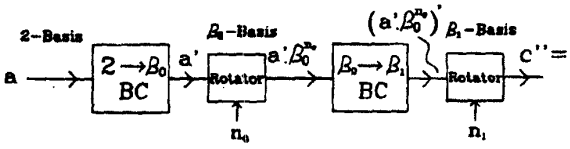

Fig 4. Concatenated BC-Based Multiplier

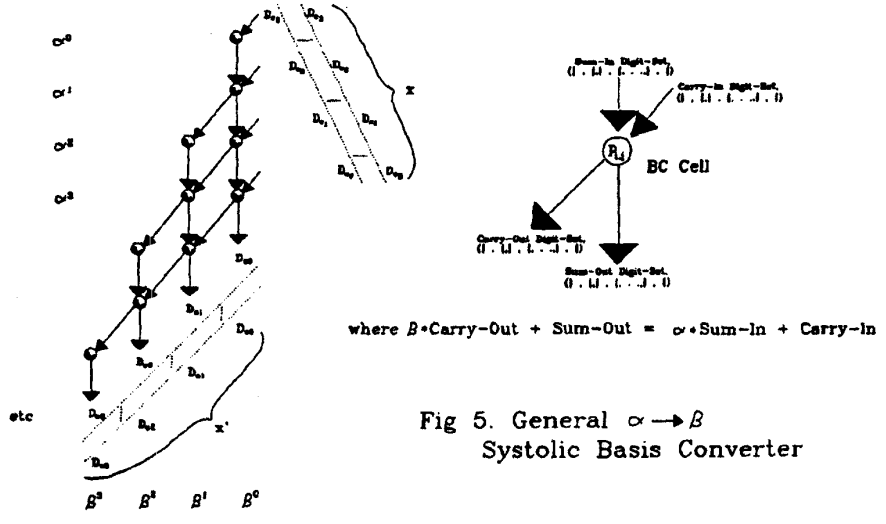

Fig 5. General $\alpha \rightarrow \beta$ Systolic Basis Converter

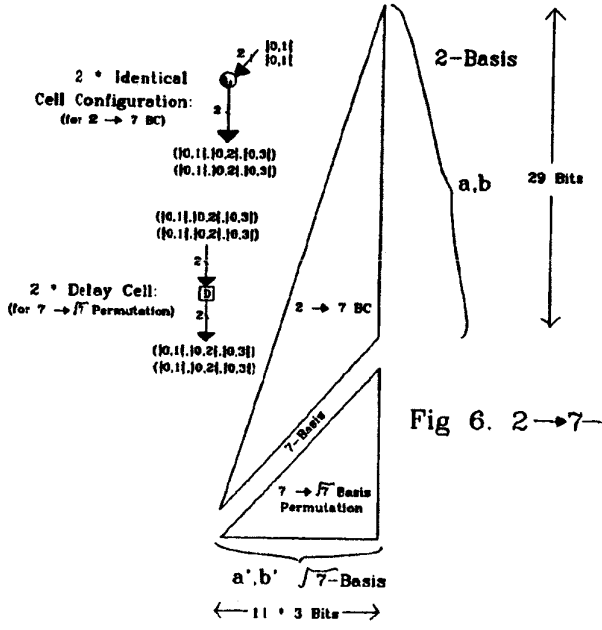

Fig 6. 2→7→√7 BC

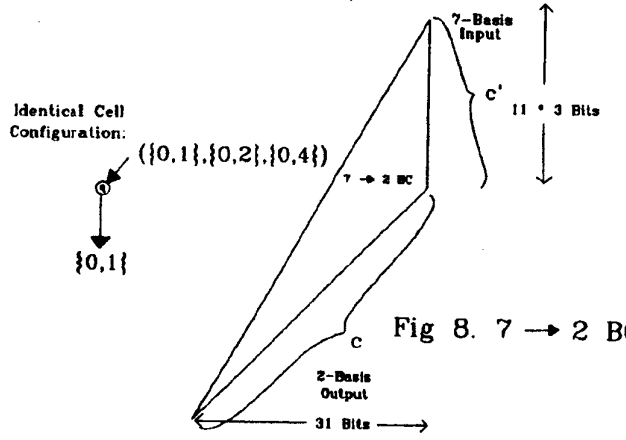

Fig 8. 7 → 2 BC

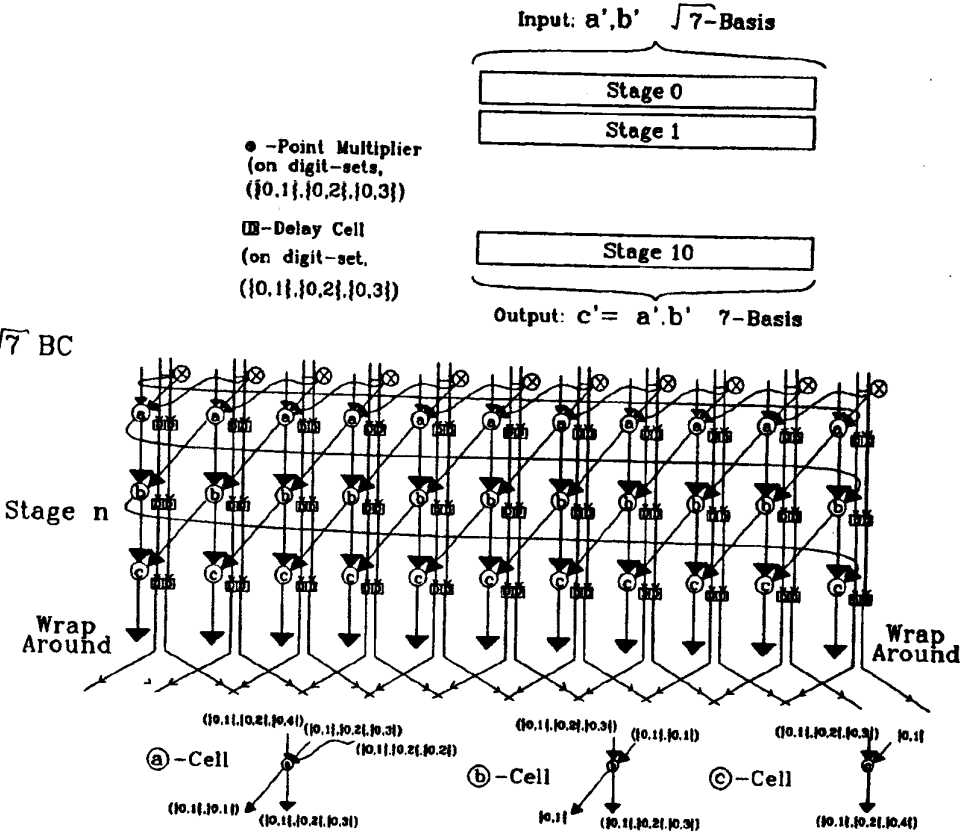

Fig 7. 7-Basis General Modular Multiplier, mod 32955457