A High-performance VLSI Architecture of the PRESENT Cipher and Its Implementations for SoCs

Abstract—**The essence of internet-of-things (IoT) and cyber-physical systems (CPS) infrastructures is primarily based on privacy and security of communicated data. In these resource-constrained applications, lightweight cryptography plays a vital role for data security. In this paper, we propose a high-performance and power-efficient VLSI architecture for the PRESENT block cipher and its integration in a system-on-chip (SoC) environment. The architecture is based on 8-bit datapath and requires 48 clock cycles for processing of 64-bit plaintext and 128-bit key. The architecture is validated using Xilinx Virtex-5 xc5vfx50 FPGA device, where it consumes 84 slices, provides 379.78 MHz maximum frequency and 506.37 of Mbps throughput. It consumes 36.57 mW of dynamic power, 57.95 nJ energy and provides 0.91 nJ/bit energy/bit. In comparison to an exiting architecture, the proposed architecture provides much improved performance. Further, an ASIC implementation of the architecture is done in SCL 180 nm technology for its usage as an intellectual-property (IP) core in SoCs. The core consumes 1785, 2-input NAND gate equivalent (GE), with 1.55 mm^2 area and can be operated up to 448 MHz clock frequency. At 100 MHz clock frequency, 0.273 mW of total power dissipation, 133 Mbps throughput, 130 nJ energy and 16.36 nJ/bit energy/bit is obtained.**

Index Terms—**Lightweight cryptography; PRESENT block cipher; VLSI architectures; ASIC; SoCs.**

1. Introduction

Recent proliferation of internet-of-things (IoT) [1], cyber-physical systems (CPS) [2] and edge computing [3] technologies enable devices to interconnect through Internet. With these enablers, a new class of applications is emerging by an amalgamation of machine learning and artificial intelligence (AI) techniques. These applications heavily rely on communicated data that can be between human-machines or their any combinations. A pervasive computing infrastructure is shown in Figure 1, where, small computing devices of credit-card sized are deployed for fulfilling the need of sensing, control, communication and computation. The ever-increasing deployment trend bring a substantial change in the design of electronic circuits and systems [1]. Accordingly, application and system developers are focused toward design of applications, related intellectual-properties (IPs) and system-on-chips (SoCs) that are optimized for resource, latency, power and bandwidth design metrics.

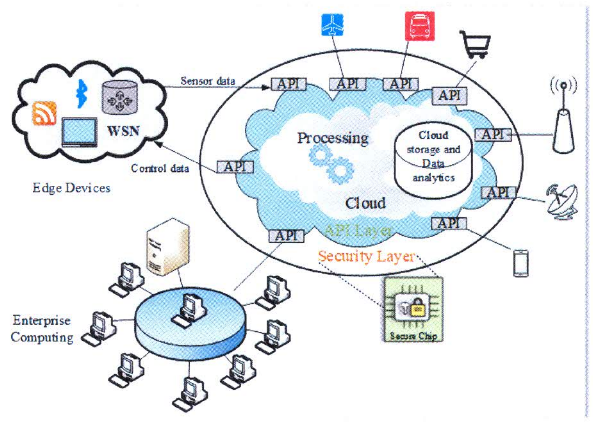

Figure 1. An infrastructure of cutting-edge IoT, CPS and edge computing technologies.

In a gamut of emerging IoTs applications, secure communication mechanism for restricting access of information is very crucial [2], [4] and [5]. Here, for securing electronic data, cryptography is very important. In encryption operation of cryptography, data is converted into a secure form which is known as ciphertext. Symmetric key cryptography algorithms and their hardware solutions that provide very low area and energy requirements for IoT applications are ideally suited [4], [6]. Efficient implementations for the criptographic algorithms are much dependent on selection of appropriate architectures as per the need of performance and energy metrics. In the lightweight cryptography context, ISO/IEC 29192-2 has standardized symmetric block cipher algorithm PRESENT in year 2012 [7]. The algorithm provides adequate level of security goals and hardware-oriented performance attributes. This makes it a preferred choice in lightweight cryptographic based applications [8].

In this paper we propose a high-performance and power-efficient custom architecture for the PRESENT block cipher and its VLSI implementations. The architecture works on the 64-bit input and of 128-bit user key. It processes 8-bit data at a time and provides ciphertext in 48 clock cycles, with registered inputs and outputs. An FPGA implementation of the proposed architecture is done in Xilinx Virtex-5 xc5vfx50 FPGA device and compared with the architecture of [9]. Here, the proposed architecture works on 50.8% increased maximum frequency, 28.8% improved throughput and 40.8% improved efficiency. Performance evaluation in terms of power, energy, energy/bit and efficiency is also performed. Further, an ASIC implementation of the architecture is done in SCL 180 nm technology for its usage as an intellectual-property (IP) core in system-on-chip (SoC) environment. Here, the chip operates on 3.3V and the core works at 1.8V. Gate equivalent (GE) of the proposed architecture is 1785 GEs with area of 1.55 mm^2, the architecture can be operates up to 448 MHz clock frequency. With 100 MHz operating frequency, a throughput of 133.333 Mbps, energy 130.880 nJ and energy/bit 16.36 nJ/bit is obtained. The efficiency of the design is 0.075. Total power consumption is 0.273 mW, where, the dynamic component is 0.256 mW and the static one is 0.016 mW.

Rest of this paper is structured as: an introduction of the PRESENT algorithm is given in Section 2. Section 3 is used to discuss some of the related work. Section 4 is use to propose the architecture for the PRESENT cipher and its detail description. Section 5 is used to provide experimental results for an FPGA and an ASIC. This section is also used to provide architectural comparison with an established architecture. Conclusion is provided in Section 6.

2. The PRESENT Cipher Algorithm

The algorithm works on 64-bit and 80/128-bit user keys. There are 31 internal rounds and structure of the cipher is based on SP-network [8]. Each rounds require an XOR operation, which is required to introduce a round key K_i for $0 \leqslant i \leqslant 31$ in which K_{31} is used for post-whitening operation. Additionally, there is a non-linear substitution layer with a 4-bit substitution box layer (S-box) and a linear bit-wise permutation layer. The pseudo-code of the algorithm is described in Figure 2. As depicted in the figure, the algorithm requires four main functions, which are explained below.

```
generateRoundKeys()
for (i=0 to 30) do
AddRoundKey(State,$K_i$)
sboxlayer(State)
p-layer (State)
end for
AddRoundKey(State,$K_{31}$)
```

Figure 2. A top-level algorithmic description of the PRESENT cipher.

2.1. Key Scheduling

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

2.2. Add Roundkey Operation (AddRoundKey)

With current state $b_{63}...b_0$ and for the given leftmost 64-bit of the round key $K_i = k^i_{63}\dots k^i_0$ (or $K_i = k^i_{127}\dots k^i_{64}$);$0 \leqslant i \leqslant 31$, the *AddRoundKey* operation is defined as $b_j = b_j \oplus k^i_j$ for $0 \leqslant j \leqslant 63$ [8].

2.3. Substitution Box Layer

The cipher requires a 4-bit S-box (S) as $\mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$ [8]. 64-bit current *State* $b_{63}...b_0$ is taken as sixteen 4-bit word $w_{15}...w_0$, where, $w_i = w_{4\times i+3} \parallel w_{4\times i+2} \parallel w_{4\times i+1} \parallel w_{4\times i}$ for i $0 \leq i \leq 15$. $S[w_i]$ provides updated value [8]. S-box is used in each round and in the key scheduling operation.

2.4. Bit Permutation (p-layer) Operation

The bit permutation layer is used to move bit *i* of the *State* to bit position $P(i)$ [8].

3. Related Work

Architecture for serial, iterative and parallel variants has been provided in [10] and [11], [12]. A serial architecture is given in [9], it requires 16 clock cycles for loading the 64-bit plaintext and 128-bit user key. To perform an intermediate round 9 clocks cycles are required. As there are 31 rounds in the PRESENT cipher, therefore, $31 \times 9 = 279$ clock cycles are required for the complete rounds. In addition, to produce the final output 8 clock cycles are needed. Thus, the total latency of the serial architecture is $16 + 279 + 8$, i.e., 303 clock cycles. A rough estimate of the gate count of the design as reported in [12] is around 1100, 2-input NAND gates, referred as gate equivalents (GEs). The architectures reported in [10] and [12], take 01 clock cycle to bring-in the input data, 31×8, i.e., 248 clock cycles are required for the complete processing of the substitution operation and 01 clock cycle is needed for the output. Thus, total 250 clock cycles are needed. The gate count of the design is around 1100 GEs. In iterative architectural category, a 64-bit datapath centric architecture has been used for encryption in [13]. It takes 64-bit input; 80/128-bit keys and consumes 57/69 slices of the Xilinx xc5vfx70t FPGA device.

The 8-bit datapath based architectures have been provided in [10] and [9]. The iterative architecture of [9] has also been analyzed by [12], here, for bringing 64-bit plaintext and 128-bit key, 16 clock cycles are needed. To process the intermediate *State* for the encryption operation, 31 clock cycles are required, and 8 clock cycles are required for producing 64-bit ciphertext. Thus, a total of $16 + 31 + 8 = 55$ cycles are required for processing of one block of data. As per [12], the design of [9] needed around 1797 GEs. The architecture of [9] when implemented in Xilinx Virtex-5 xc5vlx50 FPGA device, it takes 87 slices [9] with 47 clocks latency for producing 1-byte of the ciphertext. Maximum clock frequency is 221.64 MHz and throughput is 341.64 Mbps. The gate equivalent is 1800 GEs. In another implementation of 8-bit datapath based has been provided in [14]. The design consumes 62 slices of the Xilinx Virtex-5 XC5VLX50 device and provides latency of 295 clock cycles with a throughput of 51.32 Mbps at the maximum frequency of 236.574 MHz.

4. An Architecture of the PRESENT Cipher and Its Integration in an SoC Environment

An architecture for the PRESENT cipher is shown in Figure 3. The architecture is based on 8-bit datapath. Three main units of this architecture are an encryption engine, key scheduling and controller. The key scheduling block takes 128-bit input key and generates thirty one intermediate round keys for the cipher. The permutation/expansion is a simple bit-transposition, which is implemented by routing wires. The main building blocks of the architecture are arranged in different subsections and described below.

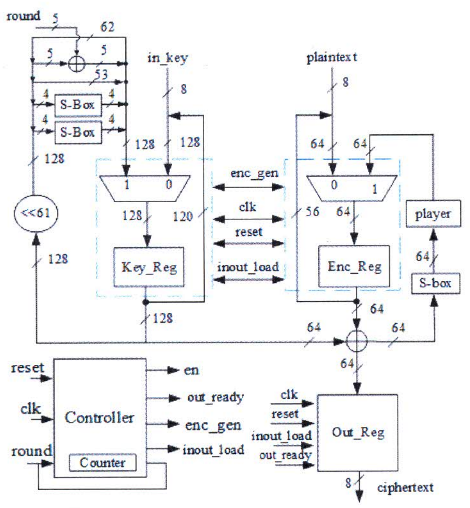

Figure 3. Proposed architecture for the PRESENT block cipher with 128-bit user key.

4.1. Datapath of the Proposed Architecture

The architecture shown in Figure 3 works on 64-bit plaintext and 128-bit user key. These inputs are provided through 8-bit input ports. In first 8 clock cycles, 64-bit plaintext and 64-bit of 128-bit key are stored in *Enc_Reg* and *Key_Reg* respectively. Subsequently, in next 8 clock cycles, remaining 64 bits input key (*in_key*) are stored in the *Key_Reg*. These registers work as a universal shift register with byte-wise I/Os and support fully parallel read and load operations. The *Enc_Reg* is utilized for storing the internal states of encryption operation. In addition, multiplexers of size 64-bit and 128-bit switch data between the load and round phases. The datapath contains sixteen S-boxes to form the *sboxlayer* and two S-boxes for the key scheduling operation. Along with this, a 64-bit and a 5-bit XOR gates, a 4-bit and a 5-bit up-counters are employed for counting needs. Data of the *Enc_Reg* is mixed with intermediate round key i.e., XORed with first 64-bit of round key.

The mixed *State* is passed to *sboxlayer* for further processing, and provides 64-bit data concurrently to *P-layer*. After that, the data is fed to the *Enc_Reg* through a multiplexer. Here, round keys are computed *on-the-fly* which are used for mixing with the *State* and the S-box is realized by the area-optimized combinational logic circuit. After completion of 31 internal rounds, 64-bit ciphertext is stored in *Out_Reg* register. This register is used to provide synchronization with the next block of input data that needs to be processed. With *out_ready*='1'and *inout_load*='1'control signals, the *Out_Reg* provides 8-bit of ciphertext and simultaneously, next set of input is stored in stored. Thus, a total of $16 + 31 + 1 = 48$ clock cycles are required for complete data processing. However, the last block of 64-bit ciphertext requires 55 clock cycles. The associated controller of the architecture is given below.

4.2. A Controller for the Encryption Operation

To control the key generation and encryption engine, a controller is designed. There are six states in the controller, and it generates *en*, *inout_load*, *en_gen* and *out_ready* control signals which are shown in Figure 4.

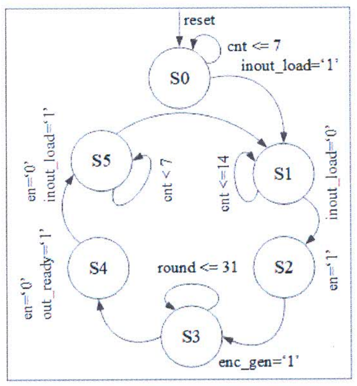

Figure 4. A FSM for the PRESENT cipher with 128-bit key.

As shown in Figure 4, in state S0, for first eight clock cycles the control signal *inout_load*='1', which enables the *Enc_Reg* and *Key_Reg* registers to load 64-bit plaintext and 64-bit of 128-bit input key. The reaming 64-bit of the key is loaded in state S1 in subsequent six clock cycles, then the machine switches to state S2. In S2, the round counter is enabled with the signal *en* is at logic 1, after that the machine goes into S3. In state S3, multiplexers are switched and the signals *en*='1'and *enc_gen*='1'. These signals are used to start the intermediate operations. The machine waits in S3 until counter value reaches to 31. Then, the machine switches to state S4 where counter is disabled by signal *en*='0'; and *enc_gen*='1', *out_ready*='1'. In the state S5, 8-bit ciphertext is available and next block of plaintext with first 64 bits of next user key are loaded into the shift registers. After that, the machine switches to state S1 for loading the remaining bits of user key and works sequentially in the states S1-S5, until the *reset* signal occurs, where the FSM restarts from state S0.

4.3. Integration of the Crypto Core in an SoC Environment

To integrate the proposed architecture as an IP core, an open source SoC environment provided by *Cobham Gaisler* is used. It provides a range of IP cores in its *GRLIB* library. Various supporting tools for testing and validating for custom SoCs have also been provided by [15]. These IP cores are vendor independent and support multiple EDA tools and target technologies both for ASICs and FPGAs. These cores communicate with a common on-chip bus interface of advanced microcontroller bus architecture (AMBA). We can interface custom cores by requisite wrapping with AMBA advanced high-performance bus (AHB) or advanced peripheral bus (APB).

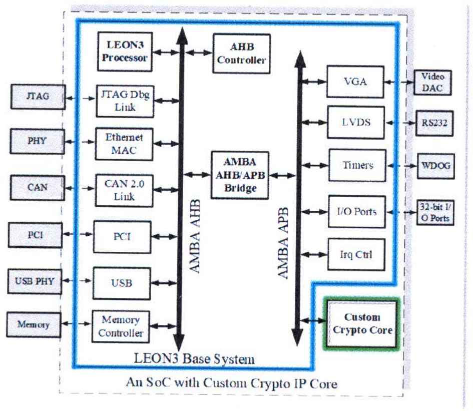

Figure 5. PRESENT crypto core in an SoC environment.

In *GRLIB* IP environment, input of the AMBA AHB/APB bridge, coming from the AHB bus is called *ahbi* signal, it uses VHDL record type *ahb_slv_in_type*. An output of the bridge is called *ahbo* that goes back to the AHB bus and uses the record type *ahb_slv_out_type*. The output of the bridge to the crypto core is *apbo* signal and it uses record type *apb_slv_out_type*. On the other hand, the bridge's input called *apbi*; it comes from the crypto core and uses the record type *apb_slv_in_type*. The core works as a slave where, a *LEON3* processor acts as a master. The master requests data from the slave by enabling the APB bus with appropriate read/write signals as per the AHB/APB protocols [15]. An interface of the PRESENT crypto core in the SoC environment is shown in Figure 5. Here, the *LEON3* base system is available from the *GRLIB* library and the developed crypto as a custom IP core is interfaced with the APB bus.

4.4. An ASIC Implementation of the Proposed Architecture

A view of the crypto chip in SCL 180 nm technology is shown in Figure 6. The chip poses 32 I/O pins, which are, clock (clk), reset, 8-bit input data (input), 8-bit user key(inkey), 8-bit output(output). Additionally, two pair of VDD (VDDO) and ground (VSSO) for the wrapper, and one pair of VDD (VDD) and ground (VSS)for the core are required. Here, the wrapper of the chip uses *pc3d01* as an input pad; *pc3o01* for output pad, *pc3c01* for the clock pad and *pfrelr* is used as corner pads. To supply 3.3 V power to the wrapper *pvda*, pad is used with *pv0a* as ground. Similarly, to supply 1.8 V power to the core, *pvdi* pad is used with *pv0i* as ground.

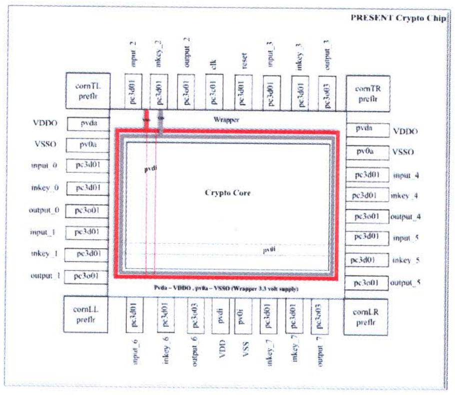

Figure 6. A view of the complete PRESENT crypto chip.

5. Experimental Results and Discussions

The proposed architecture is implemented in VHDL language and the FPGA prototype is performed in Xilinx Virtex-5 xc5vfx50ff324-1 device by using *ISE Design Suite* 14.7. In the tool, project *design goals and strategies* with synthesis and place and route (PnR) effort properties is set at high and the PnR *extra effort* is fixed at continue on impossible. After successful validation of the architecture, it has been implemented as an ASIC. Here, Synopsys *Design Compiler* is used to synthesize the architecture in SCL 180 nm technology [16]. To perform physical design, Synopsys *IC Compiler* tool and to perform power analysis, *Power Compiler* tool is utilized. Throughput is computed as $Throughput = (\max.\ freq. \times total\ bits)/latency$. Energy is obtained by $Energy = (power \times latency)/operating\ freq.$ and efficiency of the architecture is computed as $Efficiency = throughput/gate\ equivalent$. The gate equivalent (GE) is replaced by utilization of total number of slices for FPGA. The FPGA and ASIC implementation results are arranged in two different subsection which are described below.

5.1. Results of an FPGA Implementation and Architectural Comparison

An architectural compassion between the proposed architecture with [9] for Xilinx Virtex-5 xc5vfx50 ff324-1 FPGA device is performed. Resource comparison is provided in Table 1 and performance is shown in Figure 7. The selected design metrics for the comparison are: maximum frequency, throughput and efficiency.

TABLE 1. RESOURCE CONSUMPTION FOR THE PROPOSED AND [9] ARCHITECTURES USING XILINX VIRTEX-5 XC5VFX50 FPGA DEVICE.

Elements	Available Resources	Architecture [9]	Proposed Architecture
Slice LUTs	28,800	285	257
Slice Registers	28,800	200	259
Total Slices	7,200	87	84

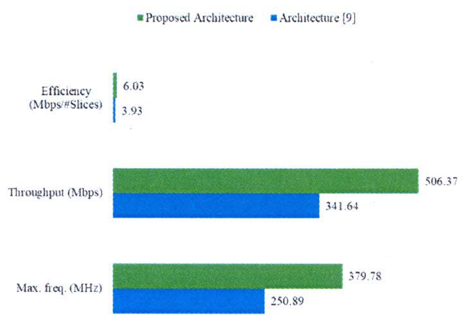

Figure 7. Performance comparison of proposed architecture with [9] using Xilinx Virtex-5 xc5vfx50 ff324-1 FPGA device.

In comparison to the implementation [9], there is a little improvement in the slice consumption. However, performance of the proposed architecture is significantly improved, which is depicted in Figure 7. In comparison to [9], the proposed architecture is able to work on an increased maximum frequency by 51.4%, which results in 48.2% increased throughput. By this, efficiency of the design is improved by 53.4% in comparison to the architecture of [9]. It is to be noted that in the proposed architecture we have provided a provision of registered output which is required in many practical systems. Similarly, a comparison with existing architectures (*PRE* and *PRE_O1*) given by [11], iterative architecture (*Iterative_Tay*) given by [14] and the serial (*Serial_Hanley*), iterative (*Iterative_Hanley*) architectures given by [9] is shown in Figure 8.

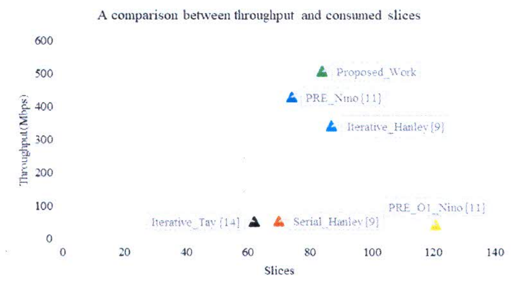

Figure 8. A comparison on throughput and slices with existing architectures of [9], [11] and [14].

Apart from computation of resource utilization and performance evaluation, an analysis on power and energy consumption is also performed. Here, *ModelSim* SE-64 10.1c HDL simulator is used for generating stimulation files in switching activity interchange format (SAIF) format, which is used for performing power analysis of the design. Xilinx *XPower Analyzer* tool is used for the computation of static and dynamic power dissipation. The tool requires native circuit description (NCD) and physical constraints (PCF) files. These files are generated by synthesizing the design for the specific FPGA using Xilinx *ISE Design Suite*. To compute dynamic power dissipation, a set of two hundred random test vectors are provided to the architecture. A result on power consumption and associated energy consumption is shown in Table 2. The architecture consumes 36.57 mW of dynamic power. The energy consumption is 57.95 nJ and energy/bit is 0.91 nJ/bit.

TABLE 2. POWER AND ENERGY COMPUTATION OF THE PROPOSED ARCHITECTURE USING XILINX VIRTEX-5 XC5VFX50 FPGA DEVICE.

Elements			Value
Power (mW)	Dynamic Power (PD)	Clock (Pc)	19.23
		Logic (Pl)	7.74
		Signals (Ps)	9.02
		IOs (Pio)	0.58
		Total (PD =Pc+Pl+Ps+Pio+Pb)	36.57
	Static Power (PS)		421.91
	Total Power (PD + PS)		458.48
Energy (nJ)			57.95
Energy/bit (nJ/bit)			0.91

Further, for performing feasibility evaluation of the proposed architecture as a suitable cryptograpic (crypto) IP candidate, the architecture is implemented as an independent ASIC component and its result is given below.

5.2. Results of the ASIC Implementation

The gate equivalent of the architecture for SCL 180 nm technology is 1785 GEs and the architecture can run at a maximum frequency of 448 MHz. However, since we are not providing clock through any phase-locked loop (PLL) and oscillator for clocking needs, we operated the design at a lower clock frequency of 100 MHz. Experimental result on utilization in terms of standard cells, pad cells, core size, pad core size and complete chip area is shown in Table 3.

TABLE 3. STANDARD CELLS UTILIZATION SUMMARY FOR SCL 180 NM TECHNOLOGY.

Elements	Area (mm^2)
Std cells	0.033
Pad cells	0.770
Core size	0.296
Pad core size	0.555
Chip size	1.550

A layout of the chip is shown in Figure 9. It is generated after placement and routing with I/O padding. there are 32 I/O pins, clock (clk); reset; 8-bit input data, user key and output. Two pair of VDD and ground for the wrapper and one pair for the core. The detailed description of the I/Os has been discussed in Section 4.4.

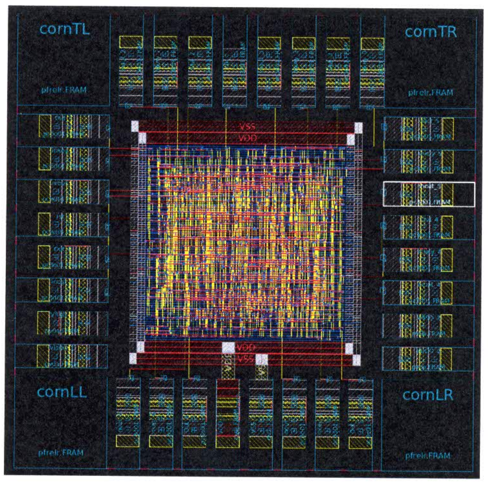

Figure 9. A layout of the proposed architecture in SCL 180 nm technology.

Here, *Synopsys IC Compiler*, *Power Compiler* and *Prime Time* tools are used for physical design and power analysis. The *Power Compiler* tool require two files. The first file is a *SAIF* file and the second is a standard parasitic exchange format (*SPEF*) file. The *SAIF* file is generated from *ModelSim* simulator by applying two hundred random vectors. The SPEF file is generated by the *IC compiler* tool after completing the place and route (*PnR*). Performance of the architecture at 100 MHz reference clock frequency for SCL 180 nm technology is shown in Table 4.

TABLE 4. PERFORMANCE OF THE ARCHITECTURE IN SCL 180 NM TECHNOLOGY.

Elements	Utilization	
Dynamic power (mW)	IO pads	0.102
	Switching	0.065
	Cell internal	0.090
	Total dynamic power	0.256
Static power (mW)		0.016
Total power (mW)		0.273
Throughput (Mbps)		133.333
Energy (nJ)		130.880
Energy/bit (nJ/bit)		16.36
Efficiency		0.075

As shown in the Table 4, total power consumption of the chip is 0.273 mW and energy/bit is 16.36 nJ/bit.

6. Conclusion

A high-performance and power-efficient VLSI architecture for PRESENT block cipher and its integration in a system-on-chip (SoC) environment has been presented. The architecture is based on 8-bit datapath and requires 48 clock cycles for processing of 64-bit plaintext and 128-bit user key. The architecture is validated as an intellectual-property (IP) core in Xilinx Virtex-5 FPGA device, where it takes 84 slices, provides 379.78 MHz maximum frequency and 506.37 Mbps of throughput. Further, an ASIC implementation of the architecture as an IP core is done in SCL 180 nm technology. The core consumes 1785, 2-input NAND gate equivalent (GE) and 1.55 mm^2 area. The chip can work up to 448 MHz frequency and we have operated it at 100 MHz. Total power consumption is 0.273 mW, where the dynamic component is 0.256 mW and the static one is 0.016 mW. A throughput of 133.333 Mbps, energy 130.88 nJ and 16.36 nJ/bit of energy/bit has been obtained.

References

[1] M. Alioto. *Enabling the Internet of Things: From Integrated Circuits to Integrated Systems*. Springer, 2017.

[2] E. A. Lee and S. A Seshia. *Introduction to embedded systems: A cyber-physical systems approach*. MIT Press, 2016.

[3] W. Shi and S. Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016.

[4] T. Xu, J. B. Wendt, and M. Potkonjak. Security of iot systems: Design challenges and opportunities. In *Proceedings of the 2014 IEEE/ACM Int'l Conf. on Computer-Aided Design*, pages 417–423. IEEE Press, 2014.

[5] Al-Sakib Khan Pathan. *Securing cyber-physical systems*. CRC Press, 2015.

[6] Thomas Eisenbarth and Sandeep Kumar. A survey of lightweight-cryptography implementations. *IEEE Design & Test of Computers*, 24(6), 2007.

[7] ISO/IEC 29192-2:2012. Information technology – security techniques – lightweight cryptography – part 2: Block ciphers.

[8] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, and et al. Present: An ultra-lightweight block cipher. In *CHES*, volume 4727, pages 450–466. Springer, 2007.

[9] N. Hanley and M. ONeill. Hardware comparison of the iso/iec 29192-2 block ciphers. In *VLSI (ISVLSI), 2012 IEEE Computer Society Annual Symp. on*, pages 57–62. IEEE, 2012.

[10] C. Rolfes, A. Poschmann, G. Leander, and C. Paar. Ultra-lightweight implementations for smart devices–security for 1000 gate equivalents. In *Int'l Conf. on Smart Card Research and Advanced Applications*, pages 89–103. Springer, 2008.

[11] C. A. Lara-Nino, M. Morales-Sandoval, and A. Diaz-Perez. Novel fpga-based low-cost hardware architecture for the present block cipher. In *Digital System Design (DSD), 2016 Euromicro Conf. on*, pages 646–650. IEEE, 2016.

[12] C. A. Lara-Nino, A. Diaz-Perez, and M. Morales-Sandoval. Lightweight hardware architectures for the present cipher in fpga. *IEEE Tran. on Circuits and Systems I: Regular Papers*, 2017.

[13] J. G. Pandey, T. Goel, and A. Karmakar. An efficient vlsi architecture for present block cipher and its fpga implementation. In *Int'l Symp. on VLSI Design and Test*, pages 270–278. Springer, 2017.

[14] J. J. Tay, M. L. D. Wong, M. M. Wong, C. Zhang, and I. Hijazin. Compact fpga implementation of present with boolean s-box. In *Quality Electronic Design (ASQED), 2015 6th Asia Symp. on*, pages 144–148. IEEE, 2015.

[15] J. Gaisler, S. Habinc, and E. Catovic. Grlib ip library users manual. *Aeroflex Gaisler*, 2010.

[16] Semi-Conductor Laboratory (SCL). Semi-conductor laboratory (scl), gov. of india, 2018.