Diminished-One Modulo $(2^n + 1)$ Multiplier Design*

NEGOVAN STAMENKOVIĆ
University of Priština, Faculty of mathematics and science

DRAGANA ŽIVALJEVIĆ AND VIDOSAV STOJANOVIĆ
University of Niš, Faculty of electronic engineering

A technique, based on the residue number system (RNS) with operands in the diminished-1 number system, has been used in several applications which include digital signal processing (DSP), implementation international data encryption algorithm (IDEA), Fermat number transform (FNT), and so on. For implementation of these techniques, several designs for modulo 2^n+1 diminished-1 arithmetic blocks have been proposed. Modulo $(2^n + 1)$ diminished-1 multiplication plays an important role in these arithmetic blocks. Existing algorithms for modulo $(2^n + 1)$ diminished-1 multiplication use either recursive modulo $(2^n + 1)$ addition, or regular binary multiplication integrated with the modulo reduction operation.

This paper proposes an enhanced approach for the design modulo $2^n + 1$ multipliers for diminished-1 operands with respect to those which have been already published. It is improved in a way that the proposed memoryless based multiplication can be decomposed into a number of small units. The architecture for the new multipliers consists of new partial product generator, inverted end-around-carry, carry-save-adder tree and one $(2^n + 1)$ adder.

1. Introduction

The Residue Number System (RNS) is a non-weighted integer system that perform decomposition of arithmetic operations into several independent sub-operations and therefore implies carry-free and high speed operations [5]. RNS is useful in several applications including Digital Signal Processing [14, 9], Image Processing [4], Fast Fourier Transform computation [10], cryptographic algorithms such as International Data Encryption Algorithm (IDEA) [7]. Moreover RNS is also inherently fault tolerant and makes diagnosis and correction of errors easier [8]. Apart from these, modulo $(2^n + 1)$ multiplier is also one of the critical component applications in the area of Cryptographic algorithms like International Data Encryption Algorithm (IDEA) which is frequently used for secured data transmission and in Fermat Number Transform (FNT).

In residue arithmetic, the moduli set $(2^n - 1, 2^n, 2^n + 1)$ has attracted attention because it is suitable for conversion of binary representation into RNS and vice versa. Using this set, for modulo $(2^n - 1)$ and 2^n operations, input operands are n - bit wide, but for the modulo $(2^n + 1)$ operations, input are $(n + 1)$ - bits wide, which makes this modulo operation difficult and calls for special attention.

The diminished-1 representation of RNS binary numbers was introduced in [6] to speed up the modulo $(2^n + 1)$ arithmetic operations. Since only n bits are required for the representation

*This work was supported in part by the Serbian Ministry of Education and Science in the frame of the project 32009TR.

of any digit in RNS, the diminished-1 representation can lead to implementations with similar delay and area complexity as for modulo $(2^n - 1)$ and 2^n representations. A lot of papers on the design of modulo $(2^n - 1)$ adders and multipliers for diminished-1 operands have already been published [13, 12, 11]. However, special treatment is required for operands equal to zero. Since this can lead to implementations with increased delay and area complexity, the efficient integration of zero handling into modulo $(2^n + 1)$ arithmetic units is an open problem.

This paper proposes an enhanced approach for the design modulo $2^n + 1$ multipliers for diminished-1 operands with respect to those which have been already published. It is improved in a way that the proposed memoryless based multiplication can be decomposed into a number of small units. The architecture for the new multipliers consists of new partial product generator, inverted end-around-carry, carry-save-adder tree and one $(2^n + 1)$ adder.

Organization of the paper is as follows. After recalling the diminished-1 arithmetic in Section 2, the architecture of diminished-1 modulo $(2^n + 1)$ multiplier is presented in section 3. Section 4 is an example and last section is conclusion.

2. Diminished-One Arithmetic

To represent all integers in RNS, using modulo (2^n+1), $(n+1)$ bits are required. The additional bit is required in order to represent the number $2^n = \langle -1 \rangle_{2^n+1}$. To overcome the problem of performing binary arithmetic with this additional bit, a modified binary number system is used in order to avoid additions and multiplications involving the additional bit. This allows the additional bit to be only 1 when the number to be represented is 0, which can be achieved by subtracting 1 from the normal binary number. The advantage of this representation is that zero is uniquely identified by MSB=1, for which all arithmetic operations are inhibited.

The normal representation and this diminished-1 representation are indicated in the Table 1 for $n = 4$. When performing arithmetic for mod $(2^n + 1)$ using diminished-1 system, all

T a b l e 1. Correspondence between normal, binary and diminished-1 representations

Normal		Binary	Diminished-1	
0		00000	1	
1		00001	2	
2		00010	3	
3		00011	4	
4		00100	5	
5		00101	6	
6		00110	7	
7		00111	8	
8		01000	9	(-8)
9	(-8)	01001	10	(-7)
10	(-7)	01010	11	(-6)
11	(-6)	01011	12	(-5)
12	(-5)	01100	13	(-4)
13	(-4)	01101	14	(-3)
14	(-3)	01110	15	(-2)
15	(-2)	01111	16	(-1)
16	(-1)	10000	0	

input operands and the corresponding results are expressed in diminished-1 form. Let x' be diminished-1 representation of normal binary number $x \in [0, 2^n]$, namely

$$x' = \langle x - 1 \rangle_{2^n+1}. \tag{1}$$

In (1), when $x \neq 0$, $x' \in [0, 2^n - 1]$ is an n-bit number, therefore $(n+1)$-bit circuits can be avoided in this case. However, when $x = 0$, $x' = 2^n$ is an $(n+1)$-bit number. This leads to the special treatment for $x' = 0$. According to this representation, number x' is represented as $x'_n X'$, where x'_n is the zero indication bit and $X' = x'_{n-1} x'_{n-2} \dots x'_0$ is the magnitude representation [3].

Ordinary addition in this number system can be obtained as follows [13]:

$$\begin{aligned} \langle x + y \rangle_{2^n+1} &= S \\ \langle (x'+1) + (y'+1) \rangle_{2^n+1} &= S' + 1 \\ \langle x' + y' + 1 \rangle_{2^n+1} &= S' \\ \langle x' + y' + \overline{C}_{out} \rangle_{2^n} &= S' \end{aligned} \tag{2}$$

where $C_{in} = \overline{C}_{out}$. Diminished-one addition can be implemented by end-around-carry adder.

Ordinary multiplication in this number system is performed as follows [2, 1]:

$$\begin{aligned} \langle x \times y \rangle_{2^n+1} &= Q \\ \langle (x'+1) \times (y'+1) \rangle_{2^n+1} &= Q' + 1 \\ \langle x' \times y' + x' + y' \rangle_{2^n+1} &= Q' \end{aligned} \tag{3}$$

Number in diminished-one arithmetic is represented by $(n+1)$ bits where the $(n+1)$-th bit is used to indicate zero.

3. Diminished-1 modulo $(2^n + 1)$ multiplication

The modulo $(2^n + 1)$ multiplication algorithm (3) can be easily adapted for the diminished-1 number representation of input operands and output product. Thereby, the two additional terms x' and y' have to be added in the modulo carry-save-adder, resulting in small increasing of area and delay. The special case of $x'y' = 0$ has to be treated separately and the constant correction term has to be adapted.

The architecture of the proposed modulo 2^n+1 diminished-1 multiplier consists of three main blocks as it is shown in Figure 1. First block is a partial product generator, second is Wallace tree for partial product addition and finally there is binary adder for product generation.

In Wallace tree, the number of operands is reduced by a factor of 2/3 at each level. Let $\lambda(l)$ be the maximal number of operands that can be added by an l-th level CSA tree, $\lambda(1) = 3$. Each CSA has three inputs and two outputs, hence the number of output times (3/2) will be the number of inputs, that is, the number of outputs in the upper level. If the number of outputs is not a multiple of 2, then $\langle \lambda(l-1) \rangle_2$ indicates the number of extra outputs in the upper level. Hence $\lambda(l)$ can be defined recursively as in Equation (4).

$$\lambda(l) = \left\lfloor \frac{\lambda(l-1)}{2} \right\rfloor \times 3 + \langle \lambda(l-1) \rangle_2. \tag{4}$$

The easier way is using the Equation (5)

$$Number\_of\_levels = \frac{\log(k/2)}{\log(3/2)}, \quad \text{for} \quad k(\frac{2}{3})^l \leq 2 \tag{5}$$

where k is number of operands.

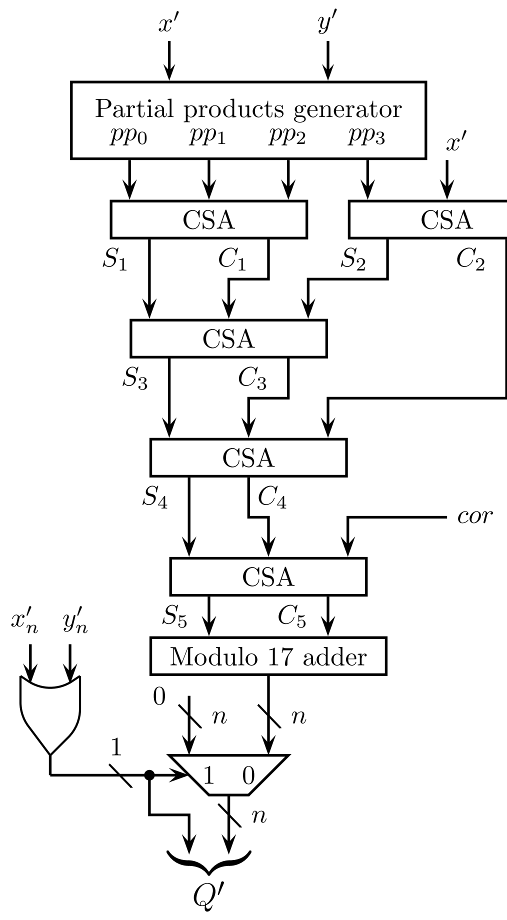

Fig. 1. Architecture of diminished-1 modulo $2^n + 1$ multiplier for $n = 4$.

T a b l e 2. Partial product generation modulo $2^4 + 1$

2^6	2^5	2^4	2^3	2^2	2^1	2^0	
			$x'_0y'_3$	$x'_0y'_2$	$x'_0y'_1$	$x'_0y'_0$	$= pp_0$
		$x'_1y'_3$	$x'_1y'_2$	$x'_1y'_1$	$x'_1y'_0$	$\overline{x'_1y'_3}$	$= pp_1$
	$x'_2y'_3$	$x'_2y'_2$	$x'_2y'_1$	$x'_2y'_0$	$\overline{x'_2y'_3}$	$\overline{x'_2y'_2}$	$= pp_2$
$x'_3y'_3$	$x'_3y'_2$	$x'_3y'_1$	$x'_3y'_0$	$\overline{x'_3y'_3}$	$\overline{x'_3y'_2}$	$\overline{x'_3y'_1}$	$= pp_3$
			x'_3	x'_2	x'_1	x'_0	$= x'$
			y'_3	y'_2	y'_1	y'_0	$= y'$
			0	0	0	0	$= cor$

If the input residues are $(n+1)$-bits wide, the partial products for modulo 2^n+1 multiple are arranged as n-bits wide vectors, because x'_n (y'_n) is a zero indicator, which is handled separately. The partial product generation for inputs of 5 bits width is shown in Table 2. Obviously, 4-bits $x'_3x'_2x'_1x'_0$ (multiplier) and 4-bits $y'_3y'_2y'_1y'_0$ (multiplicand) are required for the representation of nonzero diminished-1 binary numbers.

In this multiplication, bits with weight greater than 2^3, which are placed to the left of the straight line, are complemented and repositioned to the right of the line.

Assuming that the coefficient word length is 4-bits and input sample word length is 4-bits, in Fig. 1 is shown the hierarchical decomposition of Wallace tree logic. The partial sum are added by using five carry-save-adders (CSA) and modulo 17 adder realized as carry-propagate-adder with end-around-carry. Partial product is generated in parallel.

The principle of the proposed memoryless-based implementation of partial product generator is shown in Fig 2. It consists of n 2-to-1 multiplexers, where n is the multiplier word length. Each of the MUX consists of n MUX cells (bit-level) working in parallel, where n is the word length of the multiplier (x'). The partial product is generated by connecting zero and multiplier bit to the MUX inputs and multiplicand bit to the select input. This is well known OR logic gate implemented using 2-to-1 MUX. Finally, circular shifting $s - 1$ bits to the left of the MUX output, for $1 \leq s \leq n$, is performed. After circular shifting, $s - 1$ LSB bits are used as complement. Rotation n requires no logic: just connection of the wires appropriately.

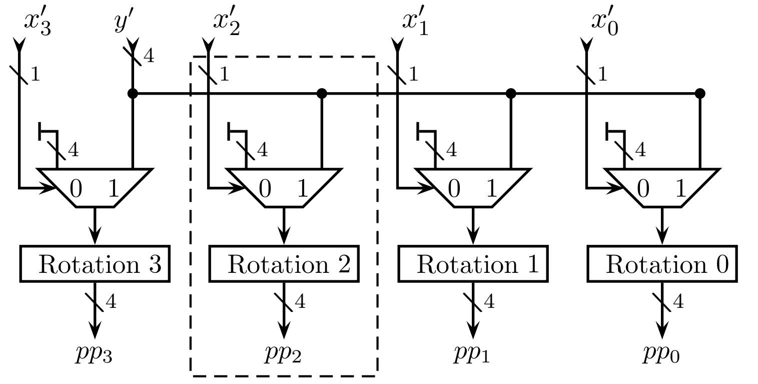

Fig. 2. Partial products generator for $n = 4$.

An implementation of the diminished-1 modulo 17 partial product generator for pp_2 is shown in Figure 3. The small circles above the register represent a complement of the input bit. As it is shown, the rotating function is incorporated in the register by writing partial product bits in proper order.

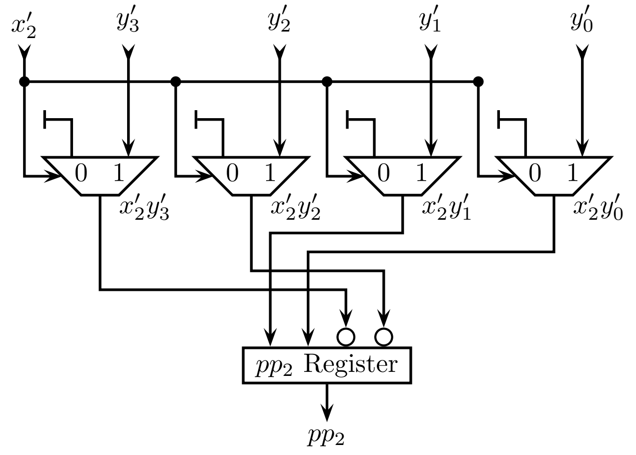

Fig. 3. A wiring diagram of partial product pp_2 generator.

The addition of the partial product can be performed by either a (n +1)- stage Carry-Save-Adder (CSA) array or by a Wallace tree, until a Carry and Sum vector pair is reached. It is well known that using a Wallace tree speed up the addition of the partial products. An implementation of the modulo $(2^n + 1)$ multiplier, based on a Wallace tree, for addition of the diminished-1 operands, is shown in Figure 4. The small circles on the carry output of the adders represent a complemented carry output.

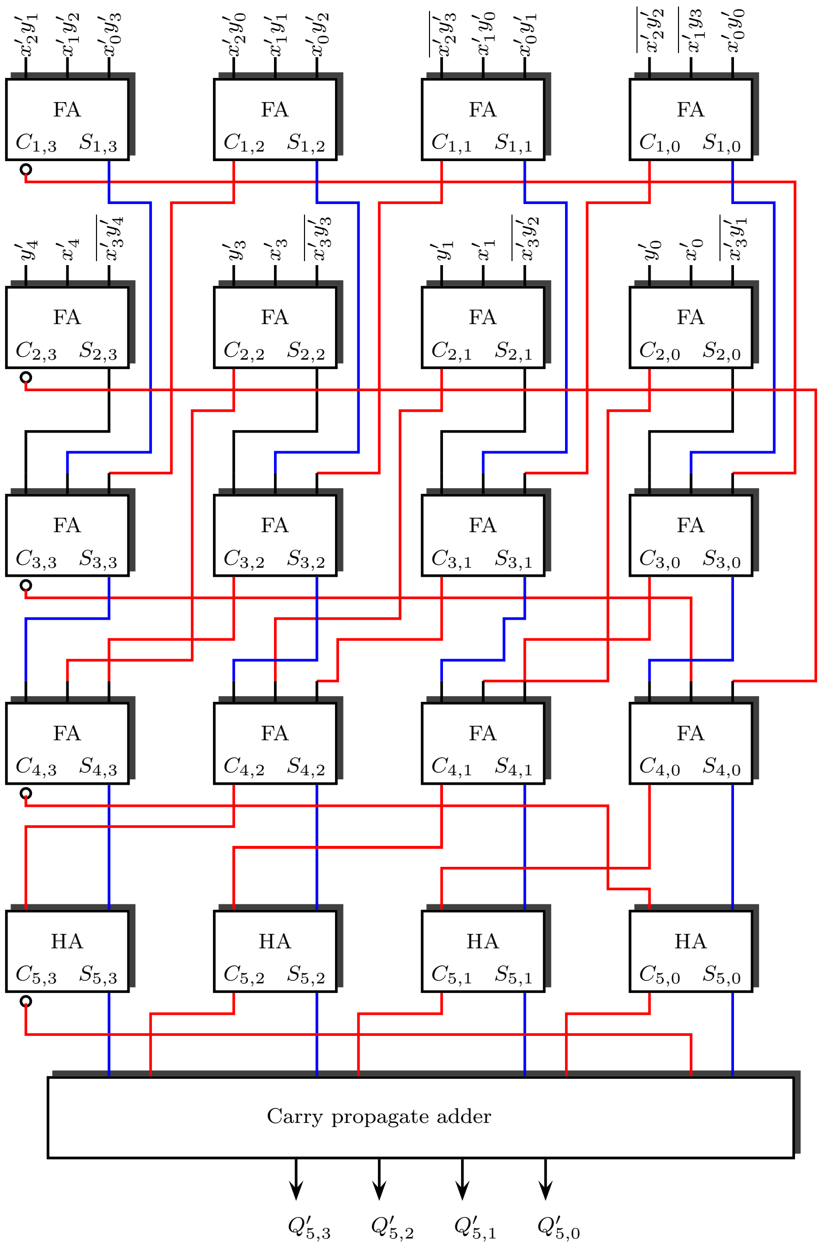

Fig. 4. A wiring diagram of modulo 17 multiplier for diminished-1 operands.

4. An Example

Validity of the above diminished-1 multiplication of two 4-bits numbers is demonstrated on following example. Let $x' = 10_{10} = 1010_2$ and $y' = 9_{10} = 1001_2$. For the modulo 17 multiplication according to proposed methodology the following operands are derived:

1010×1001

pp_0	=	1	0	1	0
pp_1	=	0	0	0	1
pp_2	=	0	0	1	1
pp_3	=	0	0	1	0
x'	=	1	0	1	0
y'	=	1	0	0	1
cor	=	0	0	0	0

The first carry-save-adder

$$\begin{array}{rcccccc} pp_0 & = & & 1 & 0 & 1 & 0 \\ pp_1 & = & & 0 & 0 & 0 & 1 \\ pp_2 & = & & 0 & 0 & 1 & 1 \\ \hline S_1 & = & & 1 & 0 & 0 & 0 \\ C_1 & = & \ulcorner 0 & 0 & 1 & 1 & \\ & & \llcorner & \longrightarrow & \longrightarrow & \longrightarrow & 1 \end{array}$$

The second carry-save-adder

$$\begin{array}{rcccccc} pp_3 & = & & 0 & 0 & 1 & 0 \\ x' & = & & 1 & 0 & 1 & 0 \\ y' & = & & 1 & 0 & 0 & 1 \\ \hline S_2 & = & & 0 & 0 & 0 & 1 \\ C_2 & = & \ulcorner 1 & 0 & 1 & 0 & \\ & & \llcorner & \longrightarrow & \longrightarrow & \longrightarrow & 0 \end{array}$$

The third carry-save-adder

$$\begin{array}{rcccccc} S_1 & = & & 1 & 0 & 0 & 0 \\ C_1 & = & & 0 & 1 & 1 & 1 \\ S_2 & = & & 0 & 0 & 0 & 1 \\ \hline S_3 & = & & 1 & 1 & 1 & 0 \\ C_3 & = & \ulcorner 0 & 0 & 0 & 1 & \\ & & \llcorner & \longrightarrow & \longrightarrow & \longrightarrow & 1 \end{array}$$

The fourth carry-save-adder

$$\begin{array}{rcccccc} S_3 & = & & 1 & 1 & 1 & 0 \\ C_3 & = & & 0 & 0 & 1 & 1 \\ C_2 & = & & 0 & 1 & 0 & 0 \\ \hline S_4 & = & & 1 & 0 & 0 & 1 \\ C_4 & = & \ulcorner 0 & 1 & 1 & 0 & \\ & & \llcorner & \longrightarrow & \longrightarrow & \longrightarrow & 1 \end{array}$$

The fifth carry-save-adder contains only half-adders since $cor = 0000$.

$$\begin{array}{rcccccc} S_4 & = & & 1 & 0 & 0 & 1 \\ C_4 & = & & 1 & 1 & 0 & 1 \\ \hline S_5 & = & & 0 & 1 & 0 & 0 \\ C_5 & = & \ulcorner 1 & 0 & 0 & 1 & \\ & & \llcorner & \longrightarrow & \longrightarrow & \longrightarrow & 0 \end{array}$$

Finally

$$\begin{array}{rcccccc} S_5 & = & & 0 & 1 & 0 & 0 \\ C_5 & = & & 0 & 0 & 1 & 0 \\ \hline & = & \ulcorner 0 & 0 & 1 & 1 & 0 \\ & & \llcorner & \longrightarrow & \longrightarrow & \longrightarrow & 1 \\ \hline Q' & = & & 0 & 1 & 1 & 1 \end{array}$$

Thus $Q' = 7$, which can be verified to be true: $Q' = \left\langle \langle 9 \times 10 \rangle_{17} + 9 + 10 \right\rangle_{17} = 7$.

5. Conclusion

In this paper, an improved method for the design modulo $2^n + 1$ multipliers for diminished-1 operands is presented. It is upgraded in a way that the proposed memoryless based multiplication has been decomposed into a number of small units.To achieve high speed, new partial product generator combining with the Wallace tree is adopted for the multipliers.

Future research includes the extension of this study to Xilinx chips, the power-figure measurement and a full characterization of each design option at layout level.

References

[1] Chen, J. W., Yao, R. H., and Wu, W. J. Efficient modulo $2^n + 1$ multipliers. *IEEE Transactions on Circuits and Systems 19*, 12 (Dec. 2011), 2149–2157.

[2] Efstathiou, C., Vergos, H. T., Dimitrakopoulos, G., and Nikolos, D. Efficient diminished-1 modulo $2^n + 1$ multipliers. *IEEE Transactions on Computers 54*, 4 (Apr. 2005), 491–496.

[3] Efstathiou, C., Vergos, H. T., and Nikolos, D. Handling zero in diminished-one modulo $2^n + 1$ adders. *Int. J. Electronics 90*, 2 (Feb. 2003), 133–144.

[4] Fernandez, P. G., Ramirez, J., Garcia, A., Parrilla, L., and Lloris, A. A new RNS architecture for the computation of the scaled 2D-DCT on field-programmable logic. In *Proc. on Conference Record of the Thirty-Fourth Asilomar Conference on Signals, Systems and Computers* (Pacific Grove, CA, USA, Oct. 29–Nov.01, 2000), pp. 379–383.

[5] Garner, H. L. The residue number system. *IRE Trans. Electronic Computer EC-8*, Issue 2 (June 1959), 140–147.

[6] Leibowitz, L. M. A simplified binary arithmetic for the Fermat number transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing ASSP-24*, 5 (Oct. 1976), 356–359.

[7] Modugu, R., Kim, Y.-B., and Choi, M. Design and performance measurement of efficient IDEA (international data encryption algorithm) crypto-hardware using novel modular arithmetic components. In *Proc. on International Instrumentation and Measurement Technology Conference (I2MTC)* (Austin, TX, USA, May 03–06, 2010), pp. 1222–1227.

[8] Pontarelli, S., Cardarilli, G. C., Re, M., and Salsano, A. Totally fault tolerant RNS based FIR filters. In *Proc. on 14th IEEE International On-Line Testing Symposium, 2008. IOLTS '08.* (Rhodes, Greece, July 07–09, 2008), pp. 192–194.

[9] Stamenković, N., and Stojanović, V. Constant-coefficient FIR filters based on residue number system arithmetic. *Serbian Journal of Electrical Engineering 9*, 5 (Oct. 2012), 100–120.

[10] Taylor, F. An RNS discrete fourier transform implementation. *IEEE Trans. on Acoustics, Speech and Signal Processing 38*, 8 (Aug. 1990), 1386–1394.

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

[12] Wang, Z., Jullien, G. A., and Miller, W. C. An efficient tree architecture for modulo $2^n + 1$ multiplication. *VLSI Signal Processing 14*, 3 (Mar. 1996), 241–243.

[13] Zimmermann, R. Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication. In *Proceedings of the 14th IEEE Symposium on Computer Arithmetic* (Adelaide, Australia, Apr. 1999), p. 158167.

[14] Zivaljević, D., Stamenković, N., and Stojanović, V. Digital filter implementation based on the RNS with diminished-1 encoded channel. In *Proc. on 35th International Conference on Telecommunications and Signal Processing (TSP), 2012* (Prague, Czech Republic, July 03–04, 2012), pp. 662–666.