 https://doi.org/10.3311/PPee.13424

Periodica Polytechnica Electrical Engineering and Computer Science, 63(4), pp. 244–253, 2019

Hardware Signature Generation Using a Hybrid PUF and FSM Model for an SoC Architecture

Jagadeesh Kokila[1*], Arjun Murali Das[1], Basha Shameedha Begum[1], Natarajan Ramasubramanian[1]

[1] Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli, Tamil Nadu, 620015, India
[*] Corresponding author, e-mail: jk.cse09@gmail.com

Received: 10 November 2018, Accepted: 26 April 2019, Published online: 14 June 2019

Abstract

Security is becoming an important issue in the recent System on Chip (SoC) design due to various hardware attacks that can affect manufacturers, system designers or end users. Major issues include hardware Trojan attack, hardware intellectual property (IP) theft, such as an illegal sale or use of firm intellectual property cores or integrated circuits (ICs) and physical attacks. A hybrid model consisting of Arbiter PUF and Butterfly PUF are used to generate random responses which are fed to a Finite State Machine (FSM). A three-level FSM was designed to generate the signature correctly to authenticate IPs. The results were obtained with the help of three Intellectual Property (IP) cores – Zedboard OLED IP, ISCAS'89 s1423 Benchmark IP and a Full Adder IP. A 16-bit arbiter PUF and Butterfly PUF have been implemented on a 28nm FPGA. The average execution time to generate hardware signature for three IP cores was found to be 4.78 seconds (5 iterations) which is considerably low.

Keywords

Physical Unclonable Function (PUF), Intellectual Property (IP), System on Chip (SoC), Finite State Machine (FSM), Zedboard

1 Introduction

A system on a chip (SoC) combines the essential electronic circuits of various computer components onto a single die. An SoC can perform analog, digital or even mixed-signal operations. It mainly consists of a graphical processing unit (GPU), a multi core central processing unit, and a system memory (RAM). SoC FPGA devices integrate both processor and FPGA architectures into a single substrate. Therefore, they provide lower power, smaller board size, higher integration, and higher bandwidth communication between the processor and FPGA logic. They also contain different peripherals, a Field Programmable Gate Array, an on-chip memory, and different types of transceivers. The advantages of using FPGA in a design are considered to be lower non-recurring engineering costs, shorter time-to-market, and higher flexibility. These reasons made FPGA a prevalent design platform for automotive, aerospace and consumer electronics applications.

An IP core is a design block that is used for product development in reconfigurable devices like ASICS or FPGAs. Due to the elements of design reuse, IP cores are essential parts of the growing electronic design automation industry. Ideally, an IP [1] core should be inserted into any vendor technology or design methodology effortlessly. Universal Serial Bus (USB), Phase Locked Loops (PLLs), Digital to Analog converters, and AMBA interfaces are some of the examples of IP cores. It can be categorized into three – hard, firm and soft cores. Hard cores are physical indicators of the IP design. These are mainly used in plug and play applications. Hard cores are less flexible and portable compared to the other two types of cores. Firm cores carry placement data that are configurable to different applications which are like hard cores. Soft cores are made of logic gates with associated interconnections. They are even available as a file written in Hardware description language like VHDL or Verilog.

IP cores are licensed and distributed like software to a system developer. Protection against unlicensed usage is a serious threat to the IP vendors which enables cloning of IP cores [2, 3]. This threat is taken into consideration when designing future embedded systems. By using IP cores, speed of product delivery can be rapidly increased. This will boost the trade with IP cores. So, the IP vendors need to address the issue of security against unlicensed usage of IP cores. It can also be found that the area

Cite this article as: Kokila, J., Das, M. A., Begum, B. S., Ramasubramanian, N. "Hardware Signature Generation Using a Hybrid PUF and FSM Model for an SoC Architecture", Periodica Polytechnica Elecrical Engineering and Computer Science, 63(4), pp. 244–253, 2019. https://doi.org/10.3311/PPee.13424

and power overhead for additional security or reliability functions will decrease with increasing chip area. Hence a secure and reliable IP core is necessary for future system development.

1.1 Hardware Signature

Modern design methodologies are based on reusable modules called IP cores. The modular nature, reduced system complexity and improved development time are some of the advantages of a reuse-based design methodology. Violation of Intellectual Property (IP) rights of the reusable modules is one of the risks in this type of design methodology. A third-party IP core vendor can sell an IP core as their own without even knowing the internal architecture or implementation. This is due to the modular nature of IP cores which enables easy integration with other components. Hence the development of an intellectual property protection (IPP) mechanisms is vital for the evolving reuse-based system design methodology. Several watermarking techniques for the protection of IP cores have been proposed in the previous years. One such protection scheme is to embed a digital signature or a hardware signature in the IP core. This watermarking technique is applied at hardware description language (HDL) level. This signature is preserved throughout the process of synthesis, placement and routing.

In this paper, the hardware signature is generated using a hybrid PUF and FSM model. The generated signature is embedded in the external IP core [4]. During verification, the BPUF key is extracted from the signature and compared with the inbuilt values in the FSM.

1.2 Physical Unclonable Function (PUF)

Physical unclonable functions (PUFs) [5 - 8] are innovative hardware primitives that are used in cryptographic applications like authentication or secret key generation. They are advantageous over existing digital storage mechanisms for several reasons:

- PUF hardware [9] is made of simple digital circuits consisting of gates and flip flops. This consumes less power and area compared to the memory solutions with antitamper facilities.
- PUF is popular due to the absence of additional cryptographic hardware like the secure hash algorithm (SHA) or an encryption algorithm involving public or private key.
- It is very difficult to perform a physical attack to extract the digital data present on the chip when the power is OFF. The chip must be powered on to store the 'secret' on a memory.
- It is hard to perform invasive attacks without modifying the physical characteristics of a PUF.
- It is very difficult to reproduce a Non-Volatile Memory (NVM) based on PUF for secret key storage.

The secret obtained from a PUF is extremely difficult to predict or extract due to the inherent randomness of a PUF. The two primary applications of PUFs are secure key generation and low-cost authentication. PUFs are broadly classified into two groups. These groups are described as strong PUF and weak PUF. Strong PUFs are mostly used for authentication [10] while weak PUFs are used for storing keys. In this project, we are using both weak and strong PUFs. We are using Butterfly PUF and Arbiter PUF in our hybrid model.

Arbiter PUF [11, 12] is a delay based PUF made of Multiplexers and a D Latch. The circuit shown in Fig. 1 takes 16-bit input called 'challenge' and produces a 1-bit 'response'. The amount of delay between two paths is determined using Multiplexers with the help of input control bits. A pair of Multiplexers is controlled by the same input bit I[i] which work as switching box. If the input control bit is zero, the Multiplexers pass through the two delay lines. Else, the top and bottom signals are interchanged. In this way, the circuit can create a pair of delay paths for each input 'I'. The output is evaluated by giving a rising signal to both the paths simultaneously. Due to the delay differences in two paths, the arbiter latch decides which signal is faster. The output is high or low depending on the speed of signal reaching data input (D) of latch.

A Butterfly PUF cell [13] is a cross-coupled circuit made of two D latches as shown in Fig. 2. The output is made to occupy any of the stable states by the circuit operation. It is very difficult to create a cross coupled ring using combinational circuits. This led to the design of combinational ring using latches present in a FPGA matrix. There are two main signals in D latch called Preset (PRE) and Clear (CLR). The

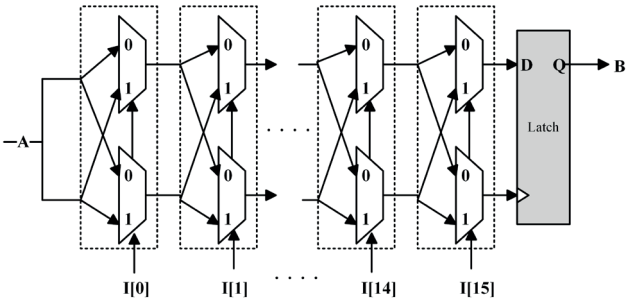

Fig. 1 Arbiter PUF

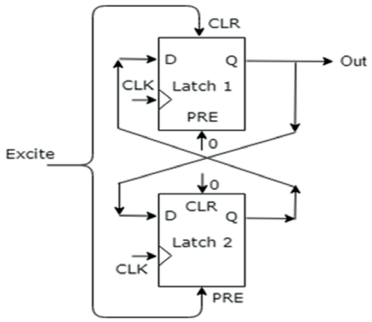

Fig. 2 Butterfly PUF

preset signal turns the output 'Q' to high on a high input. The Clear signal turns the output 'Q' to low on a high input. The output 'B' can be captured when the Clock is high. The data is transferred to 'Q'. Initially, the Preset and Clear signals are set to low. Excite signal is connected to the Clear of Latch 1 and Preset of Latch 2. Clock signal of both Latches is set to high to simulate the operation of a combinational loop. The Excite signal is made high to start the PUF operation. This brings the circuit to a highly unstable point. The Excite signal is made low after a few clock cycles. This makes the PUF circuit to attain either one of the two possible stable states, high or low, on the output 'Out'.

2 Review of Literature

Counterfeiting of IP cores is a growing concern on the global economy and it also questions the security of the critical infrastructure [14]. A very well-known impact of counterfeiting is product cloning. Overproduction of goods is another dangerous aspect which is less known to people. Flexibility of software and performance of hardware are the two critical factors concentrated on reconfigurable computing. These advantages led to the increasing attention from the industry for Reconfigurable computing. In this section we discuss some of the important works related to this article YingjieLao [14] talks about designing a two-level FSM to address the problem of Intellectual Property (IP) protection. He proposes a two level FSM architecture which is capable of authenticating IPs and correct the PUF response bit errors occurring due to environmental disparities. The cost required for this approach is very less compared to the conventional error correcting approaches (BCH codes) that are used previously for PUF based authentication.

A new IP protection mechanism to restrict the execution of IP core only on specific FPGA devices is being demonstrated by Jiliang Zhang [15]. It prevents the IP core from being reproduced. This mechanism enforces a pay-per-device licensing, which enables the system developers to purchase the IPs from the core vendors and the developer needs to pay only for the lease time. The unit price is lower compared to the normal expensive license fees. An Internet of Things architecture which facilitates run-time modifications to the hardware components is described by Anju P Johnson [16]. This Partially Reconfigurable FPGA architecture is inspired from the principles of hardware sharing and hardware mixing. This also allows on-line hardware updates which is used in enterprise IoT infrastructures sharing available resources. They evaluated the effectiveness of different threats that can originate from the IoT nodes which use the dynamic partial reconfiguration. They also proposed a possible solution based on physical unclonable function (PUF) circuits to prevent such intimidations.

To protect and validate the IP in the design there are some existing techniques such as watermarking and fingerprinting. A recent method to embed watermarking [17] in soft IP core was proposed for embedded systems and it is a sequential aware one. The simulation results have been analyzed for Xilinx Virtex-II Pro FPGA board. The main limitation of this method is sequence length. Embed the watermarks into a FPGA design at net list level by manipulating LUT. The watermark is embedded into LUTs at the net list level and fingerprint is inserted in the bit steam level to protect the IP reuse [18]. Mostly the IP protection techniques are started using PUF-FSM structure, because of its advantages and security issues as mentioned in the related works. The IP protection based mechanism started using FSM based PUF for authenticating a device and IPs with database. But in our work we have implemented a hardware signature which involves combination of two PUFs and FSM to protect specific IPs with less area, power utilization and communication overhead and no database.

3 Hardware Software Co-Design

Hardware/software (HW/SW) co-design is the concurrent development of both hardware and software sides of the system. This type of design methodology helps in embedding together the modules on hardware and software to develop an optimized solution. This enables a system programmer to design hardware and software modules with ease.

The HW/SW co-design needs segregating the specification into hardware and software. One part is implementation on hardware i.e., using hardware description language in FPGA and other part run on software such as SDK using C. For these partitions the performance evaluation is difficult. In order to achieve the above objective,

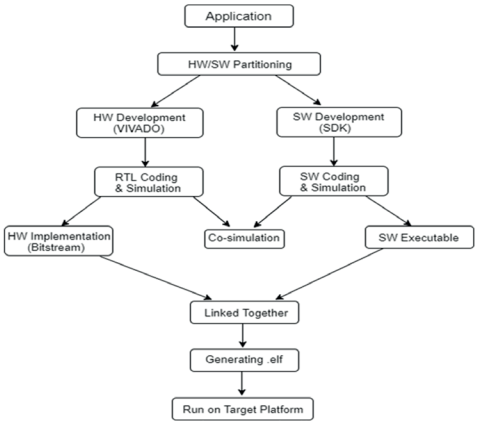

Fig. 3 HW/SW co-design overview of the design flow for ZynqSoC.

the hardware modules are mapped to compute-intensives parts of the application.

Fig. 3 shows an overview of HW/SW co-design that was used in this article. The specification of the system in terms of performance, functionality, cost and power is the initial step towards hardware-software co-design. Partitioning the application follows the initial step which is a very decision. The process of splitting the functions into hardware and software parts is termed as HW/SW partitioning. The profiling tools are used to analysis the heterogeneous application and generates the information which helps the HW/SW to take decision. After the decision of hardware, software and the interface blocks, the style of coding and simulation is being finalized. In this step, the software and hardware specification are separated independently in the implementation process to optimize the overall specification. This is succeeded by an important step called co-simulation. Validation of the system simulating the hardware and software development is undergone in this step. The co-simulation step provides the output, which is used to verify the achievement of design goals. The co-design flow will stop if the acceptance stage is reached. Else the design is not acceptable, which means some specification or design error has occurred. At this stage the Co- design will go to the previous steps where hardware and software have to be redesigned until an output that sounds a good design is attained. In the next step simulation-level implementation is achieved in hardware and software part, for which the results are obtained from the co-simulation step.

After the partitioning has been completed, the executable software and all the bit-stream generated files are linked together to create ”.elf” file that can be made to run on the target platform. For the experimental purpose, the Zedboard [19, 20] heterogeneous platform has been considered in this work. Zedboard is a Zynq evaluation development board for the designer interested in developing/testing designs. To enable a wide range of heterogeneous applications, the Zedboard includes all the necessary interfaces, communications and support functions. The most important part on this board is Xilinx Zynq 7000 All Programmable SoC (ZynqSoC os zynq board). ZynqSoC performs all computational resources for the design system. Some configuration has made computational resources, which is performed by ARM-based processing system (PS) and programmable logic (PL) based on Artix-7.

4 Proposed Design

The Flowchart described in Fig. 4 shows the different phases of our proposed hybrid model. Our hybrid model consists of two PUFs- Arbiter (APUF) and Butterfly (BPUF). The responses generated from two PUFs are given as input to the FSM. BPUF does not have an external challenge as input. ‘Excite’ signal shown in BPUF is connected to the Enable pin of D-Latch present in the BPUF cell. This signal is used to trigger the output of BPUF. It is activated only once. Whereas for APUF, there exists a challenge Response Pair (CRP). It means for every challenge there exist a unique response.

In the next step, hamming distance is found between APUF challenge and Response pair (CRP). Hamming distance (HD) along with the BPUF key is given as input to the Finite State Machine (FSM). Based on the analysis of Butterfly PUF, certain key values are fixed in FSM. Details of the FSM are described in detail in the next section. When the corresponding match occurs with the values in the FSM, a hardware signature is generated.

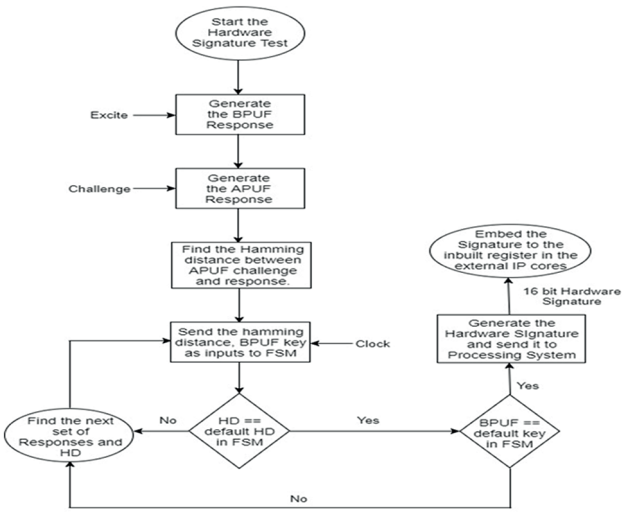

Fig. 4 Flow Chart

The HD is calculated for CRP of APUF, where the strong PUF will generate a large set of CRPs. These CRPs have been tested by IP owners and the most frequent occurring HD have been selected for specific BPUFs KEY. These BPUF KEY is unique and used to generate key for specific devices by the device vendors. The HD is acting as a hash value for IP cores and KEY is used to authenticate the device. The combination of HD, Key and logical operation such as XOR and SHIFT will generate the hardware signature. This hardware signature is configured at the time of IP execution, which needs to be verified at that time of IP integration. This signature is attached to the configuration register inside the .elf file of the requesting IP core. If this signature is not authenticated in the initial stage of the IP booting step then it is not a valid or authentic IP.

4.1 Design of Finite State Machine

A three level Finite State Machine (3-l FSM) has been proposed in this work. It is a three level design because the hardware signature is generated only after three stages. The signature is generated only if the hamming distance and BPUF key matches with the corresponding values in the FSM. Hamming distance (HD) is evaluated for 16- bit CRP of APUF which ranges from 0-15. To store this range only 4-bit is needed. The 4-bit HD is used in the first level of the FSM. A key value has been attached to corresponding HD which is checked for the second stage.

The signature generation FSM has been given in the Fig. 5. This can be made even complex and it is up to the designer. An important thing that is to be noted is these values in FSM is subject to change for other SoC board. Hence the analysis has to be performed again to find and fix the values in FSM.

5 Implementation

Xilinx Zedboard, All Programmable SoC, were used to do this experiment [16]. Three IP Cores-OLED, s1423 Benchmark IP and a Full Adder IP - were used in this design. The IP cores were selected using on board DIP switches and the status was displayed using the on board LEDs. The 32x128 pixels of OLED display panel is inbuilt on the Zedboard and is organized by the controller. The OLED controller will initialize the display panel based on the producer's guidelines and specifications. The initial configuration is archived by sending gusts of commands as bytes which are separated by measured time intervals through a Serial Peripheral Interface (SPI). Next step is to access the processing system (PS) in order to increase the performance. The OLED controller offers a PS access to the OLED display buffer through memory-mapped registers.

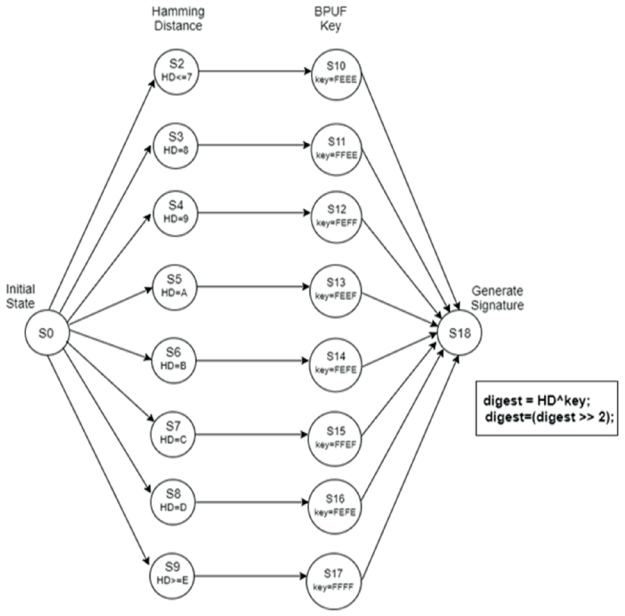

Fig . 5 FSM of the proposed Model

There are seventeen registers of 32-bits each which can be accessed through soft wares. These software access register together form a slave AXI-peripheral of the controller. In which 1 - 16 are data registers, while the 17th one is used for control. Any AXI-crossbar compliant processor system is connected to the OLED controller through AXI interface. The Zedboard OLED communicates with the display panel through SPI.

The benchmark circuits were used to evaluate the performance of the algorithms used in the areas of fault simulation, testability analysis, formal verification, logic synthesis, technology mapping, and layout synthesis. The benchmark circuits disseminated to ISCAS'89 are all described as gate level netlist. The benchmark IP distributed in ISCAS'89, s1423, is one of the IP cores used in my design. It consists of 74 D-flip flops, 167 inverters, and 490 gates. The gates are made of 197 ANDs, 64 NANDs, 137 ORs, and 92 NORs. It takes 17-bit input and produces a 5-bit output.

Full Adder is the normal 3-bit adder producing sum and carries as output. There are three inputs and two outputs for a simple full adder. A and B are the first two inputs and C-IN is the third input carry. The C-OUT is the output carry and usual output is SUM. The output carry is designated as C-OUT and the normal output is SUM. A cascade of adders can be generated using a full adder, which adds 8, 16, 32 etc. bit binary numbers

6 Results and Discussion

Fig. 6 shows the diagram of the implemented design in Zedboard. The entire logic is implemented with the help of Look Up Tables (LUTs), Flip-Flops (FFs), Block RAM (BRAM) and BUFGs.

6.1 Utilization Details

Table 1 describes the resource utilization details of the design. The column 'Available' in Table 1 tells the amount of resources available in Artix-7 based Programmable Logic (PL) present in zedboard. The logic is implemented as LUTs, LUTRAMs, Flip Flops, BRAMs, IOs and BUFGs.

Table 1 Resource utilization details

Resource	Utilization	Available	Utilization Percentage
LUT	3586	53200	6.74
LUTRAM	68	17400	0.39
FLIP FLOP	2988	106400	2.81
BLOCK RAM	0.5	140	0.36
IO	54	200	27
BUFG	1	32	3.13

6.2 Power Details

Static power of the device is the power consumed due to transistor leakage on all connected voltage rails and the circuits required for the stable operation of FPGA during the post-configuration phase. This can be obtained by programming a blank bit-stream into the FPGA device. Design power is the power consumed by the user design mainly due to the input data and the internal activities of the circuit components. It depends on voltage levels and design logic. This power is varying for each clock cycle. It is also dependent on the routing resources used.

The main part of the design power is contributed for clock managers, the static current consumed by I/O pads and the circuits which consume power when required. Power taken by off-chip devices is not taken into consideration while calculating design power.

Thermal power or total on-chip power is the power consumed by the internal components of the FPGA. This is the sum of design power and static power consumed by the device. The On-Chip Power graph, as shown in Fig. 7, tells the power dissipated in different resources. It can be seen that the Processing System (PS7) contributes most to the total power (90 %). It can be found that the PL dynamic power is 16 mW and the device static power is 163 mW.

The area and power will vary based on the type of IPs and the embedded signature. The overhead incurred in the proposed model in terms of area and power is less compared to the model without HS. The measured values are reported in the Fig. 8 with 3 different benchmark IPs.

6.3 PUF Analysis

The security metrics such as uniqueness, reliability, and randomness as referred in [21] are measured for a single 32-bit PUF and hybrid PUFs which are described below and compared in Table 2. The security metrics of PUFs with the Eq. (1)-(3) and description are explained.

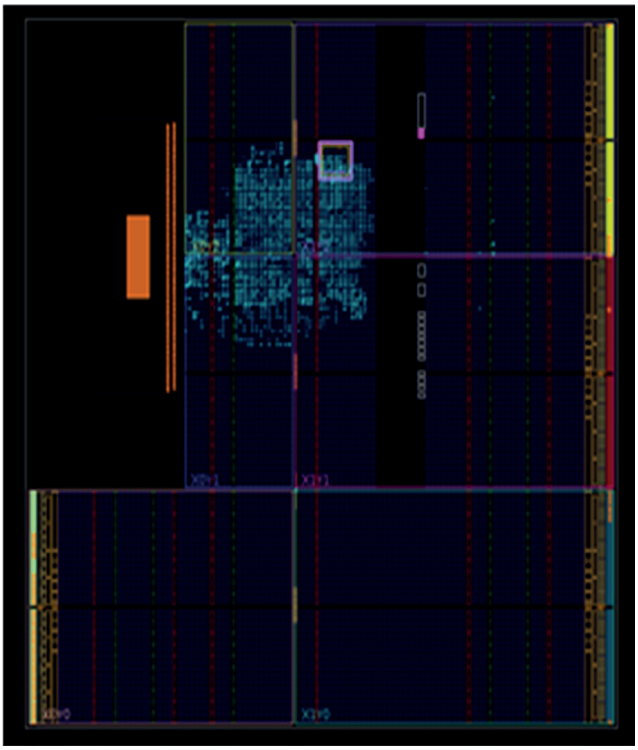

Fig. 6 Implementation Design in Zedboard

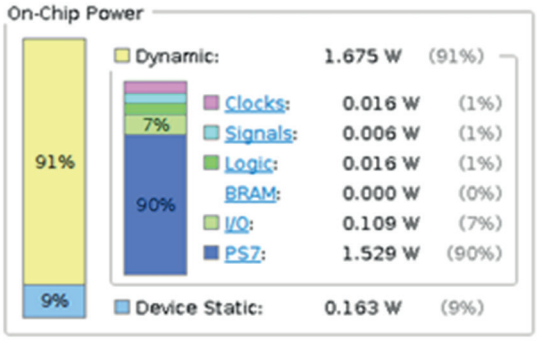

Fig. 7 On-Chip Power

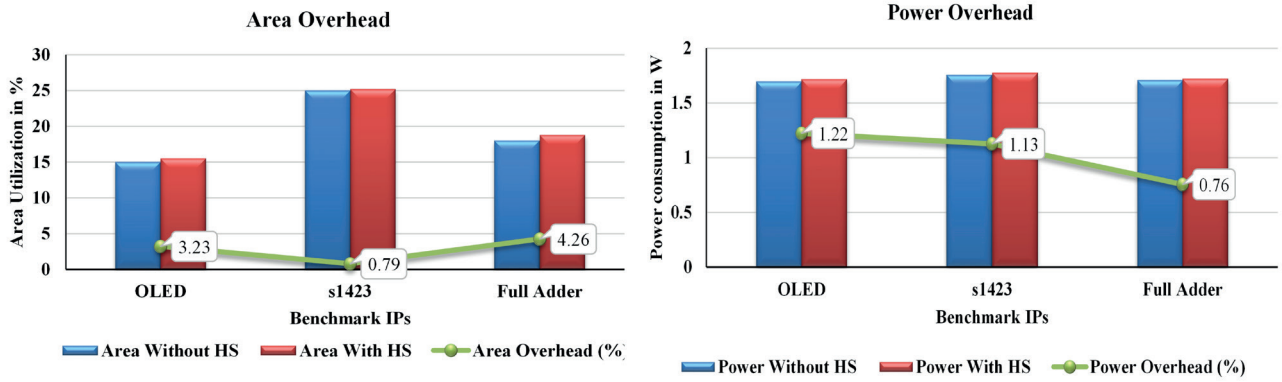

Fig. 8 Area and Power Overhead with and without HS

Uniqueness (Un)

$$U_{i,j,k} = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{HD(or_i, or_j)}{n} \times 100\ \%\ . \quad (1)$$

The average inter-chip variation between chip i and j is measured among output responses or_i and or_j for *n*-bits and k chip.

Reliability (Re)

$$\mathrm{Re}_{i,t} = 100\ \% - \frac{1}{m} \sum_{t=1}^{m} \frac{HD(or_i, or'_{i,t})}{n} \times 100\ \%\ . \quad (2)$$

The intra-chip variation based on time duration t is measured among output responses or_i for n-bit. The temperature is maintained to be 20 to 46 °C in the zynq board.

Randomness (Ra)

$$Ra = \frac{1}{L} \sum_{i=1}^{L} or_{i,j} \times 100\ \%\ . \quad (3)$$

The uncertainty in each bit is measured for L iterations, where $or_{i,j}$ is the j−th binary bit of an n−bit output response from a chip i.

While comparing with the individual PUFs, the hybrid PUF with FSM is better in terms of area, power, uniqueness, and reliability. The proposed model is implemented in SoC FPGA, which is 28 nm with heterogeneous PUF hence, its power and area vary.

6.3.1 BPUF Response Analysis

The hardware generation module is executed for 10,000 iterations, and a specific sample of 500 iterations has been selected for 4 different zynq boards, and its HD and KEY values are fixed, based on Fig. 9. The most frequently occurring HD and golden response of BPUF as KEY values have to be chosen in such a way that it is used only once and should be unpredictable. Fig. 9 includes the call out or text for specific values which helps us to select the HD and golden response KEY with the most frequency. The KEY and HD are fixed to generate a unique hardware signature for specific board and IP modules respectively. In Fig. 9 each board will have three specific KEYs and HD, to generate hardware signature for a specific device and its selected IP modules.

Table 2 PUF based metrics comparison

Metrics	Power (W)	Area (%)	Un (%)	Re (%)	Ra (%)
A-PUFs	0.959	56.21	45.67	98.4	49.6
B-PUFs	0.098	55.39	36.23	91.86	30.45
Hybrid AB-PUFs and FSM	1.675	22.66	46.23	95.06	42.49

It is based on these analyses the values are fixed in the Finite State Machine (FSM). Each key uniquely identifies IP cores for which the hardware signature has to be generated. In our design, a maximum of 8 IP cores can be incorporated. The number of IP cores that can be increased by changing the number of bits of Butterfly PUF from 8 bits to 32 bits or 64 bits.

The tabular column is shown below, Table 3, shows the KEYs, HDs and time required to generate the hardware signature for the three IP cores, which is tested in 4 Boards. It is to be noted that for the two iterations IPs are selected at random among the three. For board 3 the HD 9 is mapping to two unique BPUF KEYs, which indicates that the group of IPs can be authenticated using this method.

The time measurements are made in seconds. As described in the previous section, the signature generated is a 16-bit value. This is formed by the logical operation such as XORing the hamming distance and BPUF key

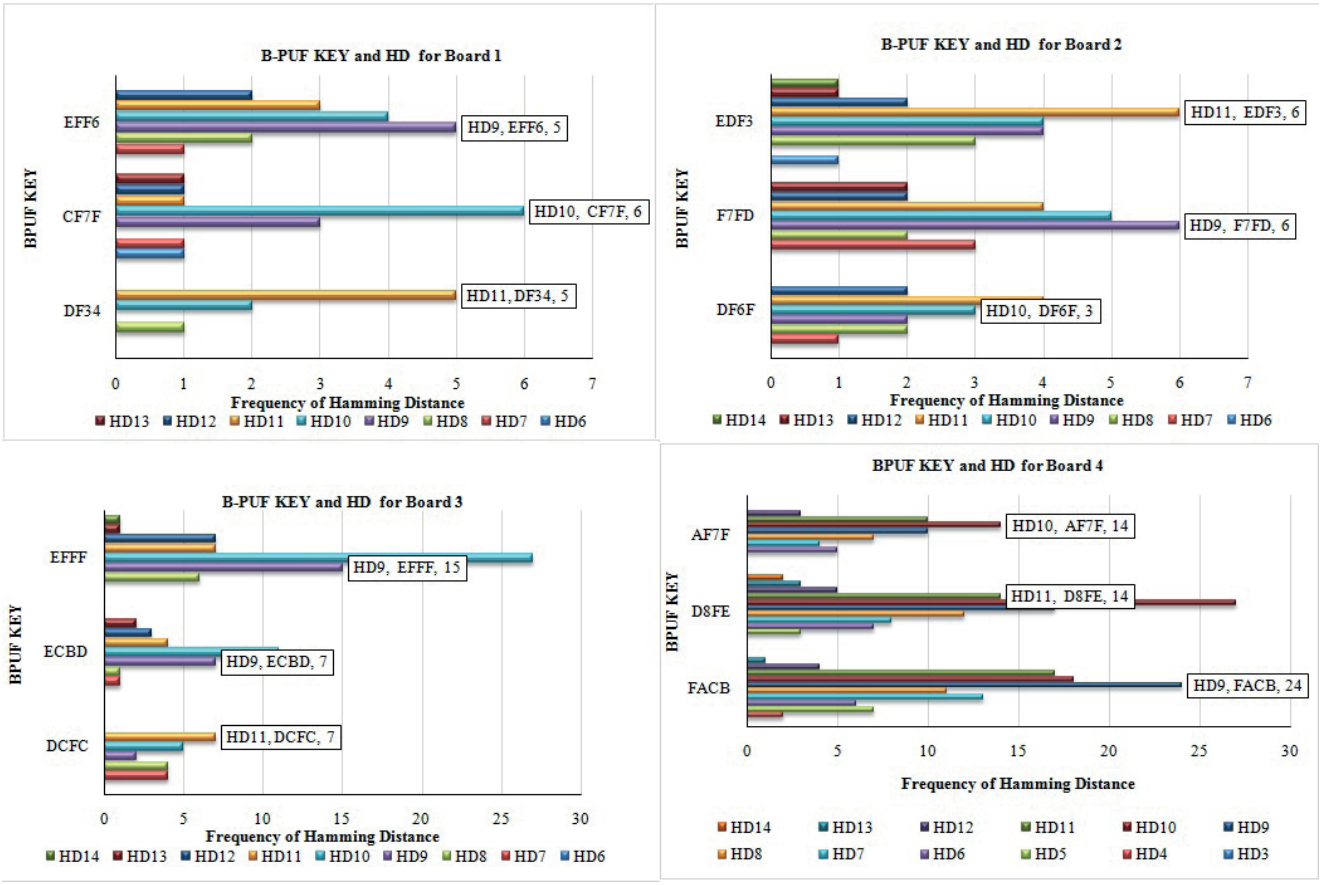

Fig. 9 BPUF Key and HD mapping for 4 independent zynq Boards

Table 3 Key, Hardware signature (HS) and Execution Time for 4 Zynq Boards

Board No:	BPUF KEY	HD	HS	Execution time (s)
1	DF34	11	37c9	2.52
	CF7F	10	19ed	5.18
	EFF6	9	077f	1.69
2	DF6F	11	37df	9.79
	F7FD	10	1efd	0.99
	EDF3	9	079f	7.56
3	DCFC	11	373b	8.53
	ECBD	9	1d95	2.11
	EFFF	9	07ef	3.36
4	FACB	11	3eb2	1.54
	D8FE	10	1b1d	1.68
	AF7F	9	057b	13.98

followed by right shifting n- bits. The logical operation and n-bit shift may vary based on the designer. A specific verification technique is needed to extract the signature from the bit file and to authenticate an IP module.

7 Conclusion and Future work

A hybrid model has been proposed, which consists of Physical Unclonable Functions (PUFs) and Finite State Machine (FSM) for an SoC architecture. It mainly addresses the issue of security in IP cores. The entire project is designed from the perspective of the IP vendor. The role of an IP vendor, like Synopsys, is to create and package design to an Intellectual Property (IP) core which is then given to a client through the network. The client uses this IP core in his design. So the vendor can incorporate this lightweight model in their design for overcoming counterfeiting of IPs. One important advantage of our design is that it can be used in a heterogeneous environment. Our model is implemented and verified in Xilinx Zedboard. Our model uses Butterfly and Arbiter PUFs to generate random responses. BPUF response is termed as the key which uniquely identifies IP cores. Hamming distance between APUF challenge and response along with BPUF key is given as input to the FSM. The output of it is a flag variable which tells whether the signature is generated or not. If it is generated the signature is returned.

The number of IP cores that can be attached to our design is limited to 8. This can be further improved by increasing the number of bits of PUFs to 64 or 128. The future works can be concentrated on implementing the same design in a networked environment. The developer can add Operating System and Networking functions to the design. As mentioned in the previous chapters BPUF is a weak PUF. So, this model can be improved by changing this to modern PUFs like Feed Forward Arbiter PUF or XOR-Arbiter PUF. This will enable a large number of IP cores to be incorporated in the design. An important feature provided by Xilinx called 'Partial Reconfiguration' can be used to reduce the area occupied and power consumed. Partial Reconfiguration enables the device to dynamically modify the logic blocks by choosing the partial bit files at run time.

Acknowledgement

The project presented in this article is supported by Visvesvaraya Ph.D. scheme (PhD-MLA/4(16)/ 2014, Dated 21.01.2014). The authors would like to acknowledge the infrastructure support provided by the Reconfigurable Intelligent System Engineering (RISE) Lab, Dept. of Computer Science and Engineering, NIT, Trichy.

References

[1] Guajardo, J., Kumar, S. S., Schrijen, G-J., Tuyls, P. "FPGA intrinsic PUFs and their use for IP protection", In: Pallier, P., Verbauwhede, I. (eds.) Cryptographic Hardware Embedded Systems, Springer, Berlin, Heidelberg, Germany, 2007, pp. 63–80. https://doi.org/10.1007/978-3-540-74735-2_5

[2] Guajardo, J., Kumar, S. S., Schrijen, G-J., Tuyls, P. "Physical unclonable functions and public-key crypto for FPGA IP protection", In: International Conference on Field Programmable Logic and Applications, Amsterdam, Netherlands, 2007, pp. 189–195. https://doi.org/10.1109/FPL.2007.4380646

[3] Trimberger, S. M., Moore, J. J. "FPGA Security: Motivations, Features, and Applications", Proceedings of the IEEE, 102(8), pp. 1248–1265, 2014. https://doi.org/10.1109/JPROC.2014.2331672

[4] Zhang, J., Lin. Y., Lyu, Y., Qu, G., Cheung, R. C. C., Che, W., Zhou, Q., Bian, J. "FPGA IP protection by binding finite state machine to physical unclonable function", In: 23rd International Conference on Field Programmable Logic and Applications, Porto, Portugal, 2013, pp. 1–4. https://doi.org/10.1109/FPL.2013.6645555

[5] Herder, C., Yu, M-D., Koushanfar, F., Devadas, S. "Physical Unclonable Functions and Applications: A Tutorial", Proceedings of the IEEE, 102(8), pp. 1126–1141, 2014. https://doi.org/10.1109/JPROC.2014.2320516

[6] Tuyls, P., Škorić, B. "Strong authentication with PUFs", In: Petković, M., Jonker, W. (eds.) Security, Privacy and Trust in Modern Data Management, Data-Centric Systems and Applications, Springer, Berlin, Heidelberg, Germany, 2007, pp. 133–148. https://doi.org/10.1007/978-3-540-69861-6_10

[7] Paral, Z., Devadas, S. "Reliable and efficient PUF-based key generation using pattern matching", In: IEEE International Symposium on Hardware-Oriented Security Trust San Diego, CA, USA, 2011, pp. 128–133. https://doi.org/10.1109/HST.2011.5955010

[8] Devadas, S., Yu, M. D. "Secure and robust error correction for physical unclonable functions", IEEE Design and Test of Computers, 27(1), pp. 48–65, 2010. https://doi.org/10.1109/MDT.2010.25

[9] Maes, R., Verbauwhede, I. "Physically unclonable functions: A study on the state of the art and future research directions", In: Sadeghi, A. R., Naccache, D. (eds.) Towards Hardware-Intrinsic Security, Information Security and Cryptography, Springer, Berlin, Heidelberg, Germany, 2010, pp. 3–37. https://doi.org/10.1007/978-3-642-14452-3_1

[10] Dailey, M. D. "Authentication schemes based on physically unclonable functions", PhD dissertation, Worcester Polytechnic Institute, 2009. [online] Available at: https://users.wpi.edu/~martin/MQP/daileyshomorony.pdf [Accessed: 20 Feburuary 2018]

[11] Suh, G. E., Devadas, S. "Physical unclonable functions for device authentication and secret key generation", In: 44th Annual Design Automation Conference, San Diego, CA, USA, 2007, pp. 9–14. https://doi.org/10.1145/1278480.1278484

[12] Majzoobi, M., Koushanfar, K., Devadas, S. "FPGA PUF using programmable delay lines", In: IEEE International Workshop on Information Forensics and Security Seattle, WA, USA, 2010, pp. 1–6. https://doi.org/10.1109/WIFS.2010.5711471

[13] Kumar, S. S., Guajardo, J., Maes, R., Schrijen, G-J., Tuyls, P. "Extended abstract: The butterfly PUF protecting IP on every FPGA", In: IEEE International Workshop on Hardware-Oriented Security and Trust, Anaheim, CA, USA, 2008. https://doi.org/10.1109/HST.2008.4559053

[14] Lao, Y., Yuan, B., Kim, C. H., Parhi, K. K. "Reliable PUF-based local authentication with Self Correction", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 36(2), pp. 201–2013, 2017. https://doi.org/10.1109/TCAD.2016.2569581

[15] Zhang, J., Lin, Y., Qu, G. "A PUF-FSM Binding scheme for FPGA IP Protection and Pay-per device Licensing", IEEE Transactions on Information Forensics and Security, 10(6), pp. 1137–1150, 2015. https://doi.org/10.1109/TIFS.2015.2400413

[16] Johnson, A. P., Chakraborty, R. S., Mukhopadhyay, D. "A PUF Enabled Secure Architecture for FPGA Based IoTApplications", IEEE Transactions on Multi-Scale Computing Systems, 1(2), pp. 110–122, 2015. https://doi.org/10.1109/TMSCS.2015.2494014

[17] Nie, T., Li, Y., Zhou, L., Toyonaga, M. "A multilevel fingerprinting method for FPGA IP protection" In: 2013 IEEE International Symposium on Circuits and Systems (ISCAS2013), Beijing, China, 2013, pp. 1789–1792.
https://doi.org/10.1109/ISCAS.2013.6572212

[18] Kufel, J., Wilson, P. R., Hill, S., Al-Hashimi, B. M., Whatmough, P. N. "Sequence-aware watermark design for soft IP embedded processors", IEEE Transactions on Very Large Scale Integration Systems, 24(1),2016, pp. 276–289.
https://doi.org/10.1109/TVLSI.2015.2399457

[19] Peterson, E. "Leveraging Asymmetric Authentication to Enhance Security-Critical Applications Using Zynq-7000 All Programmable SoCs", White Paper, Xilinx, 2015. [online] Available at: https://www.xilinx.com/support/documentation/white_papers/wp468_asym-auth-zynq-7000.pdf [Accessed: 24 February 2018]

[20] "Zynq - 7000 All Programmable SoC", Technical Reference Manual, Xilinx, 2018. [online] Available at: https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf [Accessed: 22 November 2018]

[21] Cherkaoui, A., Bossuet, L., Marchand, C. "Design, evaluation, and optimization of physical unclonable functions based on transient effect ring oscillators", IEEE Transactions on Information Forensics and Security, 11(6), 2016, pp. 1291–1305.
https://doi.org/10.1109/TIFS.2016.2524666