International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 8, No. 3, November 2019, pp.169~174
ISSN: 2089-4864, DOI: 10.11591/ijres.v8.i3.pp169-174

512 bit-SHA3 design approach and implementation on field programmable gate arrays

S. Neelima[1], R. Brindha[2]
[1]Department of ECE, Faculty of Engineering, Avinashilingam institute for home science and higher education for woman, India
[2]ECE Avinashilingam institute for home science and higher education for woman Faculty of Engineering, India

Article Info

Article history:

Received Jul 7, 2019
Revised Sep 25, 2019
Accepted Oct 11, 2016

Keywords:

Encryption and decryption
FPGA
SHA3
TPS ratio

ABSTRACT

In this work, the authors consider the newly selected Hash Secure (SHA-3) algorithm on FPGA Gateway. The design is logically optimized for zone efficiency by combining the Rho steps and the one-pass algorithm. Logically recording these three steps registers leads to usage 16% of the logical resources for all implementations. This in turn reduces the latency and increases the maximum operating frequency of the project. It uses only 240 sections and has a frequency of 301.02 MHz compared to the design results with the previous FPGA implementation described in SHA3-512, the design shows the Throughput-Per-Slice (TPS) ratio of 30, 1.

Corresponding Author:

S. Neelima,
Department of ECE,
Faculty of Engineering,
Avinashilingam institute for home science and higher education for woman, India.
Email: seethala.neelimakesav@gmail.com

1. INTRODUCTION

A hash cryptographic function is a deterministic process that is input from every data block that yields a fixed (cryptographic) value. These features were originally introduced to meet specific security and privacy requirements. New hash algorithms have been found. Persistent for attacks such as MD5, RIPEMD, SHA-0, SHA-1, and SHA-2 [1]. This algorithm was a dangerous long-term security that resulted in a new cryptographic hash request. That is why the National Institute of Standards and Technology (NIST) for the Keccak algorithm was announced in April 2014 as the new Secure Hash Algorithm (SHA-3) in 2012 [2] FIPS PUB) 202 [3]. FPGAs are a great platform for the use of cryptographic algorithms. Modern FPGAs, as well as LUTS and CLB [4] with integrated RAMS and DSL blocks, support the implementation. The different implementation of a safe hash algorithm (SHA-3) describes FPGA platforms in open literature [5]. Previous work was divided into categories in an area with a low and a high number of revolutions. The main challenge was to complete the project with the most appropriate means to balance the area and limit restrictions. Up to now, different approaches have been chosen for implementing SHA-3, depending on whether it concerns slow or fast projects. This section describes the implementation of the High-Input SHA-3 FPGA CD, which offers maximum throughput and maximum use of TPS devices.

2. SURVEY ON HASH ALGORITHMS

Many previous types of research on the implementation of SHA-3 FPGA has been published since 2012. Most implementation rules [6-8] are unprecedented for high productivity and little known for

the design compact [9]. In terms of area, the weakest hardware resources [6] of the project were designed, consisting of 188 sections operating at a frequency of 285 MHz, which significantly reduces the current by requiring clock cycles to produce output. The output of 6.07 Gbps is ready for use with a clock frequency of 143 MHz [7]. The design requires 25 clocks and 2024 tuning sections and brings TPS back to 2.99. The design used on Virtex-5 works with the maximum frequency of 238.4 and needs 24 cycles [8]. The result is designed for 1,0805 Gbps thanks to the extensive use of 1229 sections; That is why the GST value is 0.879, which is very low. The strictly published use of SHA-3 (Keccak) in [9] reported the results of SHA-3 for Virtex-6 and Spartan-6 FPGA. They show the implementation results for a 256-bit and 512-bit sales. However, consider the implementation of SHA-3 Virtex-6 for a 512-bit digestion, with a frequency of 285 MHz and a volume of 0.08 Gbps, which is much less than the results of the present paper. The application was inside G. Provelengios et al. [10] Virtex-5 operates with a maximum of 285 MHz and a large surface area of 2573 disks. The design requires 25 clock cycles and a high throughput of 5.70 Gbps, but ASP is 2.21, which is very low due to the consumption of a large area. Gives the results of implementing Keccak 512 drawings implemented in Virtex-5 times better than 6.32 Gbps sketch sketches due to shorter clock cycles [11]. This design consisted of 1197 plates and TPS project 5.27. E.Hom. et al. Transfer from 6.56 Gbps to Virtex-5 and the corresponding TOS 5.37 [12]. Previously reported projects do not offer an effective resource solution because these designs use many slides that can be effectively reduced. These projects have a very low TPS and must be improved.

3. DETAIL ANALYSIS OF KECCAK

Keccak has been identified as a new algorithm from 3 to SHA-3 [3] announced by NIST. Gilles Van Assche, Bertoni Guido, Joan Daemen Michael Peeters are worked for development of a hash function Keccak. Keccak-F is a basic component of standard Keccak hash components and supports hash variable on 224 bits, 256 bits, 384 bits and 512 bits. It consists of a round number and each round is a combination of logical operations and permutations. The error is caused by a fungal function with members of Keccak [r, c]. This is classified according to this extra function, such as interest (s) and capacity (c). The function of the additive permutation s+C to Keccak and is limited to the value indicated by 25, 50, 100, 200, 400, 800, 1600. Tim Keccak mode [1600] introduces the SHA-3 propositions with values other than 'R' and 'C'. [1600] was chosen to increase the number of cycles to provide a better safety margin. For a hash value of 256 bits=1088 rc=512. For the 512-bit hash output, the values of R and C, respectively 576 and 1024. The ground matrix comprises 1600 bits of brightness of 64 bits of matrix 5x5 words. Initially, the blocking messages must go through an inversion procedure, so this action is the last to come first and the first action must be the last. Each of the Keccak compression functions consists of 24 turns and which in turn is divided into five phases, namely Theta (Θ), rho (ρ) and Pi (π), Chi (χ) Iota (i) describes the bottom of the page.

3.1. Theta (Θ)

The Theta function consists of three equations with simple XOR movements and bit transfers. The XOR operation refers to rows (a set of 64 bits along x and at constant coordinates) for each set of five output row matrices.

$$C[X] = A[X,0] \oplus A[X,1] \oplus A[X,2] \oplus A[X,3] \oplus A[X,4] \quad 0 \le X \le 4 \tag{1}$$

Initially, the left path is used for five of these output lines, with the last line in the first and second line being the last line of (2). After moving, so that the first path to the last number and the other in the first track, then the change in which a circular variation to the left is applied at least on each track of the track was to make the position binary from a circular line on the right, (3) The only XOR content does not match the input-input matrix and the output path obtained from (2).

$$D[X] = C[X-1] \oplus ROT(C[X+1,1]) \quad 0 \le X \le 4 \tag{2}$$

$$A[X, Y]' = A[X, Y] \oplus D[X] \quad 0 \le X, Y \le 4 \tag{3}$$

3.2. Rho (p) and Pi (π)

The first two phases Rho (ρ) and pi (π) can be calculated as (4), in which a number of 5x5 emergency services for the composition of the state 'A' is calculated. Given the operation Rho (ρ) and Pi (π) and all 25 tracks, organization "A", connected to the circle, was a number of fixed values in response

to the "x" coordinates and rotating "Y", R [x, y] enters the table in [3] (the step is also called (ρ)) and then follows the tracks with different settings of the new series B (called step Pí (π)). Note that all indexes in module 5 are accepted.

$$B[Y,2X+3Y] = ROT(A[X,Y],r[X,Y]) \quad 0 \le X,Y \le 4 \tag{4}$$

3.3. Chi (*X*) phase

The Chi (*X*) phase is driven, i.e. a 64-bit word and manipulates the composition B obtained in step Rho (ρ) and Pi (π), and places itself in many states A can be said to have the phase (x) is for the path access to the place [x, y] and the logical path XOR to the address [x+1, y] and the site number [x+2, e]. The following equation describes the function of Chi (x).

$$A[X,Y] = B[X,Y] \oplus ((\text{NOT } B[X+1,Y]) \text{ AND } B[X+2,Y]) \quad 0 \le X,Y \le 4 \tag{5}$$

3.4. Iota (*i*) phase

Step *i* is the simplest step of Keccak's algorithm. It only changes XOR for a constant 64-bit operation defined in [3] with a row of the matrix [0,0] of the new state "A". The operation of a hash mode of Keccak at three different levels, initialization, is the first to be absorbed and finally printed as in Figure 1. The KECCAK offsets Cyclic shifts of R(*X*,*Y*) a shown in Table 1.

$$A[0,0] = A[0,0] \oplus RC \tag{6}$$

Table 1. The KECCAK offsets cyclic shifts of R(*X*,*Y*)

	X=3	X=4	X=0	X=1	X=2
Y=2	25	39	10	10	43
Y=1	55	20	36	44	6
Y=0	28	27	0	1	62
Y=4	56	14	18	2	61
Y=3	21	8	41	45	15

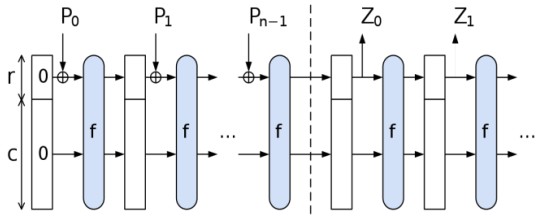

Figure 1. The Sponge function

Each matrix is started with the status 'A' to zero. In phase 2, an absorption phase, width circulation, finely packaged message blocks of the state matrix, made with 24 rounds of this Keccak permutation in succession. If the first message blocks are created, if the input block comes in that order. Finally, at the blackmail stage, the truncated length of the hashout pit can be reached and other parts of the array can reach the current hash situation of the desired length. If (bit rate) of the hash value is more necessary than E bit, the extra permutations Keccak can be active and wait for further results to reach the desired length to the hash value width.

4. IMPLEMENTATION OF SHA-512 ON FPGA

In this work, we present a SHA-3512-bit e-mail for the compact application, as shown in Figure 2. The architecture is a 128-bit input data input that includes an extra input bit. The next block in the proposed construction of fill blocks is to provide the required number of zeros in the bit data formation in 1600 data and then invert the operation for each measurement. The output of the block to the adder 2x1 multiplexer (MUX), the compression of the architecture of output data foulard Feld readers and selection of input data for the first set of data and feedback twenty-three other towers. The film that uses the control of the signal (Ctrl 1).

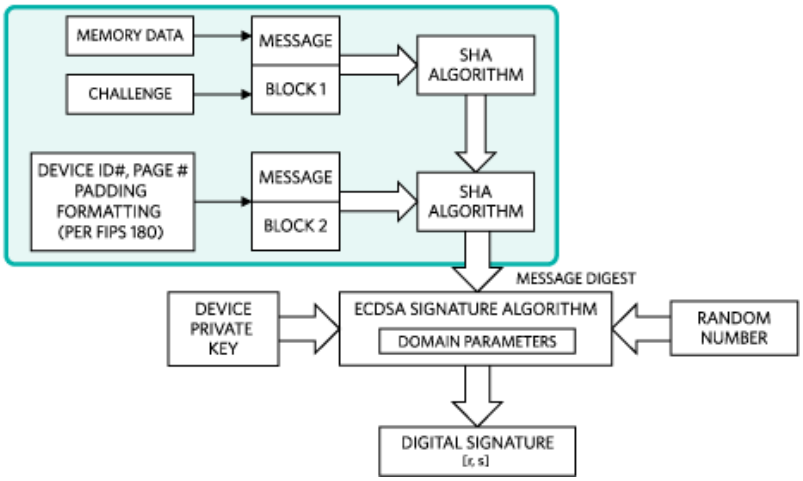

Figure 2. Sequential architecture for 128 bit Keccak

When Ctrl 1 is low, MUX selects the data entry and high, MUX selects the return data. First, the padded message is copied directly to Reg A, which has already been initialized to zero and the bit is leading to the Compression Box (C-Box) team. This is essentially the execution of the compression function of SHA-3 algorithm, comprising the steps of theta (Θ), rho (ρ), pi (π), chi (χ) and iota (i). For executions, we logically prefer to achieve our design by applying the steps rho (ρ), pi (π) and chi (χ) in one step. As a result, there are savings in material resources in the form of 48 tranches. After 24 repetitions, the last output is passed to Reg B for storage to synchronize the data route. The last component of the architecture is the interconnected component if the measurement is inverse to the output bit and then truncated according to the required hash length.

4.1. Implementation of the compressed box

4.1.1. Theta-step (Θ)

The theta-step consists of three main steps in the form of comparisons where XOR must work mainly on bits. The XOR (1) occupies between the 64-bit path in each row for each track of each line, depending on the other, so that parallel processing on the track can be applied. We used the XOR 64-bit XOR parallel operator between the traditional in every five-track parameter to reach status "A" and the result is stored in the intermediate register. The parallel XOR operation above makes our designs faster and more efficient. Step (2) a heavier rotational frame with a rotating single bit that accompanies a simple reconnection or replacement of the small pattern of each line, and then XOR previous drain paths. The results are stored in an interim program in the form of five courses. Return the line in XOR to the state matrix input A [x, y] to reshape an array of 5 x 5 states A [x, y]. Each operation is performed on module 5.

4.1.2. Integrated Rho (ρ), Pi (π) and chi (χ)

Requires Rho principle and pi permutation and each path used to make a cyclic variation of a number by the cyclical shift of fixed offset in SHA -3 FIPS PUB 202 If this phase is applied separately, we have 48 extra layers for Pi Rho and exploitation to store the disk, we do not have to create logical steps with Chi Rho and Pi to make our designs. We have all the necessary calculations for the Rho and Hand phases and we have set up the circular change on each status matrix path obtained with theta-step output and move it after calculating its new position according to (4). For example, for X=1, and Y=2, (4) can be written as :

$$B[2,2(1) + 3(2)] = ROT(A[1,2], r[1,2]) \quad (7)$$

The value of r (1,2) is 10 corresponding to the cyclic displacement offset table in FIPS PUB 202 and all operations are modulo 5, therefore (7) is converted to (8).

$$B[2,3] = ROT(A[1,2], 10) \quad (8)$$

In this way, we have calculated for all possible combinations of state matrices and placed these values directly on (5) chi-steps. For X=1 and Y=3, (5) can be displayed as shown by (9).

$$A[1,3] = B[1,3] \oplus ((\text{NOT } B[2,3]) \text{ AND } B[3,3]) \quad (9)$$

$$A[1,3] = \text{ROT}(A[0,1],36) \oplus ((\text{NOT ROT}(A[1,2],10)) \text{ AND ROT}(A[2,3],15)) \quad (10)$$

We have done all cyclical rotations manually and placed the result on (10) and then applied to the XOR, NOT and AND operations. The above logical optimization techniques allow us to minimize resources by 16%, resulting in better performance of our designs. The diagram view of the integrated Rho, Pi and Chi steps is shown in Figure 3. This shows the implementation of integrated rho, pi and chi steps, it is easy to see that the Rho and Pi permutations are directly applied during the implementation of the Chi step with simple wiring.

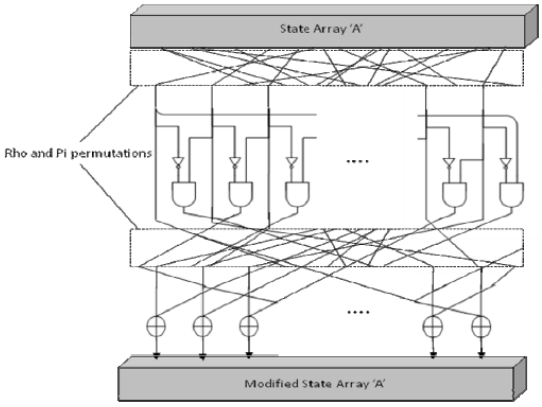

Figure 3. Integration of Rho, pie, chi

4.1.3. Iota (i)

In the Iota (i) step we used a conventional XOR 64-bit operator to perform XORing between the least significant 64-bit status array and the constant RC. The rotation constant is fixed and different for each round. This rotation constant is stored in the register.

4.2. Simulation and comparison of results

The projects have been implemented and verified on ISI Xilinx Design Suite, System Edition 14.6. The overall implementation was designed and executed on Xirtex Virtex 5 (xc5vlx330t-2ff1738). Each phase of SHA-3 is implemented as a module. These modules are included in the main code of our project to examine the results in detail. The output outputs and design performance verified by SHA-3 in the Pub 202 [3] state define the predefined test vectors defined for testing and validating the Eq of SHA-3 output standards. Table 2 shows the results of the implementation of the 3-SHA hash core in terms of area, frequency, flow rate (TP) and zone circumference (ASP) in the form of reductions. Their operating frequency is the highest reached 301.02 MHz which has an output of 7.224 Gbit/s of 240 CLB (1%) and 24 clock cycles required to achieve the final hash value.

We compared the recommended design results with a previous SHA-3 FPGA hardware design in the open literature in terms of area, frequency, processing capacity (TP) and disc feedthrough (TPS) in Table 2. Our main goal: using minimal flooring in our design with TP was enough to get the best GST. There is always an exchange between the use of the area and TP, so we have made another TPS a parameter to evaluate the effectiveness of the projects. The design is described in S. Kerckhof et al. The minimum number of territorial sources and the number of cycles of 2154 hours uses the final hash value, making it less TP than other projects [6]. Moreover, this design is TPS .425, which is inferior to the other results in the table [7]. Better than previous projects, but the material resources are much larger [8]. The GST of these designs is 2.99 and 0.879 respectively, which is very low compared to our design [11]. Shows the best TPS of 5.27 and 5.37, which is still low compared to our closure project because these projects use many disks [12]. The above comparison shows that our design is better than previous FPGA implementations for TP and ASP because our design offers the best ASP of 30.1.

Table 2. Results comparison with Keccak-512 model

Implementation	Devices	Area	Frequency (MHz)	TP (Gbps)	TPS
Our Design	Virtex-5	240	301.02	7.224	30.1
S. Kerckhof et al.	V6	188	285	0.08	0.425
A. Akin et al.	V4	2024	143	6.07	2.99
Kris Gaj et al.	V5	1229	238.4	1.0805	0.879
G. Provelengios	V5	2573	285	5.70	2.21
K. Latif et al.	V5	1197	263.16	6.32	5.27
E. Hom. et al.	V5	1220	-	6.56	5.37

5. CONCLUSION

In this work, we will present the SHA3-512 hardware editing project. We have maintained zone-to-flow negotiations, with our design showing the best possible results in terms of time and time compared to previously reported results. Our logical optimization combines the three transformations, rho, pie and chi transformation studies the parallel maximum in algorithms. This reform is usually reduced by the total LATAR, which significantly improves the performance of the system.

REFERENCES

[1] X. Wang, D. Feng, X. Lai, and H. Yu, "Collisions for hash functions md4, md5, haval-128 and ripemd," IACR, August 2004.
[2] National Institute of Standards and Technology (nist), "Cryptographic hash algorithm competition," 2007.
[3] FIPS-202, "Federal information processing standards publication fips- 202, secure hash algorithm-3 (sha-3)," 2014.
[4] Xilinx, "Virtex 2.5 V field programmable gate arrays".
[5] F.R. Henrquez, A.D. Prez, N.A. Saqib, and C.K. Koc, Cryptographic Algorithms on Reconfigurable Hardware. Signals and Communication Technology, Springer, 2007.
[6] S. Kerckhof, F. Durvaux, N.V. Charvillon, F. Regazzoni, G.M. de Dormale, and F.X. Standaert, "compact FPGA implementations of the five sha-3 finalists," Springer Berlin Heidelberg, vol. 7079, pp. 217–233, 2011.
[7] Akin, A. Aysu, O.C. Ulusel, E. Savas, "Efficient hardware implementations of high throughput sha-3 candidates keccak, luffa and blue mid night wish for single- and multi-message hashing," ACM, pp. 168–177, 2010.
[8] K. Gaj, E. Homsirikamol, and M. Rogawski, "Comprehensive comparison of hardware performance of fourteen round 2 sha-3 candidates with 512-bit outputs using field programmable gate arrays," 2nd SHA-3 *Candidate Conference*, pp 23-24, August 2010.
[9] B. Baldwin, N. Hanley, M. Hamilton, L. Lu, A. Byrne, M. ONeill, and W.P. Marnane, "FPGA implementations of the round two sha-3 candidates," The second SHA-3 Candidate Conference, 2010.
[10] G. Provelengios, P. Kitsos, N. Sklavos, and C. Koulamas, "FPGA-based design approaches of keccak hash function," 15th *Euromicro Conference*, pp. 648–653, 2012.
[11] K. Latif, M.M. Rao, A. Aziz, and A. Mahboob, "efficient hardware implementations and hardware performance evaluation of sha-3 finalists," in Proceeding of 3rd SHA-3 *Candidate Conference*, march 2012.
[12] E. Homsirikamol, M. Rogawski, and K. Gaj, "comparing hardware performance of round 3 sha-3 candidates using multiple hardware architectures in xilinx and altera fpgas," ECRYPT II Hash Workshop, pp. 1–15, 19-20 May 2011.