Multi-Voltage Low Power Convolvers Using the Polynomial Residue Number System

V. Paliouras
Electrical & Computer Engineering Dept.,
University of Patras, Greece
paliuras@ee.upatras.gr

A. Skavantzos
Electrical & Computer Engineering Dept.,
Louisiana State University, USA

T. Stouraitis
Electrical & Computer Engineering Dept.,
University of Patras, Greece

ABSTRACT

A novel approach for the reduction of the power dissipated in a signal processing application is introduced in this paper. By exploiting the properties of the Polynomial Residue Number System (PRNS) and of the arithmetic modulo (2^n+1), the power dissipation of implementing cyclic convolution is reduced up to four times. Furthermore, the corresponding power$\times$delay product is reduced up to 2.4 times, while a simultaneous reduction of area cost is achieved. The particular performance improvement becomes possible by introducing a way to minimize the forward and inverse conversion overhead associated with PRNS. The introduced minimization exploits the fact that for the conversions for particular lengths of data sequences and particular moduli, only multiplications with powers of two and additions are required, thus leading to low implementation complexity. In addition multiple supply voltages are utilized to further reduce power dissipation by more than 30% for particular cases. Formulas that return the applicable supply voltage values per PRNS channel are derived in this paper.

Categories and Subject Descriptors

B.7.1 [**Hardware**]: Types and Design Styles; B.5.1 [**Hardware**]: Design; C.3 [**Computer System Organization**] Special-purpose and application-based systems—*signal processing systems*

General Terms

Design

Keywords

Computer arithmetic, Polynomial Residue Number System (PRNS), Low power design, signal processing

1. INTRODUCTION

Recently low power consumption has emerged as a major design optimization objective due to the need for portable electronics equipment as well as a means for increasing the reliability in high-performance systems. Several design techniques have been proposed to minimize power dissipation, spanning all levels of the design abstraction [1], from system level down to the VLSI technology level. Among the various optimization techniques, optimal arithmetic selection has been shown to have an impact on overall system power dissipation [2]. In this paper, a new approach is proposed for the design of low-power digital signal processing equipment. By utilizing a number theoretic approach, it is shown that a significant reduction of power dissipation can be achieved, combined with a reduction of the system area cost, at the cost of increased delay. Further optimization is possible, by utilizing multiple supply voltages.

The Residue Number System (RNS) [3] is an integer system capable of supporting parallel, carry-free, high-speed arithmetic. The system also offers some useful properties for error detection, error correction and fault tolerance in digital systems. Important areas of application of the RNS include Digital Signal Processing (DSP) intensive computations, such as digital filtering, convolutions, correlations and DFT and FFT computations. Recent work in RNS arithmetic has resulted in the development of the Polynomial Residue Number System (PRNS) [4] which is capable of multiplying two polynomials using minimum computational complexity.

The PRNS examines the problem of multiplying two $(N-1)$-degree polynomials mod $(x^N \pm 1)$ in some modular ring $Z_m = \{0,1,\ldots,m-1\}$, a ring which is closed with respect to the operations of addition and multiplication mod m. This system can perform the above polynomial product using the minimal number of multiplications. It should be noted that the polynomial product of two $(N-1)$-degree polynomials mod (x^N-1) implements the cyclic convolution of two N-point sequences, a task which is useful in efficiently computing linear convolutions. It should also be noted that the linear convolution of two sequences is a very useful computation because it mechanizes digital filtering. Consider two N-point sequences $A = a_0, a_1, \ldots, a_{N-1}$ and $B = b_0, b_1, \ldots, b_{N-1}$.

Then their cyclic convolution is an N-point sequence $C = c_0, c_1, \ldots, c_{N-1}$, with c_i, $i = 0, 1, \ldots, N-1$ given by the coefficients of the polynomial $C(x)$, where $C(x) = \langle A(x)B(x)\rangle_{(x^N-1)}$ with $\langle P(x)\rangle_{Q(x)}$ denoting the operation $P(x)$ mod $Q(x)$ in polynomials, while $A(x) = \sum_{i=0}^{N-1} a_i x^i$, $B(x) = \sum_{i=0}^{N-1} b_i x^i$, and $C(x) = \sum_{i=0}^{N-1} c_i x^i$.

By factorizing the polynomials $x^N \pm 1$ in N distinct factors in Z_m as in

$$x^N \pm 1 = (x - r_0)(x - r_1)\ldots(x - r_{N-1}), \tag{1}$$

where $r_i \in Z_m$, $i = 0, 1, \ldots, N-1$, the product of two $(N-1)$-degree polynomials mod $(x^N \pm 1)$ in Z_m can be computed with only N multiplications instead of N^2. This is based on an isomorphic mapping f_N called the PRNS isomorphic mapping which translates polynomials of the form $A(x) = \sum_{i=0}^{N-1} a_i x^i$ into the PRNS domain. The PRNS isomorphic mapping is given by

$$\begin{aligned} A(x) &= a_0 + a_1 x + \ldots + a_{N-1}x^{N-1} \xrightarrow{f_N} A^*(x) = \\ &= (a_0^*, a_1^*, \ldots, a_{N-1}^*) \end{aligned} \tag{2}$$

with

$$\begin{aligned} a_i^* &= \left\langle \langle A(x)\rangle_{(x-r_i)} \right\rangle_m = \langle A(r_i)\rangle_m \\ &= \langle a_0 + a_1 r_i + \ldots + a_{N-1} r_i^{N-1}\rangle_m, \end{aligned} \tag{3}$$

$i = 0, 1, \ldots, N-1$ and r_i are the distinct roots of $x^N \pm 1 = 0$ in Z_m. In (3) $\langle x\rangle_m$ denotes the operation x mod m between integers. The inverse PRNS mapping f_N^{-1} is given by

$$\begin{aligned} A^*(x) &= (a_0^*, a_1^*, \ldots, a_{N-1}^*) \xrightarrow{f_N^{-1}} A(x) = \\ &= a_0 + a_1 x + \ldots + a_{N-1}x^{N-1}, \end{aligned} \tag{4}$$

where the coefficients of the polynomial $A(x)$, a_i, $i = 0, 1, \ldots, N-1$ are given by

$$a_i = \langle N^{-1}(a_0^* r_0^{-i} + a_1^* r_1^{-i} + \ldots + a_{N-1}^* r_{N-1}^{-i})\rangle_m, \tag{5}$$

$i = 0, 1, \ldots, N-1$. In (5) N^{-1} and r_j^{-i} are the multiplicative inverses of N and r_j^i in Z_m, respectively, which means $\langle N^{-1}N\rangle_m = 1$ and $\langle r_j^{-i} r_j^i\rangle_m = 1$. The main advantage of the PRNS is that it simplifies the rules of polynomial multiplication. While the rules of addition are unaffected, multiplication in the PRNS domain is performed as

$$\begin{aligned} &(a_0^*, a_1^*, \ldots, a_{N-1}^*)(b_0^*, b_1^*, \ldots, b_{N-1}^*) \\ &= (\langle a_0^* b_0^*\rangle_m, \langle a_1^* b_1^*\rangle_m, \ldots, \langle a_{N-1}^* b_{N-1}^*\rangle_m). \end{aligned} \tag{6}$$

Eq. (6) dictates that the product of two $(N-1)$-degree polynomials mod $(x^N \pm 1)$ requires N multiplications mod m if performed in the PRNS domain. The same task requires N^2 multiplications mod m if performed using the traditional technique; (non-PRNS).

Consider two polynomials $A(x) = \sum_{i=0}^{N-1} a_i x^i$ and $B(x) = \sum_{i=0}^{N-1} b_i x^i$ and consider performing $\langle A(x)B(x)\rangle_{x^N \pm 1}$ in the ring Z_m. Let C_{PRNS} and $C_{\text{non-PRNS}}$ denote the computational requirements for computing $\langle A(x)B(x)\rangle_{x^N \pm 1}$ in Z_m using the PRNS and the traditional technique respectively. Then C_{PRNS} and $C_{\text{non-PRNS}}$ are given by

$$\begin{aligned} C_{\text{PRNS}} &= (3N(N-1)+1)\text{scalings} + \\ &\quad 3N(N-1)\text{adds} + N\text{mults} \end{aligned} \tag{7}$$

$$C_{\text{non-PRNS}} = N^2\text{mults} + N(N-1)\text{adds}. \tag{8}$$

In (7) and (8) the scalings, additions and multiplications are operations mod m, while scaling means multiplication by constant.

Let r_i, $i = 0, 1, \ldots, N-1$ be the N distinct roots of $x^N \pm 1 = \langle 0\rangle_m$. If all the roots r_i, $i = 0, 1, \ldots, N-1$, their multiplicative inverses in Z_m r_i^{-1}, $i = 0, 1, \ldots, N-1$ and N^{-1} are all perfect powers of two, then all the scaling operations required by the forward and inverse PRNS mappings of (3) and (5) become multiplications by powers of two which can be implemented with simple shift operations simplifying this way the computational hardware. It can easily be shown that for several moduli of the form $m = 2^n + 1$ and several choices of N, all the N roots r_i, $i = 0, 1, \ldots, N-1$ of $x^N \pm 1 = \langle 0\rangle_{2^n+1}$, their multiplicative inverses in Z_{2^n+1} r_i^{-1}, $i = 0, 1, \ldots, N-1$ and $\langle N^{-1}\rangle_{2^n+1}$ are all perfect powers of two. If the diminished-1 system [5],[6] is used for performing arithmetic mod $2^n + 1$, then multiplications by powers of two can be implemented with leftwise rotations and complementation operations and this way very little computational hardware is required; (only the inverters responsible for complementing some bits of the number being rotated). More on diminished-1 arithmetic mod $(2^n + 1)$ will be offered in section 2 of the paper.

Lemma 1 *$x^N + 1$ can be factorized into N distinct factors in Z_m as $x^N + 1 = \langle (x - r_0)(x - r_1)\ldots(x - r_{N-1})\rangle_m$ if and only if $p_i = 2k_i N + 1$, $k_i = 1, 2, 3, \ldots$, $i = 1, 2, \ldots, L$ where N and m are positive integers with a prime decomposition of m given in terms of powers e_i of its prime factors p_i, as $m = p_1^{e_1} p_2^{e_2} \ldots p_L^{e_L}$ with $N < p_i$. Similarly for $x^N - 1$, the necessary and sufficient condition for its factorization becomes $p_i = k_i N + 1$, $k_i = 1, 2, 3, \ldots$. For both cases the factorization is not unique; there are $(N!)^{L-1}$ different ways to factorize $x^N \pm 1$ into N distinct first-degree factors [4].*

Lemma 2 *For both congruences $x^N \pm 1 = \langle 0\rangle_m$, the multiplicative inverses of their roots are also roots of the congruences, while the additive inverses are roots of the congruences only when N is even [4].*

Due to the fact that if N is even and r is a root of $x^N \pm 1 = \langle 0\rangle_m$ then $\langle -r\rangle_m$ is also a root, the number of scalings required for the forward PRNS mapping can be reduced to almost one half.

The remainder of the paper is organized as follows: In section 2 the basics of diminished-1 arithmetic are reviewed. In section 3 the area, time and power dissipation performance of a PRNS architecture that exploits the scalings by powers of two for the forward and inverse converters is quantified and compared to a non-PRNS architecture. The impact of multiple supply voltages on the PRNS convolver architecture is discussed in Section 4. Finally, conclusions are discussed in Section 5.

2. DIMINISHED-1 ARITHMETIC

Diminished-1 arithmetic has been proposed by Leibowitz [5] as an efficient means for performing arithmetic modulo 2^n+1. In diminished-1 arithmetic, the quantity $\langle x-1\rangle_{2^n+1}$ is used as an image of $x \in Z_{2^n+1}$. The particular mapping allows non-zero quantities to be represented using n bits, while zero is mapped onto the quantity 2^n, which requires $n+1$ bits for its representation.

When performing arithmetic mod (2^n+1) using the diminished-1 system, all input operands and the corresponding results are expressed in diminished-1 form.

By exploiting the diminished-1 representation, addition mod (2^n+1) is performed as an end-around carry operation, in two phases: An ordinary n-bit addition is performed, the carry out of which is negated and added back. Efficient parallel VLSI diminished-1 structures for modulo 2^n+1 two-operand addition have also been recently proposed [7].

Negation mod (2^n+1) is performed in diminished-1 system as follows: When $A \neq 0$ and $A \in Z_{2^n+1}$, $A \overset{\text{dim-1}}{\longrightarrow} A-1$. By taking the one's complement of $A-1$, the quantity $\langle -A\rangle_{2^n+1}$ in diminished-1 format is obtained.

In PRNS processing, scaling by powers of two is an important operation and it can be efficiently implemented in the diminished-1 system. In particular, $\langle 2^k A\rangle_{2^n+1}$ is computed as follows: The number $A-1$ is rotated k bits to the left, where the bits shifted out of the most significant end are complemented and shifted in the least significant end. The obtained result is expressed in diminished-1 form.

3. PERFORMANCE OF PRNS ARCHITECTURES

In the following the area, time and power dissipation performance of N-point cyclic convolution using the PRNS is quantified. The organization of the PRNS system is depicted in Fig. 1. The PRNS performance is compared to the performance of an architecture that employs conventional modular arithmetic. It is shown that the PRNS can substantially reduce both the area cost and the power dissipation of a system, even when the corresponding forward and inverse conversion overhead are taken into consideration.

The comparisons assume VLSI PRNS architectures that employ the modulo-(2^n+1) multiplier by Wang *et al.* [6], and carry-save $(3,2)$-counter Wallace trees for the N-operand additions mod (2^n+1), required by the forward and inverse converters. These structures employ diminished-1 arithmetic mod 2^n+1. The non-PRNS architectures employ the same multiplier and multi-operand adder structures for the direct computation of the cyclic convolution.

A novel observation is employed in this paper to simplify the converter design and hence reduce the PRNS implementation complexity. In particular, when certain moduli $m = 2^n+1$ are utilized for certain values of N, the roots of the polynomial x^N-1, and the multiplicative inverses of the roots and of N in Z_{2^n+1}

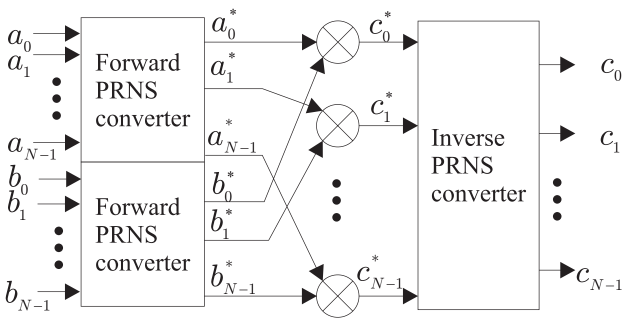
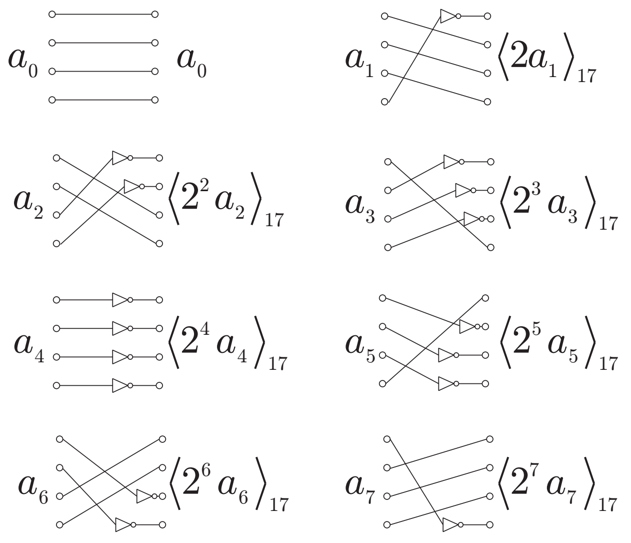


Figure 1: The organization of a PRNS system.



a_0
a_0
a_1
$\langle 2a_1\rangle_{17}$
a_2
$\langle 2^2 a_2\rangle_{17}$
a_3
$\langle 2^3 a_3\rangle_{17}$
a_4
$\langle 2^4 a_4\rangle_{17}$
a_5
$\langle 2^5 a_5\rangle_{17}$
a_6
$\langle 2^6 a_6\rangle_{17}$
a_7
$\langle 2^7 a_7\rangle_{17}$

Figure 2: Scalings in the forward PRNS converter for $N = 8$ and $m = 17$ for the computation of a_1^*, assuming diminished-1 representation.

are powers of two, either of positive or negative sign. This is demonstrated in Table 1 for several values of N and m. Hence, the scalings in (3) and (5) are reduced to scalings by powers of two, which can be efficiently implemented in diminished-1 arithmetic as rotations and bit-negation operations, with very low hardware complexity. This is demonstrated for $N = 8$ and $m = 2^4+1 = 17$ in Fig. 2, for a_1^*. The remainder of the a_i^* require scalings of similar implementation complexity.

The area, time and power dissipation performance of the PRNS-enhanced cyclic convolution architectures is summarized in Table 2, for various numbers of points N and several moduli of the form $m = 2^n+1$. The relative performance of a PRNS and a non-PRNS architecture are compared in terms of the ratios $\frac{A_{\text{non-PRNS}}}{A_{\text{PRNS}}}$, $\frac{T_{\text{non-PRNS}}}{T_{\text{PRNS}}}$, $\frac{P_{\text{non-PRNS}}}{P_{\text{PRNS}}}$, and $\frac{(PT)_{\text{non-PRNS}}}{(PT)_{\text{PRNS}}}$, where A_x, T_x, P_x, and PT_x denote the area, time, power dissipation and power$\times$delay product complexity of the architecture x, i.e., non-PRNS or PRNS. When a ratio assumes a value r larger than one, $r > 1$, then the performance of the PRNS system is r times better than the corresponding non-PRNS architecture; a value $r < 1$ denotes worse

N	m	$\{r_i, i = 0,1,\ldots,N-1\}$	$\{r_i^{-1}, i = 0,1,\ldots,N-1\}$	N^{-1}
8	2^4+1	$\{1,2,2^2,2^3,-2^3,-2^2,-2,-1\}$	$\{1,-2^3,-2^2,-2,2,2^2,2^3,-1\}$	-2
	2^8+1	$\{1,2^2,2^4,2^6,-2^6,-2^4,-2^2,-1\}$	$\{1,-2^6,-2^4,-2^2,2^2,2^4,2^6,-1\}$	-2^5
	$2^{16}+1$	$\{1,2^2,2^8,2^{12},-2^{12},-2^8,-2^4,-1\}$	$\{1,-2^{12},-2^8,-2^4,2^4,2^8,2^{12},-1\}$	-2^{13}
16	2^8+1	$\{1,2,2^2,2^3,2^4,2^5,2^6,2^7,-2^7,-2^6,$ $-2^5,-2^4,-2^3,-2^2,-2,-1\}$	$\{1,-2^7,-2^6,-2^5,-2^4,-2^3,-2^2,-2,$ $2,2^2,2^3,2^4,2^5,2^6,2^7,-1\}$	-2^4
	$2^{16}+1$	$\{1,2^2,2^4,2^6,2^8,2^{10},2^{12},2^{14},-2^{14},$ $-2^{12},-2^{10},-2^8,-2^6,-2^4,-2^2,-1\}$	$\{1,-2^{14},-2^{12},-2^{10},-2^8,-2^6,-2^4,-2^2,$ $2^2,2^4,2^6,2^8,2^{10},2^{12},2^{14},-1\}$	-2^{12}
32	$2^{16}+1$	$\{1,2,2^2,2^3,2^4,2^5,2^6,2^7,2^8,2^9,2^{10},2^{11},2^{12},2^{13},$ $2^{14},2^{15},-2^{15},-2^{14},-2^{13},-2^{12},-2^{11},-2^{10},$ $-2^9,-2^8,-2^7,-2^6,-2^5,-2^4,-2^3,-2^2,-2,-1\}$	$\{1,-2^{15},-2^{14},-2^{13},-2^{12},-2^{11},-2^{10},-2^9,-2^8,$ $-2^7,-2^6,-2^5,-2^4,-2^3,-2^2,-2,2,2^2,2^3,2^4,$ $2^5,2^6,2^7,2^8,2^9,2^{10},2^{11},2^{12},2^{13},2^{14},2^{15},-1\}$	-2^{11}

Table 1: Roots r_i of polynomials x^N-1 mod 2^n+1, and the multiplicative inverses r_i^{-1} and N^{-1} of the roots r_i and of N in Z_{2^n+1}. It can be seen that all the quantities are powers of two.

N	m	$\frac{A_{\text{non-PRNS}}}{A_{\text{PRNS}}}$	$\frac{T_{\text{non-PRNS}}}{T_{\text{PRNS}}}$	$\frac{P_{\text{non-PRNS}}}{P_{\text{PRNS}}}$	$\frac{(PT)_{\text{non-PRNS}}}{(PT)_{\text{PRNS}}}$
8	17	0.854	0.602	1.084	0.653
	257	1.481	0.567	1.820	1.032
	65537	2.462	0.556	2.912	1.619
16	257	1.671	0.559	2.136	1.194
	65537	2.991	0.553	3.717	2.055
32	65537	3.341	0.550	4.284	2.356

Table 2: Area, time, power dissipation, and power×delay performance of N-point cyclic convolution using PRNS and modulo 2^n+1 arithmetic, compared to a non-PRNS implementation, for single-modulus channel implementations.

performance of the PRNS system. The area, time and power dissipation performance of the cells that build the diminished-1 arithmetic circuits, for both PRNS and non-PRNS architectures, is obtained from a 0.7-μm CMOS library [8]. Table 2 reveals that the totally parallel implementation of the N-point cyclic convolution using PRNS, achieves significant area and power savings, at the cost of a higher delay, due to the forward and inverse conversion. As shown in the sixth column of Table 2, the power×delay product of the PRNS implementation can be up to 2.4 times better than the power×delay product of the traditional implementation, only at a fraction of the area. As the number N of points grows larger, PRNS area and power dissipation savings increase, over the corresponding performance of the non-PRNS system. The particular behavior of the experimental results is consistent with the computational complexities given by (7) and (8).

4. MULTI-VOLTAGE CONSIDERATIONS

In case of a PRNS system which includes several moduli, further optimization is possible, by using a different supply voltage for each modulo channel. The capacitance along the critical path of the employed modulo $m = 2^n+1$ diminished-one multiplier is

$$C_{\text{crit}}(m) = h(\log_2(m-1))C_{\text{FA}} + \log_2(m-1)C_{\text{FA}} + C_{\text{mux}}, \tag{9}$$

where C_{FA} is the capacitance of an 1-bit full adder, C_{mux} is the capacitance of an 1-bit two-input multiplexer, and $h(L)$ returns the height of a Wallace tree that adds L operands and it is recursively computed using [9]:

$$h(L) = 1 + h\left(\left\lceil \frac{2L}{3} \right\rceil\right) \tag{10}$$

$$h(3) = 1. \tag{11}$$

The critical path capacitance can be exploited to derive the delay along the maximum delay path of the particular multiplier architecture can be approximated by (cf. [10]):

$$T_{\text{crit}}(m) = \frac{C_{\text{crit}}(m)V}{k(V-V_{\text{th}})^2}, \tag{12}$$

where V is the supply voltage, k depends on implementation technology parameters, and V_{th} is the device threshold voltage. Eq. (12) implies that the delay of a particular modulo-m multiplier depends on m and the supply voltage V. The different delays of the various modulo channels can be balanced by properly selecting the supply voltages of each channel. The utilization of multiple supply voltages in RNS FIR filters has been proposed by Del Re *et al.* [11]. In this paper, we study the application of multiple supply voltages to PRNS architectures, and derive models and formulas that return the supply voltage value per channel. Therefore, for a PRNS system employing three moduli channels 2^4+1, 2^8+1, and $2^{16}+1$, the supply voltage for each channel can be computed by posing the requirement that the channels that correspond to smaller moduli demonstrate equal delay time to the larger moduli channels, i.e.,

$$T_{\text{crit}}(2^{16}+1) = T_{\text{crit}}(2^4+1) \tag{13}$$

$$T_{\text{crit}}(2^{16}+1) = T_{\text{crit}}(2^8+1). \tag{14}$$

Let V denote the supply voltage of the modulo $2^{16}+1$, and $V_{17} = \beta_{17}V$ and $V_{257} = \beta_{257}V$ denote the supply voltages for the channels mod 2^4+1 and mod 2^8+1. By combining (9)–(14), equations can be formed the solution of which allows the computation of the supply voltage reduction factors β_{17} and β_{257}:

$$\frac{V\,(23C_{\mathrm{FA}}+C_{\mathrm{mux}})}{k\,(V-V_{\mathrm{th}})^2}-\frac{V\beta_{17}\,(7C_{\mathrm{FA}}+C_{\mathrm{mux}})}{k\,(V\beta_{17}-V_{\mathrm{th}})^2}=0 \tag{15}$$

$$\frac{V\,(23C_{\mathrm{FA}}+C_{\mathrm{mux}})}{k\,(V-V_{\mathrm{th}})^2}-\frac{V\beta_{257}\,(13C_{\mathrm{FA}}+C_{\mathrm{mux}})}{k\,(V\beta_{257}-V_{\mathrm{th}})^2}=0. \tag{16}$$

The solution of (15) and (16) returns

$$\beta_{17}=\frac{1}{2V^2\,(23C_{\mathrm{FA}}+C_{\mathrm{mux}})}\Big(C_{\mathrm{mux}}\left(V^2+V_{\mathrm{th}}^2\right)+ C_{\mathrm{FA}}\left(7V^2+32VV_{\mathrm{th}}+7V_{\mathrm{th}}^2\right)\pm|V-V_{\mathrm{th}}|\sqrt{7C_{\mathrm{FA}}+C_{\mathrm{mux}}} \sqrt{C_{\mathrm{mux}}(V+V_{\mathrm{th}})^2+C_{\mathrm{FA}}\left(7V^2+78VV_{\mathrm{th}}+7V_{\mathrm{th}}^2\right)}\Big) \tag{17}$$

$$\beta_{257}=\frac{1}{2V^2\,(23C_{\mathrm{FA}}+C_{\mathrm{mux}})}\Big(C_{\mathrm{mux}}\left(V^2+V_{\mathrm{th}}^2\right)+ C_{\mathrm{FA}}\left(13V^2+20VV_{\mathrm{th}}+13V_{\mathrm{th}}^2\right)\pm|V-V_{\mathrm{th}}|\sqrt{13C_{\mathrm{FA}}+C_{\mathrm{mux}}} \sqrt{C_{\mathrm{mux}}(V+V_{\mathrm{th}})^2+C_{\mathrm{FA}}\left(13V^2+66VV_{\mathrm{th}}+13V_{\mathrm{th}}^2\right)}\Big). \tag{18}$$

From the two values obtained for each of b_{17} and b_{257}, the one that leads to $V_{17}, V_{257} < V_{\mathrm{th}}$ is not legitimate [10]. In order to quantify the voltage reduction factor, capacitance values taken from a 0.7-μm CMOS standard-cell library [8] are utilized as follows: Assuming $C_{\mathrm{FA}}=0.054$pF $C_{\mathrm{mux}}=0.067$pF, $V_{\mathrm{th}}=0.6$V and $V=5$V, it is obtained that $\beta_{17}=0.472$ and $\beta_{257}=0.674$. Therefore, without affecting the overall system delay, the mod 17 and mod 257 residue channels can operate at supply voltages $V_{17}=2.36$V and $V_{257}=3.37$V. The particular supply voltage reduction directly reduces the overall power dissipation of the system. Assuming a Wallace-tree based implementation of a binary multiplier, the power dissipated by an 8-point circular convolution by means of a three-modulus PRNS system which offers a dynamic range of 28 bits, is reduced by 30%. It is noted that the particular performance improvement does not affect the latency of the system. Furthermore, when compared to a full-parallel conventional binary (non-RNS) implementation of the the 8-point convolver, a five-times reduction in power is anticipated.

5. CONCLUSIONS

In this paper it has been shown that by properly selecting the modulus of operation, the conversion complexity inherent in a PRNS-based architecture can be reduced to scalings with powers of two and additions. This substantial reduction leads to significant power$\times$delay product and area savings, in comparison to a traditional architecture for the implementation of N-point cyclic convolution. In addition, in a multi-modulus PRNS VLSI architecture, the different delays displayed by the residue channels can be exploited to further reduce power dissipation. This is achieved by reducing the supply voltage of the smaller residue channels, as dictated by (17) and (18).

6. REFERENCES

[1] A. P. Chandrakasan, S. Sheng, and R.W. Brodersen, “Low-power CMOS digital design,” *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, Apr. 1992.

[2] T. Stouraitis and V. Paliouras, “Considering the alternatives in low-power design,” *IEEE Circuits and Devices*, vol. 17, no. 4, pp. 23 – 29, July 2001.

[3] M. A. Soderstrand, W. K. Jenkins, G. A. Jullien, and Fred J. Taylor, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*, IEEE Press, 1986.

[4] A. Skavantzos and F. J. Taylor, “On the polynomial residue number system,” *IEEE Transactions on Signal Processing*, vol. 39, no. 2, pp. 376–382, Feb. 1991.

[5] L. M. Leibowitz, “A simplified binary arithmetic for the Fermat number transform,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-24, no. 5, pp. 356 – 359, Oct. 1976.

[6] Zhongde Wang, G. A. Jullien, and W. C. Miller, “An algorithm for multiplication modulo (2^n+1),” in *Proceedings of 29th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, 1996, pp. 956 – 960.

[7] H. T. Vergos, C. Efstathiou, and D. Nikolos, “High speed parallel-prefix modulo 2^n+1 adders for diminished-one operands,” in *Proceedings of 15th IEEE Symposium on Computer Arithmetic*, Vail, CO, 11-13 June 2001, pp. 211–217.

[8] ES2, *ES2 ECPD07 library databook*, European Silicon Structures, Rousset, France, 1992.

[9] Behrooz Parhami, *Computer Arithmetic - Algorithms and Hardware Designs*, Oxford University Press, New York, 2000.

[10] Keshab Parhi, *VLSI Digital Signal Processing Systems,* Wiley, 1999.

[11] Andrea Del Re, Alberto Nanarelli, and Marco Re, “Implementation of carry-save residue number system,” in *Proceedings of the Asilomar Conference on Circuits, Systems and Computers*, 2001.