Circuits, Systems, and Signal Processing (2019) 38:2138–2164
https://doi.org/10.1007/s00034-018-0957-7

Area-Time-Power Efficient Maximally Redundant Signed-Digit Modulo $2^n - 1$ Adder and Multiplier

Somayeh Timarchi[1] · **Negar Akbarzadeh**[1]

Received: 14 November 2017 / Revised: 27 September 2018 / Accepted: 27 September 2018 /
Published online: 9 October 2018

Abstract
By increasing the length of input operands, standard binary number representation cannot satisfy the need for area-time-power efficient systems due to carry propagation chain problem. Redundant residue number system (RRNS) would be an appropriate solution to this demand as it divides large numbers to smaller ones on which the arithmetic operations could be performed in parallel. Maximally redundant signed-digit RNS (MRSD-RNS) has been recently presented as a low-power RRNS because the addition unit based on this number system consumes the least power among the existing RRNS encodings. In this work, a low-power MRSD-RNS multiplier for modulo $2^n - 1$ is proposed for the first time. The implementation results based on the TSMC-90 nm CMOS Technology indicate that our proposed design outperforms power, area, power-delay-product and area-delay-product in comparison with the efficient existing RRNS multipliers for the cases in which delay is not a limiting factor. It has also the least delay among the existing high-radix RRNS multipliers. Therefore, the proposed multiplier can meet the strict area-time-energy constraints which can be used as the core of signal processor in many applications.

Keywords Redundant residue number system · Modular multiplier · High-radix signed-digit · Low-power arithmetic · VLSI

1 Introduction

Multiplication is often the critical path of integrated processors, such as digital signal processing [23], cryptography [11] and communication systems [7], as they have higher latency compared to the other arithmetic circuits like addition and subtraction.

✉ Somayeh Timarchi
s_timarchi@sbu.ac.ir

Negar Akbarzadeh
ne.akbarzadeh@mail.sbu.ac.ir

[1] Department of Electrical Engineering, Shahid Beheshti University, G.C., Tehran, Iran

Moreover, many Internet of Things (IoT) applications, such as health monitoring and surveillance camera, have the limitations of power consumption and extensive computation for burst-mode signal processing. In the IoT applications, individual nodes, also called smart nodes, are often composed of sensors, signal processors and wireless transceivers [28]. To keep smart nodes working for enough time without frequent maintenance, a low-power signal processor with high energy efficiency is adopted to IoT applications [5, 13]. Therefore, it is reasonable to design more efficient multipliers in order to improve the whole device efficiency. One of the most suitable schemes is to use unconventional number systems like residue number systems (RNS) rather than conventional binary number systems. The bottleneck of binary number systems is the carry propagation chain as a function of operand width. This problem deteriorates sharply when the width of operands becomes larger. Here are the reasons which prove that RNS leads to a more efficient number system compared to its counterpart, i.e. binary number system [2–4]. RNS divides large numbers to smaller ones, called residues, based on the desired moduli set. Arithmetic operations could be done independently and in parallel on each module. Due to smaller residues, a smaller arithmetic circuit is needed for each module. There is no carry propagation chain among the moduli. These characteristics result in fast and low-power systems.

Although RNS eliminates the inter-carry propagation chain, the intra-carry propagation chain already exists, which bounds RNS speed-up [4]. Applying redundant encoding for representing each residue has turned to be the best answer to this problem [3]. In fact, the redundant number system is another way for accelerating arithmetic circuits independently. Employing the redundant number system whose digit set in radix-r number system contains more than r digits and consequently enables multiple representations for each redundant number can improve arithmetic operations like addition and multiplication through reduction or elimination of the carry propagation [8, 9, 12].

The combination of redundant and residue number systems to improve performance is called redundant residue number system (RRNS). RRNS is a special residue number system with desired moduli set which employs redundant representation to encode the residues. The basic characteristics of RRNS are fast, parallel and low-power arithmetic because of the following reasons: (1) decomposing the input operand into smaller residues, (2) lack of carry propagation among the moduli and (3) removing carry propagation inside the moduli.

The three existing RRNS encodings are binary signed-digit RNS (BSD-RNS) [29], maximally redundant signed-digit RNS (MRSD-RNS) [17] and stored-unibit-transfer RNS (SUT-RNS) [16]. When using high-radix ($r > 2$) redundant representation to encode the residues, it is called high-radix RRNS. Although BSD-RNS (with radix $r = 2$) utilizes fully redundant representation and offers faster arithmetic units, the two other ones utilize high-radix (with radix $r > 2$) redundant representation and enable a trade-off between delay and area overhead by selecting appropriate radix. Recently, MRSD-RNS was presented in [17] and also an adder structure based on this number representation was proposed. The authors indicate that MRSD-RNS adder has the least delay and power among the high-radix RRNS.

This paper extends the work of [17] and modifies the addition algorithm of $2^n - 1$ MRSD-RNS. It also explores a new modulo $2^n - 1$ MRSD-RNS multiplier for the first

time. The simulation results indicate that the proposed modulo $2^n - 1$ MRSD-RNS multiplier has the least area, power and PDP for different delays among the RRNS multipliers. It has also the least delay among the existing high-radix RRNS multipliers.

The rest of the paper is organized as follows: In Sect. 2, RNS, redundant and RRNS are described. BSD-RNS and SUT-RNS encodings are also studied in the section. Section 3 develops formal representations and algorithmic notations for maximally redundant signed-digit (MRSD) encoding and efficient addition algorithm, as well as MRSD-RNS encoding and the recently presented adder structure. In Sect. 4, we propose a radix-2^h MRSD-RNS multiplication algorithm and structure for modulo $2^n - 1$. Section 5 includes simulation results and comparison of our proposed MRSD-RNS multiplier with other RRNS multipliers. Finally, Sect. 6 concludes the paper.

2 Preliminaries

The three number systems, RNS, redundant and RRNS are first defined in this section. Then, we focus on existing RRNS encodings.

2.1 Residue Number System (RNS)

Definition 1 (*residue number system*) RNS is defined by a set of N positive and pairwise relatively prime moduli $\{m_1, m_2, \ldots, m_N\}$. A binary number R is represented in the determined residue number system as a set of N small integers $R = (R_1, R_2, \ldots, R_N)$, where

$$R_i = |R|_{m_i} \quad 0 \leq R_i < m_i \quad \text{and} \quad 1 \leq i \leq N$$

and $|R|_{m_i}$ denotes the residue of R modulo m_i. This representation is individual for any integer R in the range $[0, M-1]$, where $M = m_1 \times m_2 \times \cdots \times m_N$ is the dynamic range of the moduli set [4]. □

Therefore, RNS converts large numbers to smaller residues according to its moduli. Arithmetic operations like addition, subtraction and multiplication are implemented in parallel on the moduli without any carry propagation between them. So, RNS can support high-speed, carry-limited, low-power and parallel arithmetic.

The sections of a typical RNS architecture are depicted in Fig. 1. As shown in the figure, the RNS is composed of three sections: (1) binary to RNS $(m_1, m_2, \ldots, m_N)$ conversion, RNS modular arithmetic circuits and finally, RNS $(m_1, m_2, \ldots, m_N)$ to binary conversion. The input binary number R is converted to the residues $(R_1, R_2, \ldots, R_N)$ associated with the RNS moduli set $(m_1, m_2, \ldots, m_N)$. (2) RNS arithmetic operations are performed on $(R_1, R_2, \ldots, R_N)$ and produce the final result $\left(R'_1, R'_2, \ldots, R'_N\right)$. 3) Finally, the residues $\left(R'_1, R'_2, \ldots, R'_N\right)$ are converted to R' according to the moduli set. RNS is appropriate for applications composed of only addition, subtraction and multiplication. Division, sign detection and magnitude comparison are difficult to be computed in RNS [4].

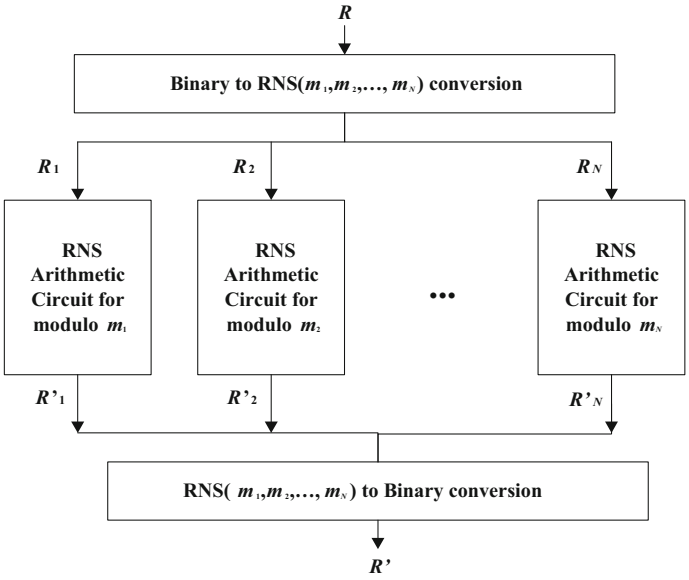

Fig. 1 Typical components of RNS architecture with the moduli set $\{m_1, m_2, \ldots, m_N\}$

2.2 Redundant Number System

A redundant number is formed by a collection of digits, each associated with a power-of-2 weight. A digit is also encoded as a collection of weighted bits. Since a redundant number exploits more bits than required to represent a binary number in a conventional binary number system, some numbers have several representations. Redundant number system allows carry-free addition independent of operand width, i.e. the addition is done without propagating a carry signal through the width of operand [1, 9, 31].

The components of a typical redundant number system architecture are illustrated in Fig. 2. As shown in the figure, the redundant number system is composed of three sections: binary to redundant conversion, redundant arithmetic circuits and finally, redundant to binary conversion.

An input operand R is first converted to redundant representation RR. Redundant arithmetic circuits operate on the input redundant operand RR and produce the output operand RR' with redundant representation. Finally, the output RR' is converted to the binary number R'.

2.3 Redundant Residue Number System (RRNS)

As mentioned earlier, utilizing redundant number system for representing each module of residue number system leads to more efficient arithmetic units due to its carry-free properties. RRNS is defined as follows:

Definition 2 (*redundant residue number system*) RRNS is defined by two parameters: (1) the moduli set parameter $m = \{m_1, m_2, \ldots, m_N\}$ associated with the RNS and

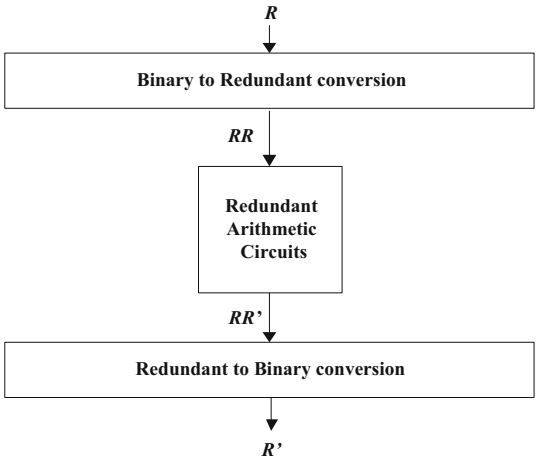

Fig. 2 Typical components of a redundant number system

(2) the parameter '*redundant*' associated with the redundant number system. So, we imply RRNS by the pair (m, *redundant*) where m is the moduli set and redundant is a desired redundant number system. □

Without losing generality, Fig. 3 presents the block diagram of a typical RRNS(m, *redundant*) with the moduli set $m = \{m_1, m_2, \ldots, m_N\}$ and every desired redundant representation. According to this figure, RRNS(m, *redundant*) includes five steps:

1. Binary to RNS conversion: The input binary number R is converted to the residues $\{R_1, R_2, \ldots, R_N\}$ according to the RNS moduli set m.
2. RNS residues to redundant conversion: The residues $\{R_1, R_2, \ldots, R_N\}$ are converted to redundant residues $\{RR_1, RR_2, \ldots, RR_N\}$ according to the defined redundant representation.
3. RRNS arithmetic circuits for each module: The final results $\{RR'_1, RR'_2, \ldots, RR'_N\}$ are produced according to the arithmetic operations.
4. Redundant to RNS residues conversion: The redundant residues $\{RR'_1, RR'_2, \ldots, RR'_N\}$ are converted to the residues $\{R'_1, R'_2, \ldots, R'_N\}$ according to the defined redundant representation.
5. RNS to binary conversion: The residues $\{R'_1, R'_2, \ldots, R'_N\}$ are converted to R' according to the RNS moduli set m.

The first two steps (binary to RNS and RNS to redundant conversions) and also, the last two ones (redundant to RNS and RNS to binary conversions) could be merged to achieve a more efficient implementation. Three RRNS encodings of BSD-RNS [18, 26, 29], MRSD-RNS [17] and SUT-RNS [15, 16, 20] have been explored in the state-of-the-art articles. Nevertheless, BSD-RNS provides the fastest and the most symmetric arithmetic circuits; MRSD-RNS and SUT-RNS (as high-radix RRNSs) offer a trade-off between area and speed through various redundancies. Each number system can be chosen regarding the worst case delay or the available resources listed to a designer. In the rest of the section, we review the BSD-RNS and SUT-RNS encodings.

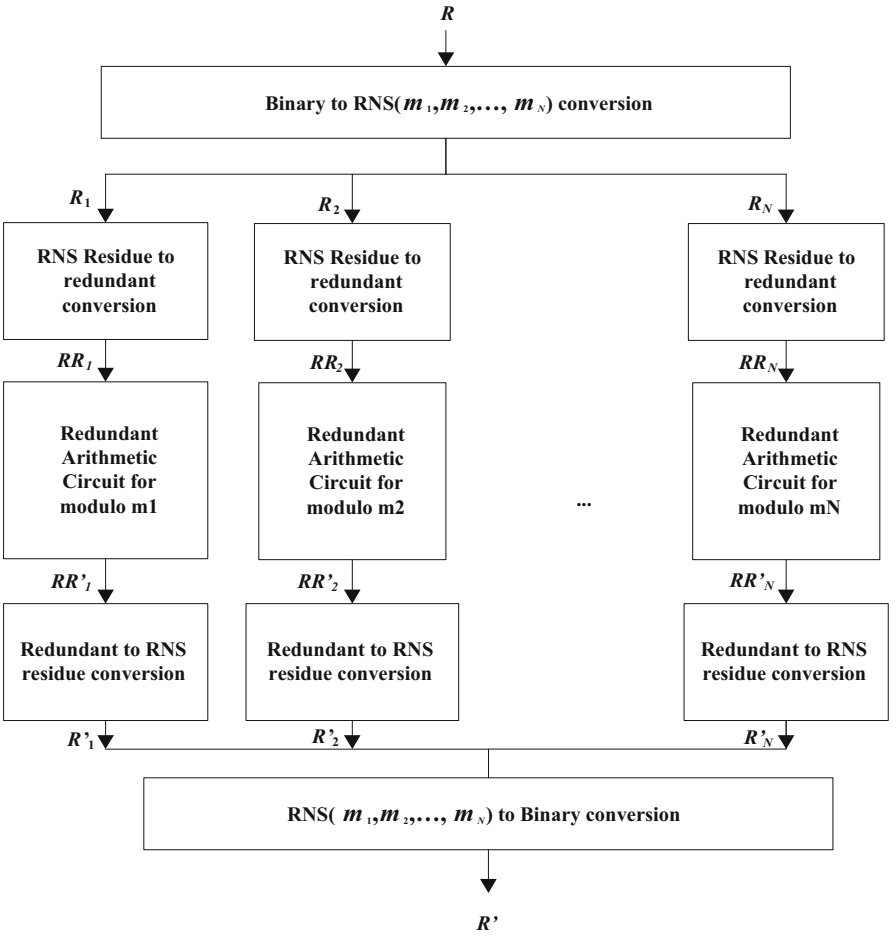

Fig. 3 Generic model of the RRNS(m, *redundant*) with the moduli set $m = \{m_1, m_2, \ldots, m_N\}$ and a defined *redundant* representation

Definition 3 (*BSD-RNS*) BSD-RNS which stands for binary signed-digit residue number system is a class of RRNS number representation which utilizes redundant binary signed-digit (BSD) number system [9, 12] to represent the residues. So, BSD-RNS is implied by RRNS(m, *BSD*) where m is the moduli set. In this number system, each redundant residue RR_i (in Fig. 3) is composed of k binary signed-digits where a binary signed-digit has three values in the range $\{-1, 0, 1\}$ and $k = \log_2 m_i$. □

To encode the three values in the range $\{-1, 0, 1\}$, one posibit and one negabit could be utilized together [21]. Posibit (x_i) is the well-known bit and has the values in the range $\{0, 1\}$. Negabit (X_i) is the negated bit and has the values in the range $\{-1, 0\}$ as depicted in Table 1. It should be mentioned that in this article, posibits and negabits are represented by small and capital words, respectively. According to the table, the posibits (negabits) are represented by black (white) circles in this article.

Table 1 Summary of two-valued digit sets: posibit, negabit and unibit [6]

Bit name	Digit set	Dot notation	Symbolic representation
Posibit	{0,1}	●	x
Negabit	{-1,0}	■	X
Unibit	{-1,1}	□	x'

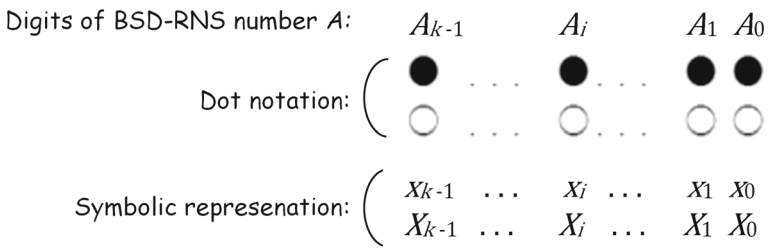

Fig. 4 Dot notation and symbolic representation of a k-digit redundant residue A in BSD-RNS [18]

Figure 4 indicates dot notation and symbolic view of a k-digit redundant residue RR_i in BSD-RNS, where BSD digits A_i $(0 \leq i < k)$ consists of a posibit x_i and a negabit X_i.

Modulo $2^k \pm 1$ BSD-RNS addition could be performed simply by adding an end-around-carry unit to the BSD adder [18, 22]. Making possible to represent results which are greater than the module value by negative range, only k digits are sufficient for representing modulo $2^k + 1$.

Modulo $2^k \pm 1$ BSD-RNS multiplication algorithm proposed in [29] is composed of three main steps: (1) partial product generation, (2) addition tree for partial products reduction and (3) final BSD-RNS modulo addition. Some efficient BSD-RNS multipliers have been reported in the articles [14, 19, 24, 26, 30].

Definition 4 (*SUT-RNS*) SUT-RNS which stands for radix-2^h $(h > 1)$ stored-unibit transfer residue number system is a class of RRNS number representation which utilizes radix-2^h SUT number system to represent the residues [20]. So, SUT-RNS is implied by RRNS(m, radix-2^h SUT). In this number system, each redundant residue RR_i (in Fig. 3) is composed of k radix-2^h SUT digits where a radix-2^h SUT digit, as presented in [6], has the values in the range $\left[-2^{h-1}-1, 2^{h-1}-1\right]$. □

Figure 5 indicates dot notation and symbolic representation of a k-digit radix-2^h redundant residue in SUT-RNS, where radix-2^h SUT digits A_i $(0 \leq i < k)$ consists of $(h-1)$ posibits $x_i^{h-2} \ldots x_i^0$, a negabit X_i^{h-1} and a unibit $x_i'^0$. Unibit is a bit in the range $\{-1, 1\}$ and is noted by a white square as shown in Table 1. It is shown in [15] that general building blocks such as full adder, half adder and compressor could be utilized to construct SUT-RNS arithmetic units.

In [15, 16, 20], the adder, subtractor and multiplier structures have been presented for the moduli set $\{2^n-1, 2^n, 2^n+1\}$ based on SUT-RNS encoding. Although SUT-RNS increases delay slightly, it outperforms area overhead of the BSD-RNS. Besides,

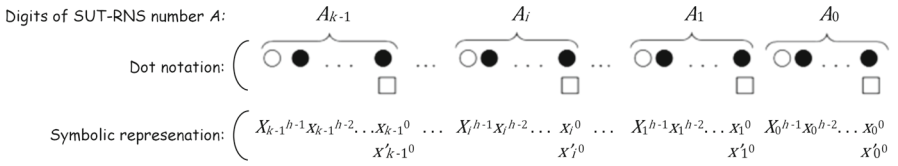

Fig. 5 Dot notation and symbolic representation of a k-digit redundant residue A in radix-2^h SUT-RNS [20]

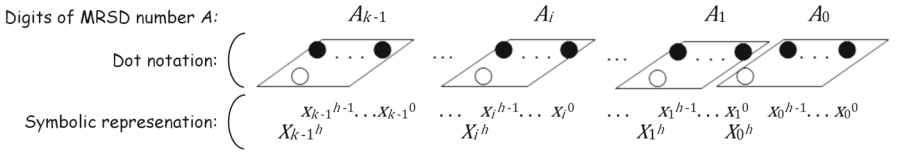

Fig. 6 Dot notation and symbolic representation of a radix-2^h MRSD number system

choosing appropriate radix value provides flexible trade-off between delay and area overhead.

3 MRSD and MRSD-RNS Encodings and Addition Algorithms

In this section, we explore formal representations for the definitions and addition algorithms of MRSD and MRSD-RNS. The algorithmic notations are developed with some modifications.

Definition 5 (*MRSD encoding*) In [13], Avezienis introduced redundant radix-r signed-digit (SD) number systems with digits in the range $[-\alpha, \alpha]$. The redundant radix-r SD with radixes greater than 2 is called high-radix signed-digit (HRSD) as shown in Fig. 6. In the figure, for digits A_i, where $0 \leq i < k$, with the intention of eliminating carry propagation chain, α should be in the following range [13]:

$$\left\lceil \frac{r+1}{2} \right\rceil \leq \alpha \leq r-1 \tag{1}$$

For the cases in which $\alpha = r-1 = 2^h-1$, the radix-2^h maximally redundant SD (MRSD) number systems lead to more efficient SD adders. In fact, the MRSD number is a HRSD number with the maximum possible range. Figure 6 indicates a k-digit radix-2^h MRSD number which consists of h posibits $x_i^{h-1} \cdots x_i^0$ and a negabit X_i^h. □

An improved addition algorithm for MRSD number system was presented in [10] and is stated by Algorithm 1.

Algorithm 1 (*Fast MRSD Addition*) Let's assume two k-digit radix-2^h MRSD numbers A and B with digits A_i and B_i whose symbolic representations are:

$$A_i = X_i^h x_i^{h-1} \cdots x_i^0,\ B_i = Y_i^h y_i^{h-1} \cdots y_i^0$$

The MRSD addition consists of two steps:

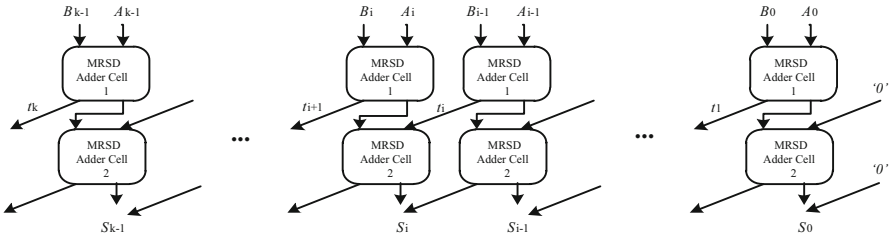

Fig. 7 Symbolic representation of MRSD addition scheme for two k-digit radix-2^h MRSD numbers A and B proposed in [10]

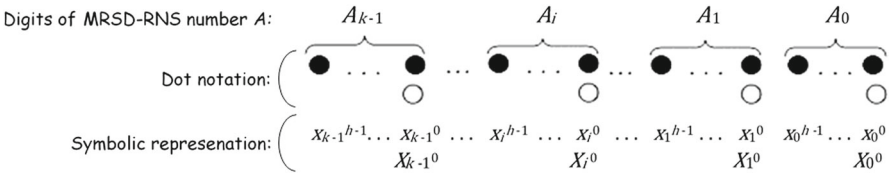

Fig. 8 Dot notation and symbolic representation of a k-digit radix-2^h redundant residue A in MRSD-RNS [17]

- Parallel calculation of transfer digit t_i by only considering digits A_i and B_i, where t_i belongs to $\{-1, 0, 1\}$ and $0 \leq i < k$.
- Parallel calculation of S_i for digits A_i and B_i where $0 \leq i < k$. □

The adder is composed of k MRSD adder cells presented in [10] as shown in Fig. 7. As proved in [10], the transfer digit t_{i+1} is calculated slightly according to the most significant bits (i.e. $X_i^h x_i^{h-1}$ and $Y_i^h y_i^{h-1}$). So, the addition is performed independent of h and a carry-free addition is exploited.

Definition 6 (*MRSD-RNS encoding*) MRSD-RNS which stands for radix-2^h ($h > 1$) maximally redundant signed-digit residue number system is a class of RRNS number representation which utilizes radix-2^h MRSD number system (Definition 5) to represent the residues. So, MRSD-RNS is implied by RRNS(m, *radix*-2^h *MRSD*). In this number system, each redundant residue is composed of k radix-2^h MRSD digits where a radix-2^h MRSD digit is composed of h posibits and a negabit defined earlier. □

Figure 8 indicates dot notation and symbolic representation of a k-digit radix-2^h redundant residue in MRSD-RNS, where radix-2^h MRSD-RNS digits A_i ($0 \leq i < k$) consists of h posibits $x_i^{h-1} \cdots x_i^0$ and a negabit X_i^0. So, the difference between a MRSD number (Definition 5) and a MRSD-RNS redundant residue (Definition 6) is their negabits position, where the negabit is considered as the most (least) significant bit in MRSD (MRSD-RNS) number system. MRSD-RNS encoding and its adder structure have been recently explored in [17]. The addition algorithm is described with some modifications in the next section.

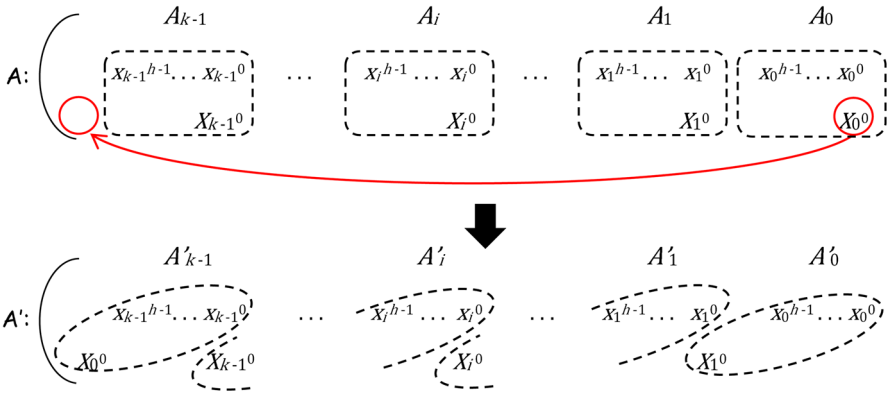

Fig. 9 Defining new digits A'_i and B'_i based on the redundant residues A and B, respectively

4 Proposed Radix-2^h MRSD-RNS Modulo $2^k - 1$ Addition and Multiplication Algorithms

This section explores the proposed addition and multiplication algorithms by Algorithms 2 and 3, respectively.

Algorithm 2 (*MRSD-RNS Modulo* $2^n - 1$ *Addition)* Let's assume two k-digit radix-2^h MRSD-RNS redundant residues A and B composed of digits A_i and B_i, where $0 \leq i < k$ and $n = k.h$.

$$A_i = {}^{x_i^{h-1} \ldots x_i^0}_{X_i^0}, \quad B_j = {}^{y_j^{h-1} \ldots y_j^0}_{Y_j^0} \tag{2}$$

Since $\left|2^n X_0^0\right|_{2^n-1} = \left|2^0 X_0^0\right|_{2^n-1}$, X_0^0 in Fig. 8 could be considered as the most significant negabit in position 2^n. Now, we define new digits A'_i and B'_i shown in Fig. 9. The redundant residues A and B are composed of A'_i and B'_i defined as follows:

$$A'_i = X_i^h x_i^{h-1} \ldots x_i^0, \quad B'_i = Y_i^h y_i^{h-1} \ldots y_i^0 \tag{3}$$

MRSD-RNS adder of A and B for modulo $2^n - 1$ is made of three parts.

1. Parallel calculation of transfer digit t_i by considering digits A'_i and B'_i, where t_i belongs to $\{-1, 0, 1\}$ and $0 \leq i < k$.
2. Parallel calculation of S_i for digits A'_i and B'_i where $0 \leq i < k$.
3. The transfer produced in the last digit (t_k) is ended around to the first digit as an input transfer. For modulo $2^n - 1$, the re-entrant carry is applied to the least significant digit position because we have:

$$\left|2^n t_k\right|_{2^n-1} = \left|(2^n - 1)t_k + t_k\right|_{2^n-1} = t_k \tag{4}$$

□

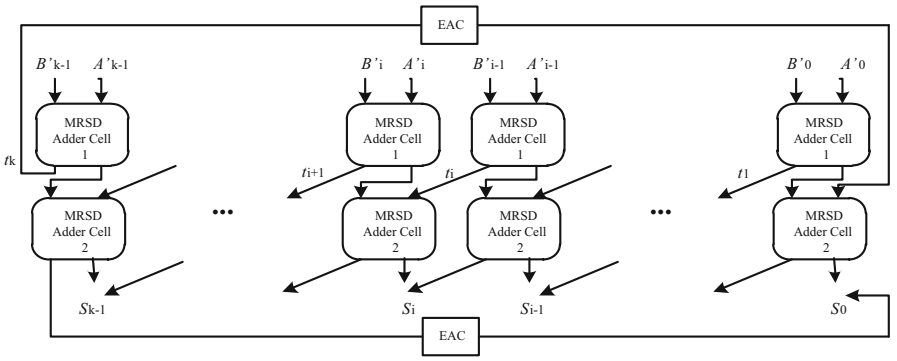

Fig. 10 Architecture of proposed MRSD-RNS modulo $2^n - 1$ adder

It should be also noted that the addition procedure of [17] has been extended by presenting new Algorithm 2 and Figs. 9 and 10. The figures demonstrate appropriately the carry-free property of MRSD-RNS addition that had not been revealed in the previous work.

As mentioned in [17], the only difference between MRSD addition (Algorithm 1) and MRSD-RNS addition (Algorithm 2) is in the third part of the above algorithm, i.e. the output transfer digit t_k is ignored in MRSD addition, whereas it re-enters as input carry to the first digit in MRSD-RNS addition. In Fig. 10, the addition of two k-digit MRSD-RNS numbers A and B for modulo $2^n - 1$ is depicted.

Algorithm 3 (*MRSD-RNS modulo* $2^n - 1$ *multiplication)* Let's assume two k-digit redundant residues A and B in radix-2^h MRSD-RNS where $n = k.h$. The proposed algorithm is composed of three main steps and is depicted in Fig. 11:

Step 1 Digit-product computation: radix-2^h digit A_i is multiplied by B_j for $0 \leq i, j < k$;
Step 2 Digit-products arrangement and rotation to generate partial products, and
Step 3 Partial product accumulation

The rest of the section explains the above three steps in detail. Finally, it is concluded by two examples.

4.1 Step 1 of the Algorithm 3: Digit-Product Computation

In the first stage, we have the following inputs and outputs:

Inputs: digits A_i and B_j in the form of MRSD-RNS digit where $0 \leq i, j < k$
Outputs: k^2 digit products $A_i \times B_j$ (shown by two-digit MRSD numbers $P'_{ji}\ P_{ji}$)

In fact, k digits of A are multiplied by k digits of B. So k^2 digit products are produced. In the following, we consider the multiplication of radix-2^h digits A_i by B_j in MRSD-RNS for $0 \leq i, j < k$. Let's assume that the symbolic representations of A_i and B_j are shown by (2).

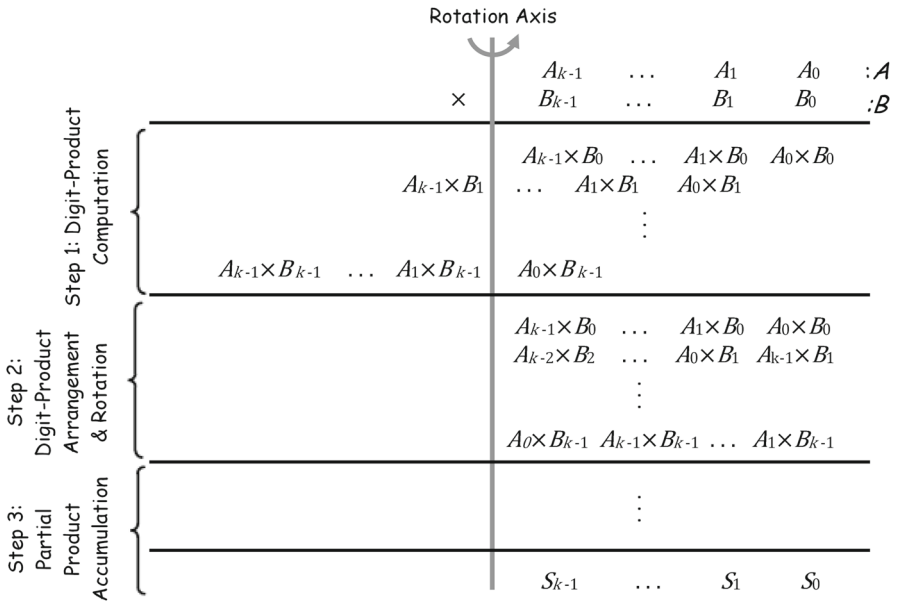

Fig. 11 Multiplying two k-digit redundant residues A and B in radix-2^h MRSD-RNS

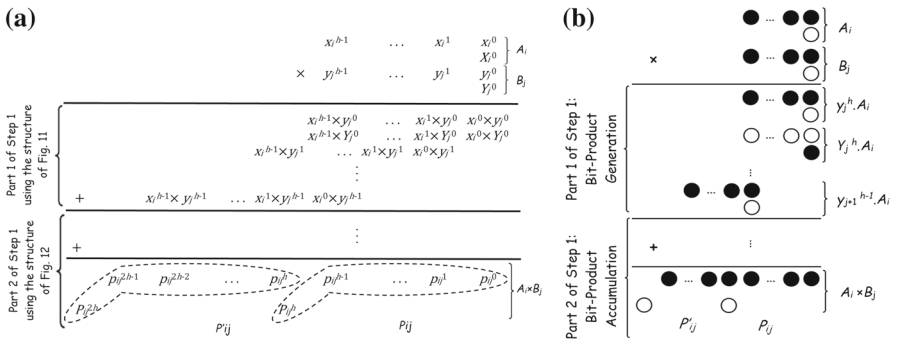

Fig. 12 a Bit representation and **b** symbolic representation of multiplying A_i and B_j according to the first step of Algorithm 3

The bit and symbolic representations of multiplying A_i and B_j are revealed in Fig. 12a, b, respectively, according to the first step of Algorithm 3. The final digit product $A_i \times B_j$ (shown by P'_{ji} P_{ji}) is computed in the form of a two-digit radix-2^h MRSD number as depicted in the figure. The digit-product computation could be divided into two parts:

Part 1) Bit-Product Generation: The first part is to derive the bit products including posibits and negabits. Figure 13 indicates symbolic representations and circuits for the bit-product derivation. There are three possible combinations of bit products as shown in Fig. 13:

- multiplying a posibit by a posibit which results in a posibit as shown in Fig. 13a,

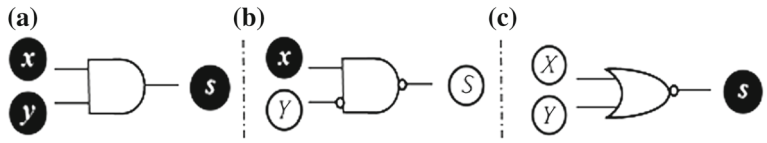

Fig. 13 Symbolic representations and required circuits for bit production as the first part of step 1 of the Algorithm 3: the production of **a** two posibits, **b** one posibit and one negabit, **c** two negabits

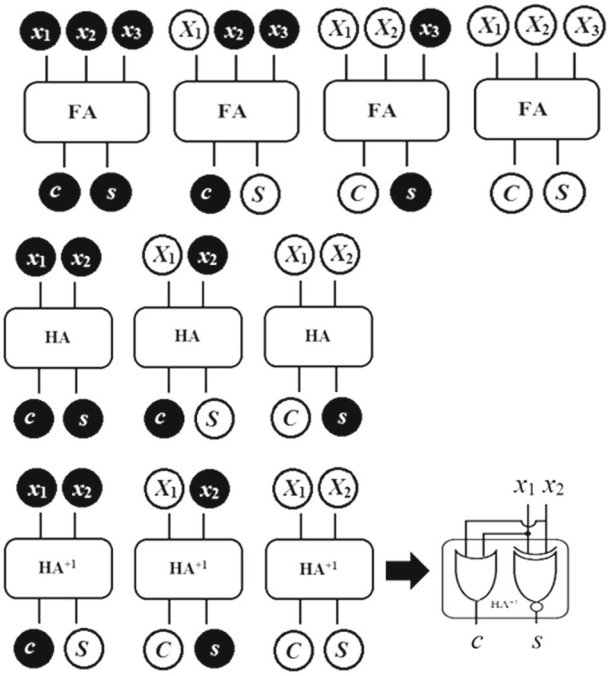

Fig. 14 Different functionalities of the full-adder and half-adder cells to accumulate various sets of input posibits and negabits exploited for the second part of step 1 of the Algorithm 3

- multiplying a posibit by a negabit which results in a negabit as shown in Fig. 13b,
- multiplying a negabit by a negabit which results in a posibit as shown in Fig. 13c.

The first type of bit production is for two posibits implemented by an and-gate as depicted in Fig. 13a. However, when multiplying a negabit Y in the range $\{-1, 0\}$ by a posibit x in the range $\{0, 1\}$, a bit in the range $\{-1, 0\}$ is achieved which is equivalent by a negabit. The bit multiplication is done by an and-gate with inverted negabit input Y and inverted negabit output S, as shown in Fig. 13b. Similarly, by multiplying two negabits in the range $\{-1, 0\}$, a bit in the range $\{0, 1\}$ is obtained which is shown by a posibit. The operation is done by a nor-gate as shown in Fig. 13c.

Part 2) Bit-Product Accumulation: In the second part, the bit products are accumulated to achieve the final digit product. To accumulate the bits, we employ adder structures. Figure 14 shows schematic representations of a full adder (half adder) used to accumulate a set of 3 (2) bits. The sum and carry of each summation are indicated

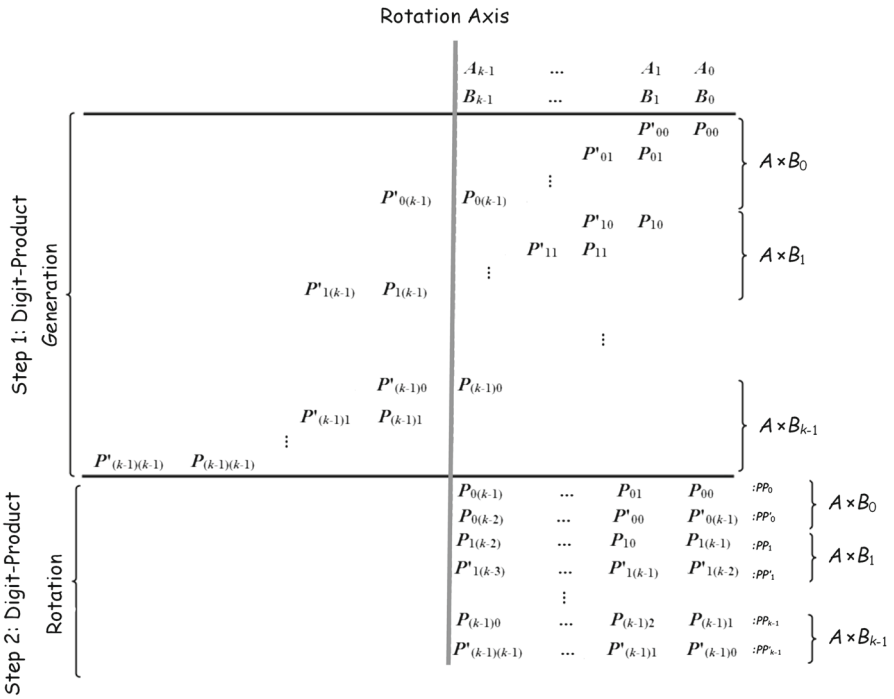

Fig. 15 The first and second steps of the proposed Algorithm 3: reducing K^2 two-digit MRSD numbers to $2k$ k-digit MRSD-RNS partial products (PP_i and PP'_i) by appropriate rotating according to the module

by capital or small s and c letters depending on the input set of posibits and negabits. It is shown in the figure that standard full adders (half adders) could be applied to accumulate three (two) combinations of negabits and posibits. Standard half adder assumes a posibit with the value zero as the third input. However, to obtain the desired final representation, two combinations of equally weighted posibits and negabits utilize a semi-half adder shown by HA^+ in the figure. The cell HA^{+1} assumes a negabit with the value of zero as the third input. So, its structure is a bit different from the standard half adder shown in Fig. 14. For bit-product summation, standard full adders (FA), standard half adders (HA) and semi-half adders (HA^{+1}) represented in Fig. 14 are used. Finally, a two-digit MRSD number is achieved at the end of first step and is shown by $P'_{ji}P_{ji}$ as depicted in Fig. 12.

4.2 Step 2 of the Algorithm 3: Digit-Product Rotation and Partial Product Generation

The inputs and outputs of the second stage are as follows:

Inputs: k^2 digit products $A_i \times B_j$ shown by two-digit MRSD numbers $P'_{ji}P_{ji}$
Outputs: $2k$ partial products in the form of k-digit MRSD-RNS numbers

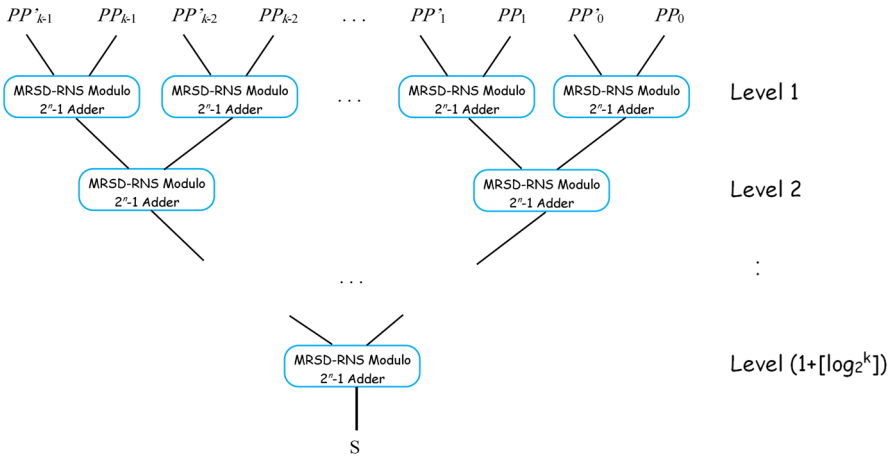

Fig. 16 MRSD-RNS adder tree structure for summing $2k$ partial products

A two-digit MRSD number $P'_{ji}P_{ji}$, produced by multiplying A_i by B_j in the first step, is located in position indices of $(i+j)$ and $(i+j+1)$. In the second step, the input k^2 two-digit MRSD numbers, obtained from the previous step, are arranged first to achieve $2k$ k-digit MRSD numbers. Figure 15 depicts two k-digit radix-2^h MRSD-RNS redundant residues, A and B. Afterwards, the digits with position indices more than n (rotation axis in the figure) are rotated based on the difference between their position indices and n. That is, the modular rotation is performed whenever the addition of two indices of P_{uv} or P'_{uv} is greater than n. let us considered that $q = u+v$. Then, P_{uv} or P'_{uv} is rotated to $|q|_k$ and $|q+1|_k$, respectively.

As indicated in Fig. 15, for multiplication of B_0 by A, as an example, the only digit whose index is greater than n is $P'_{(k-1)0}$ that should be rotated. Similarly, for e multiplication of B_1 by A, 3 digits, i.e. $P'_{1(k-1)}$, $P_{1(k-1)}$ and $P'_{1(k-2)}$, and for the multiplication of B_{k-1} by A, all digits except P_{00} have to be rotated. For modulo 2^n-1, the bits of rotating digits keep their polarity, i.e. posibits remain posibits and negabits remain negabits.

4.3 Step 3 of the Algorithm 3: Partial Products Accumulation

In this step, we have $2k$ k-digit MRSD-RNS partial products that can be accumulated simply by the proposed MRSD-RNS adder depicted in Fig. 10. So, k MRSD-RNS adders are first required to reduce $2k$ partial products to k partial products as shown in level 1 of Fig. 16. Then, $k/2$ MRSD-RNS adders are utilized for summation in the level 2. The reduction continues until the final result is achieved. The final result is achieved after $1+\log_2 k$ levels.

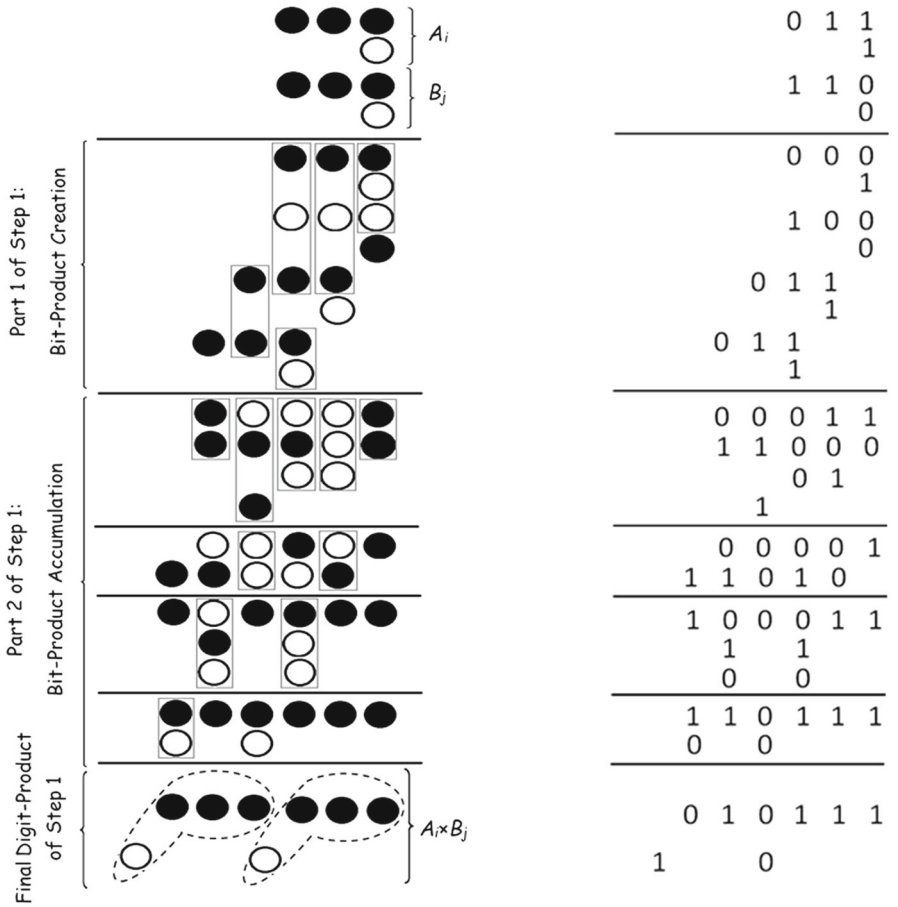

Fig. 17 Example 1: step 1 of Algorithm 3 (digit-product computation) for two radix-8 MRSD-RNS digits A_i and B_j which results in a two-digit MRSD number $P'_{ji}\ P_{ji}$ (left) the numeric example for digit product 3 by 5 which results in 15 (right)

4.4 MRSD-RNS Examples

Example 1 (*radix-8 MRSD-RNS modulo* $2^9 - 1$ *multiplication, i.e.* $h = 3, k = 3$ *and* $n = 9$) the first step of radix-8 MRSD-RNS multiplication for digits A_i and B_j results in a two-digit MRSD number as revealed in Fig. 17. The figure shows the bit representation of multiplying two radix-8 MRSD-RNS redundant residues digits A_i and B_j.

$$A_i = {}^{x_i^2 x_i^1 x_i^0}_{X_i^0}, \quad B_j = {}^{y_i^2 y_i^1 y_i^0}_{Y_i^0}$$

The first part in Fig. 17 indicated as bit product is the product of each bit of B_j multiplied by A_i. As mentioned earlier, the bit products are produced by the structures illustrated in Fig. 13 The second part includes addition of different sets of posibits

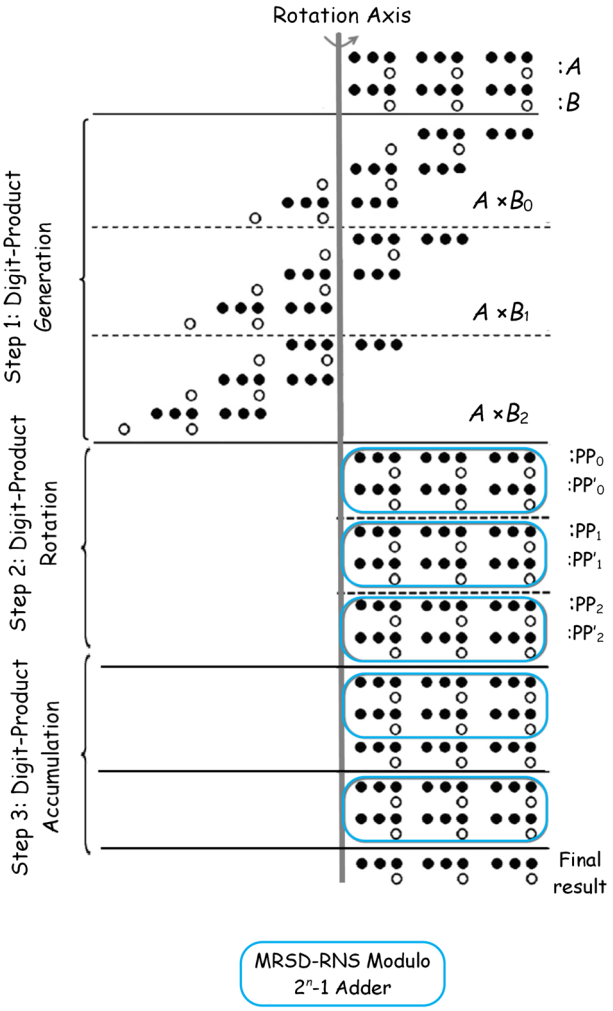

Fig. 18 Example 1: the three steps of the proposed modulo $2^9 - 1$ multiplication algorithm for two 3-digit radix-8 MRSD-RNS numbers

and negabits separated by grey rectangle utilizing the schemes shown in Fig. 14. The process of bit-product summation is done during some steps until the final result is achieved. The final result is in the form of two-digit MRSD number. As a numeric example, the digit product of 3 by 5 is also shown in the figure (right). Th process of producing the final result 15 is also traceable.

Figure 18 indicates the three steps of radix-8 MRSD-RNS modulo $2^9 - 1$ multiplication. After digit-product generation in the first step, digit products are rotated in the second step. Then, partial product accumulation for the 3-digit radix-8 MRSD-RNS multiplication is performed. The rotated partial products in the form of MRSD-RNS number are accumulated by MRSD-RNS adder.

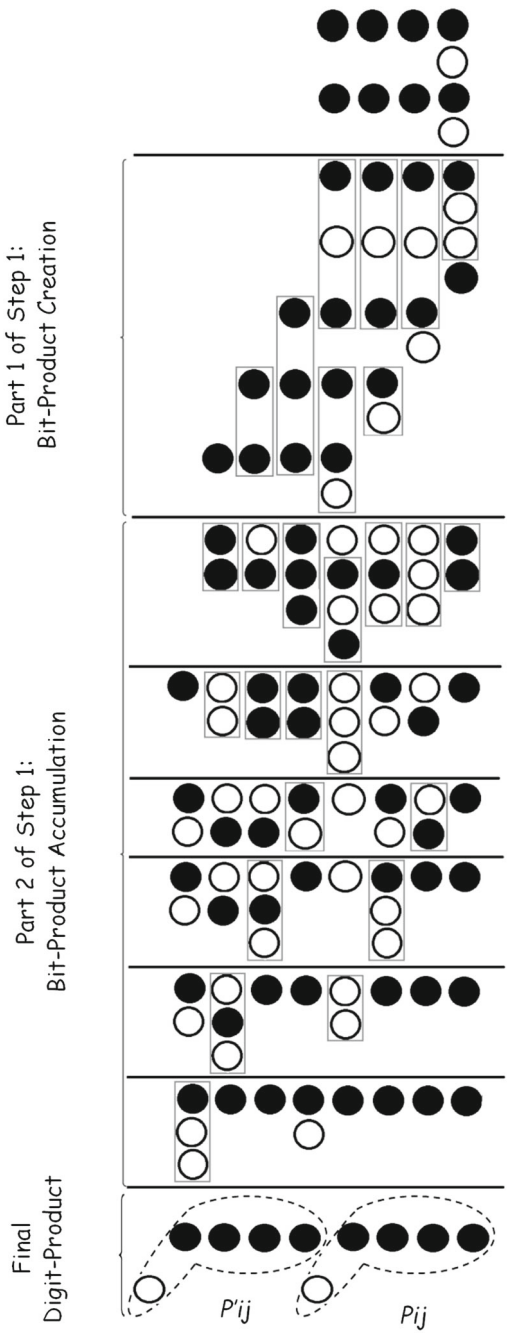

Fig. 19 Example 2: step 1 of Algorithm 3 (digit-product computation) for two radix-16 MRSD-RNS digits A_i and B_j which results in a two-digit MRSD number P'_{ji} P_{ji}

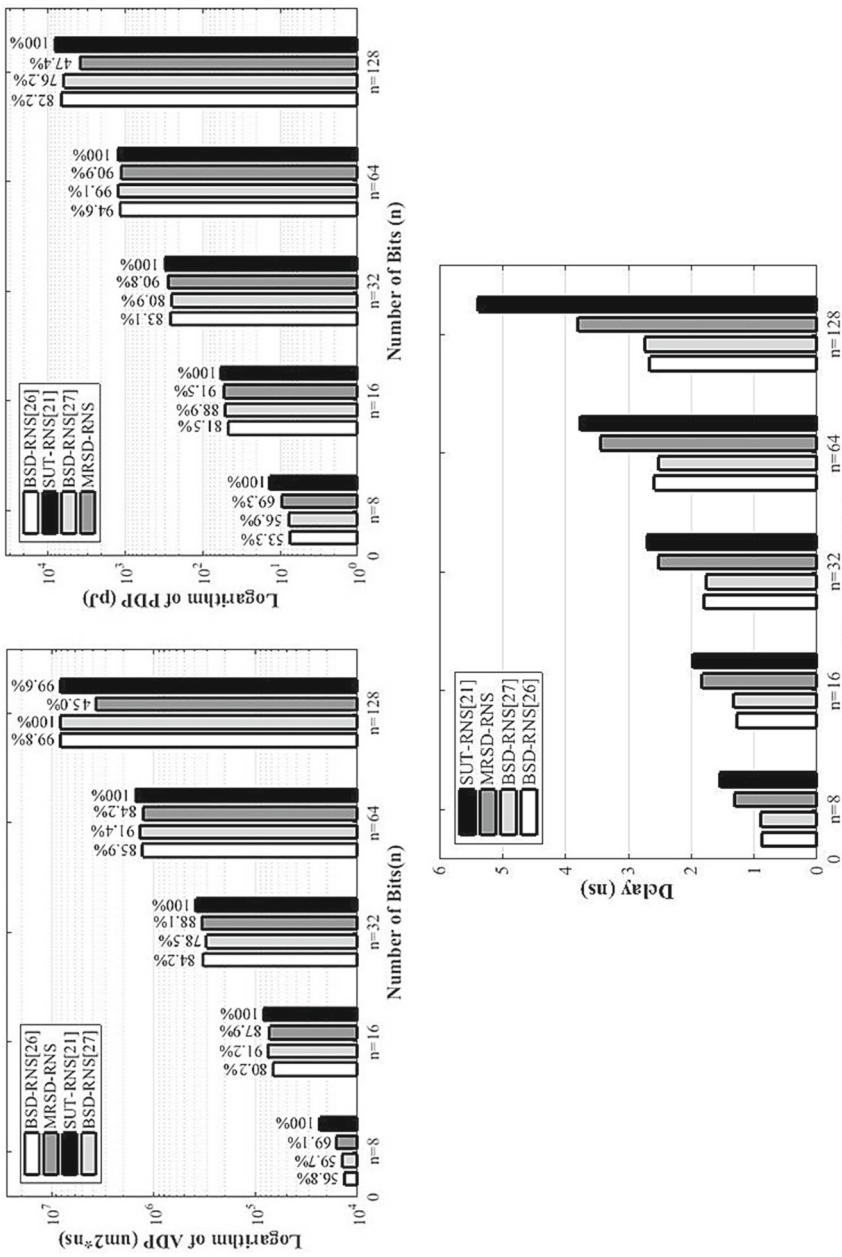

Fig. 20 Total ADP, PDP and delay of radix-16 MRSD-RNS, BSD_RNS and radix-16 SUT-RNS multipliers versus different number of digits

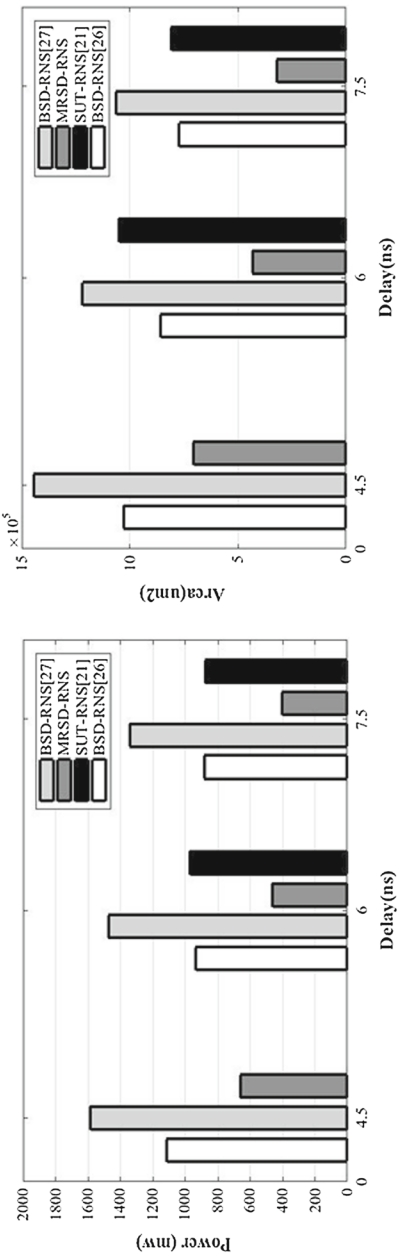

Fig. 21 Total power and area of radix-16 MRSD-RNS, BSD_RNS and SUT-RNS multipliers for modulo $2^{128}-1$

Example 2 (*radix-16 MRSD-RNS Digit Production*) Figure 19 reveals the bit representation of multiplying two radix-16 MRSD-RNS digits A_i and B_j as the first step of Algorithm 3.

$$A_i = {}^{x_i^3 x_i^2 x_i^1 x_i^0}_{X_i^0}, \quad B_j {}^{y_i^3 y_i^2 y_i^1 y_i^0}_{Y_i^0}$$

The next two steps are similar to the previous example depicted in Fig. 15.

5 Simulation Results and Comparison

To compare the proposed MRSD-RNS multiplier with the existing RRNS multipliers, first we selected the most efficient existing modulo $2^n - 1$ multipliers for BSD-RNS [24, 30] and SUT-RNS [20]. Afterwards, we described MRSD-RNS, BSD-RNS and SUT-RNS multipliers in VHDL (VHSIC—Very high speed integrated circuits—Hardware Description Language) and then synthesized them by the Synopsys Design Vision tool. The modulo $2^n - 1$ RRNS multipliers were synthesized by TSMC-90 nm CMOS technology under typical conditions (1 V, 25 °C).

Figure 20 indicates total delay, PDP and ADP of radix-16 BSD-RNS, SUT-RNS, and the proposed MRSD-RNS multipliers versus different number of bits (n). Due to great range of changes in ADP and PDP diagrams, these two figures are depicted in logarithmic scales and for better understanding, the percentage of each bar is calculated and demonstrated on top of it. Moreover, Fig. 21 indicates total area and total power consumption of the three RRNS multipliers for $n = 128$, as an example.

As depicted in Fig. 20, BSD-RNS outperforms delay of the high-radix RRNSs for radix-16. But PDP and ADP of MRSD-RNS outperform the other RRNSs for $n > 32$. It is reasonable because the main property of high-radix RRNSs is their flexibility to receive the desired area-time-power constraints with increasing the number of bits. Accordingly, simulation results of RRNS multipliers for $n = 64$ and 128 are presented in Tables 2 and 3. The tables indicate total area, total power consumption, PDP and ADP of the four radix-16 RRNS modulo $2^{64} - 1$ and $2^{128} - 1$ multipliers for their minimum possible delay. As it is reasonable, BSD-RNS multiplier offers the least delay because the carry propagates shorter path in comparison with high-radix RRNSs. However, the proposed MRSD-RNS multiplier has the least area, power, PDP and ADP.

From Fig. 20, it is concluded that MRSD-RNS offers the least ADP and PDP among all RRNS multipliers for large numbers, i.e. when n is greater than 32. Therefore, the proposed MRSD-RNS could be recognized as an efficient RRNS in terms of area-time-power for large numbers. According to Fig. 21 for the cases in which delay is not the limiting factor, our proposed MRSD-RNS multiplier outperforms area and power of the BSD-RNS and SUT-RNS multipliers versus different delays. For example, to develop a multiplier with 7.5 ns delay, radix-16 modulo $2^{128} - 1$ MRSD-RNS multiplier achieves 60.3%, 58.5% and 69.8% less area and 54.2%, 54.6% and 70.2% less power when compared with BSD-RNS [24, 30] and SUT-RNS [20], respectively.

For further enhancing the results, we utilized matrix multiplication benchmark [25, 27] which is the most suitable benchmark for comparing and approving the efficiency

Table 2 Delay, area, power, PDP and ADP of the modulo $2^{64} - 1$ RRNS multipliers for minimum delay effort

Parameters	Delay (ns)	Area (μm^2)	Dynamic power (mw)	PDP (pj)	Percentage	ADP (μm^2 * ns)	Percentage
BSD-RNS [24]	2.59	500,822.4	451.6	1169.8	94.6	1,297,129.9	85.9
BSD-RNS [30]	2.52	547,712.1	486.1	1224.9	99.1	1,380,234.6	91.4
SUT-RNS [20]	3.76	401,697.9	328.8	1236.3	100	1,510,384.3	100
MRSD-RNS (proposed)	3.45	368,461.5	325.6	1123.3	90.9	1,271,192.2	84.2

Table 3 Delay, area, power, PDP and ADP of the modulo $2^{128} - 1$ RRNS multipliers for minimum delay effort

Parameters	Delay (ns)	Area (μm^2)	Dynamic power (mw)	PDP (pj)	Percentage	ADP (μm^2 * ns)	Percentage
BSD-RNS [24]	2.66	3,106,577.6	2482.3	6602.9	82.2	8,263,496.5	99.8
BSD-RNS [30]	2.74	3,022,731.9	2235.0	6123.9	76.2	8,282,285.3	100
SUT-RNS [20]	5.39	1,530,797.8	1491.1	8037.0	100	8,251,000.2	99.6
MRSD-RNS (proposed)	3.80	978,956.5	1003.5	3813.3	47.4	3,720,034.7	45.0

Table 4 Delay, area, power, PDP and ADP of the modulo $2^{64} - 1$ RRNS multipliers for 2-by-2 matrix multiplication benchmark

Parameters	Delay (ns)	Area (μm^2)	Dynamic power (mw)	PDP (pj)	Percentage	ADP (μm^2 * ns)	Percentage
BSD-RNS [24]	5.11	1,938,469.5	2284.1	11,671.7	100	9,905,579.3	100
SUT-RNS [20]	5.85	1,530,797.8	1562.8	9142.4	78.33	8,505,600.6	85.8
MRSD-RNS (proposed)	5.91	1,340,934.3	1419.0	8386.6	71.85	7,924,921.5	80

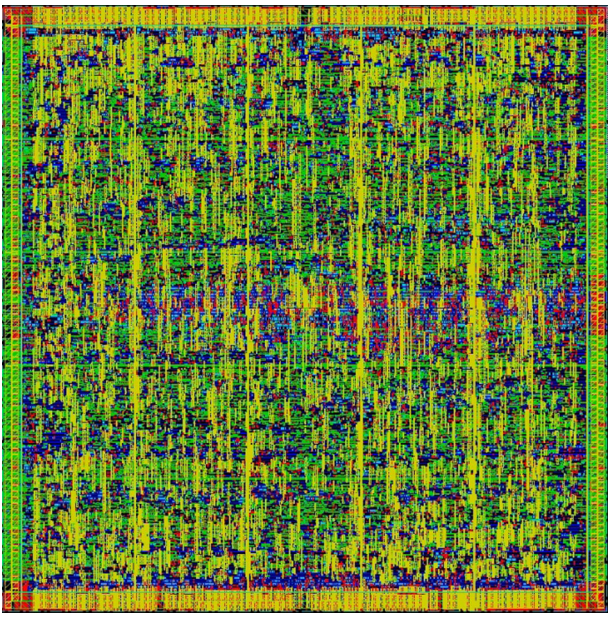

Fig. 22 Layout view of modulo $2^{64}-1$ MRSD-RNS multiplier

of multiplier architectures. Table 4 indicates the results of 2×2 square matrix multiplication for modulo $2^{64}-1$ RRNS multipliers. As it is expected, although MRSD_RNS increases delay up to 13.5%, it improves PDP and ADP by 29.2% and 20%, respectively.

Moreover, in order to retrieve design layout, netlist of modulo $2^{64}-1$ MRSD-RNS multiplier is extracted from Synopsis Design Vision tool and applied to Cadence SOC Encounter tool. SOC Encounter is an automatic place and route software from Cadence which enables quick full-chip virtual prototyping to accurately capture downstream physical or electrical impacts. Figure 22 represents the layout view of modulo $2^{64}-1$ MRSD-RNS multiplier.

6 Conclusion

In order to achieve a good trade-off between delay, area and power, we need to explore new processors to support area-time-energy limited applications. RRNS turns out to be an efficient number system as it eliminates carry propagation chain inside of the RNS moduli. The proposed MRSD-RNS is a kind of RRNS that leads to area-time-power efficient arithmetic units compared to other RRNS encodings.

In this article, we first modified the MRSD-RNS modulo 2^n-1 addition algorithm and then proposed modulo 2^n-1 multiplier for MRSD-RNS encoding. According to the synthesis results, our newly proposed modulo 2^n-1 multiplier for radix-16 MRSD-RNS has less area, power, PDP and ADP than the most efficient RRNS modulo 2^n-1 multipliers for n greater than 32. Besides, MRSD-RNS achieves the least delay

among existing high-radix RRNS multipliers. As an example, radix-16 modulo $2^{128} - 1$ MRSD-RNS multiplier outperforms the delay of SUT-RNS multiplier by %29.5, while it achieves 42.2%, 37.7% and 52.5% less PDP and 54.9%, 55.1% and 54.9% less ADP compared to BSD-RNS [24], BSD-RNS [30] and SUT-RNS [20] multipliers, respectively. Moreover, for modulo $2^{128} - 1$ multiplier, as an example, improvements in 60.3%, 58.5% and 69.8% less area and 54.2%, 54.6% and 70.2% less power are achieved for 7.5 ns delay in comparison with BSD-RNS [24], BSD-RNS [30] and SUT-RNS [20], respectively. Therefore, modulo $2^n - 1$ MRSD-RNS multiplier is a promising RRNS multiplier to realize area-time-power efficient processor.

References

1. A. Armand, S. Timarchi, Low power design of binary signed digit residue number system adder, in *24th Iranian Conference on Electrical Engineering (ICEE)* (2016), pp. 844–848
2. A. Avizienis, Signed-digit number representations for fast parallel arithmetic. IRE Trans. Electron. Comput. **10**, 389–400 (1961)
3. H. Garner, The residue number system. IRE Trans. Electron. Comput. **EC-8**, 140–147 (1959)
4. G. Jaberipur, B. Parhami, M. Ghodsi, Weighted two-valued digit-set encodings: unifying efficient hardware representation schemes for redundant number systems I. IEEE Trans. Circuits Syst. **52**, 28 (2005)
5. M. Khan, S. Din, S. Jabbar, M. Gohar, H. Ghayvat, S.C. Mukhopadhyay, Context-aware low power intelligent SmartHome based on the Internet of things. Comput. Electr. Eng. **52**, 208–222 (2016)
6. I. Koren, *Computer Arithmetic Algorithms* (Universities Press, New York, 2002)
7. X. Liu, E.S. Sinencio, An 86% efficiency 12 μW self-sustaining PV energy harvesting system with hysteresis regulation and time-domain MPPT for IOT smart nodes. IEEE J. Solid-State Circuits **50**(6), 1424–1437 (2015)
8. A.S. Madhukumar, F. Chin, Enhanced architecture for residue number system-based CDMA for high-rate data transmission. IEEE Trans. Wirel. Commun. **3**, 1363–1368 (2004)
9. H. Marzouqi, M. Al-Qutayri, K. Salah, Review of Elliptic Curve Cryptography processor designs. Microprocess. Microsyst. **39**(2), 97–112 (2015)
10. M.C. Mekhallalati, M.K. Ibrahim, New high radix maximally-redundant signed digit adder, in *IEEE International Symposium on Circuits and Systems*, vol. 1 (1999), pp. 459–462
11. K. Navi, A.S. Molahosseini, M. Esmaeildoust, How to teach residue number system to computer scientists and engineers. IEEE Trans. Educ. **54**, 156–163 (2011)
12. B. Parhami, Generalized signed-digit number systems: a unifying framework for redundant number representations. IEEE Trans. Comput. **39**(1), 89–98 (1990)
13. B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs* (Oxford University Press, Inc., Oxford, 2009)
14. D.S. Phatak, I. Koren, Hybrid signed-digit number systems: a unified framework for redundant number representations with bounded carry propagation chains. IEEE Trans. Comput. **43**, 880–891 (1994)
15. D.S. Phatak, T. Goff, I. Koren, Constant-time addition and simultaneous format conversion based on redundant binary representations. IEEE Trans. Comput. **50**, 1267–1278 (2001)
16. A. Safari, C.V. Niras, Y. Kong, Power-performance enhancement of two-dimensional RNS-based DWT image processor using static voltage scaling. VLSI J. Integr. **53**, 145–156 (2016)
17. M. Salim, A.O. Akkirman, M. Hidayetoglu, L. Gurel, Comparative benchmarking: matrix multiplication on a multicore coprocessor and a GPU, in *Computational Electromagnetics International Workshop (CEM)* (IEEE, 2015)
18. M. Saremi, S. Timarchi, Efficient modular binary signed-digit multiplier for the moduli set $\{2n - 1, 2n, 2n + 1\}$. CSI J. Comput. Sci. Eng. **9**(2 & 4(b)), 52–62 (2011)
19. M. Saremi, S. Timarchi, Efficient 1-out-of-3 binary signed-digit multiplier for the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$, in 2013 17th CSI International Symposium on Computer Architecture and Digital Systems (CADS) (IEEE, 2013)

20. S. Timarchi, K. Navi, Efficient class of redundant residue number system, in *Proc. IEEE Int. Symp. WISP* (2007), pp. 475–480
21. S. Timarchi, K. Navi, Arithmetic circuits of redundant SUT-RNS. IEEE Trans. Instrum. Meas. **58**(9), 2959–2968 (2009)
22. S. Timarchi, M. Fazlali, Generalised fault-tolerant stored-unibit transfer residue number system multiplier for moduli set $\{2n - 1, 2n, 2n + 1\}$. IET Comput. Digit. Tech. **6**(5), 269–276 (2012)
23. S. Timarchi, P. Ghayour, A. Shahbahrami, A novel high-speed low-power binary signed-digit adder, in *16th CSI International symposium on Computer Architecture and Digital Systems (CADS)* (2012), pp. 357–363
24. S. Timarchi, M. Saremi, M. Fazlali, G. Georgi, High-speed binary signed-digit RNS adder with posibit and negabit encoding, in *2013 IFIP/IEEE 21st International Conference on Very Large Scale Integration (VLSI-SoC)* (IEEE, 2013), pp. 58–59
25. S. Timarchi, N. Akbarzadeh, A. Hamidi, Maximally redundant high-radix signed-digit residue number system, in *18th CSI International Symposium on Computer Architecture and Digital Systems (CADS)* (2015)
26. E. Vassalos, D. Bakalis, Combined SD-RNS constant multiplication, in *2009 12th Euromicro Conference on Digital System Design*, *Architectures*, *Methods and Tools* (2009), pp. 172–179
27. V. Volkov, J.W. Demmel, Benchmarking GPUs to tune dense linear algebra, in *International Conference for High Performance Computing*, *Networking*, *Storage and Analysis*, *2008. SC 2008* (IEEE, 2008)
28. Z. Wang, Y. Liu, Y. Sun, Y. Li, D. Zhang, H. Yang, An energy-efficient heterogeneous dual-core processor for Internet of Things, in *IEEE International Symposium on Circuits and Systems (ISCAS)* (2015), pp. 2301–2304
29. Sh. Wei, K. Shimizu, Residue arithmetic circuits using a signed-digit number representation, in *IEEE International Symposium on Circuits and Systems*
30. Sh. Wei, J. Changjun, Residue signed-digit arithmetic and the conversions between residue and binary numbers for a Four-Moduli Set, in *2012 11th International Symposium on Distributed Computing and Applications to Business*, *Engineering & Science (DCABES)* (IEEE, 2012)
31. M. Zhang, Sh. Wei, High-speed modular multipliers based on a new binary signed-digit adder tree structure, in *Ninth International Symposium on Distributed Computing and Applications to Business*, *Engineering and Science* (2010), pp. 615–619