

# A High-performance and Area-efficient VLSI Architecture for the PRESENT Lightweight Cipher

Jai Gopal Pandey, Tarun Goel\*, Abhijit Karmakar

CSIR - Central Electronics Engineering Research Institute (CEERI)

\* Academy of Scientific & Innovative Research (AcSIR), CSIR-CEERI Campus

Pilani-333031, India

jai@ceeri.res.in

**Abstract**— Security and privacy are the prime concern in the emerging internet of things (IoT) and cyber physical systems (CPS) based applications. Lightweight cryptography plays an essential role for securing the data in this emerging pervasive computing environments. In this paper, we propose a high-performance and area-efficient VLSI architecture with 64-bit datapath for the PRESENT block cipher. The proposed architecture performs an integrated encryption/decryption operation for both 80-bit and 128-bit key lengths. The architecture is synthesized for the Virtex-5 XC5VLX110T FPGA device, available on the Xilinx ML-505 platform. It has been observed that the proposed architecture utilizes 0.73% and 0.87% of FPGA slices for 80-bit and 128-bit key lengths respectively. A throughput of 410 Mbps and power consumption is about 16 mW for both the key lengths.

**Keywords**— Lightweight cryptography; PRESENT block cipher; Integrated encryption/decryption; VLSI architecture; FPGAs.

## I. INTRODUCTION

The rapidly-growing area of internet of things (IoT) and cyber physical systems (CPS) is based on an ecosystem which eventually relies on billions of tiny interconnected computing-devices [1], [2]. The ability of selective computing, sensing, control and communication, makes these ever-ready devices effective, efficient and intelligent. Some of the day-to-day applications around these devices include car-locks, e-cash cards, electronic gadgets, digital lockers, secure communication, and many more. These tiny devices and their network create a wide-spread pervasive-computing infrastructure in emerging applications. Ever-increasing applications of these devices create an extensive demand of smart computing system and their energy-efficient field deployment. Besides design goals, security and privacy are the prime aspects of this IoT-based CPS infrastructure. Here, the field of cryptography and related ciphers provide a mechanism by which data can be efficiently secured. To secure the transmitted data through any electronic system, a variety of ciphers are used for years. The deployment trend of ciphers in electronic systems is shown in Fig. 1.

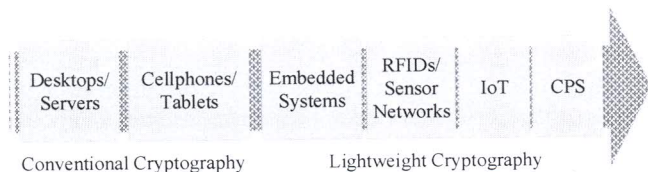


Fig. 1. Deployment trend of ciphers in electronic systems.

As shown in the Fig. 1, majority of the conventional cryptographic algorithms has been developed around desktop/server centric environments. Therefore, many of these cryptographic algorithms are generally unsuitable for implementation in constrained devices which are used in the modern-age applications. In many conventional cryptographic standards, the trade-offs between security, performance and resource requirements were optimized for desktop and server environments. This makes the implementation of conventional ciphers difficult in resource-constrained applications, and their performance may not be acceptable. The shift from desktop based applications to small-devices centric applications bring a wide range of security and privacy concerns. Lightweight cryptography provides solution tailored for resource-constrained devices and their efficient VLSI implementations [3].

Recently national institute of standards and technology (NIST) provided a report containing an overview of lightweight cryptography and an outline of NIST's plan for standardizing the lightweight cryptographic algorithms [4]. Further, a detailed taxonomy of the lightweight block ciphers can be found in [3] and [5]. Systematic survey of lightweight-cryptography ciphers and their software and hardware implementations with detailed description and related discussions can be found in [3], [5] and [6]. Here, it has been emphasized that efficient implementation of the ciphers are closely dependent on the selection of appropriate architecture, as they result in low implementation complexity and high-performance in actual realizations. To propose a new architecture for the lightweight cryptography, there is always trade-offs between the three prime objectives i.e. security, cost and performance, which is shown in Fig. 2 [7].

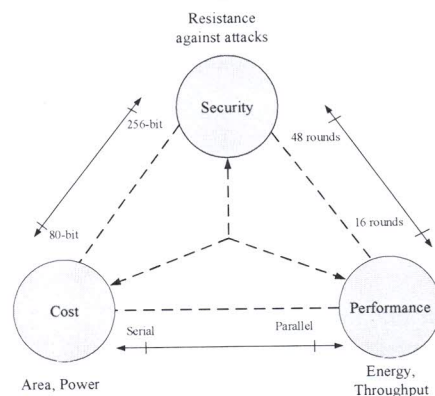


Fig. 2. Architectural trade-offs between security, cost and performance. Adapted from [7].

In the context of lightweight cryptography, ISO/IEC 29192-2 has standardized a symmetric block cipher algorithm in the year 2012, which is known as the PRESENT cipher [8]. The algorithm provides an adequate level of data security along with hardware-oriented performance attributes, which makes it a prominent choice for developing most of the secure and lightweight applications [5] and [9].

In this paper we propose a high-performance and area-efficient VLSI architecture for the PRESENT block cipher that completely integrates both encryption and decryption engines. The architecture has been implemented in the Xilinx Virtex-5 XC5VLX110T FPGA device [10]. The experimental results of the implementation show that the proposed architecture consumes a number of 126 slices for the 80-bit key and 150 slices for the 128-bit key lengths. The architecture is capable of running at a clock frequency of more than 210 MHz. The dynamic power dissipation is about 16 mW when the architecture has been operated with around two thousand random input vectors.

Rest of this paper is organized as follows: In Section II, an overview of the PRESENT algorithm is given. Section III is used to discuss the existing implementations of the PRESENT block cipher. An integrated architecture for the PRESENT block cipher along with a detailed description of various basic building blocks are proposed in Section IV. Section V is used to provide experimental results along with a comparison with an existing architecture from literature. Finally, conclusions are drawn in Section VI.

## II. THE PRESENT ALGORITHM

The PRESENT algorithm works on a block size of 64-bit and supports two key length variants of 80-bit and 128-bit. The principle of PRESENT cipher is based on the concept of substitution and permutation network (SPN). There are total 31 rounds and each round consists of an XOR operation, which is required to introduce a round key  $K_i$ , for  $0 \leq i \leq 31$ , where  $K_{31}$  is used for post-whitening operation [9]. A non-linear substitution layer operation is performed in each round and this layer consists of a 4-bit S-box which is applied 16-times in parallel. In addition to that there is a linear bitwise permutation layer. These operations are described below:

### A. The Key Schedule

The cipher requires a unique round key ( $K_i$ ) in each round, where the input key is stored in a key register  $K(k_{79}k_{78} \dots k_0)$  for 80-bit key or  $K(k_{127}k_{126} \dots k_0)$  for 128-bit key length.

### B. Add Roundkey Operation (AddRoundKey)

With current state  $b_{63} \dots b_0$  and for the given leftmost 64-bit of the round key  $K_i = k_{63}^i \dots k_0^i$  (or  $K_i = k_{127}^i \dots k_{64}^i$ );  $0 \leq i \leq 31$ , the AddRoundKey operation is defined as  $b_j = b_j \oplus k_j^i$  for  $0 \leq j \leq 63$  [9].

### C. The S-box and Inverse S-box

The PRESENT algorithm requires a 4-bit-to-4-bit S-box ( $S$ ) as  $\mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$  [9]. The 64-bit current state  $b_{63} \dots b_0$  is taken as sixteen 4-bit words  $w_{15} \dots w_0$ , where  $w_i = b_{4xi+3} \parallel b_{4xi+2} \parallel b_{4xi+1} \parallel$

$b_{4xi}$  for  $0 \leq i \leq 15$ . The output  $S[w_i]$  provides the updated state values as per [9]. The S-boxes are used in each of the rounds and in the key scheduling operation. For inverse S-box, the S-box does not satisfy  $S(S(x)) = x, x \in \mathbb{F}_2^4$ ; thus, same S-box cannot be used in encryption and decryption both. For the PRESENT cipher, a relation between the S-box and inverse S-box is given by expression,  $S^{-1}(S(x)) = x, x \in \mathbb{F}_2^4$ .

### D. The Bit Permutation (p-layer) and Inverse Bit Permutation (inv-p-layer) Operations

The bit permutation layer is used to move bit  $i$  of the state to bit position  $P(i)$  [9] and it is given by the following expression,

$$P(i) = \begin{cases} i.16 \bmod 63, & i \in \{0, \dots, 62\} \\ 63, & i = 63 \end{cases} \quad (1)$$

Similarly, the inverse bit permutation operation is defined by the below equation,

$$P^{-1} \begin{cases} i.16 \bmod 63, & i \in \{0, \dots, 62\} \\ 63, & i = 63 \end{cases} = i \quad (2)$$

In the following section, some of the work related to the implementations of the PRESENT cipher is provided.

## III. RELATED WORK

An architectural design space exploration for encryption and decryption operations can be found in [11]. In this Spartan-III FPGA device based implementation, the encryption and decryption operations require a total of 373 slices and 423 slices for the 80-bit and 128-bit key lengths respectively [11]. In another implementation, a cryptographic co-processor with encryption and decryption capabilities has been provided by [12]. In this paper the architectural exploration for serial, iterative and parallel variants of the cipher has been provided.

For encryption operation, an FPGA-based implementation of the PRESENT cipher that uses 117 slices of the Xilinx Spartan-3 XC3S50 FPGA device has been reported in [13]. Here, a throughput of 28.46 Mbps at a maximum frequency of 114 MHz has been obtained. Two different RAM-based implementations of PRESENT cipher have been provided in [14]. In the first implementation, the substitution box of the cipher has been realized into the FPGA slices. In the second implementation it has been integrated into the RAM. In these implementations, the first design occupies 83 slices and the second design consumes 85 slices of the Xilinx Spartan XC3S50 device. These realizations produce a throughput of 6.03 Kbps and 5.13 Kbps at 100 KHz system clock respectively.

Related to the encryption-only operation, one such implementation of the PRESENT cipher with 8-bit datapath has been given in [15]. This design utilizes 62 slices of the Xilinx Virtex-5 XC5VLX50 device and provide a latency of 295 clock cycles with a throughput of 51.32 Mbps at the maximum frequency of 236.574 MHz. Another implementation using 64-bit datapath has been reported in [16] that consumes 74 slices of the Xilinx Spartan-6 XC6SLX16-3CSG324C FPGA device. At maximum clock frequency of 221.63 MHz and with 33 clock latency, a throughput of 221.63 Mbps has been obtained. Similar

to this a 64-bit datapath based architecture has been synthesized on 87 slices of the Xilinx Virtex-5 XC5VLX50 FPGA device [17]. Here a latency of 47 clocks, maximum frequency of 221.64 MHz and a throughput of 341.64 Mbps have been reported.

As evident from the above, most of the authors have provided architecture for encryption-only operation and they have assumed that the decryption operation works opposite to that of the encryption operation and hence would require roughly same logic complexities and hardware resources. However, in our opinion the decryption operation is a bit complex in comparison to the encryption operation. This is due to the fact that to start the decryption operation, first we need the last round key that is generated from the key scheduling operation. In connection to this, one of the implementations in which both encryption and decryption operations have been tackled [11], assumes that last round key is available at the starting of the decryption operation. They have considered that the key is static in nature throughout all of the encryption and decryption operations for all the input blocks. In another implementation by [17], for PRESENT cipher context it has been given that the decryption operation requires almost same area as of encryption when implemented separately. In the following section, an integrated encryption and decryption architecture has been proposed and described in detail.

#### IV. AN INTEGRATED VLSI ARCHITECTURE FOR THE PRESENT LIGHTWEIGHT BLOCK CIPHER

The proposed architecture for an integrated encryption and decryption of PRESENT block cipher is shown in Fig. 3. Here, an iterative type of architecture is considered for saving the resources and computation time. To implement the PRESENT

block cipher a 64-bit datapath is chosen, mainly to implement the permutation operation efficiently.

In the proposed architecture there are three main components which are encryption/decryption engine, key scheduling unit and controller. There is a 1-bit input signal 'enc\_dec' which is used to select the encryption or decryption operations. If 'enc\_dec' is at logic '1' level then encryption operation is performed, else the decryption operation is executed. An up-down counter facilitates the integrated encryption/decryption operation. The main building blocks of the proposed architecture have been arranged in different subsections which are described below.

##### A. Datapath of the Integrated PRESENT Architecture

As shown in the Fig. 3, the architecture consists of a set of registers, multiplexers and XOR gates. The bit permutation (1) and inverse bit permutation operations (2) are simple bit-transposition, which require only simple wirings. There is one 64-bit multiplexer which is required to switch the state between the load phase (Input) and the intermediate state. After that the multiplexer passes the state to a 64-bit state register. This register is used to store the intermediate state and passes it to the 64-bit XOR gate. This gate performs the XOR operation of intermediate state coming from the state register with 64-bit round key coming from the key scheduling unit. In the architecture, both the S-box ( $S$ ) and inverse S-box ( $S^{-1}$ ) are realized by the area-optimized combinational logic implementation. To differentiate between the encryption and decryption operations, two 64-bit multiplexers are deployed in the datapath. In the proposed architecture, the inputs and output are registered. The output register is added to synchronize the output with the last round.

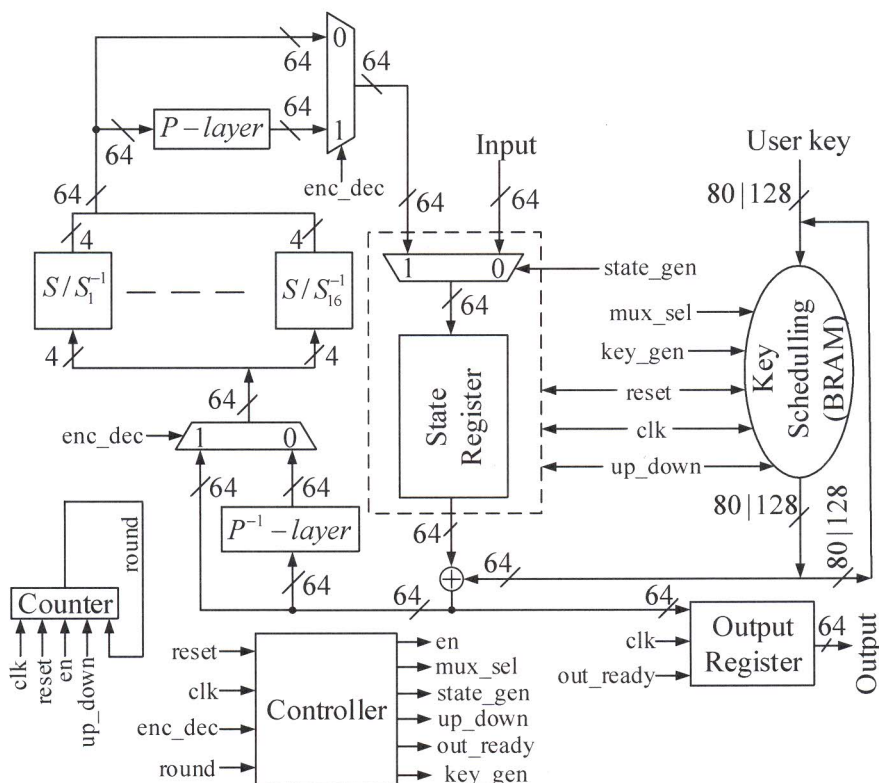


Fig. 3. A VLSI architecture of the PRESENT block cipher for an integrated encryption/decryption operation.



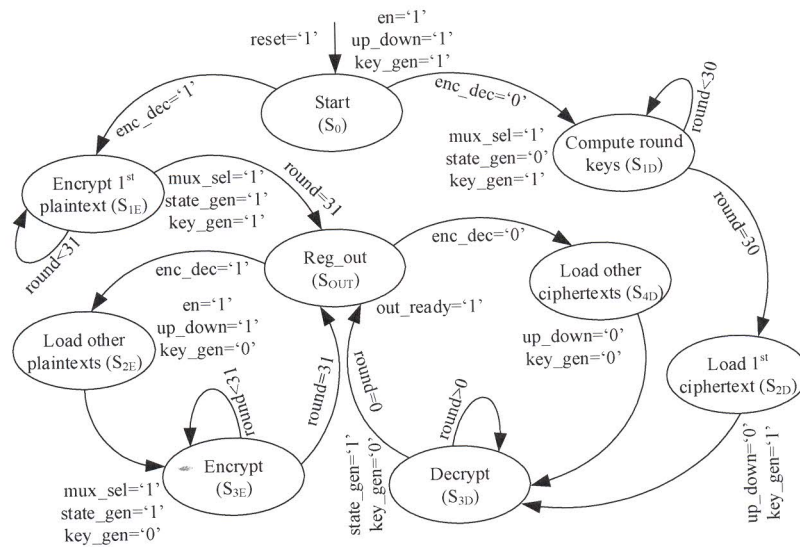


Fig. 5. Controller for the integrated encryption and decryption operations.

In the processing of the decryption operation, which is shown in the Fig. 5, the state switches at  $S_{1D}$  from the initial state  $S_0$ . In the state  $S_{1D}$ , all the round keys are computed and stored in the BRAM. In this state, the decryption process does not start as the 'state\_gen' signal is at logic '0'. When the counter value reaches to 30, the state is switched at  $S_{2D}$  state. Here, first block of ciphertext is loaded to the state register and now the counter starts to work as down counter by switching the 'up\_down' signal to logic '0'. Further, the state is switched to  $S_{3D}$  state, where the decryption process starts by assigning the 'state\_gen' signal to logic '1'. Further, the key generation unit completes its computation operation with the 'key\_gen' signal becoming logic '0'. As counter value becomes 0, the state switches to the final state  $S_{OUT}$  and first block of decrypted ciphertext is available through the output register as 'out\_ready' signal becomes logic '1'. Next, to decrypt the remaining blocks of ciphertext the state does not go back to the initial state because pre-computed round keys are available in the BRAM. So, the state moves to  $S_{4D}$  state, which is very similar to the state  $S_{2D}$  except that 'key\_gen' signal is at logic '0'. In this state, other blocks of ciphertext are loaded into the state register. To perform the decryption operation for the rest blocks, the state switches to the  $S_{3D}$  state. This cycle continues until all the blocks of ciphertext are decrypted.

## V. EXPERIMENTAL RESULTS FOR AN FPGA DEVICE

The proposed architecture is implemented in the VHDL language, and synthesized using Xilinx Design Suite 14.7 for the Xilinx Virtex-5 XC5VLX110T-1-FF1136 FPGA device on Xilinx ML-505 platform. The synthesis process for the implementation has been configured with design goal as *balanced* and strategy as *Xilinx default*. The FPGA device utilization summary of the proposed architecture is given in Table I.

As per the table, the architecture with 80-bit key ( $PRE_{80}$ ) is consuming only 0.73% slices while the proposed architecture with 128-bit key ( $PRE_{128}$ ) is consuming 0.87% slices. The  $PRE_{80}$  architecture consumes  $32 \times 80$  bit size block memory, whereas, the  $PRE_{128}$  architecture needs  $32 \times 128$  bit size of memory to store the intermediate keys.

TABLE I. DEVICE UTILIZATION SUMMARY ON THE XILINX VIRTEX-5 XC5VLX110T FPGA DEVICE.

Elements	Available Resources	Resource Utilization	Resource Utilization
		$PRE_{80}$	$PRE_{128}$
LUTs	69120	348	396
Registers	69120	137	137
Total Slices	17280	126	150

The performance of the design is evaluated in terms of power dissipation, latency, maximum frequency and throughput. The performance of the proposed architecture is given in Table II.

TABLE II. PERFORMANCE ON THE XILINX VIRTEX-5 XC5VLX110T FPGA DEVICE.

Elements	Resource Utilization	Resource Utilization
	$PRE_{80}$	$PRE_{128}$
Latency	33	33
Max. frequency (MHz)	215.42	212.13
Throughput (Mbps)	417.79	411.41
Efficiency (Mbps/#Slices)	3.32	2.74
Power (mW)	16.59	16.80

As given in the Table II, the architecture is consuming around 16 mW power at 215 MHz frequency. Both the operations have a latency of 33 clock cycles. However, the decryption operation requires an additional 33 cycles for round key generation for the first block of ciphertext. Throughput of the design is found to be around 410 Mbps for both the key lengths, that is computed for 64-bit datapath by expression,  $throughput = (\max. frequency \times total\ no.\ of\ bits) / latency$ .

To compare the proposed design with an existing design available in literature, the selected design metrics are: slice LUTs, registers and total number of consumed slices. To perform a comparison at the architectural-level, the proposed integrated architecture is tuned to match the architectural capability of [11]. Therefore, for the comparison the key scheduling unit is implemented using *on-the-fly* mode rather

