Implementation of Cryptography Hash Function Blake 64 bit using SHA-3 Algorithm

M.Rajaram
M .Tech VLSI Design Scholar,
Dept. of ECE., Kalasalingam University,
Krishnankoil, virudhunagar Dist, Tamil Nadu, India

M.Arul Then Mathi
Assistant Professor
Dept. of ECE., Kalasalingam University,
Krishnankoil, virudhunagar Dist, Tamil Nadu, India

ABSTRACT

Hash functions form an important category of cryptography, which is widely used in a great number of protocols and security mechanisms. In this paper the VLSI implementation of one of the 14 "second-round" candidates BLAKE for 64 bit and the round rescheduling technique design are proposed by using modulo 2^n adder and adiabatic multiplexer for high throughput when compared to SHA 2.

Keywords

SHA-3, BLAKE 64, low power, , cryptography hash function, Encryption

1. INTRODUCTION

A cryptographic hash function is a hash function, that is, any algorithm that takes an arbitrary block of data and returns a fixed-size bit string a hash function is generated by a function H of the form h=H(M) where M is a variable-length message and H(M) is the fixed length hash value. Hash functions are used in a multitude of protocols, be it for digital signatures within high-end servers or for authentication of embedded .All hash function operate using bit-by-bit exclusive-OR (XOR) function

$$Ci= b_{i1}\oplus b_{i2}\oplus\ldots..\oplus b_{im}$$

MD5, SHA-2, and their variants are the most popular hash algorithms [1]. They follow the Merkle Damg°ard model and use logic operations such as AND, OR, and XOR in their compression functions. Recently, collision pairs have been found for MD5, SHA-0 and SHA-1 making these algorithms vulnerable to attacks because they do not have the collision resistance property [2]. Due to security concerns of SHA-1 and recent advances in the cryptanalysis of hash algorithms a new hash algorithm standard, SHA-3, which is meant to replace SHA-2. SHA-3 is expected to have at least the security of SHA-2, and to achieve this with significantly improved efficiency besides a sufficient security level, SHA 3 should be implementable on a wide range of environments for security. BLAKE is a second round candidate in the NIST Hash Competition. BLAKE is one of the simplest designs to implement, and relies on previously analysed components such as the HAIFA structure and the Cha-ha core function. The Inner state is initialized by using salt, counter value and initial value IVi.

2. ALGORITHM SPECIFICATIONS

BLAKE-512 operates on 64-bit words and returns a 64-byte hash value. All lengths of variables are doubled compared to BLAKE-256 the chain values are 512-bit, message blocks are 1024-bit, salt is 256-bit, counter is 128-bit. It is based on the iteration of a compression function.

3. COMPRESSION FUNCTION

BLAKE's compression function is the combination of an initialization, a sequence of rounds ,and a finalization. if m is a message (a bit string), m_i denotes its i-th 16-word block, and m_ij is the j-th word of the i-th block of m . A N-block message m is decomposed as m = m0,m1 . . .mN−1, and the block m0 is composed of words m00, m01,m02, . . . ,m015 each message word contains 64 bit (ie; totally 1024 bits). The compression function takes four inputs they are as follows:

1. Chaining values $h=h_0\ldots..h_7$
2. A message block $m=m_{0\ldots.}m_{15}$
3. Salt $s=s_0\ldots.s_3$
4. Counter $t=t_0,t_1$

These inputs represent 30 words totally 1920 bits. The Salt is an optional input for randomized hashing. Randomized hashing is mainly used for digital signatures instead of sending the signature Sign(H(m)), the signer picks a random r and sends (Sign(Hr(m)), r) to the verifier. The advantage of randomized hashing is that it relaxes the security requirements of the hash function .The output of the compression function is a new chaining value h0 = h00. . . h07 of eight words. The hash value can be expressed as

$$h = compress (h, m, s, t).$$

The compression function of BLAKE-64 makes 14 rounds instead of ten, and that Gi (a, b, c, d) uses rotation distances 32, 25, 16, and 11, respectively. After ten rounds, the round function uses the permutations σ 0, . . . , σ4 for the last four rounds. The compressed function can be decomposed into initialization, round function and Finalization.

3.1 INITIALIZATION

At the initialization stage, constants and redundancy of the impose a nonzero initial state [5].The disposition of inputs implies that after the first column step the initial value h is directly mixed with the salt s and the counter tIt consists of 16 word internal states (v_0,$v_1\ldots\ldots v_{15}$) is initialized such different input produce different state. Initialization takes the input as chaining value and produces the output as internal states. It represent as 4×4 matrix

$$\begin{pmatrix} V_0 & V_1 & V_2 & V_3 \\ V_4 & V_5 & V_6 & V_7 \\ V_8 & V_9 & V_{10} & V_{11} \\ V_{12} & V_{13} & V_{14} & V_{15} \end{pmatrix}$$

The initial state is defined as

$$\begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 & h_7 \\ S_0 \oplus C_0 & S_1 \oplus C_1 & S_2 \oplus C_2 & S_3 \oplus C_3 \\ t_0 \oplus C_4 & t_0 \oplus C_5 & t_1 \oplus C_6 & t_1 \oplus C_7 \end{pmatrix}$$

3.2 ROUND FUNCTION

Once the state v is initialized, the compression function iterates a series of 14 rounds. A round is a transformation of the state v that computes

G0(v0 , v4 , v8 , v12) G1(v1 , v5 , v9 , v13)

G2(v2 , v6 , v10, v14) G3(v3 , v7 , v11, v15) and then

G4(v0 , v5 , v10, v15) G5(v1 , v6 , v11, v12)

G6(v2 , v7 , v8 , v13) G7(v3 , v4 , v9 , v14)

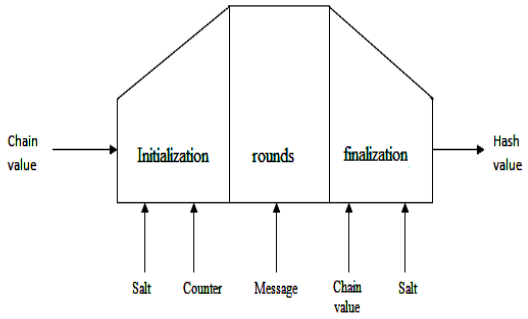

Figure 1: Block diagram of compression function

The figure 1 shows the diagram is a compression function for BLAKE 64 with input hash h, salt s, counter t and message m. The round function takes the input as message, internal states v_0 to v_{15} and output is fin_$v_{0\ to}$ fin_v_{15}.The sequence G0 to G3 is called a column step. Similarly, the last four calls G4 to G7 in update distinct diagonals and are called a diagonal step. The input message block is padded into 1024 bits.The unary operator >>> denotes rotation of words towards least significant bits.

3.3 FINALIZATION

After the rounds sequence, the new chain value h00 to h07 is extracted from the state v0, . . . , v15.With input of the initial chain value h0, . . . , h7 and the salt s0, . . . , s3:

$$HH_0 <= h0 \oplus s0 \oplus v0 \oplus v8;$$

$$HH_1 <= h1 \oplus s1 \oplus v1 \oplus v9;$$

$$HH_2 <= h2 \oplus s2 \oplus v2 \oplus v10;$$

$$HH3 <= h3 \oplus s3 \oplus v3 \oplus v11;$$

$$HH_4 <= h4 \oplus s0 \oplus v4 \oplus v12;$$

$$HH_5 <= h5 \oplus s1 \oplus v5 \oplus v13;$$

$$HH_6 <= h6 \oplus s2 \oplus v6 \oplus v14;$$

$$HH_7 <= h7 \oplus s3 \oplus v7 \oplus v15;$$

4. HASHING A MESSAGE

For BLAKE-512, message padding goes as follows: append a bit 1 and as many 0 bits until the message bit length is congruent to 895 modulo 1024. . Then append a bit 1, and a 128-bit unsigned big-endian representation of the message bit length**:**

$$m \leftarrow m||1000\ldots0001<l>_{128}$$

This procedure guarantees that the length of the padded message is a multiple of 1024. It is then processed block per block by the compression function, as described below

h0:= IV

For i = 0, . . . ,N − 1

hi+1 := compress(hi,mi, s, i)

return h N

Here, `i is the number of message bits in m0 to mi, that is, excluding the bits added by the padding. It is used to avoid certain generic attacks on the iterated hash "random salt" of BLAKE-512 is a random variable uniformly distributed over{0, 1}256, and may also mean "uniformly chosen at random" .The initial value is written as IVi.

5. HASHING A SALT

The BLAKE hash functions take as input a message and a salt. The aim of hashing with distinct salts is to hash with different functions but using the same algorithm. Depending on the application, the salt can be chosen randomly (thus reusing a same salt twice can occur, though with small probability), or derived from a counter (nonce). For applications in which no salt is required, it is set to the null value (s = 0). In this case the initialization of the state v simplifies to

$$\begin{pmatrix} V_0 & V_1 & V_2 & V_3 \\ V_4 & V_5 & V_6 & V_7 \\ V_8 & V_9 & V_{10} & V_{11} \\ V_{12} & V_{13} & V_{14} & V_{15} \end{pmatrix} \leftarrow \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 & h_7 \\ c_0 & c_1 & c_2 & c_3 \\ t_0 \oplus c_4 & t_0 \oplus c_5 & t_1 \oplus c_6 & t_1 \oplus c_7 \end{pmatrix}$$

6. IMPLEMENTATION OF BLAKE64 WITH 8G CORE

The design shows the architecture of BLAKE, with an iterative decomposition of the round process 8G. Different architectures are made possible by varying the number of integrated G modules [6]. For each round, a nonlinear function G that operates on four words is applied to columns and diagonals of the state

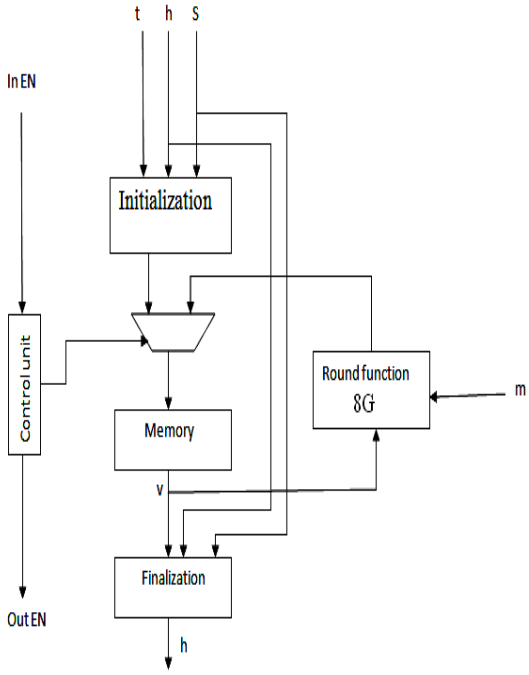

Figure 2: architecture of Blake 64 with 8G core

The figure 2 shows the architecture of Blake 64 with 8G BLAKE requires some circuitry to perform initialization and finalization for instance, w = 64 for BLAKE-64 the complete execution of initialization and finalization can be performed in the same clock cycle, when the new message block is given BLAKE uses some constant values, which are

1. The 16 round constants Ci;
2. The 14 round permutations: uses rotation distances 32, 25, 16, and 11. After 10 rounds the permutations σ0…σ4
3. The initial value IVi (eight w-bit words);

These values are used mainly by the G function; The G functions performed on the four columns and diagonals can be done in parallel, which allows convenient performance-area trade-off. The performance can be improved if more resources

a. 8-word chaining value h;
b. 16-word internal state v;
c. 4-word of the salt value s;
d. 16-word message block m.

The sequential area is thus made by 44×w registers 2816 for BLAKE 64 plus some additional registers for control unit .The only differences with BLAKE-256's Gi are the word length (64 bits instead of 32) and the rotation distances. At round r > 9, the permutation used is σr mod 10 in the last round r = 14 and the permutation σ14 mod 10 = σ5 is used

Round transformation is based on eight different functions Gi, with i = 0,…,7. G functions, in the proposed architecture operate on data with two different ways. First operates on v matrix cell, in both column and diagonal step. In the proposed architecture the first four functions Gi are computed in parallel, due to the fact that each one of them transforms different columns of the matrix. This is taken place in column transformation. On the other hand, the next four of Gi functions, G4,…,G7 work on diagonals and can be work on parallel, in an alternative way. 8G-BLAKE design corresponds to the isomorphic implementation of the round Function. Eight G function units are instantiated; the first four units work in parallel to compute the column step, while the last four compute the diagonal step.

6.1 ROUND RESCHEDULING

The G function of BLAKE 64 bit based on modified version cha-cha cipher. The normal addition with the message/constant (MC) -pair in the G function leads to an increment of the propagation delay When the round sequence function is taken over, the new chain value h' = h'0, …, h'7 is produced from the state v with inputs of the initial chain value and the salt. Some generic Gi architecture, which is basically, consists of n-bit modulo adders, XOR chains and shift operations and registers .The figure 3 shows the diagram of G function of BLAKE. In the architecture the first four functions Gi are computed in parallel, due to the fact that each one of them transforms different columns of the matrix. This is taken place in column transformation. On the other hand, the next four of Gi functions, G4,…,G7 work on diagonals and can be work on parallel

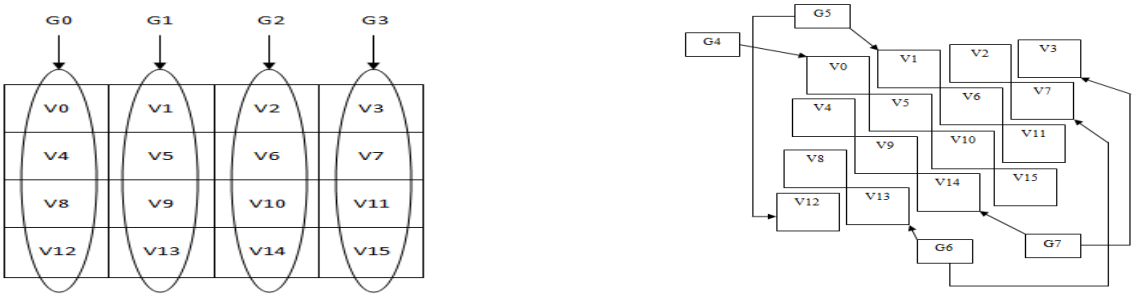

Figure 3: Column step (on the left) and diagonal step (on the right) in BLAKE

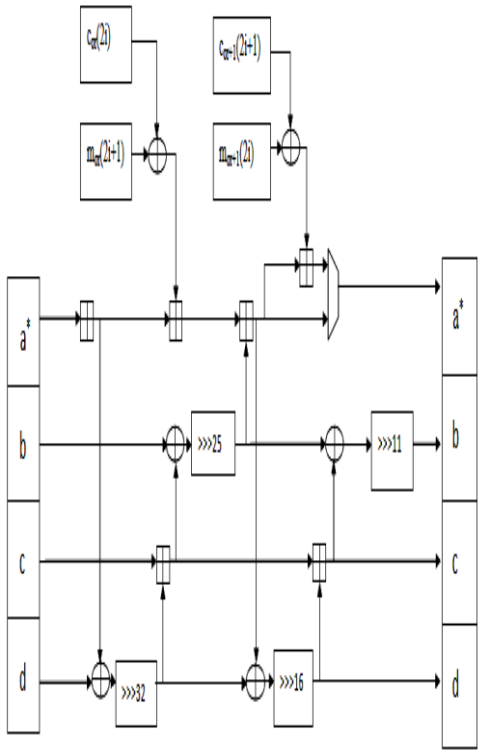

Figure 4: Block diagram of rescheduled G function

The figure 4 shows the block diagram of rescheduled G function. In this rescheduled G function consists of addition, XOR and rotation operations. Each G function operates on 4 elements of state matrix BLAKE-512 is a series of 32 half-rounds for each 128-byte block. Each half-round contains 4 parallel \G functions" each consuming 6 64-bit additions, 4 64-bit rotations, and 6 64-bit xors 6 64-bit additions per byte, 4 64-bit rotations per byte, and 6 64-bit xors per byte.

$a = a + b + (m\sigma r(2i) \oplus c\sigma r(2i+1))$

$d = (d \oplus a)>>>32$

$c = c + d$

$b = (b \oplus c)>>>25$

$a = a + b + (m\sigma r(2i+1) \oplus c\sigma r(2i))$

$d = (d \oplus a)>>>16$

$c = c + d$

$b = (b \oplus c)>>>11$

Where σr represents a permutation of integers between 0 and 15. There are ten different permutations, which are reused when the round number is ten or greater round 10 uses σ0 and round 11 uses σ1.The only differences with BLAKE-32's Gi are the word length (64 bits instead of 32) and the rotation distances. At round r > 9, the permutation used is σrmod 10

7. CONCULSION

In the Blake 64 bit architecture, the design of Blake 32 bit architecture is modified by using MOD 2 adder and adiabatic multiplexer .compare to Blake 32 bit architecture this modified design provide high throughput and low area, low power in rescheduling G function. By the method of initialization, round function and finalization in the network security application the Blake 64 bit generate hash value .In future enhancement by using carry save adder to increase the throughput and reduce the propagation delay.

8. REFERENCES

[1] Spector, A. Z. 1989. Achieving application requirements. In Distributed Systems, S. Mullender

[2] NIST, "Announcing the secure hash standard," FIPS 180-2, Technical report, 2002

[3] R. Lien, T. Grem bowski, and K. Gaj, "A 1 Gbit/s partially unrolledarchitecture of hash functions SHA-1 and SHA-512," in Topics in Cryptology - CT-RSA 2004, ser. Lecture Notes in Computer Science , vol. 2964.Springer Berlin / Heidelberg, 2004.

[4] X. Wang and H. Yu, "How to break MD5 and other hash functions, "in Advances in Cryptology - EUROCRYPT 2005, ser. Lecture Notes inComputer Science, vol. 3494. Springer Berlin / Heidelberg, 2005, pp.19–35

[5] C. D. Cannière and C. Rechberger, "Finding SHA-1 characteristics: General results and applications," in Advances in Cryptology – ASIA CRYPT2006, ser. Lecture Notes in Computer Science, vol. 4284. SpringerBerlin / Heidelberg, 2006, pp. 1–20

[6] J.-P. Aumasson, L. Henzen, W. Meier, and R. C.-W. Phan, "SHA-3 proposal BLAKE," Submission to NIST, 2008, http://131002.net/blake/

[7] Luca Henzen, Student Member, IEEE, Jean-Philippe Aumasson, Willi Meier, andRaphael C.-W. Phan, Member, IEEE "VLSI Characterization of the Cryptographic Hash Function BLAKE" Oct. 2011

[8] D. J. Bernstein, "Cha-cha, a variant of Salsa20," 2007, http://cr.yp.to/ chacha.html

[9] "Call for a new cryptographic hash algorithm (SHA-3) family," Federal Register, Vol.72, No.212, 2007, http://www.nist.gov/hash-competition