IMPLEMENTATION OF BLAKE ALGORITHM USING PIPELINING IN FPGA

[1]M.Jothi Kumar, [2]Chitravalavan
[1]Research Scholar, Department of Applied Electronics, A.V.C. College of Engineering, Mannampandal
[2]HOD, Department of Electronics and Communication Engineering, A.V.C. College of Engineering, Mannampandal
[1]jothikumar91@gmail.com, [2]chitravalavan@gmail.com

Abstract: This paper proposes the Pipelined SHA-3 BLAKE algorithm, running on an FPGA with the intention of developing the optimization in FPGA for BLAKE algorithm. Secured hash algorithm-3(SHA-3) BLAKE algorithm is a family of cryptographic hash function published by the National Institute of Standards and Technology (NIST). To implement BLAKE algorithm we have utilized VHDL, where we introduce the pipelining. Pipelining executes single task per clock cycle which improves the efficiency of the algorithm. The simulation is done with the usage of model sim and synthesis it on FPGA Spartan 3E using Xilinx software. Our VHDL implementation executed on BLAKE algorithm occupies 30% percentage of FPGA's area.

Keywords: Cryptography; SHA-3 finalist; BLAKE algorithm; Pipelining; FPGA; VHDL.

1. INTRODUCTION

Cryptography is the practice and study of techniques for secure communication in the presence of third parties (i.e.) encrypt the message using the secret key, that message is known only by sender and receiver not by the third party. Encryption is the transformation of plaintext to cipher text based on encryption algorithm using the secret key. Third parties able to identify the message only by knowing either encryption algorithm or secret key. If both sender and receiver use the same key, then it is called as symmetric key encryption and if they use the different key is called as asymmetric key encryption. All encryption algorithms are based on substitution and transposition. In plain text each element is mapped into another element is called as substitution and elements in the plaintext are rearranged in the transposition The National Institute of Standards and Technology (NIST) announced a global competition to find a new Secured Hash Algorithm (SHA) function to replace SHA-0, SHA-1 and SHA-2 in 2007. BLAKE is a cryptographic hash function submitted to the NIST hash function competition by Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and RaphaelC.W.Phan. BLAKE was chosen as one of the five finalists in SHA-3 competition announced by NIST in 2010. It is implemented using pipelining which executes single task per clock cycle instead all tasks in one clock cycle. BLAKE algorithm has a family of four hash functions, for instance: BLAKE-224, BLAKE-256, BLAKE-384 and BLAKE-512, with different sizes of output. Here, we have implemented BLAKE-256 using VHDL in FPGA Spartan 3E.

VHDL (VHSIC Hardware Description Language) is a language for describing hardware from abstract to concrete level. Firstly it describes the structure of a design that is how that structure of a design can be decomposed into sub-design and how they are interconnected. Secondly, it allows describing the function specifications of a design using programming language forms. Thirdly, as a result, it allows simulating the design before manufacturing. So the designer can quickly compare the alternatives and test for correctness.

FPGA (Field Programmable Gate array) is an integrated circuit which can be configured by a customer or designer after manufacturing, hence it is named as Field Programmable. The FPGA configuration can be specified using the HDL(Hardware Description Language).FPGA contains Configurable Logic Blocks(CLB) with Programmable Interconnect. Configurable Logic Blocks are called as slices and each slice contains 2 Flip Flops(FF),2 multiplexers and 2 four input Look Up Tables(LUTs).

2. EXISTING SYSTEM

The existing system presents the analysis of assembly instructions from SHA-3 BLAKE algorithm, running on an ARM processor, with the intention of developing a specific processor in FPGA for BLAKE Algorithm. Moreover, VHDL has been utilized here. The Synthesis and simulations of the ALU and UC were done with the usage of Altera

Quartus II 9.1. The instruction set for the ALU for BLAKE- 256 algorithm was based on the instructions from the ARM processor which is not reconfigurable.

The implementation of a control unit and an ALU, based on set of assembly instructions from the ARM (Advanced RISC Machine) processor for BLAKE algorithm (256 version), which will be implemented using VHDL(Very High Speed Integrated Hardware Description Languages) and FPGA(Field Programmable Devices). The assembly instructions for BLAKE algorithm can be obtained from the simple scalar simulation tool. This VHDL implementation executed on BLAKE algorithm occupied 43% of the FPGA's area.

3. PROPOSED SYSTEM

The Proposed system presents the implementation of SHA-3 BLAKE algorithm (version 256) in FPGA using Pipelining. This Pipelined SHA-3 BLAKE algorithm implemented in VHDL and simulated using modelsim. Finally, we synthesis it on FPGA Spartan 3E using Xilinx.

3.1 Blake-256Algorithm

BLAKE-256 algorithm is one among the family of four hash functions and it is a 32 bit version. BLAKE-256 produces a 256-bit hash which is treated as eight 32-bit word. Table I represents the some important terminologies involved in the BLAKE-256 algorithm.

Table 1: BLAKE-256 algorithm Terminologies

Symbol	Meaning
←	Variable assignment
+	Sum(module)
>>>k	K-bit rotation to the right
<<<k	K-bit rotation to the left
	Exclusive-OR(Ex-or)

BLAKE algorithm performs both encryption and compression function. The compression function of the BLAKE-256 algorithm involves three steps: i) Initialization ii) Round function iii) Finalization.

3.2 Initialization

Fig.1 shows the initialization vector of the BLAKE-256 algorithm.

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix} \leftarrow \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 & h_7 \\ s_0 \oplus c_0 & s_1 \oplus c_1 & s_2 \oplus c_2 & s_3 \oplus c_3 \\ t_0 \oplus c_4 & t_0 \oplus c_5 & t_1 \oplus c_6 & t_1 \oplus c_7 \end{pmatrix}$$

Figure 1: Initial state of the compression function

The initial state of the compression function is treated as a 4x4 matrix as shown in Fig.2. This state is initialized by the current hash h, salt value s, timer value t, and a 256-bit constant c.

3.3 Round Function

Once the matrix is initialized, next step is round function which is iteratively processed through 14 rounds. Each round consists of eight state transformations labelled as G0 through G7. Each operates on and modifies only 4 of the 16 state

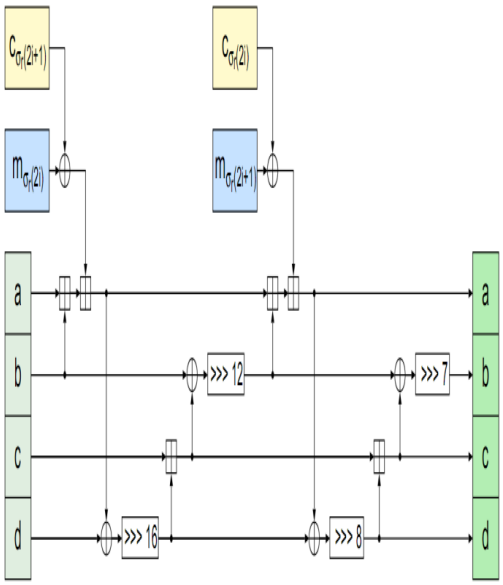

Figure 2: Visualized G transformation.

$$
\begin{aligned}
a &\leftarrow a + b + (m_{\sigma_r(2i)} \oplus c_{\sigma_r(2i+1)}) \\
d &\leftarrow (d \oplus a) \ggg 16 \\
c &\leftarrow c + d \\
b &\leftarrow (b \oplus c) \ggg 12 \\
a &\leftarrow a + b + (m_{\sigma_r(2i+1)} \oplus c_{\sigma_r(2i)}) \\
d &\leftarrow (d \oplus a) \ggg 8 \\
c &\leftarrow c + d \\
b &\leftarrow (b \oplus c) \ggg 7
\end{aligned}
$$

Figure 3: Generalized G transformation

Words, generalized as a, b, c, and d. The visual representation of G transformation is shown in Fig.2.The transformations consisting of addition, bit-rotations, exclusive or (XOR) operations.

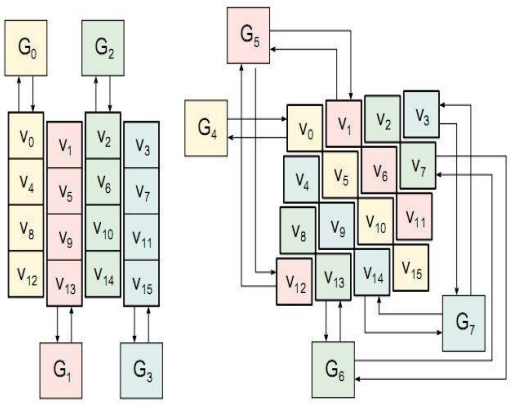

Figure 4: The parallelized column and diagonal step

The generalized form of G transformation is shown in Fig.3. The first four steps are performed on independent columns in parallel and last four steps are performed on independent diagonals in parallel as well. The parallelized column and diagonal step is shown in Fig.4. For one round, eight (4 column step [G0 to G3] and 4[G4 to G7] diagonal step) G function can be called and totally 112 G function can be called for 14 rounds.

3.4 Finalization

After fourteen rounds of G transformation, compression function performs finalization step. The final step in compression function is shown in Fig.5

$$
\begin{aligned}
h'_0 &\leftarrow h_0 \oplus s_0 \oplus v_0 \oplus v_8 \\
h'_1 &\leftarrow h_1 \oplus s_1 \oplus v_1 \oplus v_9 \\
h'_2 &\leftarrow h_2 \oplus s_2 \oplus v_2 \oplus v_{10} \\
h'_3 &\leftarrow h_3 \oplus s_3 \oplus v_3 \oplus v_{11} \\
h'_4 &\leftarrow h_4 \oplus s_0 \oplus v_4 \oplus v_{12} \\
h'_5 &\leftarrow h_5 \oplus s_1 \oplus v_5 \oplus v_{13} \\
h'_6 &\leftarrow h_6 \oplus s_2 \oplus v_6 \oplus v_{14} \\
h'_7 &\leftarrow h_7 \oplus s_3 \oplus v_7 \oplus v_{15}
\end{aligned}
$$

Figure 5: Final step in compression function

Which produces new hash value (h0' to h7') by performing EX-OR operation on sequence of new values (v0 to v15) with initial hash value (h0 to h7) and salt value (s0 to s3).

4. IMPLEMENTATIONARCHITECTURE

In this paper, BLAKE-256 algorithm is implemented using the pipelining. The basic structure of implementing BLAKE- 256 algorithm in FPGA using VHDL is shown in Fig.6. First, set the initial hash and constant value and using the iteration matrix send the A, B, C ,D input for G function. Then, set the A, B, C, D output from G function (i.e.) values in the column and diagonal are updated by fourteen rounds of G function call.

H_A_in, H_B_in, H_C_in, H_D_in are the initial hash values set to the BLAKE and the output hash values H_A_out, H_B_out, H_C_out, H_D_out from the BLAKE are sent as the G Function input (A_in,B_in,C_in,D_in) as shown in Fig.6. After performing G function (column and diagonal step), output of the G function (A_out,B_out,C_out,D_out) are set as the new hash value for BLAKE. This step is repeated continuously for 14 times (14 rounds) and finally gets the new hash value from the BLAKE.

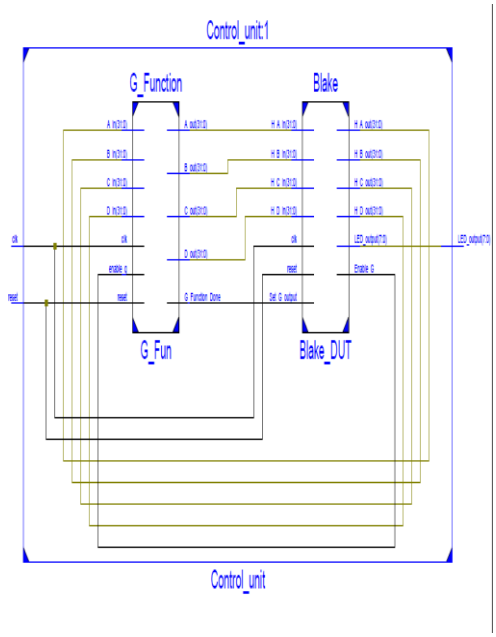

Figure 6: BLAKE algorithm implementation

5. SIMULATION AND SYNTHESIS RESULTS

The simulation result of implementing BLAKE algorithm in VHDL using modelsim is shown in Fig.7.

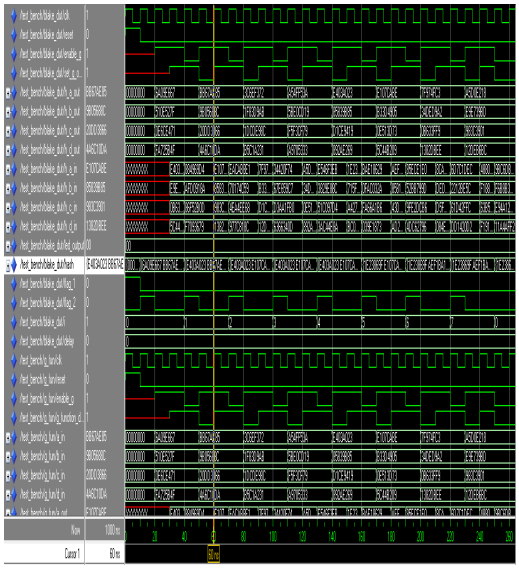

Figure 7: Simulation result

Table II shows the synthesis report, which got by synthesizing pipelined BLAKE algorithm on FPGA Spartan 3E using Xilinx.

Number of Slices	1443 out of 4656	30%
Number of Slice Flip Flops	1018 out of 9312	10%
Number of 4 input LUTs	2779 out of 9312	29%
Number of IOs	10	-
Number of bonded IOBs	10 out of	4%
Number of BRAMs	2 out of	10%
Number of GCLKs	1 out of	4%

The BLAKE algorithm occupies 1443(30%) slices out of 4656 (i.e.) 30% of the FPGA's area.

6. CONCLUSION

BLAKE algorithm is one of the security algorithms, we considered BLAKE due to its accomplishment on being one of the 5 finalist algorithms of competition prompted by NIST. BLAKE algorithm is so simple, fast in both hardware and software. The implementation of this algorithm in FPGA Spartan 3E using pipelining occupied 30% of the FPGA's area. For the future, strength of the algorithm can be tested by the security attacks, both passive and active attacks

REFRENCES

[1] J-P. Aumasson, L. Henzen, and R. C. W. Phan, "SHA-3 proposal BLAKE*" , Report NIST, December 16, 2010.

[2] A. Regenscheid, R. Perlner, S-J. Chang, J. Kelsey, M. Nandi, and S.Paul, "Status Report on the First Round of the SHA-3 Cryptographic Hash Algorithm Competition", NIST, September 2009.

[3] B. Baldwin, and W. P. Marnane, "Yet Another SHA-3 Round 3 FPGA Results Paper". IACR Cryptology

ePrint, v. 2012, p. 180.

[4] S. Kerckhof, F. Durvaux, N. Veyrat-Charvillon, F. Regazzoni, G. M.de Dormale, and F-X. Standaert, "Compact FPGA Implementations of the Five SHA-3 Finalists". In Proceeding 10th IFIP WG 8.8/11.2 International Conference - Smart Card Research and Advanced Applications (CARDIS 2011), Leuven, Belgium, pages 217-233, September 2011.

[5] J-P. Kaps, P. Yalla, K. K. Surapathi, B. Habib, S. Vadlamudi, S.Gurung, and J. Pham, "Lightweight Implementations of SHA-3 Candidateson FPGAs". In Proceeding 12th International Conference on Cryptology in India (INDOCRYPT 2011), Chennai, India, pages 270-289, December 2011.

[6] N. Sklavos, and P. Kitsos, "BLAKE HASH Function Family on FPGA: From the Fastest to the Smallest". In Proceeding of IEEE Computer Society Annual Symposium on VLSI (ISVLSI'2010), Lixouri Kefalonia, Greece, pages139-142,July2010.

[7] Cryptography and Network Security Principles and Practices,4th edition by William Stallings, November 16, 2005

[8] Pereira V.F, Moreno E.D, Dias W.R.A , q D.O.D, "Specific processor in FPGA for Blake Algorithm" ,Circuits and Systems (LASCAS), March 2013.