

Design and Performance Measurement of Efficient IDEA (International Data Encryption Algorithm) Crypto-Hardware using Novel Modular Arithmetic Components

Rajashekhar Modugu¹, Yong-Bin Kim² and Minsu Choi¹

¹Dept of Electrical & Computer Engineering, Missouri University of Science and Technology
Rolla, MO 65401, USA, {rrmt4b, choim}@mst.edu

²Dept of Electrical & Computer Engineering, Northeastern University
Boston, MA 02115, USA, ybk@ece.neu.edu

Abstract—Cryptographic algorithms such as International Data Encryption Algorithm (IDEA) have found various applications in secure transmission of the data in networked instrumentation and distributed measurement systems. Modulo $2^n + 1$ multiplier and squarer play a pivotal role in the implementation of such crypto-algorithms. In this work, an efficient hardware design of the IDEA (International Data Encryption Algorithm) using novel modulo $2^n + 1$ multiplier and squarer as the basic modules is proposed for faster, smaller and low-power IDEA hardware circuits. Novel hardware implementation of the modulo $2^n + 1$ multiplier is shown by using the efficient compressors and sparse tree based inverted end around carry adders is given. The novel modules are applied on IDEA algorithm and the resulting implementation is compared both qualitatively and quantitatively with the IDEA implementation using the existing multiplier/squarer implementations. Experimental measurement results show that the proposed design is faster and smaller and also consume less power than similar hardware implementations making it a viable option for efficient hardware designs.

Index Terms—Modulo $2^n + 1$ multiplier; International Data Encryption Algorithm (IDEA); Sparse-tree adder; Power/area/speed measurement;

I. INTRODUCTION

The demand for high security in communications channels, networked instrumentation and distributed measurement systems is ever growing rapidly. The confidentiality and security requirements are becoming more and more important to protect the data transmitted and received. This leads to the need for efficient design of cryptographic algorithms which offer data integrity, authentication, non-repudiation and confidentiality of the encrypted data across the communication channels. Various cryptographic algorithms have been studied and implemented to ensure security of these systems. In this paper, modulo $2^n + 1$ multiplier has been much focus as it has found its important role in IDEA algorithm. For example, the three major operations that decide the over all performance and delay of the IDEA [1, 4, 15] are modulo 2^{16} addition, bitwise-XOR and modulo $2^{16} + 1$ multiplication and the $GF(2^n)$ Montgomery multiplication and modular exponentiation can be implemented using repeated multiplication and squaring of

the vectors. Among these operations, improving the delay and power efficiency of the modulo $2^n + 1$ multiplication operation leads to significant increase in the performance of the entire cryptographic cipher.

Numerous hardware implementations of the IDEA algorithm are proposed in the literature using different modulo $2^{16} + 1$ multiplier architectures. The IDEA algorithm has been implemented in software [3] on Intel Pentium II 445 MHz with encryption rate of 23.53 Mb/Sec. Later, IDEA was realized on hardware chip by Curiger et al. [1] with encryption rates upto 177 Mb/sec. By using a bit-serial implementation [4], which enables the IDEA to be fully pipelined the encryption rates reached 500 Mb/sec with 125 MHz clock rate. The efficiency of the IDEA cipher can still be improved if efficient basic modules such as modulo multipliers and adders are used. The efficient implementation of the modulo $2^n + 1$ multiplier based on novel compressors and sparse tree based inverted end around carry adders is presented in [7]. Even though the architecture of the modulo multiplier is very efficiently proposed in [6], the hardware implementation and optimization are considerably improved in [7]. This is resulted by replacing the full adder arrays with the novel compressors and the final stage adder with the sparse tree based inverted end around carry adder.

The paper is organized as follows; Section II-A introduces multiplexer-based compressors. In Section II-B, the hardware implementation of modulo $2^n + 1$ multiplier is given. Section III discusses the proposed implementation of the IDEA cipher which uses modulo $2^{16} + 1$ multiplier. A comparison of our implementation to a recently proposed implementation is made in Section IV. Our conclusions are drawn in section V.

II. PRELIMINARIES AND REVIEW

Novel multiplexor (MUX) based compressors and $2^n + 1$ multiplier design have been reported in [7] and are briefly reviewed in this section as follows:

A. Compressors

1) *MUX vs XOR*: Existing CMOS designs of 2-1 MUX and 2-input XOR are shown in Fig. 1. According to [8], the CMOS implementation of MUX performs better in terms of power and delay compared to XOR. Suppose, X and Y are inputs to the XOR gate, the output is $XY + \bar{X}\bar{Y}$. The same XOR can be implemented using MUX with inputs X, \bar{X} and select bit Y . The efficient implementation of compressors [9] is achieved by using both output and its complement of these gates. This also reduces the total number of garbage outputs.

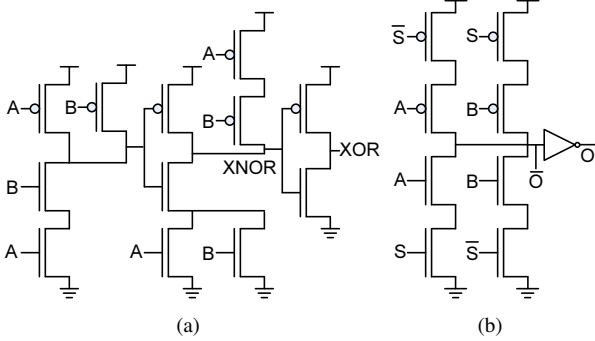


Fig. 1: CMOS implementation of 2-input (a) XOR (b) MUX

2) *Description of Compressors*: A (p,2) compressor with p inputs $X_1, X_2 \dots X_p$ and two output bits Sum and Carry along with carry input bits and carry output bits is governed by the equation:

$$\sum_{i=1}^p X_i + \sum_{i=1}^t (C_{in})_i = Sum + 2(Carry + \sum_{i=1}^p (C_{out})_i)$$

Block diagram of a 5:2 compressor is shown in Fig. 2. Efficient design of the existing XOR-based 5:2 compressor [10, 11], which takes 5 inputs and 2 carry inputs, is shown in Fig. 3(a). The critical path delay of this existing compressor is 4Δ -XOR (delay denoted by Δ).

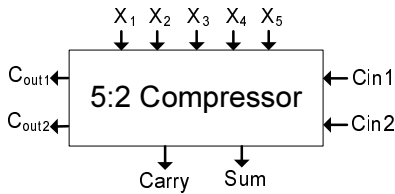


Fig. 2: Block Diagram of a 5:2 compressor

The newly designed compressors use multiplexers in place of XOR gates, resulting in high speed arithmetic. Also, as shown in Fig. 3(a) in all the existing CMOS implementations of the XOR and MUX gates both the output and its complement are available but the designs of compressors available in literature do not use these outputs efficiently. In the CMOS implementation of the MUX if both the select bit and its complement are generated in the previous stage then its output

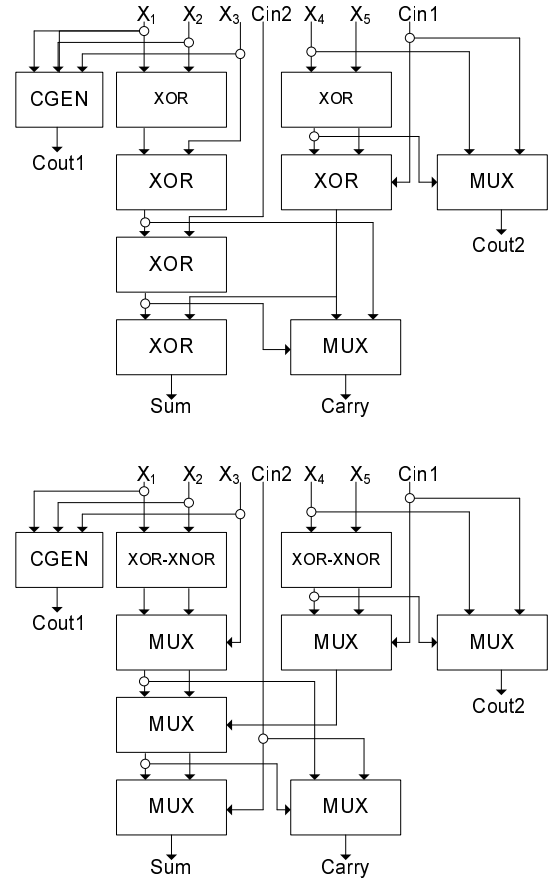


Fig. 3: 5:2 compressors: (a) existing XOR-based design; (b) new MUX-based design

is generated with much less delay because the switching of the transistor is already completed. And also if both the select bit and its complement are generated in the previous stage then the additional stage of the inverter is eliminated which reduces the overall delay in the critical path. The new MUX-based design of 5:2 compressor [9] is shown in Fig. 3(b), the delay of which is Δ -XOR+3 Δ -MUX. CGEN block used in Fig. 3(b) can be obtained from the equation $Cout1 = (x_1 + x_2) \cdot x_3 + x_1 \cdot x_2$ and the CMOS implementation is given in Fig. 4.

B. Hardware Implementation of the mod $2^n + 1$ Multiplier/Squarer

The hardware implementation of the modulo multiplier consists of three modules. First module is to generate partial products, second module is to reduce the partial products to two final operands and the last module is to add the Sum and Carry operands from partial products reduction to get the final result.

1) *Partial products generation*: The $n \times n$ partial products matrix is obtained from the $n + 1$ -bit input vectors. This partial product matrix is generated after repositioning the bits of the initial partial product matrix based on several observations presented in [6]. The final partial products matrix after applying all the observations is shown in Fig. 5. The

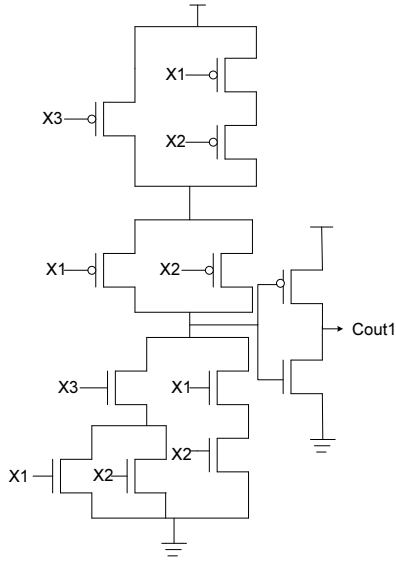


Fig. 4: CMOS implementation of carry generator block (CGEN) for the proposed design.

2^{n-1}	2^{n-2}	2^{n-3}
$PP_0 = p_{n-1,0} \vee q_{n-1}$	$p_{n-2,0}$	$p_{n-3,0}$
$PP_1 = p_{n-2,1}$	$p_{n-3,1}$	$p_{n-4,1}$
$PP_2 = p_{n-3,1}$	$p_{n-4,2}$	$p_{n-5,2}$
....
$PP_{n-2} = p_{1,n-2}$	$p_{0,n-2}$	$p_{n-1,n-2} \vee q_{n-3}$
$PP_{n-1} = p_{0,n-1}$	$p_{n-1,n-1} \vee q_{n-2}$	$p_{n-2,n-1}$

2^2	2^1	2^0
$p_{2,0}$	$p_{1,0}$	$p_{0,0} \vee q_{n-1} \vee p_{n,n}$
$p_{1,1}$	$p_{0,1}$	$p_{n-1,1} \vee q_0$
$p_{0,2}$	$p_{n-1,2} \vee q_1$	$p_{n-2,2}$
....
$p_{4,n-2}$	$p_{3,n-2}$	$p_{2,n-2}$
$p_{3,n-1}$	$p_{2,n-1}$	$p_{1,n-1}$

Fig. 5: Final $n \times n$ partial product matrix showing 2^{n-1} , 2^{n-2} , 2^{n-3} ... 2^2 , 2^1 and 2^0 columns.

partial product bits can be computed from AND, OR and NOT gates. The most complex function of partial product generation module is $p_{n-1,n-1} \vee q_{n-2}$, where $p_{i,j} = a_i b_j$ and $q_i = p_{n,i} \vee p_{i,n}$.

2) *Partial products reduction*: This is the most important module which largely determines the critical path delay and the overall performance of the multiplier. Hence this module needs to be designed so as to get minimum delay and consume less power.

The implementations from the literature [5, 6, 13] use full adders (FA) and half adders (HA) to construct this module. The series of full adders in any column can be replaced by the novel compressors that take the same number of inputs. In the proposed implementation use of suggested compressors is

done which not only reduces the delay and power consumption but also the area of the circuit. For a modulo $2^{16} + 1$ multiplier in IDEA cipher the Carry Save Adder (CSA) array implementation using Full Adders requires fifteen full adders in series in any column, these fifteen full adders can be replaced by two 7:2 compressors, one 5:2 compressor and two 3:2 compressors.

Correction factor computation is an important step while generating the partial products matrix. The full adder implementation [6] and the compressor based implementations [7] result in the same value. Because of the space constraints, computation of the correction factor COR for full adder implementation [6] is not given in this paper. COR computation for compressor implementation involves computing only COR2, because COR1 is obtained based on repositioning of the partial product term, which is same for both implementations. The correction factor COR2 computation for FA implementation which has $n-1$ stages of additions is shown in [6]. And the COR2 computation for the proposed multiplier implementation using the compressors also yields the same result. Since, any $(p, 2)$ compressor can be primarily designed using $p - 2$ FAs which give $p - 2$ carry outs with 2^n weight. Hence, the overall correction factor COR computation for CSA array FA implementation and compressor implementation yield the same result i.e., 3 as shown in [5].

3) *Final Stage addition*: The partial product reduction module gives one n -bit carry vector and one n -bit sum vector which need to be added in the final stage addition module. Very efficient parallel prefix adders are designed to do this operation [2].

Suppose S and C are sum and carry vectors produced after the partial product reduction section. As it is shown in the work of Zimmerman [2] that:

$$|S + C + 1|_{2^{n+1}} = |S + C + \overline{Cout}|_{2^n} \quad (1)$$

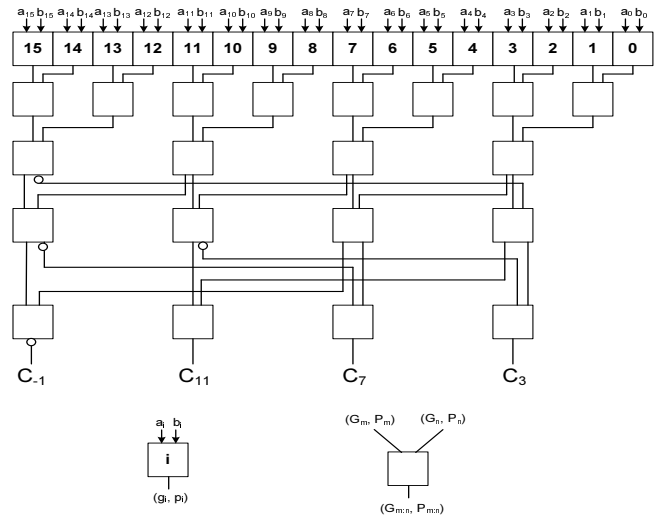


Fig. 6: Inverted EAC adder implemented using sparse tree structure

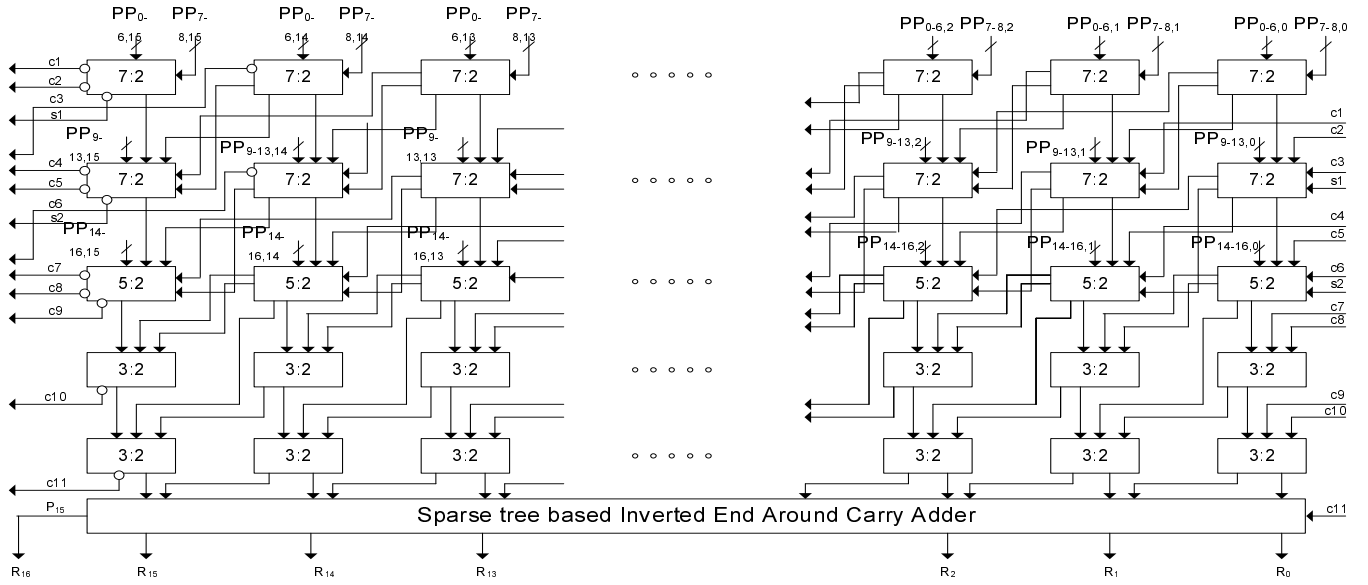


Fig. 7: Novel implementation of the modulo $2^{16}+1$ multiplier using efficient compressors

Equation (1) can be implemented using an inverted End-Around-Carry adder [2, 5, 6]. Even though the propagation delay of this adder is in the order of $\log_2 n$, it has a drawback of high interconnect complexity and high fan-out. This can be overcome by sparse tree adder [12, 16] based on the prefix network logic. The sparse tree adder generates the carry for every four bits instead of generating it at every stage and using a carry select block for selecting the final carry after the prefix network. This sparse tree adder was proven to be much more efficient in terms of both delay and power when compared with the existing prefix tree based adders [14]. Hence this sparse tree can be used to design Inverted-End-Around-Carry adder. The newly designed Inverted-End-Around-Carry adder using sparse tree adder structure is shown in Fig. 6. This Interted-EAC adder is used in the final stage addition of the modulo $2^n + 1$ multiplier. The proposed implementation of the modulo $2^{16} + 1$ multiplier for IDEA cipher is shown in Fig. 7 and $R_{16}R_{15} \dots R_2R_1R_0$ represents the final product of the modulo $2^{16} + 1$ multiplier.

III. NOVEL IMPLEMENTATION OF IDEA CIRCUIT USING THE PROPOSED MODULO $2^n + 1$ MULTIPLIER/SQUARER

The modulo $2^n + 1$ computation is an integral part of the International Data Encryption Algorithm (IDEA) where $n = 16$ [1, 4, 15]. Three major operations that decide the overall delay and performance of IDEA cipher are:

- 1) Modulo 2^{16} addition;
- 2) Bitwise-XOR;
- 3) Modulo $2^{16} + 1$ multiplication/squaring.

As the first two operations take less time and are easy to implement, the delay and power efficiency of the the entire IDEA cipher depends significantly on the modulo $2^{16} + 1$ multiplication/squaring operation. Hence, the IDEA cipher is implemented using the proposed modulo multiplier and compared with the existing implementations.

To encrypt a data block using IDEA cipher, the data should be processed through three modulo multiplication operations in a single round and the manipulated data again should pass through seven such rounds iteratively and a final output transformation to produce the final encrypted output. The IDEA cipher takes 64-bit input data and produces a 64-bit cipher text with a 128-bit key. The encryption and decryption algorithms in IDEA are almost identical except they utilize two different sets of subkey generated by the same key with different processes. The IDEA encryption and decryption processes consist of eight rounds of data manipulation using subkeys and a final output transformation stage. In this cipher, all the operations are carried out on 16-bit sub-blocks. In the encryption process, the input data block of 64-bits is divided into 4 sub blocks of 16-bits each (X_1, X_2, X_3, X_4). 52 subkeys for the encryption process are generated from the original 128-bit key by shifting a part of it. Out of the 52 subkeys, six different subkeys (i.e., $Z_1^{(r)}, Z_2^{(r)}, Z_3^{(r)}, Z_4^{(r)}, Z_5^{(r)}$ and $Z_6^{(r)}$, where r is the round number) are used for each round and the remaining 4 subkeys are used in the final output transformation stage. The 16-bit outputs at each round are represented as $Y_1^{(r)}, Y_2^{(r)}, Y_3^{(r)}, Y_4^{(r)}$ and W_1, W_2, W_3, W_4 are the outputs of the final output stage transformation. The 52 subkeys used for the decryption process are obtained using a different algorithm [17]. As shown in Fig. 8, the critical path consists of three modulo $2^{16} + 1$ multiplication operations, two modulo 2^{16} addition operations and two 16-bit XOR operations in each round. In the final output transformation stage, critical path consists of a single modulo $2^{16} + 1$ multiplication operation. The throughput of the IDEA cipher can be improved, if the delay of the modulo $2^{16} + 1$ multiplication operation is reduced in the pipelined implementation of the IDEA cipher. Fig. 8 shows the datapath of encryption process of the IDEA cipher and datapath of a single round with 4 pipeline stages with the

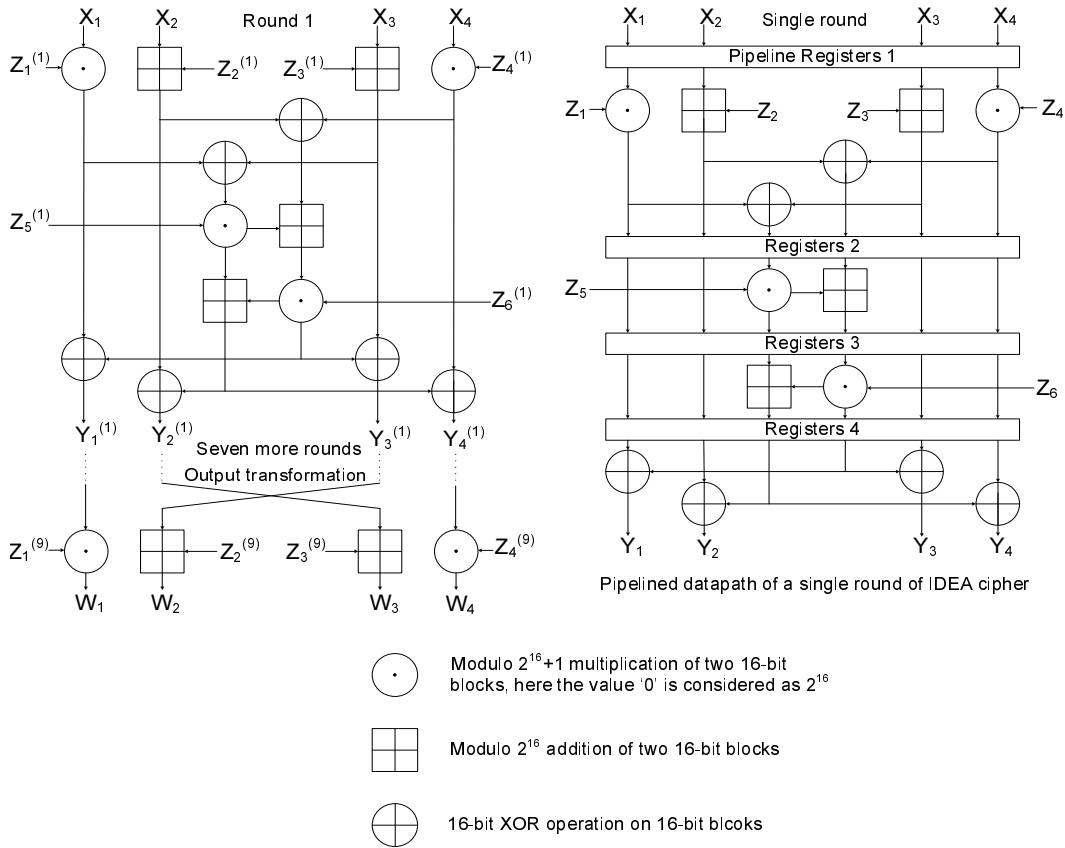


Fig. 8: Datapath of IDEA cipher with 4 pipeline stages

proposed modulo multiplier.

IV. EXPERIMENTAL SIMULATION AND RESULTS

The proposed design of the IDEA cipher with four pipeline stages using novel modulo $2^n + 1$ multipliers is used to analyze and compare with the well known IDEA cipher implementations. The use of the novel modulo multiplier improves the throughput and performance of the IDEA cipher significantly.

A. Simulation environment

All the simulations have been carried out using Mentor Graphics ASIC (Application-Specific IC) design suite. The proposed IDEA cipher design is specified using verilog HDL and the multiplier descriptions are mapped on a $0.18 \mu\text{m}$ CMOS standard cell library using Leonardo Spectrum synthesis tool from Mentor Graphics. The design is optimized for high speed performance. Netlists generated from synthesis tool are passed on to standard route and place tool, the layouts are iteratively generated to get the circuits with minimum area. The calculation of power and delay are carried out using the Eldo simulation tool. The proposed experimental simulation has been performed at 1.8V with all inputs fed at a frequency of 25 MHz.

B. Simulation results

The IDEA cipher is implemented using both the proposed multiplier and the multipliers presented in [6]. Various per-

formance measurements including encryption rate, delay and area for the IDEA cipher using both the proposed multiplier and the existing multiplier are parametrically obtained and listed in Table I. As expected, the proposed IDEA circuit implementation achieves significant improvements in terms of throughput (i.e., encryption rate), latency (i.e., critical path delay) and area (i.e., circuit area).

TABLE I: Comparison of the performance measurements for IDEA cipher

Performance Measurement	Using proposed multipliers	Using the multipliers in [6]	% Improvement
Encryption Rate (Mb/sec)	460.25	412.15	11.25
Critical path delay (nS)	4.372	5.168	15.4
Area of the cipher (mm^2)	3.68	4.22	12.79

V. CONCLUSION

An hardware implementation of the IDEA cipher using novel modulo $2^n + 1$ multipliers is presented in this paper. It is shown that the proposed modulo $2^n + 1$ multiplier improves the performance of the various cryptographic algorithms used in secure communication systems of networked instrumentation and distributed measurement systems. Efficient compressors

and sparse tree based inverted end around carry adders are used to reduce the delay and complexity of the multiplier. Simulations are performed on the known implementation and the proposed implementation. The presented implementation is proven to perform better than the existing one in various aspects, (i.e., throughput and critical path delay).

REFERENCES

- [1] R. Zimmermann, A. Curiger, H. Bonnenberg, H. Kaeslin, N. Felber and W. Fichtner, A 177 Mb/s VLSI implementation of the international data encryption algorithm, *IEEE J. Solid-State Circuits*, 1994, 29, (3), pp. 303-307
- [2] R. Zimmerman, Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication, *IEEE trans. Comput.*, 2002, 51, pp. 1389-1399.
- [3] O. Cheung, K. Tsoi, P. Leong and M. Leong, Tradeoffs in parallel and serial implementations of the international data encryption algorithm IDEA, *Lecture Notes in Computer Science*, vol. 2162, pp. 333-340, 2001.
- [4] M. Leong, O. Cheung, K. Tsoi, and P. Leong, A bit-serial implementation of the international data encryption algorithm IDEA, 2000 IEEE Symposium on Field-Programmable Custom Computing Machines, pp. 122-131, 2000.
- [5] C. Efstathiou, H. Vergos, S. Dimitrakopoulos and D. Nikolos, Efficient diminished-1 modulo $2^n + 1$ multipliers, *IEEE Trans. Comput.*, 2005, 54, pp. 491-496.
- [6] H. Vergos and C. Efstathiou, Design of efficient modulo $2^n + 1$ multipliers, *IET Comput. Digit. Tech.*, 2007, 1, (1), pp. 49-57.
- [7] R. Modugu, N. Park and M. Choi, A Fast Low-Power Modulo $2^n + 1$ Multiplier Design, 2009 IEEE International Instrumentation and Measurement Technology Conference, pp.951-956, May 2009.
- [8] R. Zimmermann and W. Fichtner., Low-power logic styles: CMOS versus pass-transistor logic, *IEEE J. Solid- State Circuits*, vol. 32, pp. 1079-1090, July 1997.
- [9] S. Veeramachaneni, L. Avinash, M. Rajashekhar and M. Srinivas, Efficient Modulo $(2^k \pm 1)$ Binary to Residue Converters System-on-Chip for Real-Time Applications, The 6th International Workshop on Dec. 2006 pp.195 - 200.
- [10] C. Chang, J. Gu and M. Zhang, Ultra low-voltage lowpower CMOS 4-2 and 5-2 compressors for fast arithmetic circuits, *IEEE J. Circuits and Systems I*, Vol. 51, No. 10, pp. 1985- 1997, 2004
- [11] M. Rouholamini, O. Kavehie, A. Mirbaha, S. Jasbi and K. Navi, A New Design for 7:2 Compressors, *Computer Systems and Applications*, 2007. AICCSA '07. IEEE/ACS International Conference on 13-16 May 2007 Page(s):474 - 478.
- [12] S. Mathew, M. Anders, R. Krishnamurthy and S. Borkar, A 4-GHz 130-nm address generation unit with 32-bit sparse-tree adder core, *IEEE Journal of Solid-State Circuits*, Volume 38, Issue 5, May 2003 Page(s):689 - 695.
- [13] Zhongde Wang, Graham A. Jullien, William C. Miller., An efficient tree architecture for modulo $2^n + 1$ multiplication, *VLSI Signal Processing* 14(3): 241-248 (1996).
- [14] P. Kogge and H. S. Stone., A parallel algorithm for the efficient solution of a general class of recurrence equations, *IEEE Trans. Comput.*, vol. C-22, pp. 786-793, Aug 1973.
- [15] A. Curiger et. al., VINCI: VLSI Implementation of the New Secret-keyBlock Cipher IDEA, *Proc. of the Custom Integrated Circuits Conference*, San Diego, USA, May 1993
- [16] Yan Sun, Dongyu Zheng, Minxuan Zhang, and Shaoqing Li High Performance Low-Power Sparse-Tree Binary Adders, *8th International Conference on Solid state and Integrated Circuit Technology*, ICSICT 2006.
- [17] Yi-Jung Chen, Dyi-Rong Duh, Yunghsian Sam Han: Improved Modulo $(2n+1)$ Multiplier for IDEA, *J. Inf. Sci. Eng.* 23(3): 911-923 (2007)