A Survey on Hardware Implementation of IDEA Cryptosystem

Sourav Mukherjee
Dept. of Computer Science and Engg.
National Institute of Technology Rourkela
Rourkela India-769008

Bibhudatta Sahoo
Dept. of Computer Science and Engg.
National Institute of Technology Rourkela
Rourkela India-769008

***Abstract*—The main goal of hardware implementation of a cryptosystem is to make it compatible for high speed networks. The cryptographic algorithms are very much computationally intensive and to achieve a high speed execution, hardware is necessary. In this paper, various hardware implementations of the IDEA cipher is discussed. The hardware implementation involves both ASIC and FPGA implementations and IDEA has been implemented quite a several times in hardware. But a complete survey of all the previous implementations has not been presented before. In each of these implementations, the focus has been made on the data throughput, area requirements and the architecture of the algorithm proposed.**

***Index Terms*—FPGA,ASIC,IDEA, Block Cipher, Throughput ,Multiplication modulo $(2^n + 1)$.**

I. Introduction

Cryptography is the art of keeping data secure from unauthorized access so as to guarantee that only the intended users can access it. As computer technologies are getting advanced, more and more cryptographic applications are used. They are mainly used to support other applications which are very much sensitive to data security such as smart cards and commercial data exchange over a network. Not only for personal use but cryptographic algorithms are also very important in every aspect of professional activities. A cryptographic algorithm generally consists of some specialized arithmetic computations which are complicated in terms of time complexity. It is because of the fact that these algorithms work with large amount of data either in blocks or simply in streams. Although a single traditional CPU is enough for performing these computations, but for a machine which works as a server in a huge network gets millions of client requests for performing cryptographic operations for them individually. This makes the workload huge. The computational resources may also be limited for example in smart-cards, mobile phones, hand-held computers, etc. Moreover if the associated network is of high speed, the speed of the necessary cryptographic computations also needs to be taken into account. For example in transmitting audio and video data for cable TV, pay TV, video conferences and sensitive financial and commercial data, the speed of the cryptographic module to be embedded ,needs to be very high. Moreover for security related issues in wireless and sensor networks, there is a need for separate hardware device with very high processing rate because of limited battery of the nodes and for optimizing the bandwidth efficiency. So from the viewpoint of high speed and throughput, traditional software implementations of these complicated cryptographic algorithms are not efficient in real time applications like ATM, VPN, etc. This forces the system designers to go for hardware implementation of the cryptosystems. Traditionally hardware implementations are based on ASIC technology, but they are not quite affordable every time especially in monetary terms. Moreover these ASICs are not adaptable to new changes once the hardware is built. The more efficient and convenient method is to use FPGA platforms which provides sufficient logics and storage elements on which any complex algorithm can be implemented. They are adaptable to new changes and their granularity matches quite well with the cryptographic algorithms.

Implementing a design in hardware is a type of resource allocation problem where the goal is to optimize the balance between silicon area requirements in a FPGA,operation throughput and the power consumption by the design. To provide this optimal balance, the regularity of the design must be high. But in cryptographic applications,high irregularity guarantees more security. So for the sake of complete functionality of the design and for high throughput, a trade off between area and time delay is often made.

Rest of the paper is organized as follows. Section 1 gives the basic idea of the IDEA cipher in details. Section 2 discusses the various implementations of the IDEA cryptosystem. Section 3 discusses the detailed approach of all the existing designs for IDEA implementations in hardware along with the brief description of the architectures and performances. The analysis and comparison of these designs along with the limitations are covered in Section 4. Finally the conclusion is drawn in Section 5.

II. International Data Encryption Algorithm

The proposed Encryption Standard (PES) is a block cipher introduced by Lai and Massey [1], [2]. It was then improved by the Lai, Massey and Murphy in 1991. This version, with stronger security against differential analysis and truncated differentials, was called the Improved PES (IPES). IPES was renamed to be the International Data Encryption Algorithm (IDEA) in 1992. Claims have been made that the algorithm is the most secure block encryption algorithm in the public domain.

A. Basic Structure

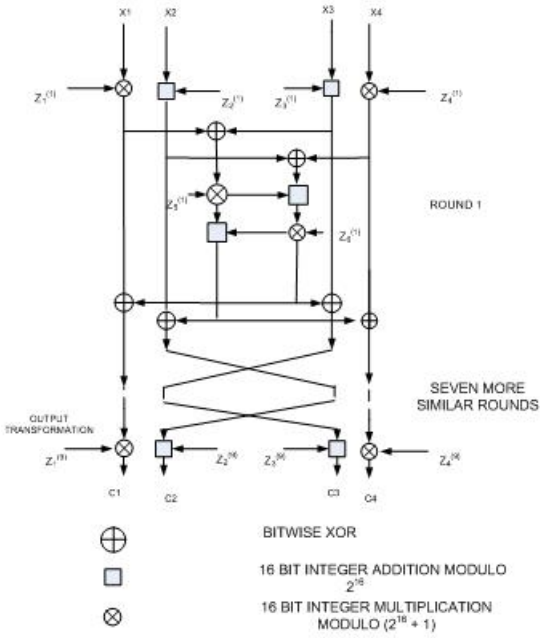

Fig. 1. Data flow of IDEA Cipher

IDEA is a symmetric, secret-key block cipher. The keys for both encryption and decryption must be kept secret from unauthorized persons. Since the two keys are symmetric, one can divide the decryption key from the encryption one or vice versa. The size of the key is fixed to be 128 bits and the size of the data block which can be handled in one encryption/decryption process is fixed to 64 bits. All data operations in the IDEA cipher are in 16-bit unsigned integers. When processing data which is not an integer multiple of 64-bit block, padding is required. The security of IDEA algorithm is based on the mixing of three different kinds of algebraic operations: XOR, addition and modular multiplication. IDEA is based upon a basic function, which is iterated eight times. The first iteration operates on the input 64-bit plain text block and the successive iterations operate on the 64-bit block from the previous iteration. After the last iteration, a final transform step produces the 64-bit cipher block.The data flow graph is shown in Figure 1. The algorithm structure has been chosen such that, with the exception that different key sub-blocks are used, the encryption process is identical to the decryption process. IDEA uses both confusion and diffusion to encrypt the data. Three algebraic groups, XOR, addition modulo 2^{16}, and multiplication modulo $(2^{16}+1)$, are mixed, and they are all easily implemented in both hardware and software. All these operations operate on 16-bit sub-blocks.

B. Key Generation

The key generation phase of IDEA generates 52 sub-keys from the 128 bit input key. The block diagram for key generation is shown in Figure 2. The basic steps of generating the encryption keys are:

- All the sub-keys are named as $Z_1^{(1)}$,...,$Z_6^{(1)}$,$Z_1^{(2)}$,...,$Z_6^{(2)}$,....,$Z_1^{(8)}$,...,$Z_6^{(8)}$, $Z_1^{(9)}$,...,$Z_4^{(9)}$.
- From the input 128 bit key, eight sub-blocks of 16 bits are partitioned and are assigned to $Z_1^{(1)}$,...,$Z_2^{(2)}$ directly.
- Now the original 128 bit key block is rotated by 25 bits and a new 128 bit block is formed. Now another eight sub-blocks are generated from this new block.
- The rotation procedure is repeated until and unless sub-blocks used in previous rounds are found.

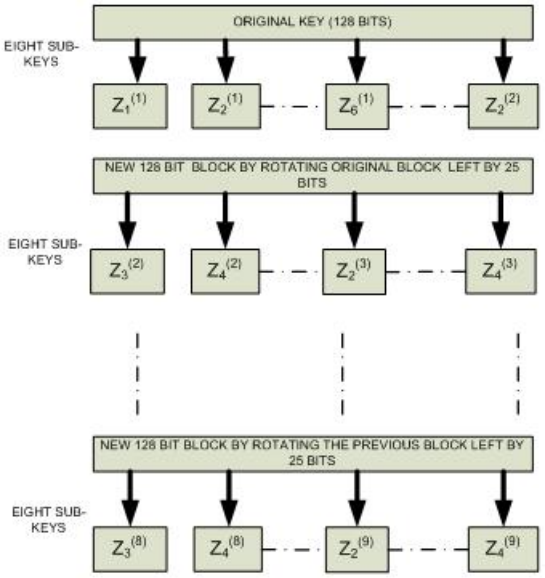

Fig. 2. IDEA Encryption key Generation

Once the encryption keys are generated, the decryption keys can be generated directly by taking their additive inverse modulo 2^{16} and multiplicative inverse modulo $(2^{16}+1)$ as required.

III. Various Implementations of IDEA

In high speed applications, where there is a need of protection of data, cryptographic algorithms are necessary. Data rates in such applications are very high and such computation cryptographic algorithms need to be run on real time so as to provide the quality of service. In this scenario, a software implementations of such algorithm using general purpose processors due to delay in instruction processing. But such speed can be easily achieved when implemented in hardware.Although the software implementation is less costly than hardware implementation, the speed up in hardware is very high. So for flexibility,availability and high functionality, there is a need of incorporating a separate cryptographic module in such applications.

Although IDEA involves only simple 16-bit operations, software implementations of this algorithm still cannot offer the encryption rate required for on-line encryption in high-speed networks.IDEA has been previously implemented in hardware using various FPGA devices and even ASIC. Like other renowned symmetric key block ciphers, IDEA contains

no S-Boxes or P- Boxes. So there is a less memory overhead.Instead it has some basic building blocks like XOR, addition modulo 2^n and multiplication modulo 2^n+1. Among these basic operations, the XOR and the addition modulo 2^n implementations are very straightforward.The multiplication module is the most computational intensive module and it needs a lot of effort to design it efficiently. In each round of IDEA, four such modulo multipliers are needed.So the performance of IDEA in hardware i.e. the throughput rate and the area and cost efficiency depends a lot on efficient design of the multiplier.

IDEA was first implemented and verified in VLSI by Bonnenberg [3] where the data encryption and decryption was performed on a single hardware unit which was a 1.5 μm double metal n-well CMOS with a maximum clock frequency of 33 MHz and data throughput rate of 44 Mb/s. In this implementation, the key management module and the inversion module was not performed on chip. The main goal was to achieve the highest possible throughput along with a hardware support to verify whether the design was cryptographically correct in terms of functionality and availability. So the task was to go for efficient design of the cipher data path along with managing off chip data traffic which was done by incorporating pipelining. By that time , some effective architectures for modulo $(2^n + 1)$ multipliers were proposed by Bonnenberg and Curiger [4]. Among those architectures, Bonnenberg's scheme[3] used the $(n + 1) \times (n + 1)$ multiplication scheme with a pipeline of two stage. With a computation speed of 60 ns per multiplication, a two multiplier round architecture using pipelining for IDEA was used. The speciality of this approach is that, the architecture is made of one encryption/decryption unit and an input/output interface unit with each unit containing a RAM. Proper clock was used to match their speeds. The drawback of this design is that extra overhead is associated due to huge data transfer from on chip RAMs as well as regulating off chip traffic. Moreover, the design was not supportive for all standardized modes of the cipher. Although it is the first VLSI implementation of IDEA, the data throughput rate was found to be twice than that of a DES chip at that time.

Bonnenberg's design [3] was found to be a prototype for a VLSI circuit , which was made essentially to speed up the cryptographical tests. But there was still a demand for a real time application hardware that can handle data traffic in high speed networks. The goal was to design an efficient basic building block with a high throughput data path architecture with an efficient interface that can handle off-chip data traffic.

Curiger's implementation [5][6] of IDEA was done on double-metal CMOS 1.2 μm which was suitable for all standardized modes. One of the speciality of this implementation is that, the data encryption and decryption was implemented on a single hardware unit.With a system clock frequency of 25 MHz the data throughput rate was found to be 177 Mb/s. This was the first silicon block which was found compatible for online encryption in high speed networks. The design was made using eight pipelining stages, containing a single round to achieve temporal parallelism.

As usual, the design of modulo $(2^n + 1)$ multiplier was crucial for the performance of the cipher. Various multiplication schemes were defined in [4]. Curiger's design used the multiplication scheme with modulo $(2^n + 1)$ adders in which one of the operands (say X) was in diminished-1 representation proposed in [7] and another operand was in normal weighted form which can be given as:

$$Z = \chi Y \ mod(2^n + 1) = (\sum_{i=0}^{n-1} 2^i \chi_i \ . \ \sum_{i=0}^{n} 2^i y_i) \ mod(2^n + 1) \tag{1}$$

$$= (\sum_{i=0}^{n} y_i (2^i \chi \ mod \ 2^n + 2^i \overline{\chi} \ div \ 2^n + 1) \ mod(2^n+1)) \ mod(2^n+1)$$

where

$$\overline{\chi} = \sum_{i=0}^{n-1} 2^i \overline{\chi_i}$$

and χ is a diminished-1 representation of $X = \sum_{i=0}^{n} 2^i x_i$, i.e. $\chi = X - 1$.

Later, in [6] a new approach was taken to avoid high computation time and area. A modified Booth recoding multiplication and a fast carry select additions for the final modulo correction were used as two stages of the multiplier in a pipeline structure. Four such modulo multipliers were used in each round for optimizing the performance of computational units. Each of the multiplication units used two stages of the eight stage pipeline. The design was made on a single hardware chip where the sub-keys were generated internally along with necessary computation of additive and multiplicative inverses. The multiplicative inverses were calculated using square and multiply method. Only the master key was loaded onto the chip at the beginning. So the speciality of this design was that, no off chip data traffic was needed to manage through buffers. The overall architecture of Curiger's design [6] contains two on chip buffer memory for implementing the different modes of operation of the cipher. In each buffer, a 8×64 bit shift register is used for implementing eight stage pipeline.

In VLSI circuits, arrival of temporary or permanent faults are very common which creates error in encryption. To get rid of these faults, necessary fault detection tests are required. These tests can be off-line or online. But if these tests are periodic, it consumes unnecessary clock cycles and degrades the speed. For testing the overall functionality during encryption,an online built in self test scheme was added which was done by incorporating a fifth multiplier in the pipeline circuitry. The drawback in this approach was that, this hardware redundancy resulted in a large extra hardware. Moreover if the time between two tests was long, there was a probability of some short-lived error to creep in. Although, the proposed design was not the fastest single chip implementation but it was the first design which was found compatible for use in high speed networks.

Although the design proposed in [6] was compatible for

real time encryption in high speed networks, there was still a demand for hardware with faster encryption ability. Moreover Curiger's design [6] was not capable of detecting all possible errors during its normal operation. Wolter's design [8] of a new hardware for IDEA was motivated by the requirement of higher data rate and online testing of circuit. The design was done by implementing one round of IDEA in a 0.8 μm CMOS and a data throughput of 355 Mb/s was obtained. The characteristics of the architecture was that, all the standardized modes of the cipher were capable to processing data with equal speed. The design of modulo $(2^{16}+1)$ multiplier was based on low high theorem of Lai and Massey [1] where modified Booth encoded multiplication algorithm and wallace tree were used. Here a 10 stage pipelining was used where two stages were reserved for performing online test.

For detecting faults, both off-line and online built in self test schemes were used in this scheme.The off-line test was performed using pseudo-random data encryption. Two online tests were performed, one based on information redundancy i.e mod 3 residue code and the second test was based on redundant test words. Although this scheme has a data throughput rate of 355 Mb/s by implementing a single round, due to overhead of online tests on chip, some additional area requirements was required in this design.

The next implementation of IDEA was made by Salomao [9] on an Application Specific IC named HiPCrypto using a 0.7 μm two metal, which was oriented towards computer network applications (like VPN) demanding high throughput. A single chip was used and the operation frequency was 53 MHz clock. This design was made to meet the requirement of applications in current and future high speed data networks. For this, temporal and spatial parallelism was exploited on the main design. No built in self test schemes were incorporated in this design for detecting faults. The modulo (2^n+1) multiplication was designed using a two stage pipelined multiply unit. Four small RAMs were used for storing the sub-keys. By using a single HiPCrypto device, a data throughput rate of 424 MB/s was achieved by the design. The disadvantage of this scheme was that, the HiPCrypto chip was not able to handle sub-keys derived from multiple keys.

A paper design of IDEA processor using four xilinx XC4020XL devices was proposed by Mencer [10] and that proposed design achieved a data throughput rate of 528 Mb/s. The design was done for comparing the parameters like performance, programmability and power for ASICs, FPGAs and normal processors. During the FPGA implementation, 56 stage pipelining was exploited for performance improvement and a custom designed Konstant coefficient multiplier was used which was based on look-up tables. The limitation of this design was the prior loading of keys before encryption.

Leong [11] implemented the IDEA cipher using a bit serial architecture[12]. Due to the bit serial architecture, the algorithm of the cipher was deeply pipelined. The operation frequency of this design was 125 MHz and a Xilinx Virtex XCV300-6 device was used. The data throughput rate was found to be 500 Mb/s which was as usual compatible for online encryption for high speed networks. The advantages of this implementation were :

- High degree of fine grain parallelism.
- Scalable and thus the trade-offs between data conversion rate and the area can be addressed.
- High clock rate.
- Compact implementation.

The design for the modulo $(2^{16}+1)$ was done using the approach proposed by Meier and Zimmerman and described by Curiger[5] in which, modulo 2^n adders were used along with bit-pair recoding algorithm. To increase the throughput, a 16 stage pipelined version of Lyon's serial parallel multiplier was used. The overall design of the cipher was done using four parallel to serial converters and four four serial ro parallel converters. The key storage and subkey generations in each round was done using shift registers. The proposed design was found to be scalable using more resources.

For incorporating efficiency in reconfigurable computing, Goldstein[13] implemented the IDEA cipher on Piperench architecture and achieved a data throughput of 1013 Mb/s. Although, the design was more suitable for stream based applications,the speciality of the Piperench architecture was the improvement of compilation and reconfiguration time from normal FPGAs by means of an advanced computing technique called pipeline reconfiguration. This feature is one type of a hardware virtualization in which, the compiler is free from hardware constraints. The simulation of [13] was done by dataflow intermediate language.

Ascom, the patent holder of IDEA, implemented a commercial design of IDEA cipher called IDEACrypt kernel on 0.25 μm CMOS technology and achieved a throughput rate of 720 Mb/s.

Mosanya[14] implemented IDEACore, an encryption core for International Data Encryption Algorithm as a modular and reconfigurable cryptographic coprocessor. The goal of that design was to accelerate cryptographic operations on a host system.The system was implemented using VHDL and it exploited the property of partial reconfiguration for a normal FPGA. In the multiplication module, bit parallel multiplier was used and for modulo $(2^{16}+1)$ correction, low-high algorithm [1] was used. For the overall design, a scalable pipeline was used where the number of pipeline stages were decided during compilation time. The design achieved a throughput rate of 1500 Mb/s. The drawback of the scheme is that, the area requirements is not fixed every time due to variation of pipelined stages. Moreover, due to session initialization and key calculation, the overall performance is slightly low in this scheme.

Cheung and Leong [15] further implemented IDEA on a bit parallel architecture[12] on Xilinx Virtex XCV300-6 FPGA and achieved a data throughput of 1166 Mb/s at a system clock rate of 82 MHz.The implementation was runtime reconfigurable and by direct modification of bitstream downloaded to the FPGA, the key scheduling was done. Moreover, the implementation was scalable with increased resource require-

TABLE I
COMPARISON OF THE EXISTING DESIGNS IN TERMS OF OPERATING FREQUENCY AND THROUGHPUT.

Design	Device	Multiplier Used	Operating Frequency	Throughput
Bonnenberg[3]	1.5 μm double metal CMOS	using $(n+1) \times (n+1)$ bit multiplier	33 MHz	44 Mb/s.
Curiger[5], [6]	1.2 μm double metal CMOS	using $(2^n + 1)$ adders	25 MHz	177 Mb/s.
Zimmerman [6]	1.2 μm double metal CMOS	using modified Booth's recoding	25 MHz	177 Mb/s.
Wolter[8]	0.8 μm CMOS	using Low-High Theorem and wallace tree	21 MHz	355 Mb/s.
Salomao[9]	0.7 μm two metal ASIC (HiPCrypto)	2 stage pipelined multiplier	53 MHz	424 Mb/s.
Mencer[10]	Xilinx XC4020XL FPGA	Konstant Coefficient Multiplier (KCM)	132 MHz	528 Mb/s.
Leong[11]	Xilinx Virtex XCV300-6 FPGA	modulo 2^n adders with bit pair recoding	125 MHz	500 Mb/s.
Cheung and Leong[15]	Xilinx Virtex XCV300-6 FPGA	Bit parallel multiplier	82 MHz	1166 Mb/s.
Thaduri[16]	EPF10K70RC240 ASIC	modulo 2^n adders with bit pair recoding	10 MHz	700 Mb/s per chip.
Hamalainen[17]	Xilinx XCV1000E-6BG560 FPGA	Diminished one multiplier using CSA	105.9 MHz	6.78 Gb/s.
Granado[18]	Xilinx Virtex 2 XC2V6000-6 FPGA	KCM multipliers using pipeline	164.3 MHz	14.7 Gb/s.

ments. With a full hardware support, a throughput of 5.25 Gb/s was estimated using this design. With a fully pipelined approach, IDEA was implemented by Hamalainen[17] using Xilinx XCV1000E-6BG560 FPGA and the throughput of 6.78 Gb/s throughput was achieved by the design. The modulo multiplier used diminished-1 number representation[7] and the multiplication schemes used in [19] [20], [4], [21] were implemented and compared. Finally, the multiplication scheme of [20] was chosen. For cyclic left shifts, extra combinational logic was used and Carry save adders were used for multi-operand addition. The entire design was made using loop unrolling architecture but it was slightly less efficient in terms of area requirements.

Gonzalez[22] achieved a throughput of 8.3 Gb/s while implementing the IDEA cipher using Xilinx Virtex XCV600-6 device. The speciality of this design was that, all the operational units were replaced by constants and a partial reconfiguration was used along with super-pipelining. The only drawback for this scheme is that, not many devices support partial reconfiguration.

Using embedded multipliers, IDEA was implemented by Pan and a throughput of 6 Gb/s was achieved but the design was costly in terms of area efficiency. An efficient VLSI implementation of IDEA was done by Thaduri[16] using Altera FPGA, where the modulo multiplier was optimized by using Wallace trees and carry look ahead adders and a deep temporal parallelism was exploited. The speciality of the design is that, the sub-keys are generated internally once the original key is fetched. Moreover, the design did not use any additional RAM for storing the subkeys.Using a clock frequency of 10 MHz, a throughput greater than 700 Mb/s was achieved by the design. In terms of scalability, a throughput of 7.8 Gb/s was achieved using scaling.

Finally Granado's design in [18] was based on dynamic and partial reconfiguration which achieved a very high encryption speed of 14.7 Gb/s. Till now this is the highest throughput achieved so far for IDEA implementation in hardware. The design is made partially reconfigurable by using Constant Coefficient Multipliers (KCM) and Constant Coefficient Adders (KCA). The multipliers were made up of small look up tables and the modulo correction is done by using Low High lemma. Moreover a dual port RAM memory and a series of pipelined registers were used for reducing the storage complexity and increasing the throughput. A 3 stage pipeline was used for each multiplier inside a single round. To decrease the development time, Handel C was used along with VHDL. The target device was chosen as Xilinx Virtex 2 XC2V6000-6 for this design.

IV. COMPARISON AND ANALYSIS

In the previous section, we have discussed all the existing implementations of the IDEA cipher. In each of these implementations, the main goal was to reduce the round complexity by reducing the sub modules of the overall design so as to optimize the performance parameters. The parameters which were considered mainly in these existing designs are:

- **Maximum Frequency:** The maximum frequency achieved by the design.
- **Throughput:** The amount of data (in bits/ bytes) processed per unit interval of time by the design.
- **Area Consumption:** The area consumed by the design (slices or CLBs).

In this section, we have made a detailed comparison of all the existing designs of IDEA in terms of operating frequency and throughput in Table I. As the efficient design of the modulo multiplier is an important issue for the IDEA cipher, the

TABLE II
EXISTING IMPLEMENTATION ISSUES AND LIMITATIONS

Design	Overview	Limitations
Bonnenberg[3]	Used as a prototype	Extra overhead for on chip RAM and off chip traffic.
Curiger[5], [6]	Achieved better throughput	Unable to detect all possible errors during normal operation.
Zimmerman [6]	8 stage pipeline used	Efficient but incompatible for high speed network applications.
Wolter[8]	10 stage pipeline used	Additional area requirements due to on chip data management.
Salomao[9]	RAMs were used for storing subkeys	Unable to work with multiple keys.
Mencer[10]	56 stage pipeline used	Prior loading of keys before encryption.
Leong[11]	Bit Serial and deeply pipelined	Overall efficient, scope for further improvement
Cheung and Leong[15]	Bit parallel design	Highly efficient because of super pipelining.
Thaduri[16]	Optimized temporal parallelism	Area consumption is high compared to throughput.
Hamalainen[17]	Super-pipelined design	Multiplier complexity is high compared to the overall design.
Granado[18]	Dynamic and partial reconfiguration	Highly efficient in terms of throughput.

corresponding multiplication approaches are also mentioned in the table for easy understanding. The design issues and limitations (if any) of each of these designs are given in Table II.

In all these implementations, the power consumption issue is not taken into account for simplicity. But in real time high speed networks, apart from the fact that the line speed is in the order of Gb/sec, such complex hardware modules consumes a substantial amount of power, and for situations where the power supply or battery life is limited (in case of Ad hoc or wireless sensor networks), the design criteria and performance will be totally different [23]. Moreover, for FPGA implementations, there is still a large scope to exploit the reconfigurability and concurrency behavior of FPGA. For designing the multiplier module, Booth's algorithm using higher radices can be a good option for reducing the number of partial products and increasing the speed of multiplication process. Moreover, there is always as option to increase the throughput by introducing more number of pipelined registers inside the main design either in the form of Basic Looping or Full Loop Unrolling Architecture as mentioned in [24]. This can be further extended by using both Inner Round as well as Outer Round pipelining architecture or an intermix of both based on situation and design requirements.

V. CONCLUSION

Thus we have seen that there are various ASIC and FPGA implementations of IDEA cryptosystem and each implementation has certain limitations in terms of area requirements and time delay. The main goal of a hardware implementation is to optimize these parameters but at certain point, a trade-off between them needs to be maintained. While implementing in ASIC, the objective is to reduce silicon area to make it less costly but in FPGA, there is less restriction of area requirements because, the main goal is to optimize the term $A \times T^2$ where A is the area and T is the time required. Bonnenberg's design verified that IDEA can be implemented in hardware and from then the subsequent implementations are made for decreasing the constraints in one way or the other. The implementations Thaduri[16], Cheung[15], Hamalainen[17] and Granado[18] was found to be scalable in terms of throughput and hardware resources. The design of the modulo $(2^n + 1)$ multiplier in each case, was the most crucial for optimizing the performance of the entire cipher.

REFERENCES

[1] Xuejia Lai and James L. Massey. A proposal for a new block encryption standard. In *Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*, pages 389–404, New York, NY, USA, 1991. Springer-Verlag New York, Inc.

[2] Xuejia Lai, James L. Massey, and Sean Murphy. Markov ciphers and differential cryptanalysis. In *Advances in Cryptology – CRYPTO '91*, pages 17–38. Springer-Verlag, 1991.

[3] H. Bonnenberg, Andreas Curiger, Norbert Felber, Hubert Kaeslin, and Xuejia Lai. Vlsi implementation of a new block cipher. In *Proceedings of the 1991 IEEE International Conference on Computer Design on VLSI in Computer & Processors*, ICCD '91, pages 510–513, Washington, DC, USA, 1991. IEEE Computer Society.

[4] A.V. Curiger, H. Bonnenberg, and H. Kaeslin. Regular vlsi architectures for multiplication modulo (2n+1). *Solid-State Circuits, IEEE Journal of*, 26(7):990 –994, July 1991.

[5] A. Curiger, H. Bonnenberg, R. Zimmermann, N. Felber, H. Kaeslin, and W. Fichtner. Vinci: Vlsi implementation of the new secret-key block cipher idea. In *Custom Integrated Circuits Conference, 1993., Proceedings of the IEEE 1993*, pages 15.5.1 –15.5.4, May 1993.

[6] 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. *Solid-State Circuits, IEEE Journal of*, 29(3):303 –307, March 1994.

[7] L. Leibowitz. A simplified binary arithmetic for the fermat number transform. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 24(5):356 – 359, October 1976.

[8] Stefan Wolter, Holger Matz, Andreas Schubert, and Rainer Laur. On the vlsi implementation of the international data encryption algorithm idea. In *ISCAS*, pages 397–400, 1995.

[9] S.L.C. Salomao, V.C. Alves, and E.M.C. Filho. Hipcrypto: a high-performance vlsi cryptographic chip. In *ASIC Conference 1998. Proceedings. Eleventh Annual IEEE International*, pages 7 –11, September 1998.

[10] O. Mencer, M. Morf, and M.J. Flynn. Hardware software tri-design of encryption for mobile communication units. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 5, pages 3045 –3048 vol.5, May 1998.

[11] M. P. Leong, Ocean Y. H. Cheung, Kuen Hung Tsoi, and Philip Heng Wai Leong. A bit-serial implementation of the international data encryption algorithm idea. In *FCCM*, pages 122–131, 2000.

[12] Tsoi Kuen Hung. Cryptographic primitives on reconfigurable platforms, 2002.

[13] Seth Copen Goldstein, Herman Schmit, Mihai Budiu, Srihari Cadambi, Matt Moe, R. Reed Taylor, and R. Reed. Piperench: A reconfigurable architecture and compiler. *Computer*, 33:70–77, 2000.

[14] Emeka Mosanya, Christof Teuscher, Héctor Fabio Restrepo, Patrick Galley, and Eduardo Sanchez. Cryptobooster: A reconfigurable and modular cryptographic coprocessor. In *CHES*, pages 246–256, 1999.

[15] Ocean Y. H. Cheung, Kuen Hung Tsoi, Philip Heng Wai Leong, and M. P. Leong. Tradeoffs in parallel and serial implementations of the international data encryption algorithm idea. In *CHES*, number Generators, pages 333–347, 2001.

[16] M. Thaduri, S.-M. Yoo, and R. Gaede. An efficient vlsi implementation of idea encryption algorithm using vhdl. *Microprocessors and Microsystems*, 29(1):1–7, 2005.

[17] Antti Hämäläinen, Matti Tommiska, and Jorma Skyttä. 6.78 gigabits per second implementation of the idea cryptographic algorithm. In *FPL*, pages 760–769, 2002.

[18] José M. Granado, Miguel A. Vega-Rodríguez, Juan M. Sánchez-Pérez, and Juan A. Gómez-Pulido. Idea and aes, two cryptographic algorithms implemented using partial and dynamic reconfiguration. volume 40, pages 1032–1040, Amsterdam, The Netherlands, The Netherlands, June 2009. Elsevier Science Publishers B. V.

[19] Reto Zimmermann. Efficient vlsi implementation of modulo $(2^n + 1)$ addition and multiplication. In *IEEE Symposium on Computer Arithmetic*, pages 158–167, 1999.

[20] Yutai Ma. A simplified architecture for modulo $(2^n + 1)$ multiplication. *IEEE Trans. Computers*, 47(3):333–337, 1998.

[21] Zhongde Wang, Graham A. Jullien, and William C. Miller. An efficient tree architecture for modulo 2+1 multiplication. *VLSI Signal Processing*, 14(3):241–248, 1996.

[22] Ivan Gonzalez, Sergio López-Buedo, Francisco J. Gómez, and Javier Martínez. Using partial reconfiguration in cryptographic applications: An implementation of the idea algorithm. In *FPL*, pages 194–203, 2003.

[23] Nicolas Sklavos and Odysseas G. Koufopavlou. Asynchronous low power vlsi implementation of the international data encryption algorithm. *proceedings of 8th IEEE International Conference on Electronics, Circuits and Systems*, 3:1425–1428, September 2001.

[24] Paris Kitsos, Nicolas Sklavos, Michalis D. Galanis, and Odysseas G. Koufopavlou. 64-bit block ciphers: hardware implementations and comparison analysis. *Computers & Electrical Engineering*, 30(8):593–604, 2004.

Sourav Mukherjee: Sourav Mukherjee received the B.Tech. degree in Computer Science and Engineering from WBUT, Kolkata, India in 2008. He completed his M.Tech. in CSE from NIT Rourkela, India in 2011.

Bibhudatta Sahoo: Bibudatta Sahoo received the ME in CSE from NIT Rourkela, INDIA, in 1999. He is currently an assistant professor in the Dept of CSE, NIT Rourkela,India.