A New Unified Modular Adder/Subtractor for Arbitrary Moduli

Tay, Thian Fatt; Chang, Chip-Hong

2015

Tay, T. F., & Chang, C.-H. (2015). A New Unified Modular Adder/Subtractor for Arbitrary Moduli. 2015 IEEE International Symposium on Circuits and Systems (ISCAS), 53-56.

https://hdl.handle.net/10356/80988

https://doi.org/10.1109/ISCAS.2015.7168568

A New Unified Modular Adder/Subtractor for Arbitrary Moduli

Thian Fatt Tay and Chip-Hong Chang
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

Abstract—**Efficient modular adders and subtractors for arbitrary moduli are key booster of computational speed for high-cardinality Residue Number Systems as they rely on arbitrary moduli set to expand the dynamic range. This paper proposes a new unified modular adder/subtractor that possesses a regular structure for any modulus. Compared to the latest modular adder/subtractor, which works for modulus in the forms of $2^n \pm k$, the proposed design is on average 10.81% faster and consumes 15.85% less hardware area and 2.51% lower power for n ranging from 4 to 8.**

I. Introduction

Residue Number System (RNS) has become a promising alternative number system for digital system implementation in recent years. The key success for RNS-based computations is its carry-free additions and subtractions in residue domain. Besides, RNS-based computations are also inherently fault-tolerant. Attempts to leverage RNS for the acceleration of fast Fourier transform (FFT) and discrete cosine transform (DFT) computations have been made [1]-[3]. The design in [1] implements modular subtraction by subtracting the subtrahend from the corresponding modulus followed by the modular addition. This method avoids the use of fused adder/subtractor but requires additional constant subtraction, which impacts the speed. The data paths of designs [2] and [3] are heavily occupied by additions and subtractions. The overall speed of the system is thus predominated by how well these modular adders and subtractors are optimized.

Modular adder and subtractor for arbitrary modulus are generally more complex than their standard binary counterparts, which lead to the long-standing interest in the moduli set $\{2^n-1, 2^n, 2^n+1\}$. Due to the end-around and complementary end-around carry properties, hardware implementations of modular $2^n \pm 1$ addition and subtraction can be made as efficient as their binary counterparts. The problem with this special moduli set is its limited dynamic range or parallelism. In order to expand the dynamic range of the RNS with minimal negative impact on the arithmetic speed, extra coprime moduli of comparable word-length have to be added. This makes obtaining balanced high-cardinality moduli set without considering other forms of moduli impossible. It becomes apparent that a uniform and efficient implementation of modular adder and subtractor for arbitrary moduli are very important for large dynamic range applications. As modular addition and subtraction may not be used concurrently, it will be beneficial to unify them into a single functional unit with an implementation cost lower than the combined cost of a modular adder and a modular subtractor.

Several unified modular adders/subtractors have been reported [4]-[6]. The unified modular adder/subtractor in [4] is designed for arbitrary moduli. It is constructed using decoders, modified barrel shifters, and read-only-memory (ROM). The rotate logic consumes enormous hardware area, which is highly inefficient for large moduli. The main contribution of [5] is the proposal of parallel prefix diminished-1 modulo 2^n+1 subtractor that can handle zero operand. The proposed subtractor can be easily transformed into a unified adder/subtractor with additional logic gates. It has simple and regular structure but is dedicated to modulo 2^n+1 operations only. The state-of-the-art unified modular adder/subtractor [6] is designed for moduli in the forms of $2^n \pm k$. The modulo 2^n-k adder/subtractor has exactly the same architecture as that of modulo 2^n+k operator, consisting of multiple carry save adders (CSAs), multiplexers (MUXs) and binary adders, except for the correction factors. Its overall speed is limited by the multi-level carry save additions.

In this paper, a new unified modular adder/subtractor for arbitrary moduli is proposed. The proposed modular adder/subtractor can be easily modified to implement modulo addition/subtraction for different moduli with the same n by changing the input configurations of MUXs. Owing to its simplicity, the proposed design is faster than the less versatile state-of-the-art design [6] for all moduli of n = 4 to 8.

II. Proposed Unified Modular Adder/Subtractor

A. Background

In an RNS formed by N coprime integers $\{m_1, m_2, ..., m_N\}$, an integer X can be represented by using an N-tuple $(x_1, x_2, \ldots, x_N)$, where m_i and x_i are known as modulus and residue digit, respectively. x_i is computed by finding the least non-negative remainder of X divided by m_i ($x_i = |X|_{m_i}$). Let Z be the result of an arithmetic operation acted upon integers, X and Y. Then

$$(z_1, z_2, \ldots, z_N) = \left(|x_1 \Diamond y_1|_{m_1}, |x_2 \Diamond y_2|_{m_2}, \ldots, |x_N \Diamond y_N|_{m_N} \right) \quad (1)$$

where $\Diamond$ denotes addition, subtraction or multiplication.

B. Algorithm Formulation

Let m be an arbitrary positive integer in the range $[2^{n-1}+1, 2^n-1]$, where $n = \lceil \log_2 m \rceil$ and $\lceil . \rceil$ is the greatest integer function. The subtraction of y from x modulo m can be formulated as:

$$|x-y|_m = \begin{cases} x-y+m & \text{if } 1-m \le x-y < 0 \\ x-y & \text{if } 0 \le x-y \le m-1 \end{cases} \quad (2)$$

where x and y are integers in the range $[0, m-1]$.

The valid range of any modulo m operation has to be [0, m–1]. Upon detecting that $x - y$ is less than 0, m is added to map the result into the range [0, m–1]. Equation (2) can thus be transformed into the modular addition of three integers as follows:

$$\begin{aligned} |x-y|_m &= \left|x+\left(2^n-1-y\right)-\left(2^n-1\right)\right|_m \\ &= \left|x+\overline{y}+\left(2m-2^n+1\right)\right|_m \\ &= \left|x+\overline{y}+c\right|_m \end{aligned} \tag{3}$$

where $\overline{y}$ is the one's complement of y and $c = 2m–2^n+1$ is a constant.

Since $x+\overline{y}+c \in [m+1, 3m-1]$, (2) can be rewritten as:

$$|x-y|_m = \begin{cases} x+\overline{y}+c-2m & \text{if } 2m \le x+\overline{y}+c < 3m \\ x+\overline{y}+c-m & \text{if } m+1 \le x+\overline{y}+c < 2m \end{cases} \tag{4}$$

The detection of the range of $x+\overline{y}+c$ in (4) is difficult to implement on hardware. It involves the comparisons of multiple bits depending on the word-length of m.

Lemma 1: $|x-y|_m$ can be computed by:

$$|x-y|_m = \begin{cases} \left|x+\overline{y}+1\right|_{2^n} & \text{if } x+\overline{y}+2^n+1 \ge 2^{n+1} \\ \left|x+\overline{y}+m+1\right|_{2^n} & \text{if } 2^n < x+\overline{y}+m+1 < 2^n+m \end{cases} \tag{5}$$

Proof:

When $x+\overline{y}+2^n+1 \ge 2^{n+1}$,

$$\begin{aligned} \left|x+\overline{y}+1\right|_{2^n} &= \left|x+\overline{y}+2^n+1\right|_{2^n} \\ &= x+\overline{y}+2^n+1-2\times 2^n \\ &= x+\overline{y}+c+\left(1-2^n-2m+2^n-1\right) \\ &= x+\overline{y}+c-2m \end{aligned} \tag{6}$$

When $2^n \le x+\overline{y}+m+1 < 2^{n+1}$,

$$\begin{aligned} \left|x+\overline{y}+m+1\right|_{2^n} &= x+\overline{y}+m+1-2^n \\ &= x+\overline{y}+c+\left(m+1-2^n-2m+2^n-1\right) \\ &= x+\overline{y}+c-m \end{aligned} \tag{7}$$

Hence, the ranges for (4) can be substituted by those of (5). The range detection circuit can be further simplified by noticing that when $x+\overline{y}+2^n+1 \ge 2^{n+1}$, $x+\overline{y}+m+1$ will never fall into the range (2^n, 2^n+m), which can be proved by

$$\begin{aligned} x+\overline{y}+2^n+1 &\ge 2^{n+1} \\ x+\overline{y} &\ge 2^n-1 \\ x+\overline{y}+m+1 &\ge 2^n+m \end{aligned} \tag{8}$$

Therefore, (5) can be simplified into:

$$|x-y|_m = \begin{cases} \left|x+\overline{y}+1\right|_{2^n} & \text{if } x+\overline{y}+2^n+1 \ge 2^{n+1} \\ \left|x+\overline{y}+m+1\right|_{2^n} & \text{otherwise.} \end{cases} \tag{9}$$

The conditions of (9) can be detected by the most significant bit (MSB) of $\left(x+\overline{y}+2^n+1\right)$, which can be easily implemented on hardware.

On the other hand, the modulo m addition of x and y can be formulated by [7]:

$$|x+y|_m = \begin{cases} \left|x+y+2^n-m\right|_{2^n} & \text{if } x+y+2^n-m \ge 2^n \\ x+y & \text{othersiwe.} \end{cases} \tag{10}$$

When $x + y + 2^n - m < 2^n$, $|x+y|_m = |x+y|_{2^n}$ is always true. Thus, (10) can be rewritten as:

$$|x+y|_m = \begin{cases} |x+y|_{2^n} & \text{if } x+y+\overline{m}+1 < 2^n \\ \left|x+y+\overline{m}+1\right|_{2^n} & \text{othersiwe.} \end{cases} \tag{11}$$

where $\overline{m}+1 = 2^n - m$.

The conditions of (11) can be easily detected by the MSB of $x+y+\overline{m}+1$.

Comparing (9) and (11), it is observed that the computations of $|x-y|_m$ and $|x+y|_m$ share some common terms. By introducing a binary variable s, the modular addition and subtraction can be unified into:

$$\left|x+(-1)^s y\right|_m = \begin{cases} |w|_{2^n} & \text{if } s = \text{'0'} \text{ and } v < 2^n \\ & \text{or } s = \text{'1'} \text{ and } w \ge 2^{n+1} \\ |v|_{2^n} & \text{otherwise.} \end{cases} \tag{12}$$

where $w = x+(y \oplus s)+\left(2^n \oplus s\right)+s$ and $v = x+(y \oplus s)+(\overline{m} \oplus s)+1$. Modular addition is performed by setting s = '0' and modular subtraction by s = '1'. The operation '$a \oplus s$' performs a bitwise exclusive-or of every bit of a with s.

Example 1: Consider m = 11. $n = \lceil \log_2 m \rceil = 4$. Let x = 8 and y = 5. If s = '0', then w = 8 + 5 + 16 = 29 and v = 8 + 5 + 4 + 1 = 18. From (12), since s = '0' and $v > 2^n = 16$, $|v|_{2^n} = |18|_{16} = 2$ is computed, which is the same as $|x+y|_m = |8 + 5|_{11} = 2$. If s = '1', then w = 8 + 10 + 16 + 1 = 35 and v = 8 + 10 + 11 + 1 = 30. Since s = '1' and $w > 2^{n+1} = 32$, $|w|_{2^n} = |35|_{16} = 3$ is selected, which is equal to $|x-y|_m = |8 – 5|_{11} = 3$.

C. Circuit Architecture

Fig. 1 depicts the computations of w and v from (12) for m = 11 and n = 4. The terms in dotted-line boxes are used only for the detection of the conditions of $v < 2^n$ and $w \ge 2^{n+1}$. They are not involved in the addition operations for $|w|_{2^n}$ and $|v|_{2^n}$. The computations of w and v consist of two levels of additions. The first level involves the additions of the first three terms, i.e., $x+(y \oplus s)+\left(2^n \oplus s\right)$ and $x+(y \oplus s)+(\overline{m} \oplus s)$, for the computation of w and v, respectively. Since one of the terms is a constant, this first level of additions can be implemented using half-adder-like (HAL) cells proposed in [8]. The implementation is shown in Stage 1 of Fig. 2, where a_i^j and b_{i+1}^j are the sum and carry terms, respectively and j is the constant bit at position i.

The second level of addition in Fig. 1 involves a_i^j for $0 \le i \le n$–1, b_i^j for $1 \le i \le n$–1, and the fourth term in w and v computations, which is s for w computation and 1 for v computation. It can be implemented using two parallel prefix adders (PPAs) as shown in Stage 2 of Fig. 2. The PPA for v computation requires additional 2n units of 2-to-1 input MUXs because the sum and carry terms to be added when s = '0' are different from those when s = '1', as shown in Fig. 1.

The final stage of the unified modular adder/subtractor selects either $|w|_{2^n}$ or $|v|_{2^n}$ as output. The conditions in (12)

can be implemented by simple logic gates depicted in Stage 3 of Fig. 2. The unified modular adder/subtractor architecture has a regular structure. It can implement modular addition or subtraction for any m in the range $[2^3+1, 2^4-1]$ by changing the input configurations of the MUXs in Stage 2.

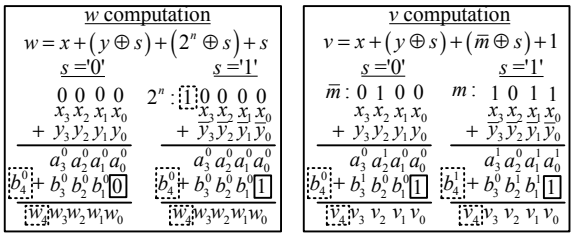

Fig. 1: Computation of w and v for m=11 and n=4.

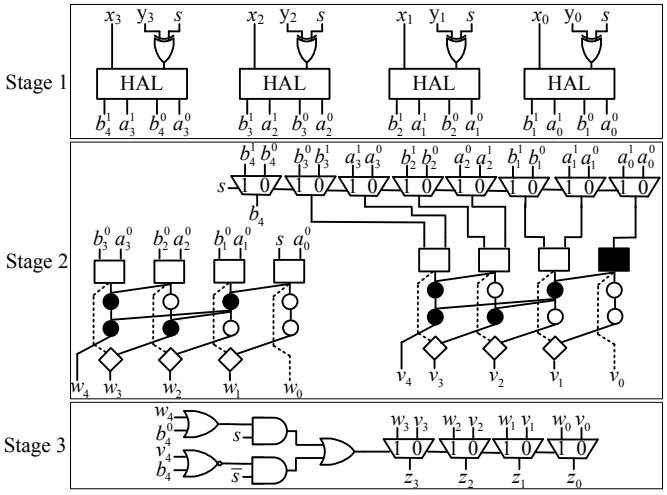

Fig. 2: The proposed unified modular adder/subtractor for m=11 and n=4.

III. Evluation and Comparison

In this section, the proposed unified modular adder/subtractor depicted in Fig. 2 is evaluated and compared against the latest design [6]. The designs are first analyzed using unit-gate model [9] before they are synthesized and optimized. Two designs are proposed in [6], which are *Adder/subtractor I* and *Adder/subtractor II*. The former can be implemented for moduli in the forms of $2^n - k$ and $2^n + k$ but the latter is limited to only moduli of form $2^n + k$ for odd k and has no speed and area advantage when n is less than 12 according to the synthesis result in Fig. 6 of [6]. Therefore, the *adder/subtractor I* for moduli $2^n - k$ is implemented for comparison. The performances of the proposed design and *adder/subtractor I* are analyzed with n = 4, 5, 6, 7, and 8. For each n, three moduli in the forms of $2^{n-1} + 3$, $2^n - 2^{n-2}$ and $2^n - 3$ are chosen.

A. Unit-gate Analysis

The unit-gate analysis is performed based on the model [9], where a two-input monotonic logic gate, such as *AND*, *OR*, *NAND* and *NOR*, is considered to have one unit of area and one unit of delay; Both *XOR* gate and MUX have two units of area and two units of delay; The area and delay of an inverter are assumed to be negligible [9]. In addition, the area and delay of a full adder are counted as seven units and four units, respectively [10].

Each HAL cell in Stage 1 of the proposed unified adder/subtractor has four units of area and two units of delay each [8]. Since Stage 1 consists of n HAL cells and n *XOR* gates, its total area and delay are $6n$ units and 4 units, respectively. In Stage 2, two PPAs are required for w and v computations. Since there is only one input at the least significant bit (LSB) position, the propagate, generate and half-sum generation of PPA for v computation can be simplified. Thus, the total area and delay of Stage 2 inclusive of the $2n$ MUXs are $3n\lceil \log_2 n \rceil + 16n - 8$ units and $2\lceil \log_2 n \rceil + 5$ units, respectively. Stage 3 consists of two *OR* gates, two *AND* gates, one *NOR* gates and n MUXs. Thus, Stage 3 has $2n$+5 units of area and 3 units of delay. All in all, the proposed unified modular adder/subtractor has an overall area of $3n\lceil \log_2 n \rceil + 24n - 3$ units and a delay of $2\lceil \log_2 n \rceil + 12$ units as shown in Table I.

Table I: Unit-Gate Area (A) and Delay (D) of the Proposed Design.

	Stage 1		Stage 2			Stage 3		Total
	XOR	HAL	MUX	PPA (for w)	PPA(for v)	gates	MUX	
A	$2n$	$4n$	$4n$	$1.5n\lceil \log_2 n \rceil + 6n - 2$	$1.5n\lceil \log_2 n \rceil + 6n - 6$	5	$2n$	$3n\lceil \log_2 n \rceil + 24n - 3$
D	2	2	2	$2\lceil \log_2 n \rceil + 3$		3		$2\lceil \log_2 n \rceil + 12$

Table II shows the unit-gate area and delay of *adder/subtractor I* calculated based on the architectures depicted in Fig. 1 and Fig. 2(b) of [6]. As suggested in [6], the binary adder in add 3:1 can either be implemented using PPA or carry propagate adder (CPA). The PPA in *adder/subtractor I* is assumed to have the same unit gate area and delay as our proposed design. From Table II, the total area and delay of the PPA model are $3n\lceil \log_2 n \rceil + 41n + 7$ units and $2\lceil \log_2 n \rceil + 24$ units, respectively, whereas the total area and delay of the CPA model are $45n + 4$ units and $4n - 19$ units, respectively.

By comparing Tables I and II, the proposed design is $17n + 4$ units and $21n - 3n\lceil \log_2 n \rceil + 7$ units smaller than the PPA and CPA implementations of *adder/subtractor I*, respectively, and 12 units and $4n - 2\lceil \log_2 n \rceil + 7$ units faster than the PPA and CPA versions of *adder/subtractor I*, respectively .

Table II: Unit-Gate Area (A) and Delay (D) of *Adder/Subtractor I* [6]

	MUX	*XOR*	add 3:1					Total
			CSA	Logic A	MUX	Adder	Logic B	
A (PPA)	$2n$	$2n$	$21n$	2	$6n$	$3n\lceil \log_2 n \rceil + 12n - 4$	10	$3n\lceil \log_2 n \rceil + 41n + 7$
A (CPA)						$14n - 8$		$45n + 4$
D (PPA)	2		8	2	4	$2\lceil \log_2 n \rceil + 3$	5	$2\lceil \log_2 n \rceil + 24$
D (CPA)						$4n - 2$		$4n + 19$

B. Logic Synthesis Results

The proposed design and the PPA and CPA versions of [6] are specified in Verilog HDL at gate level, functionally verified using ModelSim, and synthesized and technology mapped to STM 65nm CMOS technology standard cell library

using Synopsys Design Compiler. Each design is independently and recursively optimized for minimum achievable delay.

The synthesized areas and delays of the proposed design, [6] (PPA), and [6] (CPA) are tabulated in Table III. Compared with [6] (PPA), the proposed design is at least 14.48%, 16.12%, 13.89%, 13.18% and 20.74% smaller for n = 4, 5, 6, 7 and 8, respectively. Compared with [6] (CPA) with moduli in the forms of $2^{n-1}+3$ and 2^n-3, the proposed design is at least 15.08%, 9.85%, 9.32%, 6.04% and 0.37% smaller for n = 4, 5, 6, 7 and 8, respectively. For moduli in the form of 2^n-2^{n-2}, according to (9) of [6], the constants k_1 and k_2 are in the form of 2^l, where l is an integer less than n. Since k_1 and k_2 are fed as one of the three inputs, the structures of the CSAs and MUXs in add 3:1 can be easily optimized to achieve smaller area. Furthermore, CPAs generally consume smaller hardware area compared to the PPAs implemented in our proposed design. These explain the smaller area of [6] (CPA) than our proposed design for moduli 2^n-2^{n-2} when n = 6, 7 and 8. In term of delay, our proposed design is at least 10.96%, 6.76%, 7.79%, 7.69% and 6.33% faster than [6] (PPA) for n = 4, 5, 6, 7 and 8, respectively. The speed improvements of the proposed design over [6] (CPA) are more than 7.25%, 5.56%, 7.89%, 8.86% and 9.76% for n = 4, 5, 6, 7 and 8, respectively.

Table III: Comparison of Synthesized Areas and Delays (Value in Parenthesis Indicates the Percentage Reduction of Proposed Design Over the Contenders)

n	m	Synthesized Area (μm^2)			Synthesized Delay (ns)		
		This	[6] (PPA)	[6] (CPA)	This	[6] (PPA)	[6] (CPA)
4	11	565	874(35.36)	666(15.08)	0.64	0.73(12.33)	0.74(13.51)
	12	614	718(14.48)	651(5.69)	0.64	0.73(12.33)	0.69(7.25)
	13	541	788(31.29)	652(16.98)	0.65	0.73(10.96)	0.72(9.72)
5	19	708	1067(33.67)	830(14.78)	0.68	0.75(9.33)	0.78(12.82)
	24	758	903(16.12)	778(2.61)	0.68	0.76(10.53)	0.72(5.56)
	29	752	1059(29.01)	834(9.85)	0.69	0.74(6.76)	0.76(9.21)
6	35	881	1251(29.52)	1024(13.96)	0.70	0.77(9.09)	0.82(14.63)
	48	931	1082(13.89)	876(–6.29)	0.70	0.77(9.09)	0.76(7.89)
	61	876	1283(31.74)	970(9.32)	0.71	0.77(7.79)	0.82(13.41)
7	67	1020	1465(30.41)	1085(6.04)	0.72	0.79(8.86)	0.88(18.18)
	96	1055	1215(13.18)	1037(–1.65)	0.72	0.78(7.69)	0.79(8.86)
	125	1037	1470(29.47)	1141(9.16)	0.72	0.80(10.00)	0.86(16.28)
8	131	1268	1600(20.74)	1272(0.37)	0.73	0.80(8.75)	0.91(19.78)
	192	1149	1521(24.48)	1102(–4.25)	0.74	0.79(6.33)	0.82(9.76)
	253	1192	1632(26.98)	1231(3.17)	0.73	0.80(8.75)	0.90(18.89)

Table IV shows the total power dissipation and leakage power of the proposed design and the contenders. The power consumption are measured using Synopsys PrimeTime PX at the same supply voltage of 1V and the same clock rate determined by the slowest design of Table III for each m. Monte Carlo simulation method [11] is adopted where a finite number of randomly generated test patterns are inserted to estimate the average power dissipation with 99% confidence that the error is bounded below 3%. From Table IV, the total power consumptions of the proposed design are more than those of [6] (PPA) and [6] (CPA) with moduli 2^n-2^{n-2} due mainly to the more favorable logic optimization of [6] (PPA) and [6] (CPA) architectures explained earlier. Nonetheless, for moduli $2^{n-1}+3$ and 2^n-3, the proposed design consumes less power compared to the contenders for all n. In term of leakage power, the proposed design consume less power compared to [6] (PPA) for all m except for m = 12. Compared with [6] (CPA), the leakage power of the proposed design is lower for all m except for m = 12, 48 and 192.

Table IV: Comparison of Total and Leakage Power (Value in Parenthesis Indicates the Percentage Reduction of Proposed Design Over the Contenders)

n	m	Total Power (μw)			Leakage Power (μw)		
		This	[6] (PPA)	[6] CPA	This	[6] (PPA)	[6] (CPA)
4	11	847	1244(31.89)	940(9.91)	32.15	52.35(38.55)	39.83(19.28)
	12	908	863(–5.11)	815(–11.37)	36.66	36.60(–0.16)	35.50(–3.27)
	13	794	1008(21.25)	863(8.06)	31.38	45.99(31.77)	36.26(13.46)
5	19	1004	1258(20.23)	1021(1.72)	40.68	63.16(35.59)	51.22(20.58)
	24	1124	1116(–0.72)	986(–14.00)	43.57	49.29(11.60)	47.11(7.51)
	29	1151	1287(10.57)	1146(–0.44)	45.73	65.73(30.43)	52.02(12.09)
6	35	1185	1521(22.11)	1304(9.18)	49.89	75.63(34.03)	60.61(17.69)
	48	1391	1238(–12.36)	1057(–31.60)	54.36	61.47(11.57)	50.39(–7.88)
	61	1315	1600(17.80)	1332(1.28)	51.05	77.06(33.75)	59.13(13.66)
7	67	1257	1551(18.99)	1257(0)	57.42	90.00(36.20)	59.39(3.32)
	96	1491	1321(–12.87)	1227(–21.52)	61.42	68.04(9.73)	62.75(2.12)
	125	1404	1727(18.71)	1529(8.19)	59.12	86.61(31.74)	70.88(16.59)
8	131	1635	1678(2.59)	1528(–6.98)	75.23	93.78(19.78)	75.83(0.79)
	192	1643	1690(2.78)	1256(–30.81)	66.10	86.04(23.18)	61.62(–7.29)
	253	1530	1825(16.16)	1553(1.48)	67.26	97.14(30.76)	73.70(8.74)

IV. CONCLUSION

By changing the range selection criteria of modular subtraction, modular addition and subtraction for any modulus can be elegantly merged. The simplification of range detection criteria leads to a regular unified modular adder/subtractor architecture. In most cases, the proposed design is faster, smaller and consumes less power than the latest design [6].

References

[1] R. B. Are and K, Rajan, "An RNS based transform architecture for H.264/AVC," in *2008 IEEE Region 10 Conf. (TENCON 2008)*, Hyderabad, India, Nov. 2008, pp. 1-6.

[2] F. J. Taylor, G. Papadourakis, A. Skavantzos, and A. Stouraitis, "A radix-4 FFT using complex RNS arithmetic," *IEEE Trans. Comp.*, vol. C-34, no. 6, pp. 573–576, Jun. 1985.

[3] P. Fernandez, A.Garcia, J. Ramirez, L. Parrilla, and A. Lloris, "A RNS based matrix-vector-multiply FCT architecture for DCT computation," in *Proc. 43rd IEEE Midwest Symp. Circuits Syst.*, Lansing, MI, Aug. 2000, vol. 1, pp. 350–353.

[4] G. Lakhani, ''VLSI design of modulo adders/subtractors,'' in *Proc. IEEE Int. Conf. Comp. Design: VLSI Comps. & Processors (ICCD 92),* Cambridge, MA , Oct. 1992, pp. 68–71.

[5] C. Efstathiou, I. Voyiatzis, "Handling zero in diminished-1 modulo 2^n+1 subtraction", in *Proc. of 3rd Int. Conf. Signals, Circuits and Systems (SCS 09)*, Medenine, Tunisia, Nov. 2009, pp. 1-6.

[6] P. Matutino, H. Pettenghi, R. Chaves, and L. Sousa, ''RNS arithmetic units for modulo $\{2^n \pm k\}$,'' in *2012 15th Euromicro Conf. Digital System Design (DSD)*, Izmir, Turkey, September 2012, pp. 795 –802.

[7] R. Patel, M. Benaissa, N. Powell, and S. Boussakta, ''Novel power-delay-area-efficient approach to generic modular addition,'' *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 6, pp. 1279–1292, Jun. 2007.

[8] A. A. Hiasat, ''High-speed and reduced-area modular adder structure for RNS,'' *IEEE Trans. Comput.*, vol. 51, no. 1, pp. 84–89, Jan. 2002.

[9] H. T. Vergos, C. Efstathiou, and D. Nikolos, ''Diminished-one modulo 2^n+1 adder design,'' *IEEE Trans. Comput.*, vol. 51, no. 12, pp. 1389-1399, Dec. 2002.

[10]R. Zimmermann, ''Efficient VLSI implementation of modulo $(2n \pm 1)$ addition and multiplication,'' in *Proc. 14th IEEE Symp. Computer Arithmetic,* Adelaide, Australia, pp. 158-167, Apr. 1999.

[11]R. Burch, F. N. Najm, P. Yang, and T. N. Trick, ''A Monte Carlo approach for power estimation,'' *IEEE Trans. VLSI Syst.*, vol. 1, no. 1, pp. 63-71, Mar. 1993.