Implementation of SPONGENT Variants Using FPGA

[1]Neethu Sebastian,[2]Anjali V
[1]Student,[2]Assistant Professor
[1]ECE Department,
[1]Mangalam College of Engineering, Ettumanoor,India

***Abstract*— The design of secure cryptographic algorithm is a fundamental problem of cryptography. Recently a new technique has been adopted. i.e, Lightweight Cryptography. Lightweight cryptography is a cryptographic protocol or algorithms which are employed in constraint environments in order to alleviate the implementation difficulties. Also Cryptograhic hash function which play a vital role in cryptographic applications. This paper proposes SPONGENT hash function having hash size 88,128 and 160.These are implemented using Verilog Hardware Description Language. The Verilog code is synthesized on Spartan 3 FPGA using Xilinx ISE 14.7 software tool.**

***IndexTerms*— FPGA,Hash function,lightweight cryptography,SPONGENT, Xilinx.**

I. INTRODUCTION

Due to the advancement of pervasive computing, the need for the security in RFID and sensor networks is tremendously increasing which require efficiently implementable cryptographic primitives likes hash function. In a constrained environment area and power consumption are becoming important issues and the implementation of standard algorithm is expensive. So go for lightweight cryptography. Once this problem was identified, the cryptographic community designed a number of lightweight algorithms to address this challenge. They are HIGHT [2], DESL and PRESENT etc. This paper reveals the design space of lightweight hash functions based on sponge construction instantiated with PRESENT type permutation. The resulting family of such hash function is called SPONGENT.

Basically a Hash function is a function which map an arbitrary lengthy message into a fixed length output. The footprint of hash function [1] is determined by the number of state bits as well as the size of the functional and control logic used in a round function. The main properties of hash functions are fixed length output, preimage resistance, second preimage resistance and collision resistance. Preimage resistance means one could not be capable of recovering the original message from its hash value. Whereas second preimage resistance or weak collision resistant means that given one message, can't find another message that has the same message digest And collision resistant or strong collision resistance means that can't find any two different messages with the same message digest.

The most prominent security application targeted by a hash function are Lightweight signature schemes, RFID security protocols, Random number generation etc. Also hash algorithm can be used to verify the data integrity. After receiving the data one can compute the hash of the original data and can compare it with the original one that comes from a trusted source. If they are same it results in high confidence level. It means that message was not modified during the transmission. Indeed, hash functions are quite useful in determining whether any alteration has been made to message. It can also be used for the purpose of authentication. In this case hash value [3] of the user's password instead of user's password is transmitted and compared by the server. When computing the hash, password may be concatenated with a random value generated by the server. Hence the hashes are different every time, preventing the attacker sniffing on the network traffic from reusing the old hash. Hash algorithm can also be used in hash based message authentication code algorithms. They can be used in many systems in order to provide data integrity [4] and message authentication, especially in constraint environments. Hence a good cryptographic hash function is expected to behave like a random oracle. These application shows that cryptographic hashing is crucial for a wide spectrum of cryptographic applications ranging from symmetric key to public key cryptography.

This paper proposes sponge construction based on permutation function. In this case, r be the rate means that number of bits input or output per one permutation call, c be the capacity means that internal state bits not used for input or output and n be the hash length in bits. The resulting family of hash function is called SPONGENT. The parameterization of SPONGENT as –n/c/r. Where n denotes the hash size, c denotes the capacity and r denotes the rate. SPONGENT is a hermetic sponge [5] means that it doesn't allow underlying permutation to have any structural distinguishers.

II. DESIGN OF SPONGENT

SPONGENT depends on sponge construction instantiated with PRESENT type permutation function π_b. It is a simple iterated design that takes a varible length input and produce output of any fixed length. The design of SPONGENT follows hermatic sponge strategy.

A. *Permutation-based sponge construction*

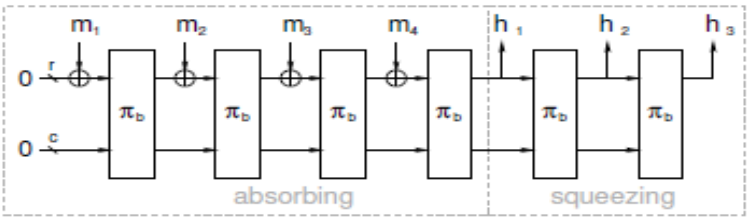

Figure. 1 Sponge construction based on a b-bit permutation π_b with capacity c bits and rate r bits. m_i are the r bit message blocks. h_i are parts of the hash value

SPONGENT is a simple iterated design. This takes a variable length input and produce output of any fixed length based on the permutation function π_b operating on internal state having b bits . The size of the internal state $b=r+c \geq n$ is called width, where r is the rate and c the capacity. The sponge construction follows three phases. They are initialization phase ,absorbing phase and the squeezing phase.

-**Intialization phase:** the message is padded by bit 1 followed by the number of 0 bits in order to make multiple of r bits.Then it is cut into r bit message blocks.

-**Absorbing phase:** the r bit message blocks are exored with first r bit of the state followed with the application of permutation function π_b.

Squeezing phase: the first r bits of the state are returned as output followed with permutation function π_b, until n bits are returned. In SPONGENT,the b-bit 0 is the initial value before the absorbing phase.Except SPONGENT 88/80/8,the other two has hash size n equals the capacity.

B. *Parameters*

This table propose three variants of SPONGENT with three different hash output at multiple security levels. It is shown in Table 1.

Table 1 SPONGENT Variants

Variants	n bit	b bit	c bit	r bit	R no. of rounds	security (bit)		
						pre.	2nd pre.	col.
88/80/8	88	88	80	8	45	80	40	40
128/128/8	128	136	128	8	70	120	64	64
160/160/16	160	176	160	16	90	144	80	80

C. *PRESENT-type permutation*

The permutation function π_b is an R-round transform of the input state b-bits. It is outlined as follows.

for i=1 to R **do**

state←retnuoC1$_b$ (i) ⊕ state ⊕ 1counter$_b$(i)

state←sBoxLayer$_b$(state)

state←pLayer$_b$(state)

end for

where sBoxLayer$_b$ and pLayer$_b$ indicates how the state evolves.Here only b/4 permutation is allowed. The number R of rounds depends on the block size b. 1counter$_b$(i) is the state of an LFSR dependent on b at time i, which is added to the rightmost bits of state. retnuoC1$_b$ (i) is the value of 1counter$_b$(i) with its bits in reversed order and is added to the leftmost bits of state. The sBoxLayer$_b$ and pLayer$_b$ constitute the main building block of *PRESENT* structure.

*sBoxLayer$_b$:*It denotes the use of a 4-bit to 4-bit S-box S. Here permutation is applied is b/4 times in parallel. The value of the S-box are given in hexadecimal notation. It is shown in Table 2.

Table 2 S box

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S[x]	E	D	B	0	2	1	4	F	7	A	8	5	9	C	3	6

pLayer$_b$: It is an extension of inverse PRESENT bit-permutation and moves bit j of state to $P_b(j)$, where

$$P_b(j)=\begin{cases} j.b/4 \bmod b-1, if\ j \in \{0,\ldots.,b-2\} \\ b-1, if\ j=b-1. \end{cases} \quad (1)$$

and can be seen in Fig. 2.

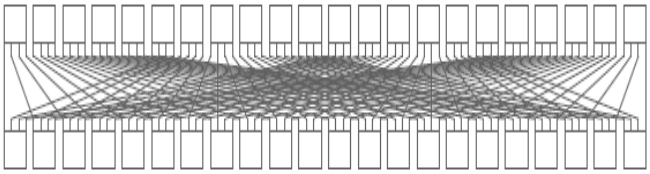

Figure. 2. The bit permutation layer of SPONGENT-88 at the example of pLayer$_{88}$

1counter$_b$: This is a $\log_2$R-bit LFSR. The LFSR is clocked once every time its state has been used and its final value is all ones. The n-bit LFSRs are defined by the primitive polynomial are shown in Table 3.

Table 3 Size and Initial Values of all the LFSRs

Variants	**LFSR size (bit)**	**Initial value (hex)**	**Primitive Polynomial**
88/80/8	6	05	$\alpha^6+\alpha^5+1$
128/128/8	7	7A	$\alpha^7+\alpha+1$
160/160/16	7	45	$\alpha^7+\alpha+1$

III. RESULTS AND OBSERVATION

SPONGENT coding were done using Verilog ISE Design Suite 14.7 for direct implementation in Spartan 3 FPGA kit. This paper proposes three variants of SPONGENT having hash size 88,128 and 160. This variants are capable of giving multiple security levels based on the size of hash value. Power ,Delay and Area analyses were carried out and are shown in Table 4.

Table 4 Comparision Table of Variants

Variant	**Power(mW)**	**Delay(ns)**	**Power*Delay (J)**
88/80/8	33	10.66	3.51E-10
128/128/8	43	10.70	4.60E-10
160/160/16	79	12.67	1.00E-9
SHA 1	86	12.09	1.03E-9

A. *Simulation Results*

The performance analysis of SPONGENT variants are analysed and it is simulated using ModelSim simulator . It is shown in Fig. 3,Fig. 4 and Fig. 5.

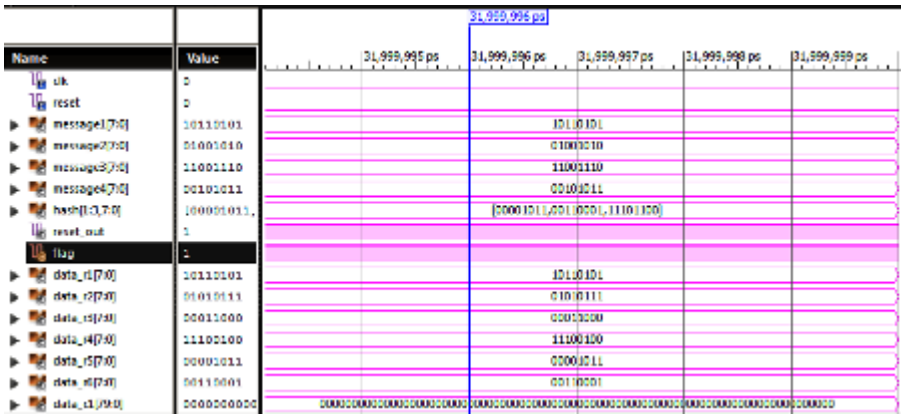

Figure. 3 Simulated waveform of SPONGENT 88/80/8

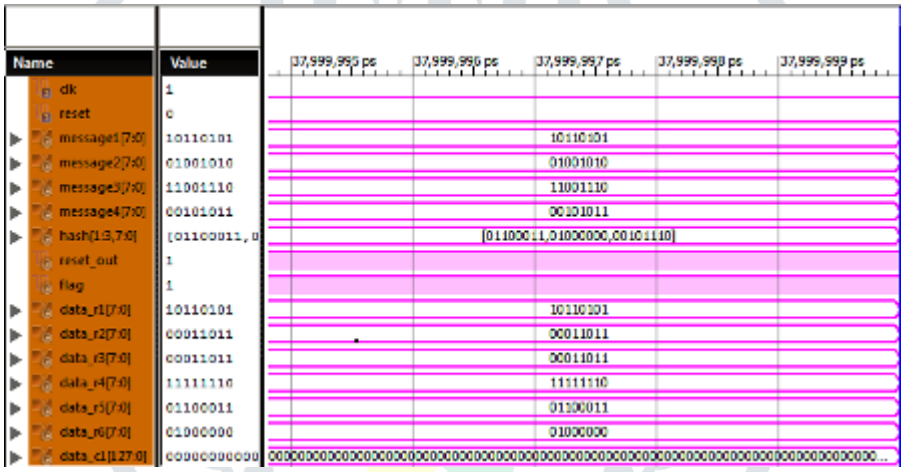

Figure. 4 Simulated waveform of SPONGENT 128/128/8

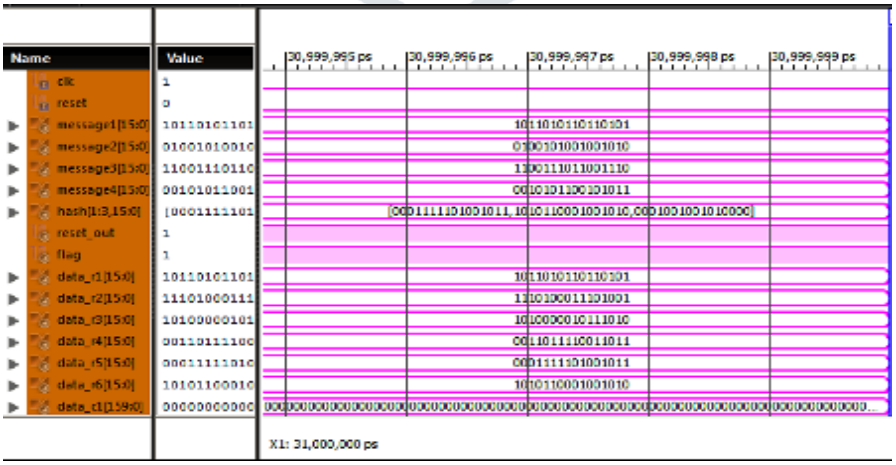

Figure. 5 Simulated waveform of SPONGENT 160/160/16

.

IV. CONCLUSIONS

Cryptographic hash function are the fundamental building blocks of many applications that are used in our daily lives. This paper proposed the design space of lightweight cryptographic hash function. It is applicable in resource constrained areas. From the design itself it is clear that its resourse useage is small as compared to other standard algorithm.

V. ACKNOWLEDGMENT

I am grateful to Ms. Anjali V,Assistant Professor,Department of Electronics and Communication Engineering ,Mangalam College of Engineering, for her innovative ideas in proceeding with this proposed system.

REFERENCES

[1] Bertoni G,Daemen.J,Peeters.M,Van Assche, G:Sponge-BasedPsuedo-RandomNumberGenerators ,In:Mangard.S, Standaert, F.X.(eds) CHES. LNCS, vol. 6225, pp.33-47. Springer 2010.

[2] Bertoni.G,Daemen.J,Peeters.M,VanAssche.G:On the indifferentability, of the Sponge Construction .In:Smart .P(.eds.)EUROCRYPT'08.LNCS, vol.4965, pp.181-197. Springer 2008.

[3] Aumasson, J.P, Henzen.L, Meier.W ,Naya-Plasenica. :Quark:A Lightweight Hash .In:Mangard.S,Standaert,F.X.(eds) CHES. lncs,vol.6225, pp.1- 15. springer 2010.

[4] Yang,B.,Wu,K.,Karri,R:Scan Based Side channel Attack on Dedicated Hardware Implementation of Data Encryption Standard .International Test Conference pp.339-344,2004.

[5] A. Bogdanov, M. Knezevic, G. Leander, D. Toz, K. Varici, and I. Verbauwhede, "SPONGENT: Lightweight Cryptographic Hash function," In:Preneel,B., Takagi,T. (eds.) CHES'11. LNCS, vol 6917, pp.312-325,Springer 2011.