Analysis and Design of Photon Protocol Architecture for High Speed Applications

R. Gayathri[1] M. Nisha Angeline[2] Dr. S. Valarmathi[3]
[1]PG Scholar [2]Assistant Professor (Sr.Gr.) [3]Professor and Head
[1,2,3]Department of Electronics Communication Engineering
[1,2,3]Velalar College Of Engineering and Technology, Erode, Tamilnadu, India

Abstract— In recent years, communication plays a vital role in the world, whereas the secured communication lags in some applications such as military and message authentications. PHOTON light weight hash function is an attractive modeling tool to improve the security and it is under the SHA-3 algorithm. Hash function takes a string of length as input and produces a fixed length value. The PHOTON hash function family can be built with VLSI technologies. The existing PHOTON protocol architecture is modified for better security and high speed applications. The modified high speed architecture can be obtained by performing the pipelining and parallel processing techniques. The proposed architecture will provide high speed and secured communication.

Key words: Lightweight, Hash Function, Sponge Function, AES

I. INTRODUCTION

Security is the big issue for all networks in today's environment. Many methods have been developed to secure the network infrastructure and communication over the Internet. Many new security protocols were proposed for RFID applications that includes hash functions. The lightweight hash function is the current method for secured communication. It is feasibly impossible for two different messages to be given to the same string of bits.

Hashing differs from encryption because the resulting hash information is normally smaller than the original data, whereas the hashed document is a similar size. Encryption and hashing are similar in the way that they both take a string of useful text and convert it into something very different.

Hash functions are functions which take a variable input message and compute a fixed-length message digest for each such message. This digest serves as a digital fingerprint allowing a receiver to check that the original message has not been altered during transmission. They can be combined with a secret key to produce a message authentication code.

Hash algorithms are used widely for cryptographic applications that ensure the authenticity of digital documents, such as digital signatures and message authentication codes. These algorithm take a large document file and generate a short " message digest" (output message) as a sort of digital message of the given document. Any change in the original message, must cause a change in the message digest, it must be infeasible for a forger to create a different file with the same digest. Hash function with output 160-bits needed to provide RFID tag security.

PHOTON is a secure light weight hash algorithm developed by Jian Guo, Thomas Peyrin and Axel Poschmann. It is one the light weight hash function, available in different flavors and uses a sponge framework[8] as an extension algorithm in the existing system. It produces 128 bit message digest and suitable for passive RFID tags. PHOTON is very simple to analyze and drastically lowers the area requirement. The sponge functions framework is preferred in order to reduce the memory registers in hardware. The sponge framework is extended, to provide a trade-offs in hardware.

The serialized architecture of PHOTON is modified as parallelized architecture with pipelining and parallel processing techniques. These techniques are applied to LFSR. LFSRs are simple to construct and are useful for a wide variety of applications and the drawback of LFSR is that the input should be serial.

II. EXISTING SYSTEM

PHOTON is one of the light weight hash function under SHA-3(Secure Hash Algorithm). The internal permutations of PHOTON can be seen as AES-like primitives.

A. An AES-Like Internal Permutation:

An AES-like function has a fixed internal permutation P, is applied to an internal state of d2 elements of s bits each and it can be represented as a (dxd) matrix. The permutation P consist of Nr rounds.

Each round has 4 layers: Add Constants(AC), Sub Cells(SC), Shift Rows(ShR), Mix Column Serial(MCS). The internal permutation of one round, is shown in figure.1.

Shift Rows. Since the internal state is a square of cells, it is simple to perform shift rows operation. It rotates the position of the cells in each of row. Therefore row i can be rotated by i positions left.

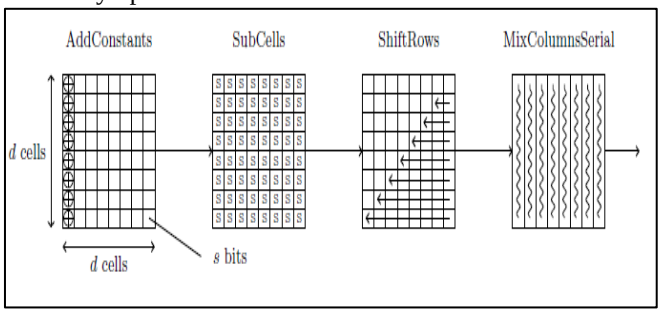

Fig. 1: PHOTON Permutation

Add Constants. It simply means that adding fixed values to the cells of the internal state. The constant has been choosen such that each of the Nr round computation are different and the classical symmetry between columns in AES-like designs are destroyed. The round constants can be generated by a combination of compact Linear feedback shift registers. Only the first column of the internal states are taken for computation.

Sub Cells. It applies an s-bit Sbox to each of the cells. The size of the Sbox should not be odd, because this

leads to odd message block size or capacity when d is also odd. The 4 bit Sboxes can be very compact in hardware while the acceptable upper limit on the cell size is s=8. If s=4 or 8 the software computation is faster and simpler.

Mix Columns Serial. Mix Columns Serial, linearly mixes all the columns. The matrix under the AES Mix Columns Function is a circulant matrix with low hamming weight co-efficients. The byte serial implementation of the function is not compact. AES Mix Columns matrix is the composition of d operations each updates a single byte at a time in a serial way, the coefficients of these d matrices will be very bad for small area implementations.

Moreover, AES-like permutations allows to derive very simple proofs on the number of active Sboxes over four rounds of the primitive. If, the matrix is under the Mix Columns Serial layer is Maximum Distance Separable(MDS), the smallest AES hardware implementation requires 2400 GE[10]. It is possible to implement Mix Columns of AES in a byte-by-byte fashion which requires only 81 GE to calculate one byte of the output column.

III. PROPOSED SYSTEM

The PHOTON family of each variant is defined by its hash output size 64<=n<=256, its input and its output bitrate is r and r'. The PHOTON hash function is represented as n/r/r'. The internal state size (t=c+r) is depends on the hash output size and can take only 5 distinct values: 100,144,196,256 and 288 bits. The five different families of PHOTON uses the internal permutations as P_{100}, P_{144}, P_{196}, P_{256}, P_{288} respectively.

A. Internal Permutations:

The internal permutation Pt, is defined as t €{100,144,196,256,288}.The internal state of the Nr-round permutation is viewed as the circular (dxd) matrix of s-bit cells and corresponding values given in the following table.,

	t	d	s	N_r	$IC_d(\cdot)$	irr. polynomial	Z_t coefficients
P_{100}	100	5	4	12	[0,1,3,6,4]	x^4+x+1	(1,2,9,9,2)
P_{144}	144	6	4	12	[0,1,3,7,6,4]	x^4+x+1	(1,2,8,5,8,2)
P_{196}	196	7	4	12	[0,1,2,5,3,6,4]	x^4+x+1	(1,4,6,1,1,6,4)
P_{256}	256	8	4	12	[0,1,3,7,15,14,12,8]	x^4+x+1	(2,4,2,11,2,8,5,6)
P_{288}	288	6	8	12	[0,1,3,7,6,4]	$x^8+x^4+x^3+x+1$	(2,3,1,2,1,4)

Table 1: The Internal Permutations of PHOTON P_t

B. Hardware Architecture:

The serialized hardware architecture of PHOTON can be implemented in VHDL and simulated using Mentor Graphics ModelsimXE6.4b.The serialized architecture performs one cell per clock cycle and area requirement is smaller. The architecture consist of six modules; MCS, State, IO,AC,SC and Controller.

IO allows to initialize with all "0" vector, absorb message chunks, forward the output of the State module to the AC module. Here two NAND and one XOR gate are used to forward the output of state to the AC module.

MCS calculates the A_t in one clock cycle. The result stored in the State module, that is in the last row of column 0, has been shifted upwards at the same time. After d clock cycles the Mix Column Serial operation is applied to an entire column. The array is rotated by one position to the left and the next column is processed. The d.(d+1) clock cycles are required to perform MCS.

State comprises d.d array of flipflop cells storing s bits each. Every row constitutes a shift register uses the output of the last stage, i.e. column 0, as the input to the first stage of the same row and the next row.

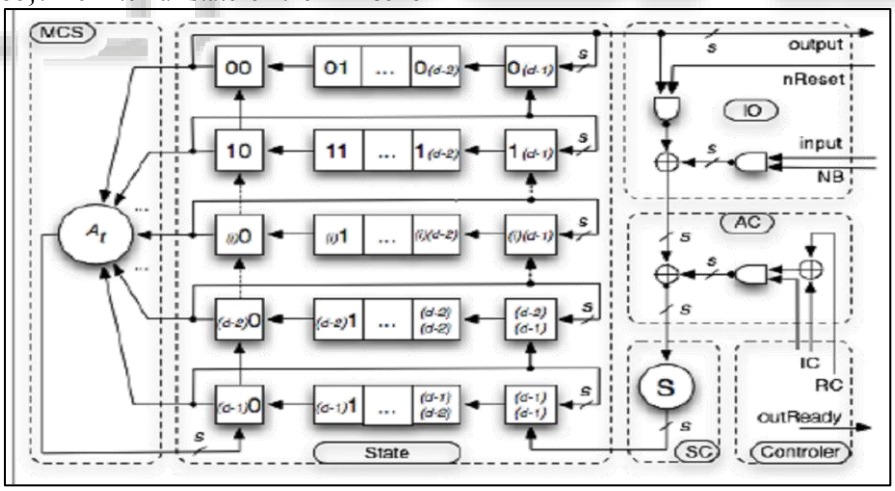

Fig. 2: Hardware Architecture of PHOTON

AC performs the Add Constant operation by XORing the sum of the round constant RC with the current internal constant IC. AC is only applied to the first column, the input to the XNOR gate is gated with a NAND gate.

SC performs the Sub Cells operation and consist of a single instantiation of the corresponding Sbox. For s = 4 we used an optimized Boolean representation.

Controller uses a Finite State Machine(FSM) to generate all control signals. The round constants and the internal constants are generated within this module. The FSM consists of one idle state , one state foe the combined execution of AC and SC, d-1 states for ShR and two states for MCS.

IV. RESULTS ANALYSIS

The message bits is to be padded if it is necessary for the operation. The above architecture is implemented using Xilinx-FPGA and the simulated results are given:

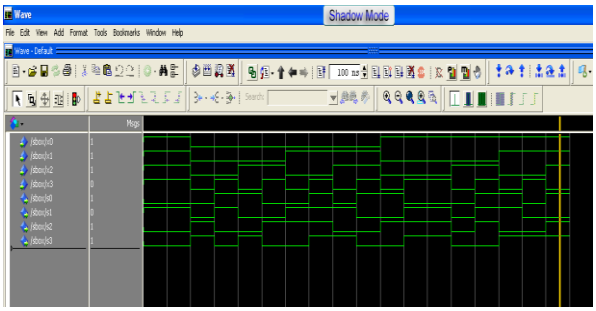

Fig. 3: Output of S-Box

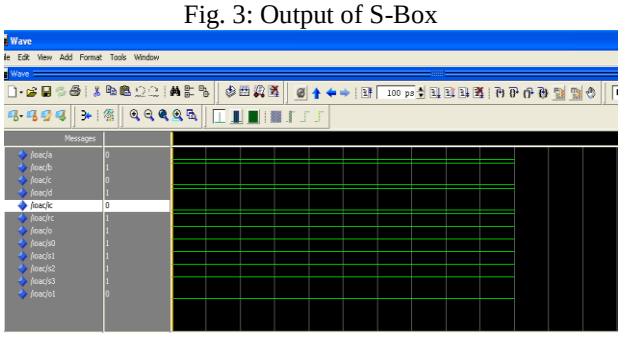

Fig. 4: Output of IO And AC Module

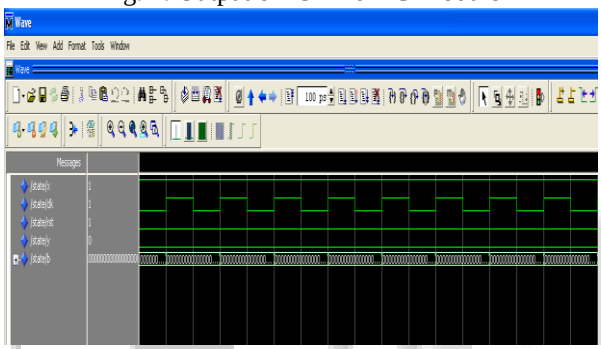

Fig. 5: Output of State Module

V. CONCLUSION

Here, PHOTON the light weight hash function is implemented with serialized architecture and it may also be implemented using pipelining and parallel processing. Pipelining is a technique in which multiple processing are considered, where the output of one unit is the input to next one. The main advantage of the technique is that it takes quicker time of execution and increases the performance. Parallel processing is also as parallel computing, where the number of bits given is increased by reducing the width of clock pulse.

REFERENCES

[1] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, Maria Naya-Plasencia."Quark: a lightweight hash". In Mangard and Standaert, pages 1-15. Springer, 2012.
[2] Jian Guo, Thomas Peyrin, Axel Poschmann. "The PHOTON Family of Lightweight Hash Functions". In Rogaway, P.(ed.) CRYPTO. LNCS, vol.6841,pp.222-239. Springer, 2011.
[3] Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, Ingrid Verbauwhede. "SPONGENT: The Design Space of Lightweight Cryptographic Hashing". Vol.62, no.10. IEEE, 2013.
[4] Guido Bertoni, Joan Daemen, Michael Peeters, Gilles Van Assche. "On the indifferentiability of the Sponge construction". In Nigel P. Smart, editor, EUROCRYPT, vol.4965 of LNCS, pp.181-197. Springer, 2008.
[5] Christophe De Canniere, Orr Dunkelman, Miroslav Knezevic. "KATAN and KTANTAN- a family of small and efficient hardware-oriented block ciphers". In Clavier and Gag [30], pp.272-288.
[6] Martin Hell, Thomas Johansson, Alexander Maximov, Willi Meier. "A stream cipher proposal: Grain-128". In IEEE International Symposium on Information Theory, 2006.
[7] Martin Hell, Thomas Johansson, Willi Meier. " Grain: a stream cipher for constrained environment". IJWMC, 2(1): 86-93, 2007.
[8] Guido Bertoni, Joan Daemen, Michael Peeters, Gilles Van Assche. "Sponge functions". Ecrypt Hash Workshop 2007, May 2007.
[9] Guido Bertoni, Joan Daemen, Michael Peeters, Gilles Van Assche. "Sponge-Based Pseudo-Random Number Generator". In Mangard and Standaert [47], pp.33-47. Springer, 2010.
[10] Amir Moradi, Axel Poschmann, San Ling, Chrstof Paar, Huaxiong Wang. "Pushing the limits: A Very Compact and a Threshold Implementation of the AES". In Paterson [59].
[11] Andrev Bogdanov, Knudsen,L.R., Gregor Leander, Paar,C., Poschmann,A., Robshaw,M.J.B., Seurin,Y., Vikkelsoe,C. "PRESENT: An Ultra-Lightweight Block Cipher". In CHES'07. LNCS, vol.4727, pp.450-466. Springer, 2007.

