The CSI Journal on
Computer Science and Engineering
Vol. 14, No. 2, 2017
Pages 60-67
Regular Paper

Implementation of Efficient Modulo 2^n+1 Squarer for $\{2^n-1, 2^n, 2^n+1\}$ Based Residue Number System

Horialsadat Hossein Sajedi[1] Keivan Navi[2] Ali Jalali[2]

[1]Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
[2]Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran

Abstract

In order to design special-purpose digital signal processing, modulo 2^n+1 squarer is the core element that is widely employed in algorithms. Among the 3n dynamic ranges, the $\{2^n-1, 2^n, 2^n+1\}$ moduli set has received significant attention in residue number systems. Since modulo 2^n+1 is the most critical one, in this paper we present a novel architecture of modulo 2^n+1 squarer for weighted binary representation. Experimental results show power, delay and area improvements compared to the latest 2^n+1 squarer architectures.

Keywords: Modulo Squarer, Multiplier, Residue Arithmetic, Residue Number System.

1. Introduction

Squaring an operand means multiplying it by itself. Several applications such as adaptive filtering [1-6] and image recognition [7-9] have very large number of operations in which dedicated squaring circuit can perform more computations per second. Arithmetic operations can be sped up, by decomposing large operands of such applications into several smaller residues. Residue number computations are performed in different coprime modulo channels in parallel. Using special moduli such as 2^n-1, 2^n and 2^n+1 has advantages because of efficient implementation of modulo arithmetic units and reverse converters for RNS applications [10, 11].

The execution delay of the three moduli set $\{2^n-1, 2^n, 2^n+1\}$ is limited to modulo 2^n+1 channel, because among the others, it has (n+1)-bit wide operands. To alleviate this problem, the diminished-1 representation was introduced [12]. In diminished-1 representation each operand is decremented by one, compared to its weighted one.

The diminished-1 has main drawbacks including extra hardware for handling zero operands and conversion between weighted and diminished-1 forms. Consequently, using the diminished-1 representation is efficient for computations with fewer arithmetic operations in contrast with using weighted representations. Several architectures for designing modulo 2^n+1 squarer have been presented in literatures [13, 14].

Several approaches have been introduced for squarer design. One is carried out by applying a modular multiplier whose all inputs are determined by the same operands. The mentioned approach is more complicated and slower. Another one is performed using minimized logic functions which can be suitable for small input operands [15]. In [13] a new design of modulo 2^n+1 squarer is presented using compressors in partial product reduction stage instead of full adders. The final stage modular adder has less complexity using sparse tree. Although squarer in [13] includes conversion between diminished-1 and weighted number systems which leads to overhead, the overheads will be excluded in implementations to compare fairly with the proposed one. Another method which has been given in

researches is combinational squaring consisting of three stages: partial product generation, partial product reduction stage (with carry save trees) and finally two input modular adder. Optimization of partial products can be gained by periodic properties of modulo 2^n+1 in any stages. In [14] such method is specified using periodic properties of modulo 2^n+1 squarer to reduce the number of columns by rearranging bits in the columns above the period in the lower columns in which a constant correction word is needed.

In this paper, a new combinational approach is used for implementing such arithmetic operation. The proposed squarer outperforms the weighted squarer described in [13, 14], since it leads to shallower matrix, the number of bits and also height of the partial products matrix. The resulting matrix is regular for odd and even value of n.

The rest of this paper is organized as follows: The novel squarer will be presented in Section 2. The analysis and experimental result will be discussed in Section 3 and finally, Section 4 concludes the paper.

2. Proposed Design

The three main components of a modular squarer consist of partial product matrix generate on, carry save adder, to produce two final operands, and a modular adder.

2.1. The Partial Product Formation:

Let $X = x_n x_{n-1} x_{n-2} ... x_0$ denote (n+1)-bit operands of squarer in the range $[0, 2^n]$.It can be optimized by $x_a.x_a = x_a$. It is evident that if x_n is 1 then other bits of X are zero. This fact permits us to eliminate any partial products having the combination of $x_n x_i$ which $i \in [0, n-1]$. The partial product matrix can be divided into four groups, A, B, C and D shown in figure 1. As mentioned earlier, groups B and D are zero. Terms of groups A and C cannot be 1 at the same time.

The term weighted 2^{2n} in modulo 2^n+1 can be substituted by one term and ORed with x_i in group 'A' at the position of $i \in \{0\}$ because:

$$\left|x_n 2^{2n}\right|_{2^n+1} = \left|x_n 2^n.2^n\right|_{2^n+1} = \left|x_n.(-1)(-1)\right|_{2^n+1} = \left|x_n\right|_{2^n+1} = \left|x_n\right|_{2^n+1} \quad (1)$$

According to equation 2, each term such as c in figure 1, which is placed to the left of column (n-1), must be inverted and repositioned n-bits to the right of column (n-1). The gray colored part of figure 2 shows these modifications.

$$\begin{aligned} & n \le i \le 2n-1 \\ & \left|c2^i\right|_{2^n+1} = \left|c2^n 2^{i-n}\right|_{2^n+1} = \left|-c2^{i-n}\right|_{2^n+1} \\ & = \left|(2^n+1-c)2^{i-n}\right|_{2^n+1} = \left|(2^n+\bar{c})2^{i-n}\right|_{2^n+1} \\ & = \left|2^n 2^{i-n} + \bar{c}2^{i-n}\right|_{2^n+1} \end{aligned} \quad (2)$$

Moreover, we must consider the correction factor of $2^n 2^{i-n}$, which is imposed by each repositioning and complementation. From figure 2 we notice that the first row (row 0) does not need any correction, whereas every other rows (row j) impose $2^n(2^j-1)$ correction factors. Calculation the sum of correction factors in all rows (COR_{pp}) is given by:

$$\begin{aligned} COR_{pp} &= 2^n\left(0+(2^1-1)+(2^2-1)+...+(2^{n-1}-1)\right) \\ &= 2^n\left(2+2^2+...+2^{n-1}-(n-1)\right) \\ &= 2^n\left(2^n-2-n+1\right) = 2^n\left(2^n-n-1\right) \end{aligned} \quad (3)$$

The partial product formation of modulo 2^8+1 squarer is depicted in figure 2.

2.2. Carry Save Adder

In figure 1 each pair of terms in the form of ($x_a x_b$, $x_b x_a$) can be rewritten as two terms in the same form which is called identical pairs in this paper. Since there are identical pairs in every column, we can reduce each pair to one and shift to the left column. So, the partial products will be halved. Another correction factor must be computed because identical pairs in the leftmost column, according to (2), can be completed and transferred to the rightmost column.

$$COR_{identical} = \begin{cases} 2^n.(\frac{n}{2}) & even \quad n \\ 2^n.(\frac{n-1}{2}) & odd \quad n \end{cases} \quad (4)$$

For an even value of "n" in the proposed modulo 2^n+1 squarer, the number of partial product bits in odd columns is 3 less than even columns. The even columns have partial product combinations as shown in table 1. Instead of applying n/2-bit CSA to even columns, we can reduce it into two partial product bits.

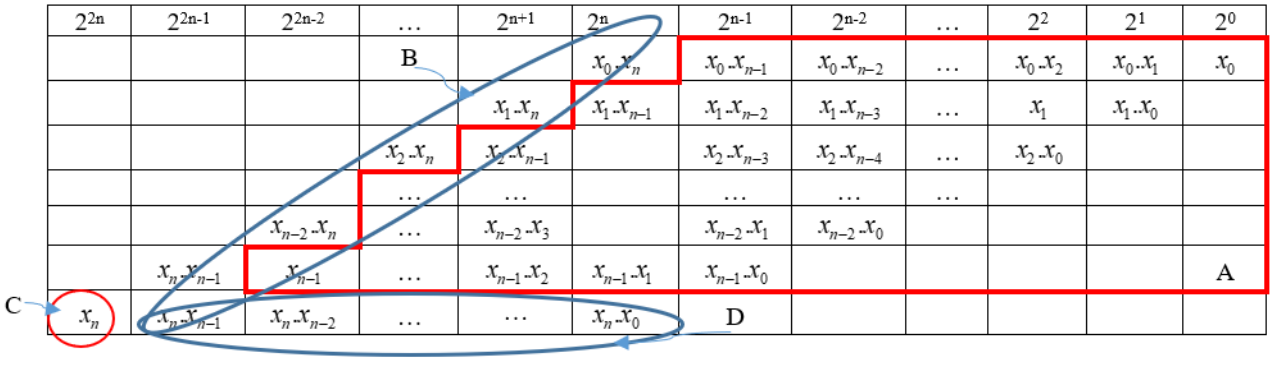

Figure 1. Four parts of partial product (n+1)(n+1) matrix

Therefore, the number of partial product bits in even columns is decreased two units and in odd columns is increased one unit. As a result, the number of partial product bits in all columns is equal to $n/2+1$.

Each column of the proposed modulo 2^n+1 squarer, for an odd value of "n", has three terms (f, g, h) as depicted in figure 3. These terms generate sum and carry which is shown in table 2. So, one level of inverted EAC CSA (End-Around Carry CSA) will be simplified into two partial product bits. The number of partial products in each column will be diminished from $(n+1)/2+1$ to $(n+1)/2$. This process imposes one correction factor of 2^n, according to (2), which will be referred to as $COR_{Odd}=1$ in equation 5.

Table 1. Full adder reduction for even n for even columns

x	y	$x\bar{x}\bar{y}+\bar{x}+0$
0	0	10
0	1	10
1	0	01
1	1	00
		Sum=$x+\bar{y}$ Carry= $\bar{x}$

The partial products matrix must be added until two final summands are fed to modulo 2^n+1 adder. This process can be done using Dadda tree [16]. In order to reduce partial product matrix, inverted EAC CSA will be applied in each stage. It is worth mentioning that carry output of the last column will be inverted and fed as an input of the subsequent stage and produce a correction factor of 2^n.

Modulo 2^n+1 squarer requires $(n/2+1)-2$ stages for even value of n and $(n+1)/2-2$ stages for an odd value of n. Therefore, another correction factor is calculated.

$$COR_{EACCSA}=\begin{cases} 2^n(\frac{n}{2}+1)-2 & even \quad n \\ 2^n(\frac{n+1}{2}-2+COR_{Odd}) & odd \quad n \end{cases} \qquad (5)$$
$$=\begin{cases} 2^n(\frac{n}{2}-1) & even \quad n \\ 2^n(\frac{n+1}{2}-1) & odd \quad n \end{cases}$$

Final correction factor will be obtained as a result of adding equation 3 to 5:

$$COR=COR_{pp}+COR_{identical}+COR_{EACCSA}$$
$$COR=\begin{cases} 2^n(2^n-n-1+\frac{n}{2}+(\frac{n}{2}-1)) & even \quad n \\ 2^n(2^n-n-1+\frac{n-1}{2}+(\frac{n+1}{2}-2)+1) & odd \quad n \end{cases}$$

$$COR=\begin{cases} 2^n(2^n-1-1) & even \quad n \\ 2^n(2^n-1-1)) & odd \quad n \end{cases}=\begin{cases} 3 & even \quad n \\ 3 & odd \quad n \end{cases} \qquad (6)$$

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	COR_{pp}
$x_0..x_7$	$x_0..x_6$	$x_0..x_5$	$x_0..x_4$	$x_0..x_3$	$x_0..x_2$	$x_0..x_1$	x_0+x_8	0
$x_1..x_6$	$x_1..x_5$	$x_1..x_4$	$x_1..x_3$	$x_1..x_2$	x_1	$x_1..x_0$	$\overline{x_1..x_7}$	2^n
$x_2..x_5$	$x_2..x_4$	$x_2..x_3$	x_2	$x_2..x_1$	$x_2..x_0$	$\overline{x_2..x_7}$	$\overline{x_2..x_6}$	$(1+2)2^n$
$x_3..x_4$	x_3	$x_3..x_2$	$x_3..x_1$	$x_3..x_0$	$\overline{x_3..x_7}$	$\overline{x_3..x_6}$	$\overline{x_3..x_5}$	$(1+2+2^2)2^n$
$x_4..x_3$	$x_4..x_2$	$x_4..x_1$	$x_4..x_0$	$\overline{x_4..x_7}$	$\overline{x_4..x_6}$	$\overline{x_4..x_5}$	$\overline{x_4}$	$(1+2+2^2+2^3)2^n$
$x_5..x_2$	$x_5..x_1$	$x_5..x_0$	$\overline{x_5..x_7}$	$\overline{x_5..x_6}$	$\overline{x_5}$	$\overline{x_5..x_4}$	$\overline{x_5..x_3}$	$(1+2+2^2+2^3+2^4)2^n$
$x_6..x_1$	$x_6..x_0$	$\overline{x_6..x_7}$	$\overline{x_6}$	$\overline{x_6..x_5}$	$\overline{x_6..x_4}$	$\overline{x_6..x_3}$	$\overline{x_6..x_2}$	$(1+2+2^2+2^3+2^4+2^5)2^n$
$x_7..x_0$	$\overline{x_7}$	$\overline{x_7..x_6}$	$\overline{x_7..x_5}$	$\overline{x_7..x_4}$	$\overline{x_7..x_3}$	$\overline{x_7..x_2}$	$\overline{x_7..x_1}$	$(1+2+2^2+2^3+2^4+2^5+2^6)2^n$
COR								

Figure 2. Partial product matrix for modulo 2^8+1 squarer

Table 2.Simplified full adder for odd value of n in each column

x	y	f+g+h = $\overline{xy}+\bar{x}+1$	f+g+h = $\overline{xy}+\bar{x}+0$	f+g+h = $x+\overline{xy}+0$	f+g+h = $x+xy+0$
0	0	11	10	01	00
0	1	11	10	01	00
1	0	10	01	10	01
1	1	01	00	01	10
		Sum = $\bar{x}+y$ Carry = $\bar{x}+\bar{y}$	Sum = $x+\bar{y}$ Carry = $\bar{x}$	Sum = $\bar{x}+y$ Carry = $x\bar{y}$	Sum = $x+\bar{y}$ Carry = xy

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
$\overline{x_7}$	$x_6 + \overline{x_7}$	$\overline{x_6}$	$x_5 + \overline{x_6}$	$\overline{x_5}$	$x_4 + \overline{x_5}$	$\overline{x_4}$	$x_3 + \overline{x_4}$
$x_2 . x_4$	x_3	$\overline{x_5 . x_6}$	x_2	$\overline{x_4 . x_5}$	x_1	$x_1 . x_7$	$x_0 + x_8$
$x_1 . x_5$	$x_2 . x_3$	$\overline{x_1 . x_3}$	$x_1 . x_2$	$x_0 . x_2$	$x_0 . x_1$	$\overline{x_2 . x_6}$	$\overline{x_0 . x_7}$
$x_0 . x_6$	$x_1 . x_4$	$\overline{x_0 . x_4}$	$x_0 . x_3$	$\overline{x_3 . x_7}$	$\overline{x_2 . x_7}$	$\overline{x_3 . x_5}$	$\overline{x_1 . x_6}$
0	$x_0 . x_5$	0	$\overline{x_4 . x_7}$	0	$\overline{x_3 . x_6}$	1	$\overline{x_2 . x_5}$

Figure 3. Partial product matrix of the proposed modulo 2^8+1 squarer

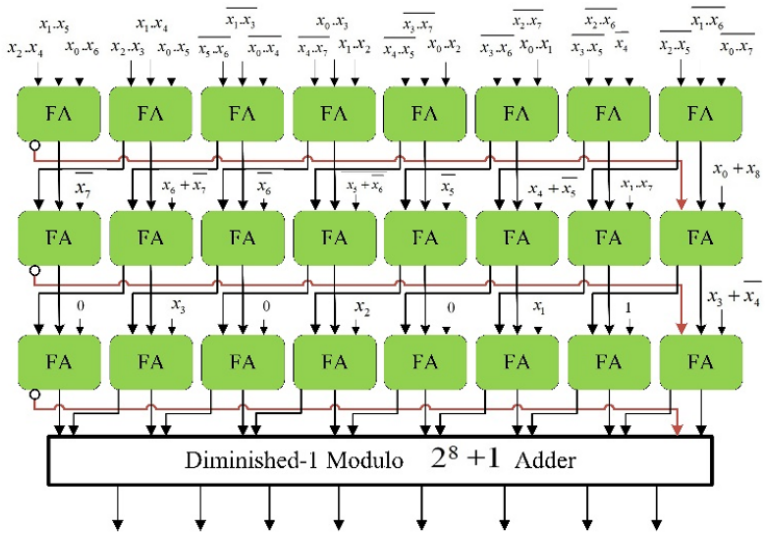

Figure 4. Circuit implementation of the proposed modulo 2^8+1 squarer

This finding reveals that final correction factor is constant and can be added as an extra partial product. Figure 3 indicates partial products for n=8 which shows remarkable partial products reduction.

Formation of the proposed partial products matrix is independent from the value of n whereas the architectures [13] and [14] have different forms depending on whether n is odd or even.

Table 3. Number of bits and maximum height of the partial matrix

Even n	proposed	[14]	[13]
number of bits	$\frac{n^2+n}{2}$	$\frac{n^2+3n}{2}$	$\frac{n^2+5n}{2}$
maximum height	$\frac{n}{2}+1$	$\frac{n}{2}+3$	$\frac{n}{2}+4$

For odd value of n, the maximum height of all columns is the same in each architecture. Although maximum value of height is different in each one. For even value of n, the total number of partial product bits and the maximum height of all columns for each architecture are shown in table 3. It indicates that the total number of bits and the maximum height is significantly reduced compared with others. So, the proposed modulo 2^n+1 squarer leads to regular architecture for both odd and even value of n while others have regular architecture only for odd value of n. Regular array is well-suited for VLSI implementation.

2.3. Modular Adder

A mod 2^n+1 adder is used to sum the two final CSA tree outputs. Weighted and diminished-1 adders are two choices, but the latter is faster. When both inputs of diminished-1 adder are weighted integers instead of diminished-1 representation, consequently the output is $|X+Y+1|_{2^n+1}$.

This means that an extra one is added to the result of the sum. Thus, the correction factor can be reduced by one. This technique was applied in [17]. The correction factor must be modified to 2. Circuit implementation of the proposed modulo 2^8+1 squarer is shown in figure 4.

3. Area-Delay Analysis Results

In this section, the proposed squarer will be compared with the work of [13, 14] which is the most efficient squarer. The area and delay are analyzed quantitatively by unit-gate model proposed in the work of [18]. Delay and area of all 2-input gates compute as one gate equivalent except XOR/XNOR gate, which count as two equivalents. The area and delay of the proposed squarer consist of three sections: partial product formation, the CSA tree and the diminished-1 modular adder.

Considering that each partial product term is produced by either AND or NAND gate and there are identical pairs in each column, equation 7 shows area of AND/NAND gates and one more OR gate that are required for transferring the x_{2n} bit.

$$A_{partial-product} = \begin{cases} n(n/2)+1 & even \quad n \\ n(n+1)/2+1 & odd \quad n \end{cases} \tag{7}$$

The area of CSA tree stage is calculated in terms of unit gate.

$$A_{CSA} = \begin{cases} 7n(n-2)/2 & even \quad n \\ 7n(n-3)/2 & odd \quad n \end{cases} \tag{8}$$

Area occupied by the diminished-1 mod 2^n+1 adder of [19] is $\frac{9}{2}n.\log_2 n+\frac{n}{2}+6$ unit gate. The total area of the proposed squarer is calculated by adding equation (7) to equation (8). Therefore, area of the new architecture is computed by the following equation:

$$A_{total} = \begin{cases} n\left(\frac{n-1}{2}\right)+1+7n\left(\frac{n-2}{2}\right)+\frac{9}{2}n.\log_2 n+\frac{n}{2}+6 & even \quad n \\ n\left(\frac{n+1}{2}\right)+1+7n\left(\frac{n-3}{2}\right)+\frac{9}{2}n.\log_2 n+\frac{n}{2}+6 & odd \quad n \end{cases} \tag{9}$$

The delay of the proposed squarer is computed in the same way. One unit-gate is needed to generate the partial products. The delay of CSA stage is calculated based on the height of the Dadda tree [16]. The modulo 2^n+1 diminished-one parallel prefix adder has $2\log_2 n+3$ unit-gate delay.

$$D_{pro} = 1+4H\left\lfloor\frac{n}{2}+1\right\rfloor+2\log_2 n+3 \tag{10}$$

Table 4. Unit gate model for different value of n

n	Area			Delay		
	Proposed	[14]	[13]	Proposed	[14]	[13]
16	1207	1263	1291	28	32	35
32	4599	4711	4903	39	43	51

Table 4 shows areas and delays of all architectures based on unit gate model for some values of n.

Implementation has been done in VHDL by Modelsim software. Synthesize is tested on ISE 14.6 with default settings, using Xilinx Spartan series 6 XC6SLX4 FPGA chip (smallest device in Spartan-6 family).The analysis is run at room temperature and 1.2V supply voltage. Various key performance metrics such as number of slice registers, number of slice LUTs displaying the occupied area, maximum frequency and total delay are estimated for each circuit. The simulation results for n=8, 12 are shown in table 5. Simulation results indicate that using the proposed squarer leads to better area and delay compared to [13, 14]. The slice resources are the most essential and fundamental to an FPGA design. The total number of slice registers and LUTs are added up to compare the areas. Reductions in area and delay achieved by the proposed squarer are shown in table 6. Examining table 6 shows that the proposed squarer compared to [13], on average achieved 18.1% area reduction and 22.8% delay reduction. Comparing novel squarer to [14] illustrates that the average reduction in area and delay are 17.5% and 6.8% respectively. The maximum frequency is illustrated in figure 5.

Table 6. Area and delay reductions of the proposed squarer for modulo 2^n+1 in Xilinx

N	Proposed squarer compared to	% Total number of slice registers and LUTs reduction	% Delay reduction
8	[14]	21.4	6.1
	[13]	20.8	22.5
12	[14]	13.7	7.5
	[13]	15.4	23.2

It should be mentioned that the result shown in table 5 has excluded the conversion between diminished-1 and weighted binary representation.

Extensive simulations are also conducted for all designs in the same conditions by Synopsys Design Complier tool under nominal synthesis environment, i.e. 25^0C and 1.62V based on TSMC 0.18 μm CMOS standard-cell library. These results have been shown in table 7. The area, power and delay metrics for all squarer circuits have been calculated which demonstrate the superiority of the proposed design.

Table 5. Xilinx implementations result for n=8 and n=12

	n=8			n=12		
	Proposed	[14]	[13]	Proposed	[14]	[13]
No. of slice registers	47	63	58	98	115	107
No. of slice LUTs	59	72	76	116	133	146
Total memory usage(KB)	213964	254776	262172	224988	255180	287640
Total Delay (ns)	14.31	15.25	18.47	15.65	16.92	20.38
Max Frequency (MHz)	464.76	448.88	384.24	437.66	418.57	343.31

Table 7. Simulation results in Synopsys for n=8 and n=12.

	n=8			n=12		
	Proposed	[14]	[13]	Proposed	[14]	[13]
Area (µm2)	102093.4	108626.3	110083.5	203576.6	204779.4	207324.2
Power (µw)	2.18	2.88	3.4	6.58	8.14	8.95
Delay (ns)	27.02	34.93	38.66	36.52	43.86	47.32

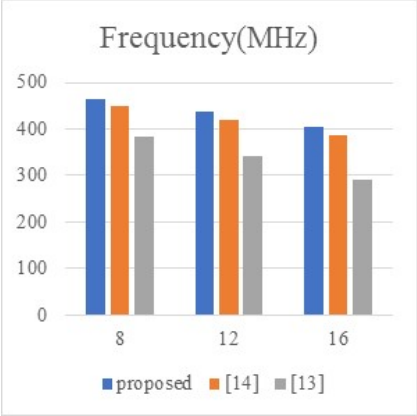

Figure 5. Maximum frequency results in Xilinx for n=8,12,16

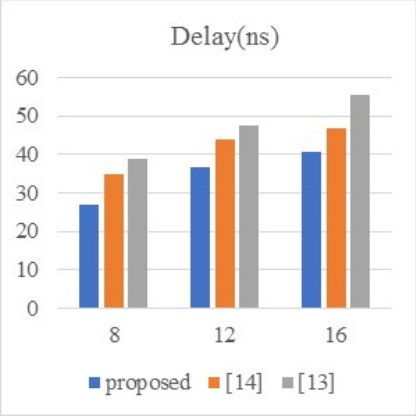

Figure 6. Delay results in Synopsys for n=8,12,16 for

Figure 7. Power results in Synopsys for n=8,12,16

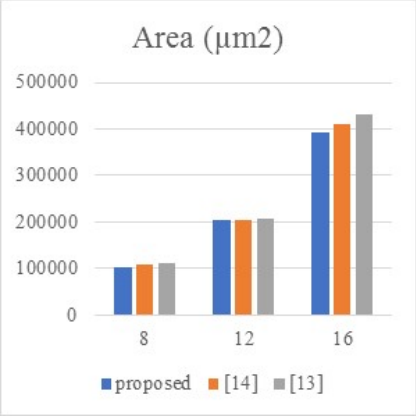

Figure 8. Area results in Synopsys for n=8,12,16

The delays, powers and areas of different architectures are shown in the figures 6, 7 and 8 respectively for various values of n.

4. Conclusion

In this paper, an efficient modulo 2^n+1 squarer is proposed that uses normal binary representation. By reducing all partial product bits in various stages with a constant correction factor significant partial product reduction has been achieved. One level of EAC CSA in Dadda tree is replaced with two partial product bits, which leads to regular architectures for both odd and even value of n. The area, delay and power of the proposed squarer are computed based on unit-gate model, implemented with VHDL and synthesized on FPGA and ASIC. The result shows that the new architecture leads to higher performance than the other mentioned architectures.

References

[1] S. Das, and J.-C. Giomi, "Synthesis of Fast Squarer Functional Blocks," ed: Google Patents, 2015.

[2] B. K. Mohanty, P. K. Meher, and S. K. Patel, "LUT Optimization for Distributed Arithmetic-Based Block Least

Mean Square Adaptive Filter," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, pp. 1926-1935, 2016.

[3] B. K. Mohanty, and S. K. Patel, "Efficient Very Large-scale Integration Architecture for Variable Length Block Least Mean Square Adaptive Filter," *IET Signal Processing*, vol. 9, pp. 605-610, 2015.

[4] S. Y. Park, and P. K. Meher, "Low-Power, High-Throughput, and Low-Area Adaptive FIR Filter Based on Distributed Arithmetic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, pp. 346-350, 2013.

[5] B. Shao, and P. Li, "Array-Based Approximate Arithmetic Computing: A General Model and Applications to Multiplier and Squarer Design," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, pp. 1081-1090, 2015.

[6] Y. Yu, H. Zhao, and B. Chen, "Steady-State Mean-Square-Deviation Analysis of the Sign Subband Adaptive Filter Algorithm," *Signal Processing*, vol. 120, pp. 36-42, 2016.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, pp. 1904-1916, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770-778.

[9] E. Vassalos, D. Bakalis, and H. T. Vergos, "RNS Assisted Image Filtering and Edge Detection," in *Digital Signal Processing (DSP), 2013 18th International Conference on*, 2013, pp. 1-6: IEEE.

[10] A. Hiasat, "A Residue-to-Binary Converter for the Extended Four-Moduli Set {2n-1, 2n+ 1, 2 2n+ 1, 2 2n+ p}," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2017.

[11] S. Kumar, C.-H. Chang, and T. F. Tay, "New Algorithm for Signed Integer Comparison in{2^{n+ k}, 2^{n}-1, 2^{n}+ 1, 2^{n\pm 1}-1\}and Its Efficient Hardware Implementation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, pp. 1481-1493, 2017.

[12] L. Leibowitz, "A Simplified Binary Arithmetic for Fermat Number Transform", IEEE Trans. Acoust., Speech, Signal Process, vol. 24, pp. 356-359, Oct. 1976.

[13] R. Modugu, Y.-B. Kim, K. K. Kim, and M. Choi, "Modulo 2 n+ 1 Squarer Design for Efficient Hardware Implementation," in *SoC Design Conference (ISOCC), 2014 International*, pp. 17-18, 2014.

[14] R. Muralidharan, C.-H. Chang, and C.-C. Jong, "A Low Complexity Modulo 2n+ 1 Squarer Design," in *Circuits and Systems, 2008, APCCAS 2008, IEEE Asia Pacific Conference on*, pp. 1296-1299, 2008.

[15] S. J. Piestrak, "Design of Squarers Modulo A with Low-Level Pipelining," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, pp. 31-41, 2002.

[16] L. Dadda, "Some Schemes for Parallel Multipliers," *Alta frequenza*, vol. 34, pp. 349-356, 1965.

[17] S. Menon, and C.-H. Chang, "A Reconfigurable Multi-Modulus Modulo Multiplier," in *Circuits and Systems, 2006, APCCAS 2006, IEEE Asia Pacific Conference on*, pp. 1168-1171, 2006.

[18] A. Tyagi, "A Reduced-Area Scheme for Carry-Select Adders," *IEEE Transactions on Computers*, vol. 42, pp. 1163-1170, 1993.

[19] H. T. Vergos, C. Efstathiou, and D. Nikolos, "Diminished-One Modulo 2/sup n/+ 1 Adder Design," *IEEE Transactions on Computers*, vol. 51, pp. 1389-1399, 2002.

Horialsadat Hossein Sajedi received her M.S. degree in computer engineering from Science and Research Branch of Islamic Azad University, Tehran, Iran. She is currently working towards the Ph.D. degree in computer engineering at Science and Research Branch of Azad University. Her research interests include computer Arithmetic, signal processing, computer vision and robotics.
Email: p.sajedi@srbiau.ac.ir

Keivan Navi received the B.Sc. degrees in Hardware Engineering from Beheshti University, Tehran, Iran, in 1987 and M.Sc. in Electrical Engineering (Hardware Engineering) Sharif University of Technology, Tehran, Iran, in 1990. He also received the Ph.D. degree in computer architecture from Paris XI University, Paris, France, in 1995. He is currently Professor in faculty of computer science and engineering of Beheshti University. His research interests include VLSI Design, Computer Arithmetic, Multiple Valued Logic, Fuzzy Logic, Quantum Computing, and Emerging Technologies with special emphasis on Single Electron Transistor (SET), Carbon Nanotube Transistor and Quantum-dot Cellular Automata.
Email: navi@sbu.ac.ir

Ali Jalali received the B.S. degree in electronic engineering from Sharif University of Technology, Tehran, Iran, in 1991, the M.Sc. degree in electronic engineering from Supelec University, Gif-sur-Yvette, France, in 1994, and the Ph.D. degree in electronic engineering from Rennes I University, Rennes, France, in 1998. Since 1998 he has been with the Faculty of Electrical and Computer Engineering, Shahid Beheshti University, Tehran. His research interests include low-power and low-voltage analog

and digital integrated circuits, digital audio broadcasting, and VLSI design.
Email: a_jalali@sbu.ac.ir

Paper Handling Data:

Submitted: 13.05.2017
Received in revised form: 15.10.2017
Accepted: 21.04.2018
Corresponding author: Dr. keivan Navi,
Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran.