LOW POWER SIGNAL PROCESSING ARCHITECTURES USING RESIDUE ARITHMETIC

Manish Bhardwaj, Arjun Balaram
Microelectronics Design Center
Telecommunication System ICs
Siemens Components Private Limited
166 Kallang Way, Singapore 349249.

ABSTRACT

Recent trends like increasing operating frequencies, larger die sizes and demand for greater portability make power reduction a hard taskmaster. It is acknowledged that the greatest returns come from optimizations at the architectural and technology level. In this paper, we present, for the first time, residue architectures that reduce power by more than 70% *without* changes in technology. This reduction is achieved without sacrificing performance and with minimal sacrifice in area (less than 60%). The key to such low power solutions is trading-off the speed gained by parallelism for lower power. Existing proposals that achieve similar trade-offs demand an area increase of more than a factor of two and also increase control complexity. Other benefits of using residue arithmetic for low power is the significant reduction in peak current and increased design locality. The role of the number of computations per forward (or reverse) conversion in determining the power characteristics of the system are also analyzed and explained. The effectiveness of the methodology is illustrated using a system that extracts a 256-point FFT of the input signal.

1. INTRODUCTION

Concerns for low power and low voltage have traditionally taken a backseat to those of high performance and low die size. This is changing fast, mainly due to increasing frequency, increasing number of devices per chip and the demand for portability.

Most current research on low power system stresses the need for action at all levels of hierarchy, starting from architectural and down to technology [1, 5, 9]. It is also accepted that greatest improvements can be made at the highest i.e. architectural and algorithmic level (and for power, also at the lowest i.e. technology specific level)[1,10].

Before we move on to discuss the previous proposals for architectural reduction of power, we present a quick review of residue number systems and residue arithmetic.

1.1 Overview of Residue Arithmetic (RA)

The Residue Number System (RNS) is a "carry-free" non-weighted number system that exploits parallelism by representing numbers as sets of remainders. [3,7,12].

The base of RNS is a set $\{m_1, m_2, \ldots, m_p\}$, where each member of the set is called a *modulus* and the set elements are pair-wise relatively prime. For any given moduli set, the residue representation of an integer X is a p-tuple, $(x_1, x_2, \ldots, x_p)$, where $\{x_i\}$ are least positive residues of X modulo m_i, and are defined by

$$x_i = |X|_{m(i)}, \qquad i = 1, 2, \ldots, p.$$

Beginning with zero, all numbers up to and excluding M given by,

$$M = \prod_{i=1}^{p} m_i$$

may be unambiguously encoded in RNS. This is often called the dynamic range of the given system.

The advantage of residue arithmetic is that arithmetic operations of addition, subtraction and multiplication can be performed for each residue independent of all other residues. The individual arithmetic operations can thus be done in parallel and the resulting speedup is close to the number of moduli.

This tremendous speedup rarely translates to system speed-up due to the need to convert weighted binary numbers into and out of the residue domain. These *forward* and *reverse* converters have to be carefully crafted to retain the advantages of parallel arithmetic.

1.2. Low Power Architectural Techniques

The key to power reduction using super-scalarity is the realization, that in VLSI CMOS, speed can be traded off for power by reducing the operating power-supply voltage. To maintain the original performance, some technique must be employed to reduce delay. Broderson et. al.'s work first demonstrated how super-scalar implementations could trade area for power [6]. The significance of this proposal was that it highlighted the enormous potential to reduce power without any changes to existing technology. On the other hand, it was impractical to achieve as presented, mainly because --

- It calls for an area increase of more than twice, which is usually unacceptable to system architects.
- The increase in control complexity and the resulting area and power increase were not taken into account.
- The issue of multiple voltages and their effect on the system was not considered.

- In deep submicron designs, an area doubling might have adverse effects on the interconnect that could potentially negate the power advantage.

2. LOW POWER THROUGH RESIDUE ARITHMETIC

Since architectural techniques rely on some kind of parallelism to reduce power, it is quite natural to investigate low power residue arithmetic architectures. In other words, the speed advantage of residue arithmetic can be traded for a significant power advantage. Note that there are fundamental differences in the kinds of parallelism that Broderson et. al. propose and what we propose through the use of RA --

- There is *no duplication* of hardware resources. The same resource is being made faster. Hence the operating frequency is *not* reduced. It stays *f*.
- The speed gain is roughly proportional to the number of moduli i.e. *s*. Hence, if the old datapath delay is taken to be *t* ns, the new delay is *t/s* ns. Power savings are realized by reducing V_{dd} so that the delay is *t* ns again. In Broderson's scheme, reducing delay by *s* times requires an *s* times increase in area. As we see later, our scheme roughly requires a 54% increase in area.
- In RA-based low-power architectures, the increase in area and hence capacitance comes about due to --

  1. Use of forward and reverse converters.
  2. Extra logic for modulo correction.

  Unlike Broderson's proposal, the area does not increase linearly with speed-up. For sufficiently large RA implementations, the increase in converter area or delay with *s* is negligible [11].
- Unlike super-scalar based implementations, power reduction through RA-based architectures is a strong ratio of computation to conversion (that is to say, the number of computations performed per invocation of the forward and reverse conversion). We formally address the problem of determining optimal ratios in a latter section.
- Use of residue datapaths has powerful implications for deep sub-micron (DSM) design. For one, it allows acceptable performance at low voltages. Secondly, the peak currents drawn in residue architectures are at-least 5-6 times lower than those in conventional architectures. Thirdly, splitting a monolithic datapath into channels increases locality -- an architectural feature coveted in DSM.

We now explore the low-power aspect of residue systems in detail. The main issues that must be considered are --

1. The speed, power and area tradeoffs in modular operations.
2. The area, performance and power penalties of the forward and reverse converter.
3. The power savings obtained by increasing the computations per conversion ratio.

The residue technique is chiefly applicable in arithmetic circuits where the adder is unarguably the work-horse. Hence we use a 32-bit adder to illustrate the low power methodology. We assume that a normal 32-bit Carry Propagate Adder is used in the conventional system. A simple three moduli set $\{2^{11}-1, 2^{11}, 2^{11}-1\}$ with corresponding modular adders for the three parallel channels is chosen for a residue implementation. All data provided below have been obtained through Timemill and Powermill simulations of back-annotated SPICE netlists. A 0.45µ, V_{dd}(nominal) = 3.3 V process is employed. All measurements are at 298 K.

2.1. Reducing power in the RA datapath

SPICE simulations of the conventional and residue datapaths yielded the following results --

	Predicted Delay	Measured Delay(@ 3.3V, 298 K)
32-bit adder	$32\ t_{FA}$	28.67 ns
Mod 2^{11} adder	$11\ t_{FA}$	8.4 ns
Mod 2^{11}+1 adder	$12\ t_{FA}$ + $t_{CORRECTION}$ + t_{MUX}	10.77 ns
Mod 2^{11}-1 adder	$11\ t_{FA}$ + $t_{CORRECTION}$ + t_{MUX}	9.83 ns

Table 1 : Datapath operator delays at nominal V_{dd}

All adders are driven by and are driving similar adders to correctly simulate a large system. The correspondance between the predicted and measured values is close. Also, of the three parallel channels, the mod 2^n+1 channel is the slowest. Hence, the speedup obtained by the use of residue arithmetic is 2.66 (=28.67/10.77).

Next, V_{dd} was decreased such that the critical path delay of the residue implementation matched the conventional one. To find such a V_{dd}, we swept the latter around a rough prediction we obtained using the following relation which is accurate for most sub- to deep-sub-micron technologies --

$$\text{Delay} \propto 1/(V_{dd}-V_t).$$

Operating Vdd	Delay (loaded with a mod 2^n-1 adder)
1.2	43.25
1.3	37.15
1.4	32.84
1.5	29.85
1.55	**28.65**
1.6	27.55
1.7	25.95

Table 2: Delay for a mod-2^n-1 adder over a range of supply voltages

This scaling of V_{dd}, contributes to a power reduction factor of roughly $(3.3/1.55)^2$ i.e. 4.532. The operating frequency remains the same since both data-paths (conventional and

residue) have the same delays. The other effect to be considered is the capacitance difference between the normal and residue implementations. Residue implementations incur an additional capacitance due to the mod-correction circuitry that must follow ordinary residue addition.

	Measured Av. Current (μA)	V_{dd} (V)	Measured Av. Power (μW)	Delay (ns)
32-bit adder	10.92	3.3	36.036	28.67
Mod 2^{11} adder	0.925	1.55	1.434	20.8
Mod 2^{11}-1 adder	3.47	1.55	5.379	28.65
Mod 2^{11}-1 adder	2.069	1.55	3.206	22.5

Table 3: Decrease in data-path power due to low V_{DD}

Tabulated above is the power consumption of the various adders at a frequency of 1 Mhz for worst-case (maximum) switching.

Power reduction factor (PRF)

Conventional adder power/Total RA datapath power

= 36.036/10.019 3.596

Note that for the datapath alone, we have achieved a 72% reduction in power at the expense of some area as we see above.

2.2 Performance/area penalties in RA datapath

Firstly, note that, by construction, there are no performance penalties when we reduce V_{dd}. The table below gives the increase in area when we change from a conventional to a RA datapath --

	Predicted Area	Measured Area (10^{-3} sq. mm.)
32-bit adder	32 A_{FA}	31.179
Mod 2^{11} adder	11 A_{FA}	9.949
Mod 2^{11}-1 adder	12 A_{FA} + $A_{CORRECTION}$ + A_{MUX}	20.206
Mod 2^{11}-1 adder	11 t_{FA} + $A_{CORRECTION}$ + A_{MUX}	18.12

Table 4: Area of residue and conventional data-paths

Adding the area for the three mod adders and comparing it to that of the 32-bit adder, we see that a residue datapath will be 54.8% costlier in terms of real estate.

2.3 Performance/area penalties in converters

A particularly thorny issue in RA based designs is the cost of forward and reverse conversion [2,4,8,12]. These converters, especially the reverse converter, are considered both "large" and "slow". The table below provides some area statistics of a 32-bit reverse converter for the $\{2^{11}-1, 2^{11}, 2^{11}+1\}$ set --

	Predicted Area	Measured Area (10^{-3} sq. mm.)
32-bit adder	32 A_{FA}	31.179
Reverse Converter	79 A_{FA} + A_{MUX}	96.892
Mod 2^{11}-1 forward conv.	25 A_{FA} + $A_{CORRECTION}$ + A_{MUX}	30.544
Mod 2^{11}-1 forward conv.	23 A_{FA} + $A_{CORRECTION}$ + A_{MUX}	26.758

Table 5: Area of forward and reverse converters

Note that no circuitry is needed to forward convert a number mod 2^n. We see from the table that the area cost due to conversion is usually insignificant for large signal processing systems. In conclusion, the area overheads mainly come from the residue datapath.

The other penalty is the conversion time penalty. Some statistics on this follow in the table below --

	Predicted Delay	Measured Delay (ns)
32-bit adder	32 t_{FA}	28.67
Reverse Converter	$13t_{FA}$ + t_{MUX} + $3t_{add}$	11.5
Mod 2^{11}-1 forward conv.	$12t_{FA}$ + $t_{CORRECTION}$	13.8
Mod 2^{11}-1 forward conv.	$11t_{FA}$ + $t_{CORRECTION}$	10.1

Table 6: Delay of forward and reverse converters

The forward conversion rate of 72.5 Mconversions/s is usually sufficient for most applications today. The reverse converter used here is extremely fast and can deliver nearly 90 Mconv/s. From a signal processing perspective, this rate is quite high when compared to the typical data bandwidth. Hence the reverse converter presented here is not on the critical path.

2.4 Peak current

The table below illustrates how residue implementations significantly lower the peak current consumption --

	Peak Current (mA)
32-bit adder	14.84
Mod 2^{11} adder	0.77
Mod 2^{11}+1 adder	2.57
Mod 2^{11}-1 adder	1.65

Table 7: Peak current of various data-paths

With the advent of deep submicron and intensified electromigration phenomenon, the decrease of peak current is a marked advantage.

2.5 Viable computation/conversion ratios

An important characteristic of residue based implementations is the number of computations per forward conversion - g_f and per reverse conversion g_r.

Consider a N 256-point FFT implementation. Here, we need $4N\log_2 N$ real multiplications and $2N\log_2 N$ real additions. Clearly,

$$g_r = 6N\log_2 N / N + 6\log_2 N = 6(8) \quad 48.$$

Similarly, g_f also comes out to be 48. The g-ratios strongly influence the power characteristics of a residue implementation. This is formally expressed thus -

$$P_{original} = N_c \cdot P_{data\text{-}path} / f_{compute}$$

where N_c number of computation units

$P_{data\text{-}path}$ Power/MHz of the original data-path.

$f_{compute}$ Operating frequency.

For low power residue systems as described above.

$$P_{new} = N_c P_{residue\text{-}path} / f_{compute} + N_r P_{rconv} / f_{rconv} + N_f P_{fconv} / f_{conv}$$

where N_r number of reverse converters

N_f number of forward converters

$P_{residue\text{-}path}$ = Power/MHz of the residue data-path.

P_{rconv}, P_{fconv} = Converter powers in W/MHz.

f_{rconv}, f_{fconv} Operating frequency of converters.

Hence, the power-gain.

$$P_{gain} = 1 - [P_{new}/P_{orig}]$$

$$= 1 - [1/P_{gain}(residue\text{-}path)] - [1/g_r][N_r/N_c][P_{rconv}/P_{data\text{-}path}] - [1/g_f][N_f/N_c][P_{fconv}/P_{data\text{-}path}]$$

For the case of the 256-point FFT

$$P_{gain} = 1 - 0.28 - [1/g_r][N_r/N_c][1.16] - [1/g_f][N_f/N_c][0.48].$$

$$1 - 0.28 - [1/48][1/48][1.16] - [1/48][1/48][0.48]$$

$$1 - 0.28 - 0.00069 - 0.000208$$

$$0.7191$$

Note that such gains were only possible due to the predominance of the residue datapath. Residue based applications with poor g_r and g_f may in fact end up using more power than conventional architectures.

3. SUMMARY

In this paper, a new methodology for residue arithmetic-based low power design has been developed. Using the parallel properties of residue arithmetic, power savings of over 72% have been shown for a mere 54% increase in data-path area. Unlike previously proposed super-scalar schemes, no modifications are needed in the scheduling strategy. Using residue arithmetic, locality is significantly enhanced while peak currents drop substantially. Both these are important virtues in deep-submicron. We have also analysed and presented the viable design space of low-power residue architectures by employing the concept of computations per conversion ratios. It is clear that the low power residue architectures are well suited to signal processing applications like the FFT, which possess high computation/conversion ratios.

4. REFERENCES

[1] A. Bellaouar and Mohammed I. El-Masry, "Low Power Digital VLSI Design," Kluwer Academic Publishers, June 1995.

[2] F. Barsi and M.C. Pinotti, "A Fully Parallel Algorithm for Residue to Binary Conversion," Information Processing Letters, vol. 50, pp. 1-8, 1994.

[3] M. A. Bayoumi, G. A. Julien and W. C. Miller, "A Look-up Table VLSI Design Methodology for RNS Structures used in DSP Applications," IEEE Trans. CAS, vol. CAS-34, pp. 604-615, June 1987.

[4] R. M. Capocelli and R. G. Carlo, "Efficient VLSI Networks for converting an integer from Binary System to Residue Number System and vice-versa," IEEE Trans. CAS, vol. 35, pp. 1425 - 30, Nov. 1988.

[5] A. P. Chandrakasan and R. W. Brodersen, "Low Power Digital CMOS Design," Kluwer Academic Publishers, June 1995.

[6] A. P. Chandrakasan, S.Sheng and R. W. Brodersen, "Low-Power CMOS Digital Design," IEEE Trans. On Solid-State Circuits,No. 4, pp. 473-483, April 1992.

[7] S. Oddvar, Y. Lundh and J.K. Hagene, "Very High Performance Signal Processing using the Residue Number System", *Proc. 6 MIT Conference - Advanced Research in VLSI*, ed. W. J. Dally, MIT Press, pp. 18-32, 1990.

[8] B. Premkumar, M. Bhardwaj and T. Srikanthan , "High Speed and Low Cost Converters for the (2n-1, 2n, 2n+1) Moduli Set," *Accepted for publication in IEEE Trans. CAS-II.*

[9] J. M. Rabaey and M. Pedram, "Low Power Design Methodologies," Kluwer Academic Publishers, Sept. 1995.

[10] K. Roy, S. Prasad and R. Roy, "Low-Power CMOS VLSI Design," John Wiley.

[11] T. Srikanthan, M. Bhardwaj and C.T. Clarke, "An Approach to Implementing Area-Time Efficient VLSI Residue to Binary Converters," *To be presented at the IEEE Intl. Conf. On Signal Processing Systems : Design and Implementation*, Leicester, 1997.

[12] N. S. Szabo and R. I. Tanaka, "Residue Arithmetic and its applications to Computer Technology", McGraw-Hill, 1967.