Generalised fault-tolerant stored-unibit-transfer residue number system multiplier for moduli set $\{2^n - 1, 2^n, 2^n + 1\}$

S. Timarchi[1] M. Fazlali[2]
[1]Electrical Engineering Department, Shahid Beheshti University, Tehran 1983963113, G.C., Iran
[2]Computer Science Department, Shahid Beheshti University, Tehran 1983963113, G.C., Iran
E-mail: s_timarchi@sbu.ac.ir

Abstract: Residue number system (RNS) which utilises redundant encoding for the residues is called redundant residue number system (RRNS). It can accelerate multiplication which is a high-latency operation. Using stored-unibit-transfer (SUT) redundant encoding in RRNS called SUT-RNS has been shown as an efficient number system for arithmetic operation. Radix-2^h SUT-RNS multiplication has been proposed in previous studies for modulo $2^n - 1$, but it has not been generalised for each moduli lengths (n) and radix ($r = 2^h$). Also, SUT-RNS multiplication for modulo $2^n + 1$ has not been discussed. In this study the authors remove these weaknesses by proposing general radix-2^h SUT-RNS multiplication for the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$. Moreover, the authors demonstrate that our approach enables a unified design for the moduli set multipliers, which results in designing fault-tolerant SUT-RNS multipliers with low hardware redundancy. Results indicate that the proposed general SUT-RNS multiplier for the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ is a fast fault-tolerant multiplier which outperforms area, power and energy/operation of existing RRNS multiplier.

1 Introduction

Multiplication is a critical operation in digital signal processing [1], cryptography [2] and communication [3] because of its high latency in comparison to addition and subtraction. Although techniques such as pipelining can be employed to accelerate multipliers, there is still a necessity to improve multiplication performance. An important scheme to do this is utilising unconventional number representations specially residue number system (RNS) [4]. RNS is presented by a moduli set which represents a large number by a set of small residues (reminders) of each module. In this context, integer operands are represented by residues, and arithmetic operations are independently performed on the residues. The most important characteristic of RNS is carry propagation limited inside each module [5]. Thus, RNS can accelerate arithmetic operations specially addition and multiplication by employing this parallelism. Another way to accelerate digital arithmetic circuits is employing redundant number system whose digit set in radix-r system contains more than r digits [6]. So, a redundant number can have more than one representation and using this capability results in significant improvements in performance through reduction or elimination of carry propagation. Therefore utilising a proper representation for digits among various redundant representations can improve arithmetic operations such as multiplication [7].

Although RNS lowers carry propagation, it cannot eliminate carry propagation inside each module which bounds RNS speed-up. Applying redundant encoding to each module is a suitable solution to reduce the remained carry propagation in RNS. As redundant addition can be done in a constant time independent of the operand length, combining carry-free property of redundant number system with RNS arithmetic can improve the implementation of digital arithmetic circuits. This results in redundant residue number system (RRNS). Signed-digit RNS (SD-RNS) proposed in [8] and stored-unibit-transfer RNS (SUT-RNS) proposed in [9] are two RRNS with fully and high-radix redundant representation, respectively, that have ever been used. SUT-RNS with high-radix redundant representation offers wider variety of choices to designer than the former one. For example, the area available to a designer may be limited, or, the worst case delay is determined. In these cases, the designer can select an appropriate radix-2^h that yields the most suitable implementation while satisfying area usage or time constraint. To this end, we proposed efficient Radix-2^h SUT-RNS multiplication for modulo $2^n - 1$ in [10] and here we discuss on the general multiplication for modulo $2^n + 1$. Also, here we design a fault-tolerant multiplier [11] for moduli set $\{2^n - 1, 2^n, 2^n + 1\}$. Therefore the main contributions of this paper are

1. Generalising radix-2^h SUT-RNS multiplication proposed in [10] for the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ for each moduli lengths (n) and radix (2^h).
2. Enabling a unified design for the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ multipliers to open the possibility of designing fault-tolerant SUT-RNS multipliers with low hardware redundancy.

The rest of the paper is organised as follows. Backgrounds on SD-RNS and SUT-RNS encodings and their arithmetic operations are briefly studied in Section 2. Section 3 focuses on the proposed general radix-2^h SUT-RNS multiplication algorithm and its hardware realisation. Comparisons to previously published architectures are presented in Section 4 and finally, Section 5 concludes the paper.

2 Backgrounds

2.1 Redundant SD-RNS

Applying redundant binary SD representation has been proposed in [8] as a way of eliminating the remaining carry propagation in RNS arithmetic. The resulted number system is called SD-RNS. An n-digit SD number $X = [x_{n-1} \dots x_0]_{\text{BSD}}$, $x_i \in = \{\bar{1}, 0, 1\}$ where $\bar{1} = (-1)$, has the value $|X|_{10} = \sum_{i=0}^{n-1} x_i \times 2^i$. Binary representation of a SD digit, x_i, requires two bits with the same weight, negative bit x_i^- (negabit) and positive bit x_i^+ (posibit). Illustration of posibits and negabits is depicted in Table 1. SD number system can represent a negative integer without any special sign digit. Negation of an SD number is a very simple operation performed by exchanging the bits polarities. That is, all negabits are converted to posibits and vice versa.

Modulo $2^n \pm 1$ SD-RNS addition can be performed by including an end-around-carry in the SD addition computations which has been described in [8, 12–14]. This results in modulo adders that are just a few gates larger than a normal SD adder. Using a more efficient SD adder cell makes SD modulo adders faster and smaller. Also it can make efficient multiplier scheme called SD-RNS multiplier.

SD-RNS multiplication of two n-digit SD numbers can be implemented by obtaining n partial products and accumulating partial products. A partial product is simply obtained through end-around-shift and multiplying a multiplier digit by n-digit multiplicand number. A binary tree of modulo m SD-RNS adders can be constructed for the modulo m summation of the partial products [14].

2.2 Redundant high-radix SUT-RNS

Redundant high-radix SUT-RNS, abbreviately SUT-RNS, is a combination of two number systems: SUT and RNS. That is, SUT-RNS utilises SUT digits (with radix r) in each RNS moduli $(m_1, m_2, \dots, m_p)$. Each radix-$r(r = 2^h)$ SUT digit consists of three types of two-valued digits (twits) [15]: $(h - 1)$ posibits in the set $\{0, +1\}$, a negabit in the set $\{-1, 0\}$, and a unibit in the set $\{-1, +1\}$. Posibit has a lower value equal to 0 whereas negabit and unibit have a lower value equal to -1. Furthermore, posibit and unibit have an upper value equal to 1 whereas negabit has an upper value equal to 0 [15]. All the three twits have two values, so they require one bit for representation. Dot notations, symbolic representations and binary encodings of the twits are given in Table 1. We underline negabits and draw two lines under unibits in SUT bit representation as depicted in Table 1. A negabit is encoded by using logical $\underline{1}$ to denote arithmetic value 0 and logical $\underline{0}$ to denote arithmetic value -1. A unibit is also encoded using logical $\underline{1}$ to denote arithmetic value 1 and logical $\underline{0}$ to denote arithmetic value -1.

SUT-RNS has been proposed as a redundant encoding to represent the numbers in moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ [9, 10, 16]. SUT-RNS encoding proposed in [9] is defined by two types of parameters: value of moduli (m) as well as radix of SUT (r) against the moduli. Therefore SUT with different radix, $r_1, r_2, \dots, r_p$, is generally applied to each moduli $m_1, m_2, \dots, m_p$, respectively, to create SUT-RNS$(r_1, m_1, r_2, m_2, \dots, r_p, m_p)$. Utilising the same radix $(r = r_1 = r_2 = \dots = r_p)$ in a system results in SUT-RNS $(r, m_1, m_2, \dots, m_p)$ where for each module, SUT-RNS is shown by SUT-RNS (r, m), $m \in \{m_1, m_2, \dots, m_p\}$.

In the rest of the paper, we assume SUT-RNS system with the same radix $r = 2^h$ and moduli set $\{2^n - 1, 2^n, 2^n + 1\}$. Now we discuss about SUT-RNS (r, m), $m \in \{2^n - 1, 2^n, 2^n + 1\}$ without losing generality. Consider X is a number in SUT-RNS (r, m). X is composed of k radix-2^h SUT digits $(n = k.h)$ as depicted in Fig. 1. Each radix-2^h SUT representation is composed of a radix-2^h main part $[-2^{h-1}, 2^{h-1} - 1]$ in two's complement representation and a transfer part in the set $\{-1, 1\}$. In this figure, k-digit number X in SUT-RNS encoding is shown, where, x_i, X_i and x_i' are posibit, negabit and unibit, respectively. A_i $(0 \le i \le k)$ is a radix-2^h SUT digit.

SUT-RNS addition structure is depicted in Fig. 2 which is composed of following three stages:

1. Summation of input unibits,
2. Addition of three vectors including two inputs and the vector obtained from the first stage,
3. Ripple-carry addition of two vectors achieved from the previous stage.

Equations of end-around-carry in modulo $2^n \pm 1$ addition are shown by the following equations proved in [9], the least significant bits of addition output can be obtained by the following equations

$$
\begin{aligned}
(s_0)_{\text{modulo } 2^n \pm 1 \text{ addition}} &= C_n \oplus C_n \oplus m_0 \\
(s_0')_{\text{modulo } 2^n - 1 \text{ addition}} &= m_0 C_n + c_n (C_n + m_0) \\
(s_0')_{\text{modulo } 2^n + 1 \text{ addition}} &= m_0 \overline{C_n} + \overline{c_n} (\overline{C_n} + m_0)
\end{aligned}
\tag{1}
$$

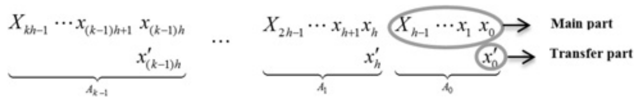
An output carry c_{out} [c_n and C_n in (1)] of modulo $2^n - 1$ SUT-RNS adder can be rotated and stored in position 0. This is justified by (2)

$$
|2^n c_{\text{out}}|_{2^n - 1} = |(2^n - 1) c_{\text{out}} + c_{\text{out}}|_{2^n - 1} = c_{\text{out}} \tag{2}
$$

Therefore if we use modulo $2^n - 1$ SUT-RNS residues, the output carries can be reentered as c_{out} in position 0. That is, equation set (1) is implemented with standard full adders (FAs) as shown in Fig. 2. End-around-carry operation for c_n and C_n leads to a posibit and a negabit in the least significant position, respectively.

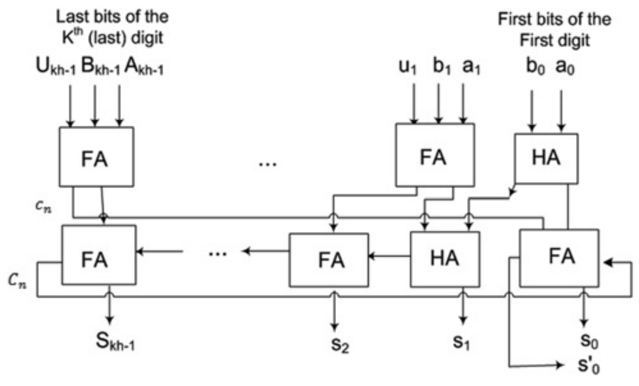
Table 1 The specifications of posibit, negabit and unibit

Bit name	Dot notation	Symbolic representation	Lower value representation	Upper value representation	Arithmetic value
posibit	●	x	0	1	X
negabit	○	X	$\underline{0}$	$\underline{1}$	$X - 1$
unibit	□	x'	$\underline{0}$	$\underline{1}$	$2x' - 1$



$X_{kh-1} \cdots x_{(k-1)h+1}\ x_{(k-1)h}$
$x'_{(k-1)h}$
A_{k-1}
$X_{2h-1} \cdots x_{h+1} x_h$
x'_h
A_1
$X_{h-1} \cdots x_1\ x_0$
x'_0
A_0
Main part
Transfer part

Fig. 1 *Symbolic representation of a k-digit radix-2^h SUT-RNS number X [16]*



Last bits of the K^{th} (last) digit
First bits of the First digit
U_{kh-1} B_{kh-1} A_{kh-1}
u_1 b_1 a_1
b_0 a_0
FA
HA
c_n
C_n
S_{kh-1}
S_2
S_1
S_0
S'_0

Fig. 2 *Redundant end-around-carry adder for two k-digit modulo $2^n - 1$ SUT-RNS residues*

We can rewrite (2) to satisfy modulo $2^n + 1$ SUT-RNS addition. An output carry c_{out} [c_n and C_n in (1)] with weight 2^n in modulo $2^n + 1$ SUT-RNS adder can be reentered as $-c_{\text{out}}$ in position 0. Equation (3) justifies this end-around operation

$$|2^n c_{\text{out}}|^n_{2+1} = |(2^n + 1)c_{\text{out}} - c_{\text{out}}|^n_{2+1} = -c_{\text{out}} \qquad (3)$$

Therefore output carry $c_n = 0$ is converted to 0 and reentered in the least-significant position and output carry $c_n = 1$ results in -1. To perform this inversion and, satisfy inverted encoding depicted in Table 1, the posibit output carry c_n is inverted first and its polarity is converted to a negabit. The same description can be represented for negabit output carry C_n. Hence, after inverting the output carries in modulo $2^n + 1$ SUT-RNS addition, their polarity are also changes. Therefore desired generic modulo $2^n + 1$ adder can be obtained by simply adding two inverters in the output carries (c_n and C_n) paths. The case is also proved in [9] and shown in (1). The adder is obtained by adding two inverters in Fig. 2 and inverting the polarity of two least significant bits.

3 Proposed radix-2^H SUT-RNS multiplication algorithm

SUT-RNS multiplication algorithm proposed in [10] is composed of four stages: creating bit-products, producing digit-products and partial-products summation through addition tree and final modulo addition. We combine third and fourth phases in the previous SUT-RNS multiplication algorithm besides adding a new phase to the algorithm. This let us well generalise multiplication algorithm for moduli set $\{2^n - 1,\ 2^n,\ 2^n + 1\}$ as well as generalising addition tree stage for the moduli set. The proposed SUT-RNS multiplication algorithm is listed in the following stages:

1. Creating bit-products,
2. Producing digit-products,
3. Rotating digit products and producing partial products,
4. Partial-products summation through addition tree.

The first stage of multiplying two SUT-RNS numbers is to derive bit products including posibits, negabits and unibits. There are six possible combinations of bit products whose circuits were proposed in [10, 16].

In the second stage, we derive the digit products. k^2 digit products are formed for n-bit radix-2^h SUT-RNS operands. This stage has been discussed in [16] just for radix-4 and radix-8 SUT digits product, and was proved generally in [10] where we demonstrated that multiplication of two radix-2^h SUT ($h \geq 3$) digits results in a two-digit SUT number with the same representation. Therefore in the second stage, after multiplying two k-digit SUT numbers, there are k^2 digit products with different weights. Each digit product is a two-digit radix-2^h SUT number. It should be mentioned that for radix-4 we suggested a different representation (a pseudo-SUT representation) to guarantee that a two-digit radix-4 pseudo-SUT number is produced.

The third stage is to rotate k^2 digit products which is proposed in the paper for general moduli length (n). The rotation rule is based on the module format and is discussed for moduli set $\{2^n - 1,\ 2^n,\ 2^n + 1\}$ in this paper. k^2 two-digit SUT numbers (obtained from previous stage) are converted to $2k$ k-digit SUT-RNS numbers, as described underneath. So, in this stage $2k$ partial products are created.

In the fourth stage, the partial products are accumulated during an addition tree. Here, we develop the method presented in [10, 16] to be utilised for modulo $2^n + 1$ multiplication. We show that there are three possible radix-2^h SUT digit additions in the addition tree which can be performed using SUT-RNS addition algorithm as well as rules that are proposed in the paper.

The first two stages (producing bit-products and partial products) which have been proposed in [10] are general and can be used in our algorithm. Here we go for generalising the rotation and producing partial products, as well as creating general addition tree. Then, we will explain how the proposed SUT-RNS structure enables fault tolerance.

3.1 Rotating digit products and producing partial products

Let us assume two k-digit radix-2^h SUT-RNS numbers as shown in Fig. 3. SUT-RNS numbers A and B composed of k SUT digits; A_i and B_i $(0 \leq i \leq k - 1)$ refer to ith digit of the numbers A and B, respectively.

During first and second stages, each digit of B (B_i) have been multiplied by A. Fig. 4 depicts digit products of multiplying the first digit of B (B_0) by number A. Each digit production has been resulted in a two-digit SUT-RNS number as proved in [10]. After producing k digit products (resulting from multiplying B_0 and A), the digit products are rotated in the third stage and two k-digit SUT-RNS partial products in the format of SUT or pseudo-SUT are produced. The pseudo-SUT refers to a SUT-RNS number whose some digits are not in the format of standard SUT.

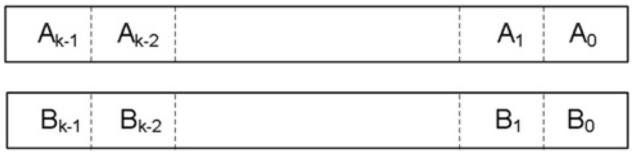


Fig. 3 *Representation of two k-digit SUT-RNS numbers, A and B*

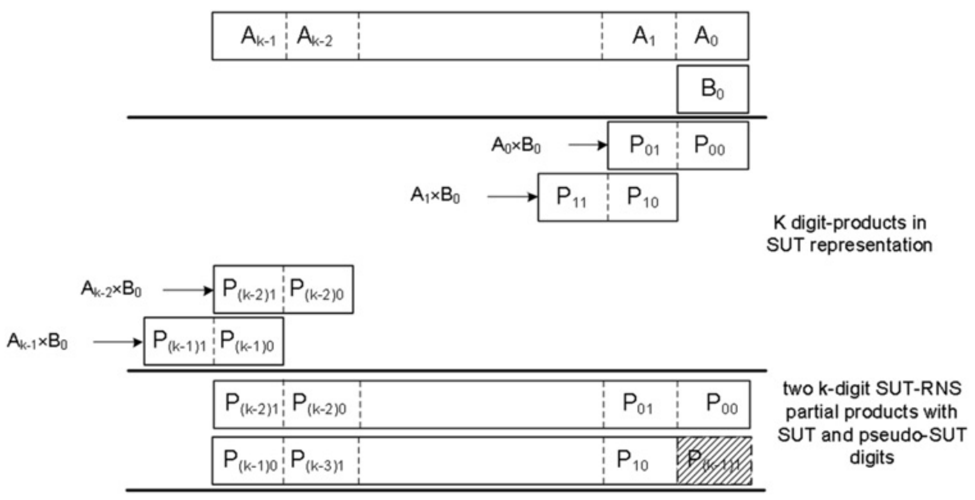


Fig. 4 *Rotating digit products resulted from multiplying B_0 by A and producing two final partial products in Stage 3*

In modulo $2^n - 1$ rotation, the format of SUT-RNS representation does not change. However, the value '-1' converts to '$+1$' and vice versa in the third stage for modulo $2^n + 1$ multiplication. To do this, for modulo $2^n + 1$ rotation, posibits, negabits and unibits of the rotated SUT digit are inverted and polarities of negabits and posibits are also reversed (as described in Section 2.2) that is forming a pseudo-SUT digit in modulo $2^n + 1$ rotation. Pseudo-SUT digit has a posibit in the highest position, a unibit in the first position and $(h - 1)$ negabits in other positions. Rotated SUT digit is shown by a shadowed block in Fig. 4. The shadowed block is still a SUT digit in modulo $2^n - 1$ multiplication and is a pseudo-SUT digit in modulo $2^n + 1$ multiplication which is called moduli rotation rules in the rest of the paper.

Underneath, we discuss our method to rotate the digit products resulted from multiplying A by B_i. The method is depicted in Fig. 5. In the figure, P_{j0} and P_{j1} indicate to double-digit SUT number which is produced by multiplying A_j and B_i, and their position numbers are $(i + j)$ and $(i + j + 1)$, respectively. In modulo $2^n - 1$ and $2^n + 1$ multiplications, the digit products are not rotated if their position numbers are less than k. Otherwise; the digit has to be rotated according to the moduli rotation rules. In this case, new position of each SUT digit is equal to its previous position minus k. For example, in Fig. 4, position of $P_{(k-1)1}$ is k [because $((k - 1) + 1) = k$]. So it is rotated and transferred to position zero (because $k - k = 0$). The rotated digits are shown by shadowed blocks in Fig. 5.

After rotating digit products, $2k$ partial products are finally created. We summarise the case in Fig. 6. To conclude, difference among the moduli set multipliers is because of their different moduli rotation rules in the third stages. Modulo 2^n multiplier does not include any rotation because it is a regular multiplier. In modulo $2^n - 1$ multiplier, the digits whose position numbers are more than k, are directly entered to their new positions in the structure. In modulo $2^n + 1$ multiplier, digits whose position numbers are more than k, are firstly inverted and polarities of posibits and negabits are changed before moving back to their new positions in the multiplier.

3.2 Partial-products summation through addition tree

Rotated SUT digits (resulted from stage 3) are shown by shadowed blocks in Fig. 6. As mentioned before, in modulo $2^n - 1$ multiplication, shadowed blocks are still SUT digits, whereas they are pseudo-SUT digits in modulo $2^n + 1$ multiplication. As in modulo $2^n + 1$ multiplication, there are both SUT and pseudo-SUT digits, we face to different types of SUT-RNS digit additions, whereas, in modulo $2^n - 1$ multiplication, we only face to normal SUT-RNS digit additions in the addition tree. There are three possible cases of digit additions in modulo $2^n + 1$ addition tree whose symbolic representation are depicted in Fig. 7. The cases include addition of: two SUT digits (Fig. 7*a*),

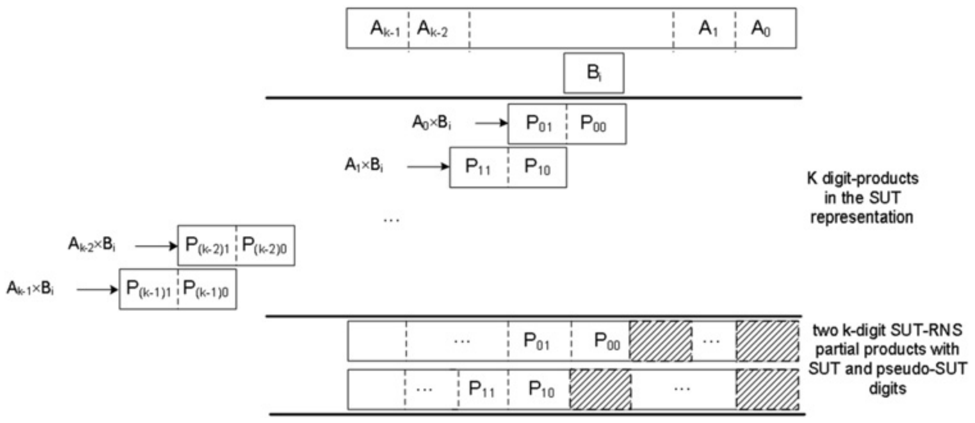
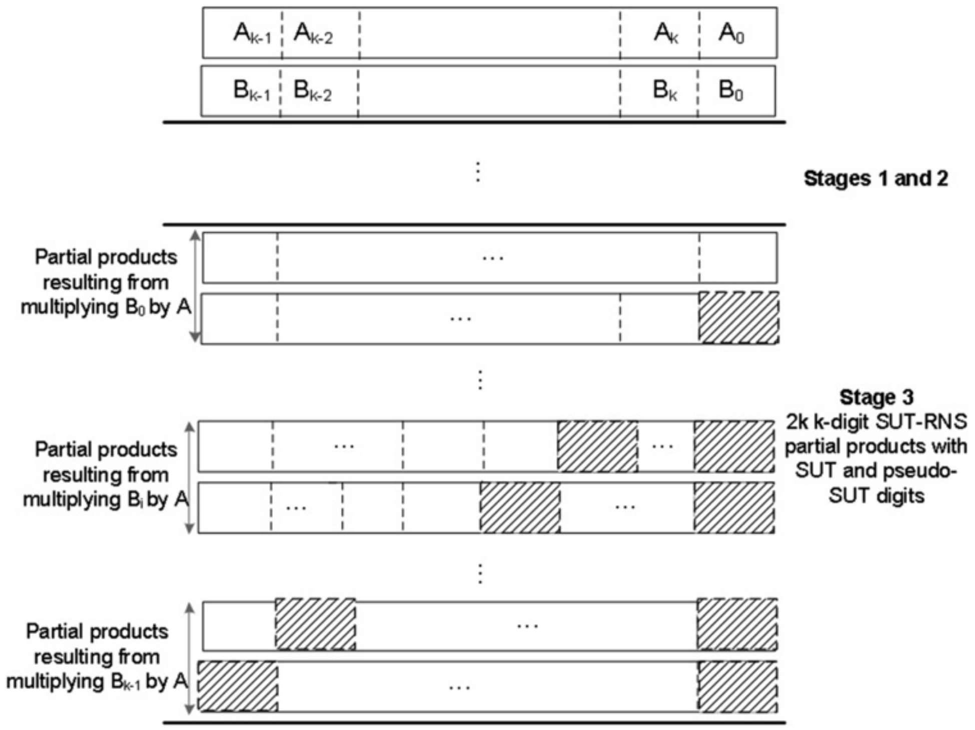


Fig. 5 *Rotating digit products resulted from multiplying B_i by A, and producing two final partial products in Stage 3*



A_{k-1}
A_{k-2}
A_k
A_0
B_{k-1}
B_{k-2}
B_k
B_0
Stages 1 and 2
Partial products resulting from multiplying B_0 by A
Stage 3
2k k-digit SUT-RNS partial products with SUT and pseudo-SUT digits
Partial products resulting from multiplying B_i by A
Partial products resulting from multiplying B_{k-1} by A

Fig. 6 *Resulted partial products from stage 3*

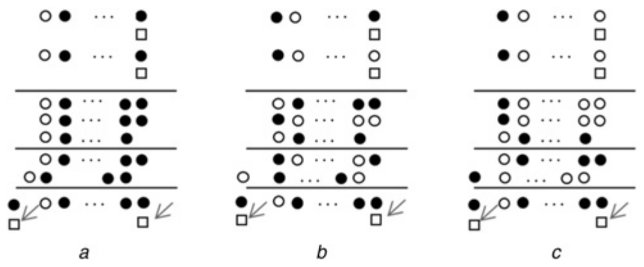


Fig. 7 *Notations of three existing digit-products additions for modulo $2^n + 1$ addition tree in stage 4*

a Two SUT digits
b One SUT and one pseudo-SUT digits
c Two pseudo-SUT digits

SUT and pseudo-SUT digits (Fig. 7*b*), and two pseudo-SUT digits (Fig. 7*c*). In the additions, firstly, two's complement addition of two unibits is performed. Then the three input operands are reduced to two numbers by *h*-bit carry-save-adder. If we apply an inverted encoding to negabits, illustrated in Table 1, standard carry-save-adder can handle any mix of equally weighted posibits and negabits [15]. Finally, two numbers are added together. The output posibit and negabit are passed to the next higher position digit. These output carries are added to the least significant posibit of the next higher position digit, to generate a posibit and a unibit as illustrated in Fig. 7. Three additions depicted in Fig. 7 are performed by a unique structure because, the output of each three different additions is a SUT digit.

Output carries resulted from the addition of *k*th digits of partial products are ignored in modulo 2^n multiplier addition tree in the last stage. However, these output carries are moved back in the least significant bit of the first digit in modulo $2^n \pm 1$. They are directly entered to modulo $2^n - 1$ multiplier as input carries, while they are moved back in modulo $2^n + 1$ multiplier after inverting.

3.3 Fault-tolerant structure

In the proposed SUT-RNS multiplication algorithm, the same components are utilised for the first two stages, that is, creating bit-products and producing digit-products, in three moduli $2^n - 1$, 2^n and $2^n + 1$. Therefore these stages can be performed for all moduli in the same way. The differences between these moduli multiplication are on rotating the digit products as well as modulo addition. However, as described above, modulo $2^n + 1$ rotation and addition are obtained by adding an inverter for each output carry coming back to the structure as a feedback. Therefore in our proposed multiplication algorithm, since we assumed SUT-RNS system with the same radix $r = 2^h$ for the moduli set, similar design strategies are applied to the moduli set. Unified design for the moduli set enables us to realise a reconfigurable multiplier which can accept operands of the three moduli. Basic structure of the reconfigurable multiplier is composed of 2^n SUT-RNS multiplier. Output carries of the multiplier are ignored, whereas these are reentered to the structure following modulo $2^n - 1$ and $2^n + 1$ multiplication rules described before. To allow 2^n SUT-RNS multiplier being able to do multiplication for modulo $2^n - 1$ and $2^n + 1$, the structure have feedback wires for both of the moduli and also inverters have to be added to modulo $2^n + 1$ multiplier. Such a reconfigurable multiplier lets fault-tolerant designs while it has a fewer hardware redundancies than full replication. In a reconfigurable SUT-RNS architecture composed of the three modular multipliers, one reconfigurable multiplier can be employed as a spare and whenever a fault is detected in one of the multipliers, the spare can be replaced to enable fault tolerance [17].

4 Simulation results and comparisons

To evaluate speed, area and power dissipation of the proposed SUT-RNS multipliers, four stage-pipelined structural VHDL

(VHSIC-Very high speed integrated circuits – Hardware Description Language) descriptions of SD-RNS and the proposed SUT-RNS multipliers for modulo $2^6 - 1$, modulo 2^6 and modulo $2^6 + 1$ with $h = 3$ have been first generated. Then, we synthesised them for 130 nm CMOS technology with the synopsys design vision tool. A typical condition (1.2 V, 25 °C) was considered. As stated in [10], radix-8 SUT-RNS modulo adder consumes significantly less power and less area than SD-RNS modulo adder. Since radix-2^h SUT-RNS multiplier is composed of radix-2^h SUT-RNS adders, radix-8 SUT-RNS is selected for implementation.

The results of total power, energy per operation and area for modulo 2^6, $2^6 - 1$ and $2^6 + 1$ multipliers are shown in Fig. 8. Delay, area and power results are given in ns, μm^2 and mw, respectively. The syntheses were performed for different delays from 2 to 5 ns.

The achieved experimental results shown in Fig. 8 reveal that power, energy/operation and area of the SD-RNS and also SUT-RNS multipliers are reduced with reducing their clock frequencies. It means that to reach the least delay (best clock frequency), total power, energy and area will be increased. However, for different clock frequencies, the proposed radix-8 SUT-RNS modulo multiplier outperforms area, power and energy/operation of the SD-RNS modulo multipliers for each module. This is because, SUT-RNS employs a hybrid redundant representation in each module whereas SD-RNS is a fully redundant representations for each moduli set. Besides, SUT-RNS multiplier delays are very close to SD-RNS. This shows that SUT-RNS representation and consequently proposed general SUT-RNS multiplier are appropriate for modular operations.

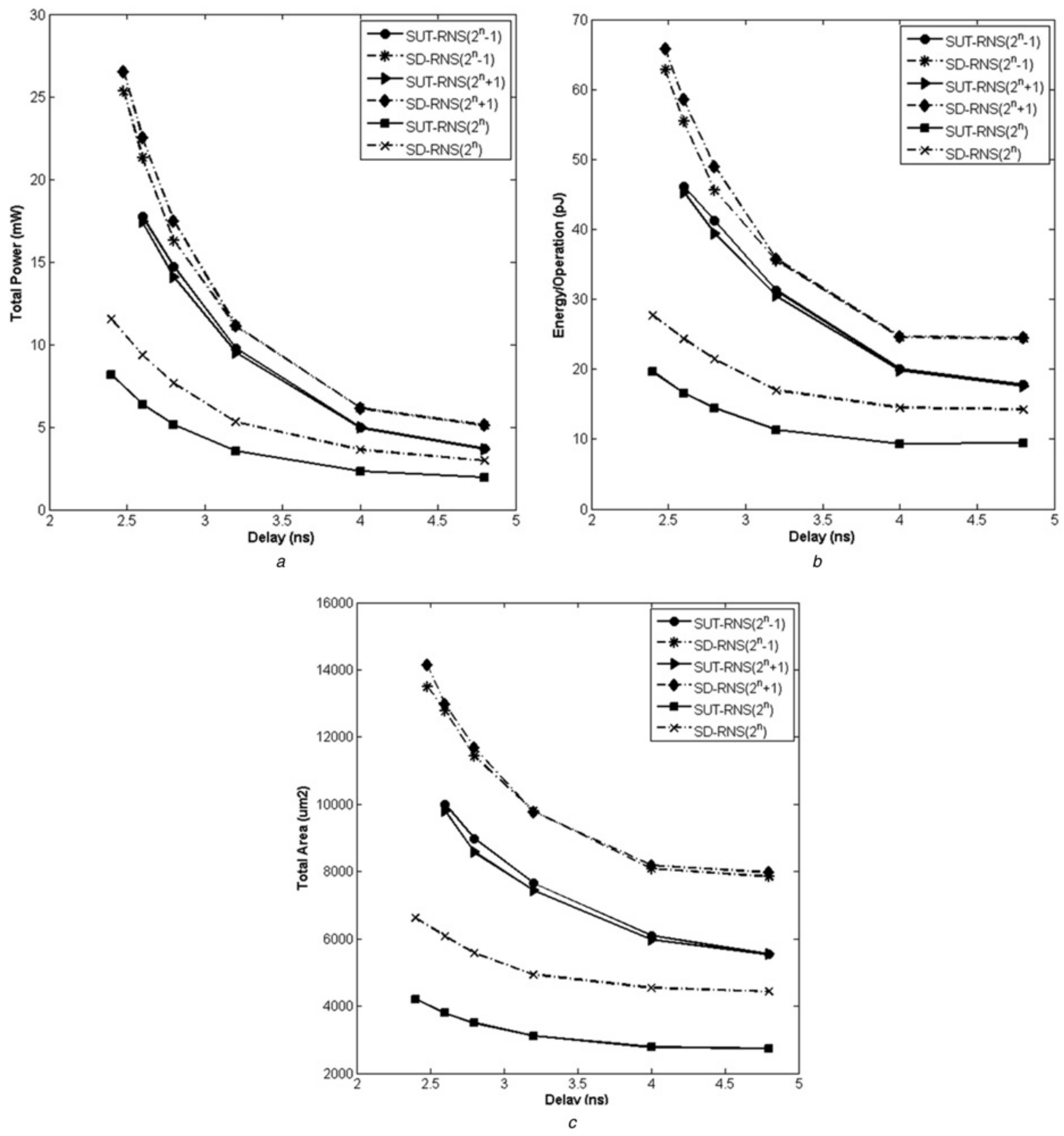


Fig. 8 *Synthesis results of SD-RNS and radix-8 SUT-RNS multipliers for moduli set $\{2^6 - 1, 2^6, 2^6 + 1\}$ against different delays*

a Total power
b Energy per operation
c Area consumption

The results indicate that radix-8 SUT-RNS multiplier outperforms power of the SD-RNS multiplier about 29, 17 and 23% for modulo 2^6, 2^6-1 and 2^6+1 multipliers in their maximum clock frequencies, respectively. Besides, SUT-RNS multiplier consume about 30, 27 and 31% less energy/operation than SD-RNS multipliers for modulo 2^6, 2^6-1 and 2^6+1 multipliers, respectively. Area comparison of the multipliers shows that modulo 2^6, 2^6-1 and 2^6+1 SD-RNS multipliers have 36, 22 and 24% more area than modulo SUT-RNS multipliers. Besides, SUT-RNS reaches the near speed of the most high-speed RRNS multiplier. The minimum delays of the proposed multipliers are about 2.4 ns for modulo 2^6, and 2.6 ns for modulo 2^6-1 and 2^6+1 SUT-RNS multipliers. Whereas, the minimum delays of the SD-RNS multiplier for modulo 2^6 is 2.4 ns (the same as SUT-RNS modulo multiplier). While the minimum delays of SD-RNS multiplier for modulo 2^6-1 and 2^6+1 are 2.48 ns which is 0.12 ns less than SUT-RNS modulo multipliers. This indicates that area usage reduction and power reduction of SUT-RNS is obtained at the cost of 0.5% speed reduction.

The results in Fig. 8*a* indicate that power consumption of SUT-RNS multipliers for modulo 2^6-1 and 2^6+1 are very comparable because of their similar structures. Similarly, energy per operation and area consumption of each SUT-RNS multiplier, shown in Figs. 8*b* and *c*, for moduli set $(2^6-1,\ 2^6+1)$ are the same comparable. This indicates that we obtain the symmetric structure for the proposed moduli multipliers.

From the proposed algorithm, it is concluded that the first and second multiplication stages are completely similar for the three moduli set of $\{2^n-1,\ 2^n,\ 2^n+1\}$ multipliers. The difference in the third stages of the moduli set multipliers that is because of their different digit rotation rules. In this stage

- Modulo 2^n multiplier does not include any rotation because it is a regular multiplier.
- In modulo 2^n-1 multiplier, the digits whose position numbers are more than k, are directly entered to their new positions in the structure.
- In modulo 2^n+1 multiplier, the digits whose position numbers are more than k, are first inverted before moving back to their new positions in the multiplier.

Similarly, the only difference among the fourth stages of the moduli multipliers is related to the rotation rules of output carries in kth digits additions. The output carries resulted from the kth digits additions

- Are ignored in modulo 2^n multiplier.
- Are directly entered to modulo 2^n-1 multiplier as input carries.
- Are moved back in modulo 2^n+1 multiplier after inverting.

We can conclude that modulo 2^n+1 multiplier has a number of inverters more than modulo 2^n-1 multiplier. Besides, modulo $2^n\pm1$ multipliers have back wires for digit rotations and end-around-carries rather than 2^n multiplier. Therefore the proposed SUT-RNS multiplication algorithm results in unified designs for the three moduli of $\{2^n-1,\ 2^n,\ 2^n+1\}$, thus provide support for fault-tolerant RRNS processors. The unified design leads to the possibility of providing reliability with low hardware redundancy by using reconfigurable modular multipliers that can process inputs for different moduli. A reconfigurable modular multiplier enables fault-tolerant designs with much lower hardware redundancy than full replication. One way to do this is to implement more number of multipliers for three moduli set $\{2^n-1,\ 2^n,\ 2^n+1\}$ and then configure them to perform three different modular multiplication, with extra ones kept as spare. If a fault occurred in one of the available multipliers, one of the spare can be configured accordingly and employed instead of the faulty multiplier [17].

5 Conclusion and summary

RRNS has been proved an appropriate number representation to accelerate frequent multiplications. SUT-RNS is a continuous choice between ordinary (non-redundant) and full redundant RNS representation that provides flexible trade-off between area and speed. In this paper, we have generalised SUT-RNS multiplication algorithm for the moduli set $\{2^n-1,\ 2^n,\ 2^n+1\}$ multipliers. From synthesis evaluation, it is concluded that the proposed generalised SUT-RNS is more efficient than SD-RNS. The results indicate that radix-8 SUT-RNS multiplier for the moduli set $\{2^6-1,\ 2^6,\ 2^6+1\}$ outperforms area, power and energy/operation of the SD-RNS multiplier. The proposed SUT-RNS multipliers for the moduli set $\{2^6-1,\ 2^6,\ 2^6+1\}$ consume at least 17 less power, 27 less energy/operation and 22% less area than SD-RNS multipliers. These achievements are obtained at cost of 5% delay increasing that shows SUT-RNS reaches nearly speed of the most high-speed RRNS multiplier. Moreover, we demonstrated our method allows a unified design for moduli set $\{2^n-1,\ 2^n,\ 2^n+1\}$ multipliers, which causes to design fault-tolerant SUT-RNS multipliers with low hardware redundancy. We conclude that the proposed generalised SUT-RNS multiplier is an appropriate RRNS multiplier to achieve an appropriate tradeoff among area, speed and power.

6 References