J. Inst. Eng. India Ser. B (February 2021) 102(1):59–74
https://doi.org/10.1007/s40031-020-00518-w

ORIGINAL CONTRIBUTION

Efficient Implementation of Cryptographic Arithmetic Primitives Using Reversible Logic and Vedic Mathematics

V. G. Kiran Kumar[1] · C. Shantharama Rai[1]

Received: 10 December 2019 / Accepted: 2 November 2020 / Published online: 8 January 2021

Abstract The concept of IOT (Internet of Things) prevalent in the era, an enormous increase in the implementation of low resource devices with limited resources to implement various arithmetic primitives for cryptographic applications. Most of the National Institute of Standards and Technology NIST approved conventional security algorithms like the IDEA, RSA etc. involve simple operations like addition, multiplication and complex operations like addition-modulo, multiplication-modulo etc. such algorithms may not be routed to the low resource devices. Even though the NIST approved algorithms may be fitted into the low resource devices their erformance may be a concern. An extensive research has been carried out on designing the cryptographic arithmetic structures over the years, the arithmetic modules have been implemented using different design styles, logic, algorithms and aspects. In this paper the reversible logic and Vedic mathematics have been combined to implement efficient cryptographic arithmetic primitives. A considerable amount of improvement in the performance parameters like Area, power dissipation and delay have been analyzed by the implementation and synthesis.

Keywords Internet of Things · Reversible logic · Computer arithmetic · Cryptography · Modular multiplication

✉ V. G. Kiran Kumar
kiranvgk@gmail.com

C. Shantharama Rai
csraicec@gmail.com

[1] Department of Electronics and Communication Engineering, A J Institute of Engineering and Technology, Kottara, Mangalore 575006, India

Introduction

With the emergence of IoT (Internet of Things) there has been increased demand for resource-constrained devices, and hence, the design of such computing devices is a great challenge [1]. The Moore's law is based on the observation that the number of transistors in a chip is doubled for every 2 years. The recent design technology has been on par with the Moore's law. Increase in transistor count, a measure of IC complexity, increases area, power consumed and power dissipated. Power consumption is proportional to the complexity of the chip. Timing also becomes a critical factor. Timing analysis is crucial while designing a system; time taken by the design to give the desired output, time taken to pass data from one chip to another. In the modern day technology there is an effective communication between high-end servers to tiny devices like sensors and RFID, thus posing another major challenge to security. Thus, a design of effectively efficient (the term effective means highly secured, while efficient means the speed of encryption of the plaintext and area and power-efficient processor) cryptographic algorithms would be a solution to the problem [2].

Cryptographic algorithms are classified as symmetric-key algorithm and asymmetric-key cryptographic algorithm [3]. Same key is used for both encryption and decryption in a symmetric-key algorithm, whereas different keys for encryption and decryption are used in asymmetric-key cryptography. Numerous public-key cryptographic algorithms are available in the literature which comprises modular arithmetic modules [4, 5], such as modular addition, multiplication, inversion and exponentiation. Recently, numerous cryptographic algorithms have been proposed based on modular arithmetic which are scalable, do word-based operations and efficient in various aspects.

The modular arithmetic is an important computation, and its implementation measures the overall performance of the cryptographic processor. Thus, better results can be obtained by designing efficient arithmetic modules such as modular addition, multiplication, exponentiation and squaring.

With reversible logic emerging one of the promising computing technologies and an important research area in the fields like quantum computation, nanotechnology low-power VLSI design and so on, benefits of implementation of reversible logic are reduction of constant inputs, garbage outputs and constant inputs [5]. Reversible logic-based design reduces heat dissipation, has high circuit densities and basic gates having higher efficiency gate level and design level by 50% and 33%, respectively [6]. An efficient implementation of cryptographic arithmetic circuits using reversible logic and Vedic mathematics proposed that the performance has been analyzed and compared with state-of-the-art literature.

Background and Motivation

Over the years abundant research has been carried out in implementation of arithmetic circuits; here few designs that have been studied have been discussed.

Banerjee et al. [7] presented a survey paper on the design of reversible logic-based arithmetic circuits like adder/subtractor, multiplier and squarer, and a comparative analysis of these circuits based on auxiliary inputs, garbage outputs and quantum costs has been made. Thakral et al. [8] presented 1-bit ALU design using optimization metrics like auxiliary inputs, garbage outputs quantum costs, number of gates and power dissipation the ALU being realized using carry save adder block. This paper also presents different reversible logic gates used in current methods of ALU designs and brief overview of these ALU implementations.

Deeptha et al. [9] proposed an 8-bit ALU based on reversible logic, the 1-bit ALU consisting of control and the adder unit is designed, and these blocks are cascaded to design an 8-bit ALU. The 1-bit ALU is designed using control output gate (COG) for the control circuitry, while the adder unit is designed using modified HNG (Haghparast and Navi Gate). The designed arithmetic unit has been compared with other works found reduced gate count and transistor count, and the propagation delay was also lesser by 33.41%. Amirthalakshmi et al. [10] designed a low-power 8-bit ALU for nanoprocessor based on reversible logic using single electron transistor (SET). The basic blocks of ALU are designed using Dual key Gate Pair (DKGP), one of the many port gates. The blocks are designed using SET technology and compared with CMOS technology. It concludes that SET technology has adopted for low-power digital applications. Thakral and Dipali [11] presented a fault-tolerant ALU based on high functionality. The proposed design compared with existing designs resulted in improvement of functionality by 25% and a quantum cost reduction of 40%. Prabhu et al. [12] proposed a 8 × 8 Vedic multiplier designed using Nikhilam sutra and Urdhva-Tiryakbhyam sutra. The proposed multiplier that has been compared with modified booth multiplier found that for a 8, 6 and 25 bases the proposed multiplier is faster by 32.54%, 37.29% and 51.28%, respectively. Deepa and Marimuthu [13] proposed a Yaduvanam sutra-Vedic square and multiplier architecture. It offers significant improvement in speed, area-delay product and power-delay product compared to other various architecture like array multiplier, braun multiplier, shift and add multiplier dada multiplier, Urdhva multiplier and nikhilam multiplier. Liu and Li [14] implemented 258-bit multiplier based on KO-3 algorithm. A 4-stage pipelined-modular-multiplier of 256-bit is implemented by combining Montgomery modular algorithm and KO-3 algorithm. The design implemented in Virtex-6 FPGA and compared with existing designs showed a better performance in terms of LUT, latency and area-delay product.

Reversible Logic Gates

Reversible logic [15] is an exceptional computational method that does not eliminate the information. The use of reversible logic gates would retrieve the inputs from the outputs. In a reversible logic circuit [16] the number of inputs and the number of outputs are equal, i.e., the logic has one-to-one mapping of input and output vectors; thus, the input vector can be recovered from the output vector.

Figure 1 shows the general reversible logic circuit which consist of n-inputs and is called an '$n \times n$' logic circuit with top k to $k-1$ lines giving the output $P = Y$ and the bottom $k-1$ lines gives the output $Q = f(X)$.

A simple two-input gate is represented as in Fig. 2.

Figure 3a depicts a 2*2 Feynman gate that implements the following logic functions:

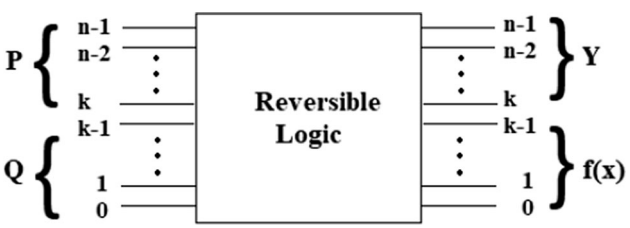

Fig. 1 Reversible logic circuit

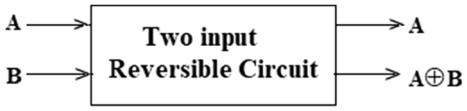

Fig. 2 Two-input reversible circuit

$P = A,$ and $Q = A \oplus B.$

Figure 3b shows 3*3 peres gate, and it implements the following logic functions:

$P = A,\ Q = A \oplus B$ and $R = AB \oplus C.$

Figure 3c shows 4*4 HNG (Haghparast and Navi Gate) gate, and it implements the following logic:

$P = A,\ Q = B,\ R = A \oplus B \oplus C$ and $S = (A \oplus B)C \oplus AB.$

Cryptographic Arithmetic Units

This section describes various cryptographic modules.

Adder Using Reversible Logic

A full adder is the standard arithmetic component in most of the cryptographic circuits and can be implemented using two peres gates as depicted in Fig. 4. Similarly, an *n*-bit adder [17] can be implemented by connecting such components in cascade.

Public-key cryptographic algorithms like idea, RSA, Diffie–Hellman use the arithmetic operations like addition modulo and multiplication modulo that has prompted to implement various arithmetic logics in this paper by combining reversible logic and Vedic maths and others in the most efficient way.

Multiplier Using Reversible Logic and Vedic Mathematics

The system performance depends on the longest delay, multiplier has the longest delay, and in cryptographic arithmetic, multiplier is the key primary operation. Designing the mathematical operation in the most efficient way is the challenge. A multiplier is implemented using reversible logic and the Vedic mathematics [18–20].

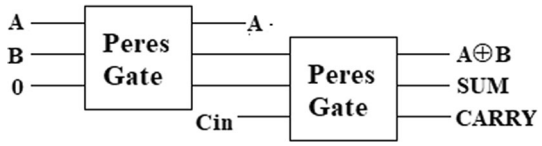

Fig. 4 Full adder using peres gates

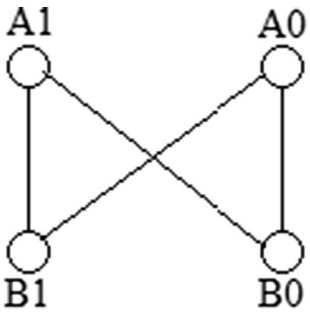

Fig. 5 A 2 × 2 Vedic

Figure 5 shows how 2 × 2 Vedic multiplier is implemented. This technique will be extended for any n-bit number.

$$P_{m0} = a_0 \times b_0$$
$$P_{m1} = a_1 \times b_0 + a_0 \times b_1$$

A 4 × 4 Vedic multiplier implemented in Fig. 6 is designed by top-down design approach. A 2 × 2 Vedic multiplier is implemented first and is the leaf module for any $N \times N$ multiplication. A 4 × 4 multiplication is implemented using the 2 × 2 multipliers, and multiplicand and multiplier are divided into two groups of 2-bits each. 2 × 2 multiplication is performed on these groups, and the partial products are obtained. These partial products are then modified to get an 8-bit product term. The adder logic using reversible logic is implemented.

The reversible multiplier [21] implemented below consists of two stages. In the first stage shown in Fig. 7 the partial products are generated, and using these partial products the final product term is obtained in the next stage shown in Fig. 8. A 4 × 4 reversible multiplier uses 16 peres gates to produce partial products and combination of 12 peres and HNG gates to compute the resultant.

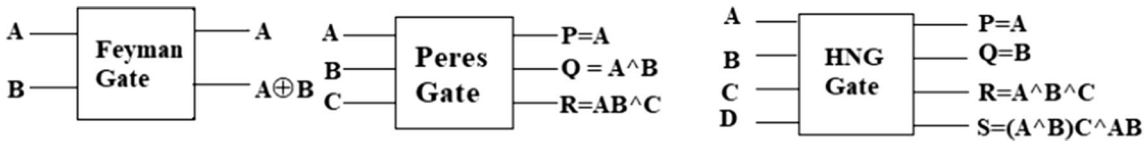

Fig. 3 Reversible logic gates

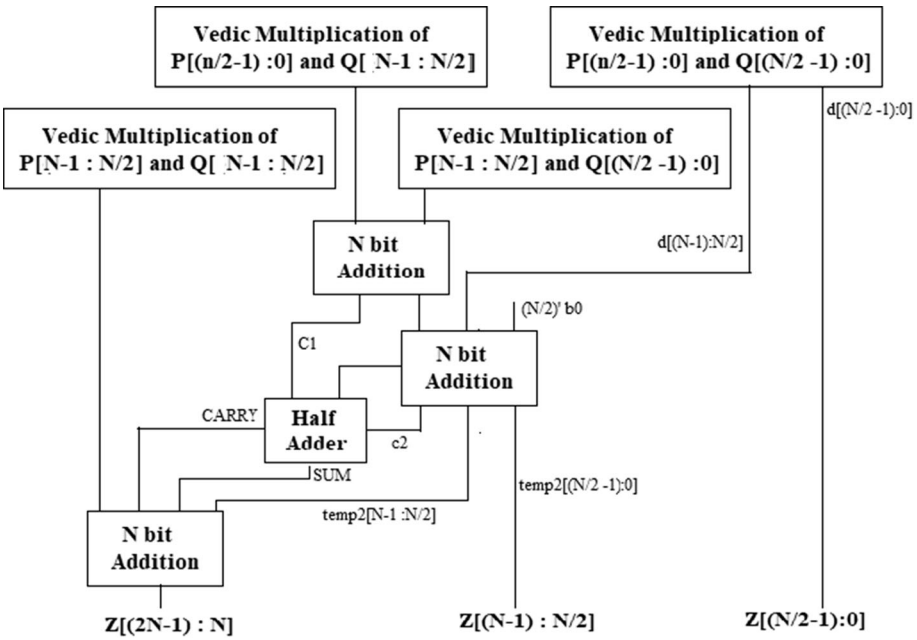

Fig. 6 A 4 × 4 Vedic multiplier using reversible logic

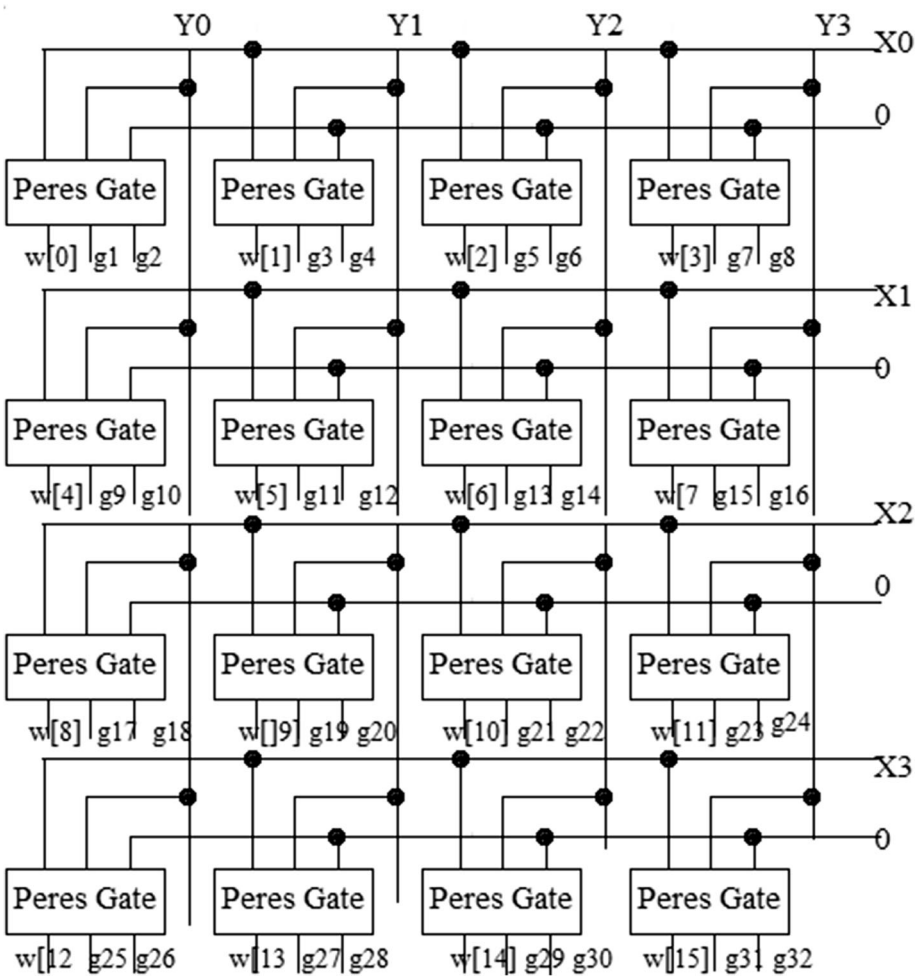

Fig. 7 The partial products generation for a 4 × 4 reversible multiplier

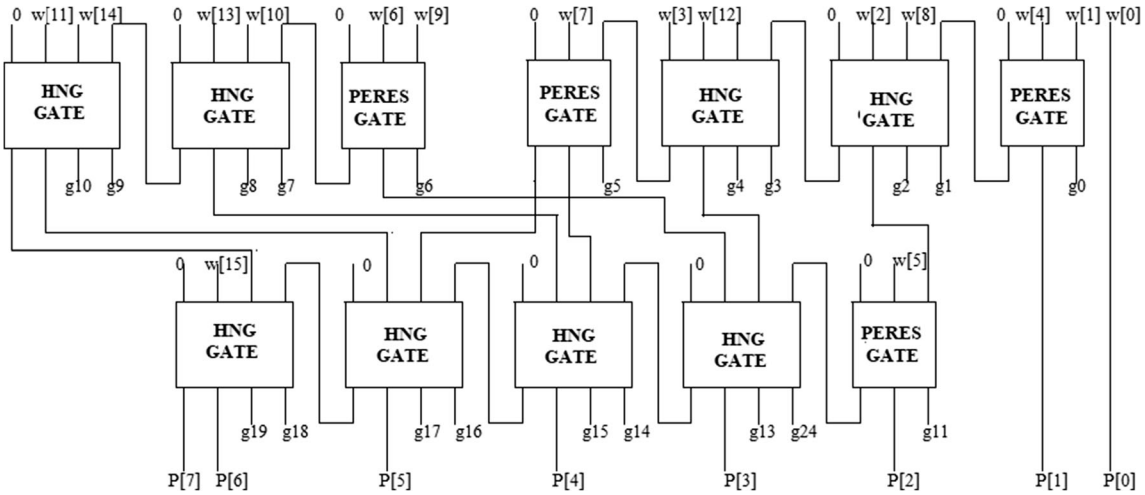

Fig. 8 Reversible multiplier product generation stage

Vedic Divider

Division is one of the essential operations of cryptographic arithmetic. The division of two common numbers is the way toward ascertaining the quantity of times one number is contained inside each other. Divider is fundamental equipment utilizing in rapid and progressed advanced signal processing (DSP) units. It is most vital part in high exactness radar innovation and cryptography. The Vedic mathematics consist of sixteen sutras. Vedic algorithms implementation is easy, and processing is done fast. These sutras help in reducing the iterations for complex circuitry. Vedic mathematics is part of four Vedas. It is part of Sthapatya Veda, an upa-veda of Atharva-Veda. The Paravartya Yojayet algorithm is applied to develop a high-performance Vedic divider [22]. This sutra reduces the total power consumption and delay to a considerable extent.

"Paravartya-Yojayet—Transpose and Apply" is used to implement division algorithm for decimal numbers shown in Table 1. Easier and logically simple implementation could be implemented through decimal division. Table 2 shows the logical implementation of Vedic divider.

Modular Operations

Most of the basic public-key cryptosystems use modular operations. In this section, few modulo multiplication algorithms that have been implemented are discussed [23–25].

The algorithms are implemented using Vedic/reversible multiplier logic for the computation of product $x \times y$ and then computed $m = (x \times y) \bmod n$.

Classical Algorithm

Classical algorithm or modulo multiplicative inverse algorithm implemented is comprised of simple operations addition, shift operation and subtraction. The time for the modulo computation is based on the computation of the remainder and thus is determined by the number of iterations in the loop. For a number in the range from 0 to 65,536 there could be 18 iterations. The multiplication

Table 1 Paravartya sutra to decimal number system

Divisor			Dividend				
1	1	3	1	3	9	0	5
	− 1	− 3		− 1	− 3		
					− 2	− 6	
						− 4	− 12
			1	2	4	− 10	− 7
			1	2	3		6
			$Q = 124 - 1 = 12$			$R = 113 - 107 = 6$	

Table 2 Vedic divider for binary

111	1	1	1	1	0	1	1	0	1
		1	0	0					
		0	0	0					
			0	0	0				
			0	0	0				
				1	0	1			
				0	0	0			
					0	0	0		
					1	0	1		
						1	0	1	
						0	0	0	
							0	0	0
							1	0	1
Result bit	1	0	1	1	1	1	1	1	0
Carry bit							1	0	1
Sign bit	0	0	0	1	0	1	0	1	

performed by the operands is implemented using Vedic multiplier/reversible logic. The division algorithm is given by Vedic divider, and thus, a subtractor is needed to perform subtraction depending on the number of iterations and also on multiple precision.

Algorithm 1: Modulo multiplicative inverse algorithm

INPUT : x,y,n with 0≤x≤n , 0≤y ≤n; |log2 x| = k
OUTPUT : p = x y mod n

Step1 : p = 0
Step 2: for 1= k-1 ; i ≥0; i=i-1 do
a. p = 2p+x.y
b. q = |p/n|
c. p = p - qn

Algorithm 1 shows the pseudocode. Table 3 shows the steps for modulo multiplication algorithm.

Barrett Reduction

The algorithm was introduced by Barrett to compute $p = x\ y \bmod n$ without resorting to the trail division.

The equation $z \bmod n$ can be approximated as

$$\left\lfloor \frac{z}{n} \right\rfloor = \frac{\left\lfloor \frac{z}{l^{k-1}} \right\rfloor \left\lfloor \frac{l^{k-1} l^{k+1}}{n} \right\rfloor}{l^{k+1}} = \frac{\left\lfloor \frac{z}{l^{k-1}} \right\rfloor \left\lfloor \frac{l^{2k}}{n} \right\rfloor}{l^{k+1}}$$

The precomputed value $\mu = |l^{2k}/n|$ is a constant which can be calculated and stored. l is the base value of operand. The x and y are both 16 bit; hence, the product is 32 bits, while n is 16 bits. The architecture consists of multipliers adders and subtractors, and division can be performed using shift operations reducing the complexity of the architecture. Algorithm 2 describes the pseudocode for the

Table 3 Numerical example of modulo multiplicative inverse algorithm

Input:
$P = (10391)_{10}$ $N = (16)_{10}$

$s1 = (16)_{10}$

$s2 = (10391)_{10}$

$d1 = (0)_{10}$

$d2 = (1)_{10}$

Q	*s1*	*s2*	*R*	*d1*	*d2*	*d*
0	*16*	*10391*	*16*	*1*	*0*	*0*
649	*10391*	*16*	*7*	*0*	*1*	*1*
2	*16*	*7*	*2*	*1*	*-2*	*-2*
3	*7*	*2*	*1*	*-2*	*7*	*7*
2	*2*	*1*	*0*			

Since Remainder = 0 and d2 is positive, $z^{-1} \bmod b = 7$

Barrett algorithm. Table 4 describes the stepwise implementation of the Barrett reduction.

Algorithm 2: Barett Reduction algorithm

INPUT : x,y,n with $0 \le x \le n$, $0 \le y \le n$; $2^{k-1} < p < 2^k$
$\mu = |l^{2k} / n|$ *and* $l > 3$
OUTPUT : $p = x\,y \bmod n$
Step1 : $r = x.y$
Step 2: $q = |r/l^{k-1}|.\ \mu$
Step 3 : $q = q/l^{k-1}$
Step 4 : $r_1 = r \bmod l^{k+1}$ $r_2 = q.p \bmod l^{k+1}$
Step 5: $u = r_1 - r_2$
Step 6: if $u < 0$ *then* $u = u + l^{k+1}$
Step 7: while $u \ge n$
a. $u = u - n$

Montgomery Algorithm

Montgomery multiplication [26, 27] is one of the most widely used algorithms for computing the result of *xy mod n*. The computation is implemented in two steps. First the result is obtained in Montgomery domain and calculated as

$$\bar{z} = \text{Montgomery}\ (x, y, n) = a \cdot b \cdot r^{-1} mod\, n$$

The second step is that the result will be transformed back to standard representation by performing an additional multiplication.

$$z = \text{Montgomery}\ \left(\bar{z}, r^2 mod\ n, n\right) = a \cdot b \bmod n$$

The multiplication operation is performed using Vedic multiplier/reversible multiplier. For Montgomery domain hardware implementation $r = 2^k$ where k—number of bits. The algorithm is modified and $r = 16^4$. Here the hardware is reduced by two multipliers by appending four hexadecimal zeros. The next step of converting the standard representation can be done by implementing the inverse operation (Table 5).

Algorithm 3: Montgomerry algorithm

INPUT : x,y,n with $0 \le x \le n$, $0 \le y \le n$; $2^{k-1} < p < 2^k$ *and* $s = 2t+1$
$r = 2^k$, $n' = -s^{-1} \bmod 2$
OUTPUT : $p = a\ b\ r^{-1} \bmod n$
Step1 : $u = 0$
Step 2: for $1 = 0$; $i < k$; $i = i+1$ *do*
a. $u = u - n$
b. $q = n' u_0 \bmod 2$
c. $u = u + qs$
d. $u = u/2$

Step 3: if $u \ge s$ *then* $u = u - s$

Simulation Results

The modeling of the blocks (arithmetic units) is carried out using Verilog HDL. The functional verification and simulations are performed using Xilinx Vivado Design Suite and Vivado Integrated Design Environment (IDE).

Figures 9 and 10 show the RTL schematic of the 32-bit adder using basic gates and reversible logic gates.

Figures 11, 12 and 13 show the RTL schematic of Vedic multiplier using standard logic and reversible multiplier and Vedic multiplier using reversible logic.

Table 4 Numerical example of Barrett reduction

Inputs:

$z = (696)_{10} = (1010111000)_2$ $p = (15)_{10} = (1111)_2 \acute{A} = \{0010, 1011, 1000\} N = 2\mu = 2^4 \mod 15 = (1)_{10} = (0001)_2$

	$H = \left\{0, \acute{A}[2]\right\} = 00010$	$H = H + \left\{0, \acute{A}[1]\right\}$	$[0] = \acute{A}$ $[0] + H = 10101\,\acute{A}[0] = 00101 + 0001 = 00110$
		$= 01101$	
		$H = H \ll 1 = 11010$	
i = 0	$H = H \ll 1 = 00100$	$H = 01010 + 0001$	
		$= 01011$	
i = 1	$H = H \ll 1 = 01000$	$H = H \ll 1 = 01110$	
	$H = H \ll 1 = 10000$		
i = 2	$H = 00000 + 0001$	$H = H \ll 1 = 01110$	
	$= 00001$	$H = H \ll 1 = 11100$	
i = 3	$H = H \ll 1 = 00010$	$H = 01100 + 0001$	
$z \bmod p = (0110)_2 = (6)_{10}$		$= 01101$	

Table 5 Numerical example of montgomery multiplication

Inputs:	$z = (ADEF)_{16}$ $y = (1348)_{16}$ $p = (2897)_{16}$
Steps:	$p' = -p^{-1} \mod b = -7 \bmod 16 = (9)_{16}$ $a = 0$
	$a = ((0 + (F \times 8))9) \bmod 10 = (8)_{16}$
i = 0	$t = \left(0 + (F \times 1348) + \frac{8 \times 2897}{10}\right)/10 = (265F)_{16}$
	$a = ((F + (E \times 8))9) \bmod 10 = (7)_{16}$
i = 1	$t = \left(265F + (E \times 1348) + \frac{7 \times 2897}{10}\right)/10 = (2507)_{16}$
	$a = ((7 + (D \times 8))9) \bmod 10 = (7)_{16}$
i = 2	$t = \left(2507 + (D \times 1348) + \frac{7 \times 2897}{10}\right)/10 = (23BD)_{16}$
	$a = ((D + (A \times 8))9) \bmod 10 = (5)_{16}$
i = 3	$t = \left(23BD + (A \times 1348) + \frac{5 \times 2897}{10}\right)/10 = (1AF8)_{16}$
$zyR^{-1} \bmod p = (1AF8)_{16}$	

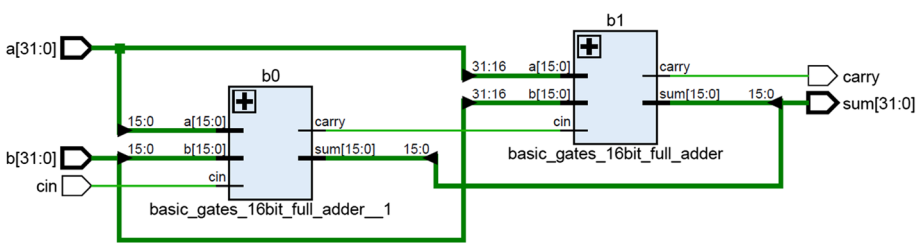

Fig. 9 RTL schematic of 32-bit adder using basic gates

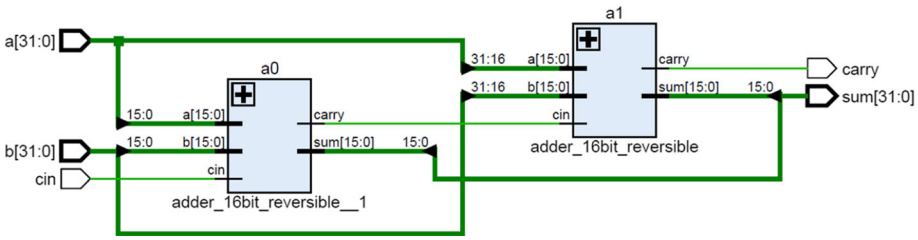

Fig. 10 RTL schematic of 32-bit adder using reversible gates

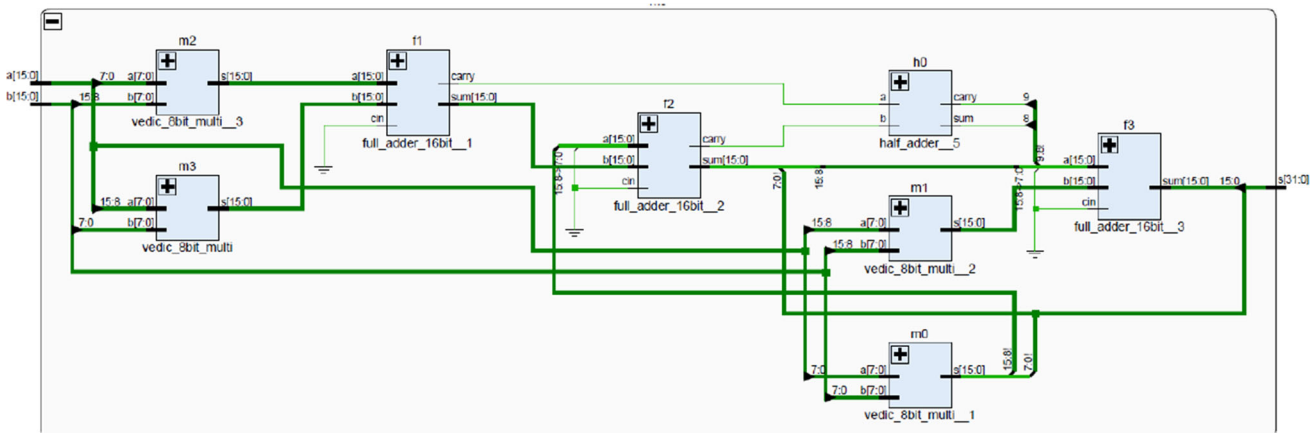

Fig. 11 RTL schematic of Vedic multiplier using standard logic

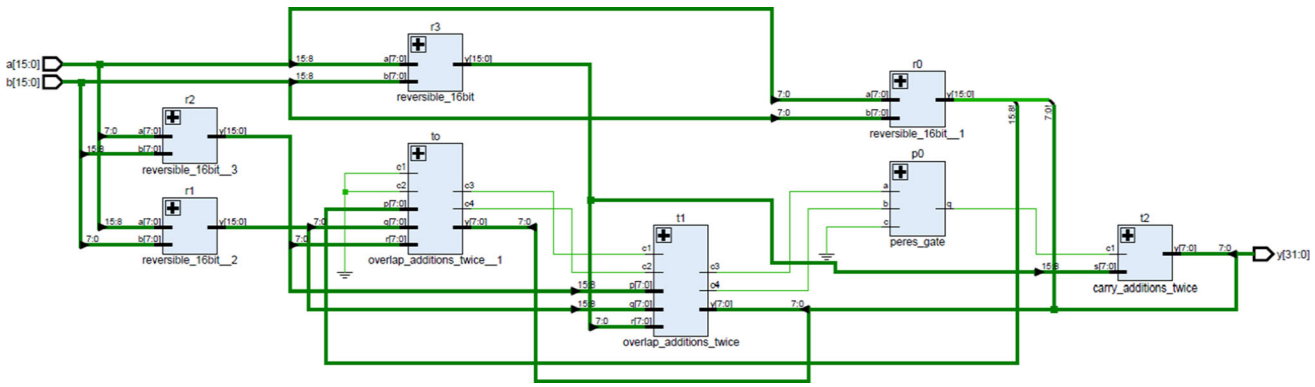

Fig. 12 RTL schematic of reversible multiplier

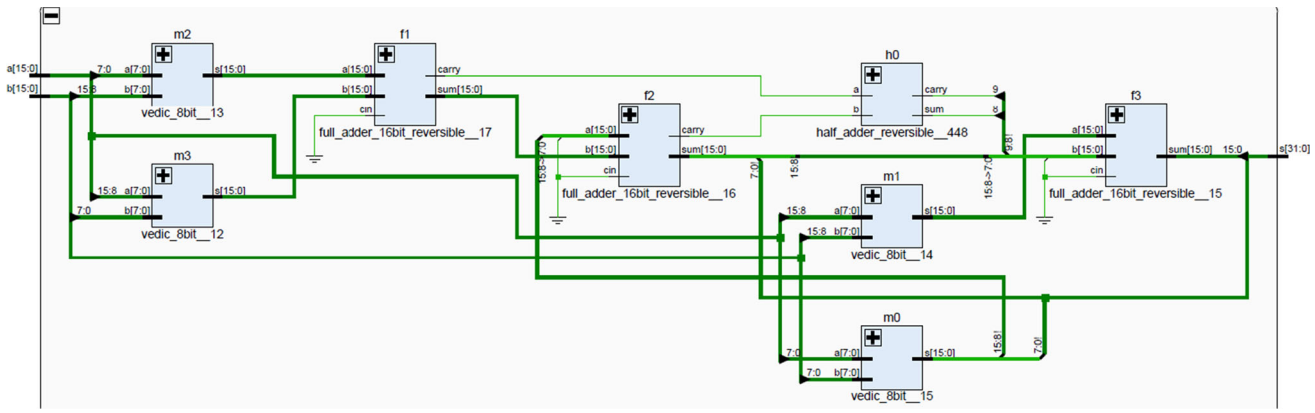

Fig. 13 RTL schematic of Vedic multiplier using reversible logic

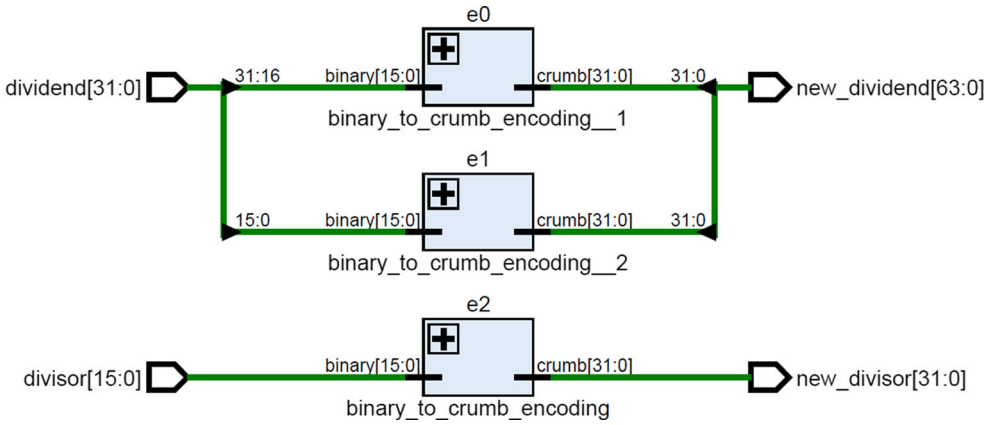

Fig. 14 RTL schematic of crumbs encoding

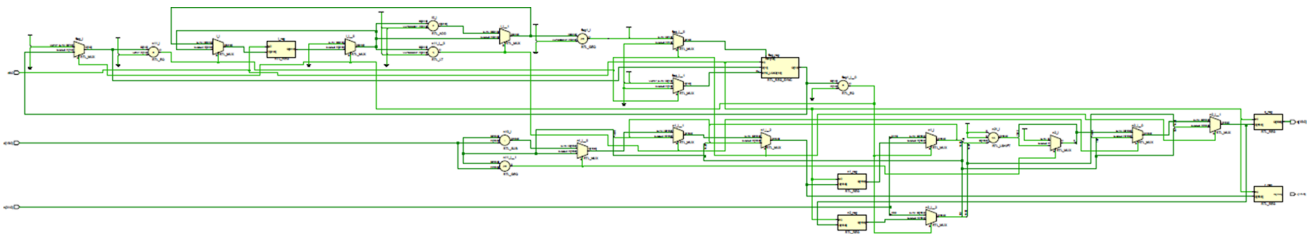

Fig. 15 RTL Schematic of Vedic divider using standard gates

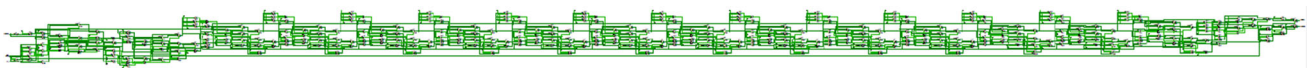

Fig. 16 RTL schematic of Vedic divider using reversible gates

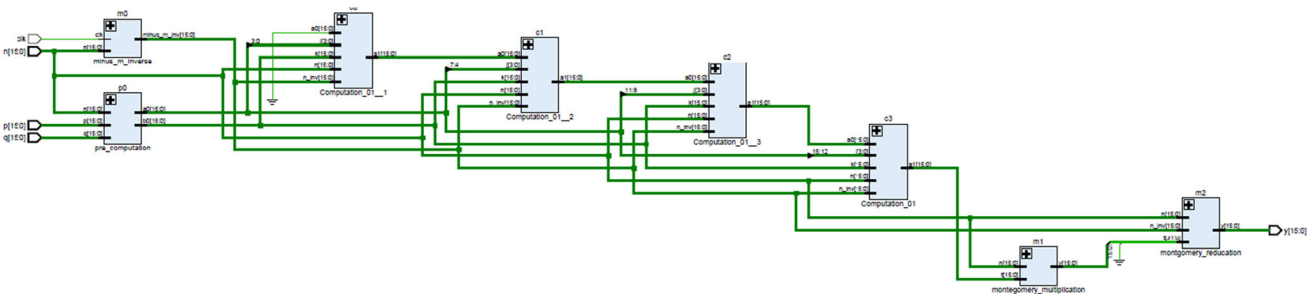

Fig. 17 RTL schematic modified montgomery modular algorithm

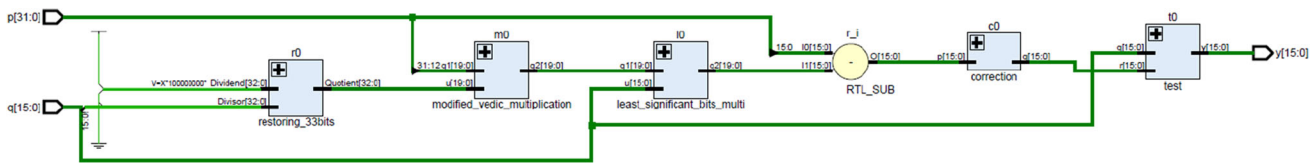

Fig. 18 RTL schematic modified Barrett reduction algorithm

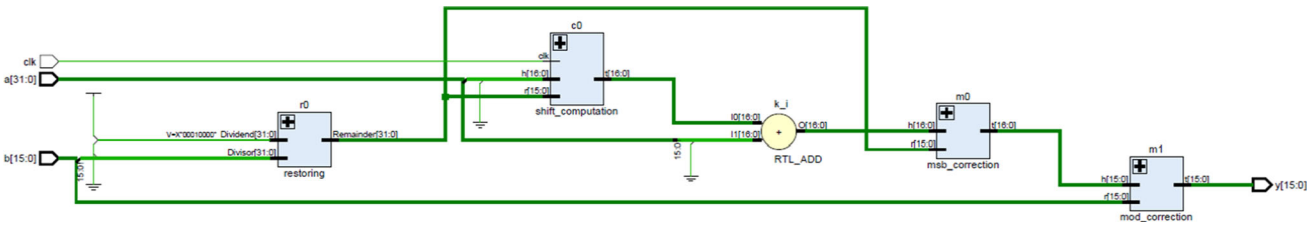

Fig. 19 RTL schematic modulo multiplicative inverse

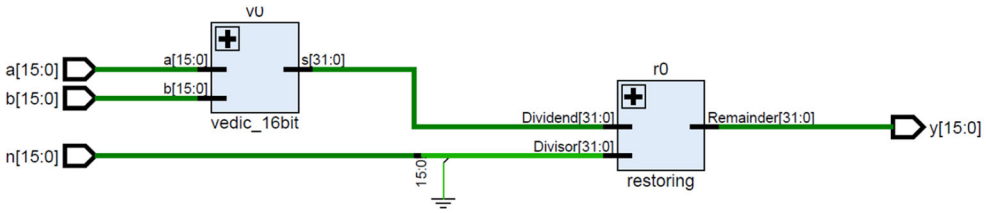

Fig. 20 RTL schematic classical modulo multiplication algorithm

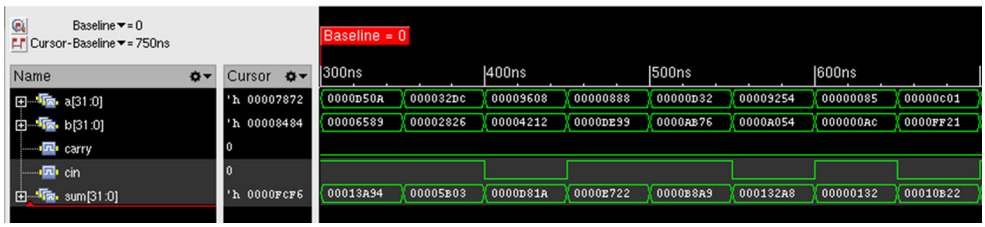

Fig. 21 Simulation result of 32-bit adder using basic gates

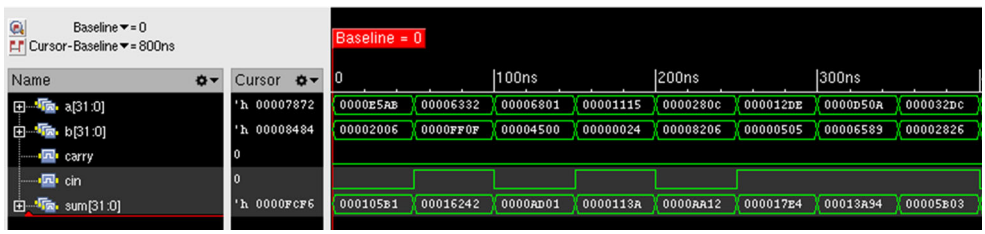

Fig. 22 Simulation result of 32-bit adder using reversible gates

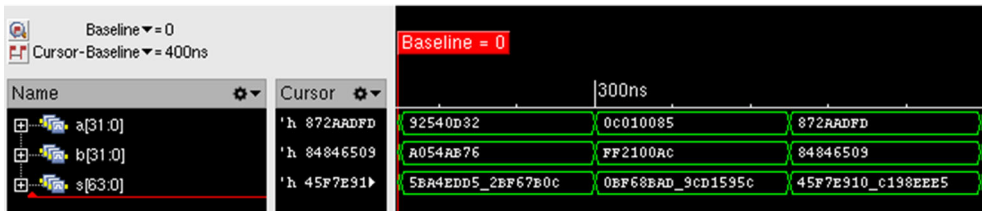

Fig. 23 Simulation result of Vedic multiplier using standard logic

Figures 14, 15 and 16 show implementation of dividers using crumbs encoding, standard logic and reversible gate logic.

Figures 17, 18, 19 and 20 show implementation of modular algorithms using modified Montgomery, Barrett reduction, modulo multiplicative inverse and classical modulo multiplication algorithm.

Figures 21 and 22 show the simulation results of the implemented 32-bit adders using standard logic and reversible logic.

Fig. 24 Simulation result of reversible multiplier

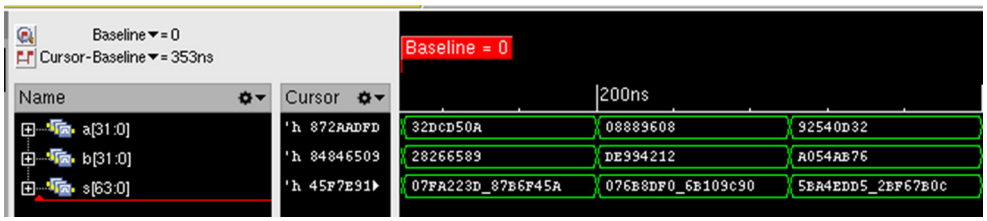

Fig. 25 Simulation result of Vedic multiplier using reversible gates

Fig. 26 Simulation result of Vedic divider using standard gates

Fig. 27 Simulation result of Vedic divider using reversible gates

Fig. 28 Simulation result of crumbs division

Figures 23, 24 and 25 show simulation results of Vedic multiplier using standard logic and reversible multiplier and Vedic multiplier using reversible logic.

Figures 26, 27 and 28 show simulation results of Vedic divider using standard logic and reversible logic and Vedic multiplier using reversible logic.

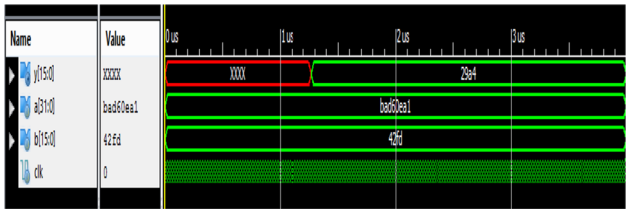

Fig. 29 Simulation result of classical modular

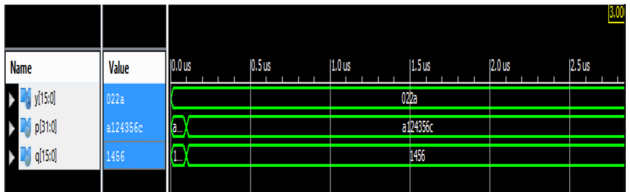

Fig. 30 Simulation result of Barrett reduction

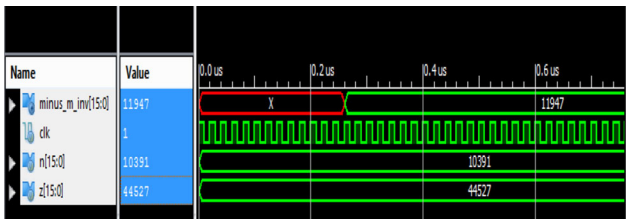

Fig. 31 Simulation result of modulo multiplicative inverse

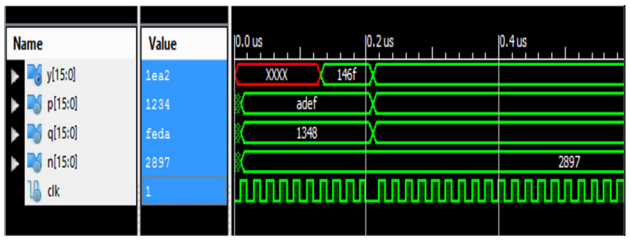

Fig. 32 Simulation result of montgomery modular algorithm

Figure 29, 30, 31 and 32 show simulation results of modular algorithms using classical modular, Barrett reduction modulo multiplicative inverse and modified Montgomery algorithm, respectively.

Implementation Results

The proposed blocks have been modeled using Verilog HDL, and the simulations and synthesis are performed using Xilinx Vivado verified on the Nexys 4 Artix-7 FPGA and Oasys RTL Tool (45-nm technology). The results design summary is obtained in Table 6 showing timing/critical path delay (logic delay + net delay) Slice LUTS, Registers and IOB's and total on chip power (Dynamic + Static) in terms of Watts as obtained using Xilinx Vivado Tool and area in terms of micrometer square and power (internal + Switching + leakage power) in terms of microwatts as obtained using Oasys RTL tool (45-nm technology). The proposed logic blocks have lesser power dissipation and lesser delay compared to the logic implemented in [7, 19, 20, 23].

Adder using reversible logic has lesser power reduction of 50% than Ref. [27] as seen in Fig. 9. Multipliers using reversible logic have less power reduction of 23% compared to Ref. [19] and about 10% reduction compared to Vedic multiplier using reversible logic, similarly area and timing reduction by about 10–25% in both adders and multipliers as seen in Fig. 33. Vedic divider using reversible logic has power and area reduction by 50% compared to Ref. [22] as shown in Fig. 34. Proposed modified Montgomery algorithm has area and timing reduction by about 25–30% compared to other algorithms and that implemented in Ref [23] as shown in Fig. 35.

Table 6 Implementation results (Artix 7 FPGA and Oasys RTL tool—45-nm technology)

Cell name	Nexys 4 Artix-7 FPGA					Oasys RTL tool—45-nm technology	
	Timing/path delay (ns)	Area			Power (W)	Area μm^2)	Power (μW)
		Slice registers (15,850)	Slice LUT's (63,400)	IOB's (210)			
32-bit adder							
32-bit adder using reversible logic	23.477	24	56	98	23.393	157	0.830,495
32-bit adder using standard logic	18.710	16	32	98	22.422	187	0.965919
Ref. [7]	7.88	–	–	–	–	–	11.39
32-bit multiplier							
Vedic multiplier using reversible gates	42.923	528	1940	128	123.154	6688	11377.84
Vedic multiplier using standard gates	43.59	592	2030	128	128.55	7099	11678.35
Reversible multiplier	26.527	112	378	64	38.629	1600	3168.57
Ref. [7]	21.44	–	–	–	–	–	299.87
Ref. [19]	4.932	–	–	–	–	1972	15,621.12
32-bit divider							
Vedic divider using standard gates	101.21	321	1093	97	56.342	2395	3437.402
Vedic divider using reversible gates	205.946	180	65	81	16.65	766	327.2207
Crumbs division	7.242	0	0	144	12.148	16,769	16,354.396
Ref. [22]	1380.68	–	–	–	–	–	1409.
Modular algorithm							
Modulo multiplicative inverse	161.548	106	1557	65	32.637	5988	5079.74
Modified montgomery modular	238.428	70	4462	65	138.351	1793	92.70
Barrett reduction	176.33	18	2465	64	108.29	9700	11,304.02
Classical modular	177.981	560	1962	64	72.91	7701	8253.96
Ref. [23]	303.3	–	–	–	–	2695	–

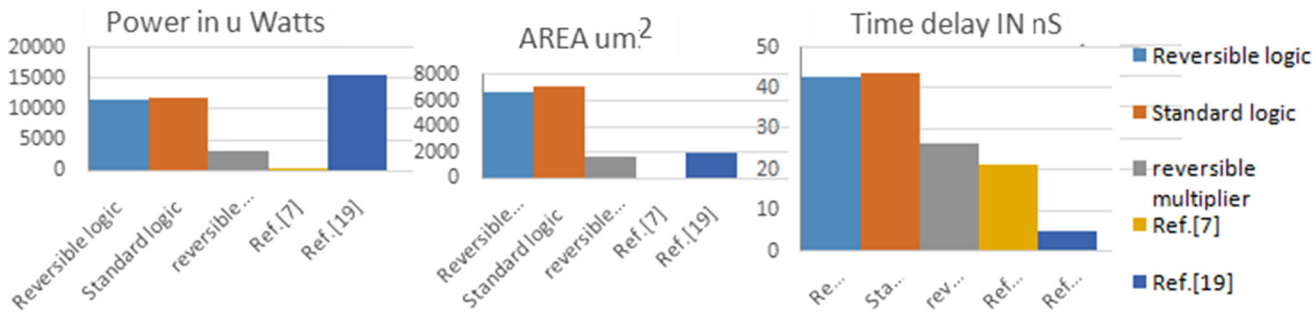

Fig. 33 Power, area and delay comparison of existing and proposed multipliers

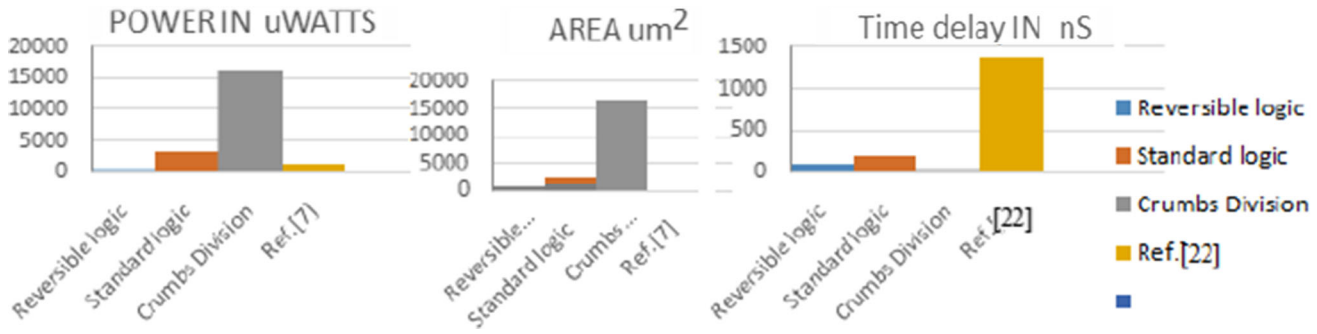

Fig. 34 Power, area and delay comparison of existing and proposed division algorithms

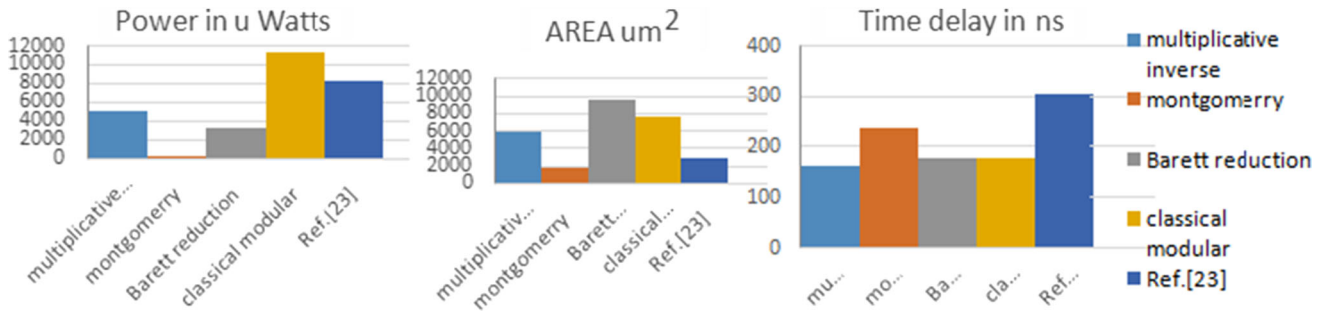

Fig. 35 Power, area and delay comparison of existing and proposed modulo multiplication algorithms

Conclusion

In this paper, an efficient implementation of cryptographic arithmetic units is proposed. The modules implemented combining reversible logic and Vedic mathematics has better reduction in power delay and area that has been compared with various implementations. Thus the algorithms implemented above can be applied to conventional algorithms so as to suit for resource-constrained devices. The future work is to implement the reversible logic and Vedic mathematics to the conventional cryptographic algorithms.

Acknowledgements The authors would like to thank the Department of Electronics and Communication and Engineering, Canara Engineering College, Mangalore, and Visvesvaraya Technological University, Belagavi, for the support for carrying out the research work.

References

1. Anish Bhimani, Securing the commercial internet. Commun. ACM **39**(6), 29–35 (1996)
2. K.A. McKay, L. Bassham, M.S. Turan, N. Mouha, Report on lightweight cryptography, draft nistir 8114. Technical report, National Institute of Standards and Technology (2016)
3. V.G. Kiran Kumar, C. Shantharama Rai, Implementation and analysis of cryptographic ciphers in FPGA, in *Emerging Technologies in Data Mining and Information Security. Advances in Intelligent Systems and Computing*, vol. 755, ed. by A. Abraham, P. Dutta, J. Mandal, A. Bhattacharya, S. Dutta (Springer, Singapore, 2019)
4. S. Vollala, V.V. Varadhan, K. Geetha, N. Ramasubramanian, Efficient modular multiplication algorithms for public key cryptography, in *2014 IEEE International Advance Computing Conference (IACC)*, pp. 74–78
5. W.D. Pan, M. Nalasani, Reversible logic. IEEE Potentials **24**(1), 38–41 (2005). https://doi.org/10.1109/MP.2005.1405801
6. H.M. Gaur, A.K. Singh, U. Ghanekar, In-depth comparative analysis of reversible gates for designing logic circuits. Procedia Comput. Sci. **125**, 810–817 (2018). https://doi.org/10.1016/j.procs.2017.12.103
7. A. Banerjee, D.K. Das, Arithmetic circuits using reversible logic: a survey report, in *Advanced Computing and Systems for Security. Advances in Intelligent Systems and Computing*, vol. 995, ed. by R. Chaki, A. Cortesi, K. Saeed, N. Chaki (Springer, Singapore, 2020)
8. S. Thakral, D. Bansal, S. Chakarvarti, Implementation and analysis of reversible logic based arithmetic logic unit. TELKOMNIKA Telecommun. Comput. Electron. Control **14**(4), 1292–1298 (2016)
9. D.M. Deeptha, M. Dhrithi, M. Pratiksha, B.S. Kariyappa, Design and optimization of 8 bit ALU using reversible logic, in *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, pp. 1632–1636 (2016)
10. T.M. Amirthalakshmi, S.S. Raja, Design and analysis of low power 8-bit ALU on reversible logic for nanoprocessors. J Ambient Intell Human Comput (2018). https://doi.org/10.1007/s12652-018-1074-y
11. S. Thakral, B. Dipali, Novel high functionality fault tolerant ALU. TELKOMNIKA Telecommun. Comput. Electron. Control **18**, 234 (2020). https://doi.org/10.12928/telkomnika.v18i1.12645

12. E. Prabhu, H. Mangalam, P.R. Gokul, A delay efficient vedic multiplier. Proc. Natl. Acad. Sci. India Sect. A Phys. Sci. **89**, 257–268 (2019). https://doi.org/10.1007/s40010-017-0464-4
13. A. Deepa, C.N. Marimuthu, Design of a high speed vedic multiplier and square architecture based on *Yavadunam Sutra*. Sādhanā **44**, 197 (2019). https://doi.org/10.1007/s12046-019-1180-3
14. R. Liu, S. Li, A design and implementation of montgomery modular multiplier, in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, Sapporo, Japan, pp. 1–4 (2019)
15. K. Morita, *Theory of reversible Computing. Monographs in theoretical computer science* (Springer, Berlin, 2017)
16. S.M.R. Taha, *Reversible Logic Synthesis Methodologies with Application to Quantum Computing, Studies in Systems, Decision and Control*, vol. 37 (Springer, Berlin, 2016)
17. J.W. Bruce, M.A. Thornton, L. Shivakumaraiah, P.S. Kokate, X. Li, Efficient adder circuits based on a conservative reversible logic gate, in *Proceedings IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI* (2002)
18. M. Haghparast, S.J. Jassbi, K. Navi, O. Hashemipour, Design of a novel reversible multiplier circuit using HNG gate in nanotechnology. World Appl. Sci. J. **3**(6), 974–978 (2008)
19. R. Anitha, N. Deshmukh, S.K. Sahoo, S. Prabhakar, A 32 BIT MAC unit design using vedic multiplier and reversible logic gate, in *International Conference on Circuit, Power and Computing Technologies [ICCPCT]* (2015)
20. R. Katreepalli, T. Haniotakis, Power-delay-area efficient design of vedic multiplier using adaptable manchester carry chain adder, in *2017 International Conference on Communication and Signal Processing, 6–8 April* (2017)
21. H.G. Rangaraju, A.B. Suresh, K.N. Muralidhara, Design of efficient reversible multiplier. Adv. Intell. Syst. Comput. (2013). https://doi.org/10.1007/978-3-642-31600-5_56
22. R. Vemula, K. Manjunatha Chari, Design and implementation of 64 bit divider using 45 nm CMOS technology. Int. J. Pure Appl. Math. **118**(5), 293–301 (2018)
23. X.D. Yan, S.G. Li, Modified modular inversion algorithm for vlsi implementation, in *Conference on ASIC—ASICON 2007*, pp. 90–93 (2007)
24. A. Bosselaers, R. Govaerts, J. Vandewalle, Comparison of three modular reduction functions, in *Advances in Cryptology, CRYPTO 1993, Lecture Notes in Computer Science*, vol. 773 (Springer, 1993), pp. 175–186
25. M.A. Kaihara, N. Takagi, A hardware algorithm for modular multiplication/division. IEEE Trans. Comput. **54**(1), 12–21 (2005)
26. G. Saldamli, Y. Baek, Uniform montgomery multiplier. J. Cryptogr. Eng. **9**, 333–339 (2019). https://doi.org/10.1007/s13389-019-00213-7
27. K. Arunachalam, M. Perumalsamy, C.K. Sundaram, J. Senthil Kumar, Design and implementation of a reversible logic based 8 bit arithmetic and logic unit. Int. J. Comput. Appl. **36**, 49–55 (2015)