Design of 512 bitKeccak Hash Function Encryption for High Throughput Core

[1] Ms. Sheetal C. Deshmukh, [2] Ms.ApekshaV. Sakhare

Department of Computer Science and Engineering, G.H.Raisoni College of Engineering
Nagpur, Maharashtra, India.

Department of Computer Science and Engineering, G.H.Raisoni College of Engineering
Nagpur,Maharashtra,India

Abstract - Security has become a curial aspect in the design and use of computer system and network. Hash functions are used for many applications in cryptography mainly in digital signatures and message authentication code and in network security. Keccak hash function has been submitted to SHA-3 competition. In this paper has implement "SHA-3 512 bit" hash function and high throughput cone designed to work in high clock frequency dedicated to ASIC or expensive FPGA (virtex-7). The maximum clock frequency supported by the design 267.2 MHz.

Keywords- **FPGA, cryptography, hash functions, Xilinx IES 13.1.**

1. Introduction

Cryptography is used to adders many security issues the most pertinent of which are integrity, cryptography encompass, confidentiality and authentication proposes. Security has become a crucial aspect in the design and use of computer system and network. Cryptanalysis entails the analysis and evaluation of cryptographic algorithms including protocol and primitives. The important of hash function. In cryptographic protocols the modern cryptography is clearly proven by different application and multi-purposes.

Nowadays, the security is more important in all area. SHA-3(secure hash algorithm) is performed using various hash function.[1] In this some hash function are design and implemented on the FPGA. There are five hash algorithms and keccak is one of them. In this paper the total internal architecture of secure hash algorithm-3 using keccak hash function for 512 bit encryption are implemented. This total architecture is implemented on the FPGA.

Cryptography does nothing on its own. It is a basic and vital ingredient of any security architecture, but it is nothing more than that. Cryptography needs to be used in particular ways, it needs to be combined with other technologies, it needs to be implemented properly and it needs to be supported by the appropriate managerial processes. If any one of these aspects is deficient then it is quite likely that using cryptography does not bring the security guarantees that are being sought.vCryptographic secure hash function consists of many applications. First is digital signature are the first application of cryptographic secured hash function. Massage authentication code (MAC) is the second application, in digital signature schemes the hash function serve a dual role. They expand the domain of messages that can be signed by sachem and they are an essential element of the schemes security. The third a common method of client authentication is to require the client to present a password previously registered with the server.

Storing password of all the users on the server poses an obvious security risk. Although the server need not to know the passwords. It may store their hashes (together with some salt to frustrate dictionary attacks) and use the information to match it with the hashes of alleged passwords. Efficiency on the hardware implementation is one of the important criteria for the hash function selection. The implementation is categorized in two parts.[2] That is FPGA and standard cell ASIC implementation. Implementation on FPGA it is desirable to compare implementation on the same target device or on device of the same FPGA family.But the minimal gate lengths are required for ASIC implementation .Three different hardware implementations are used for comparison in many modules and different application.

- Autonomous implementation
- Implementation with external memory
- Implementation of core functionality

The field-programmable gate array (FPGA) is a semiconductor device that can be programmed after manufacturing. Instead of being restricted to any predetermined hardware function, an FPGA allows you to program product features and functions, adapt to new standards, and reconfigure hardware for specific applications even after the product has been installed in the field—hence the name "field-programmable". You can use an FPGA to implement any logical function that an application-specific integrated circuit (ASIC) could perform, but the ability to update the functionality after shipping offers advantages for many applications.Unlike previous generation FPGAs using I/Os with programmable logic and interconnects, today's FPGAs consist of various mixes of configurable embedded SRAM, high-speed transceivers, high-speed I/Os, logic blocks, and routing. Specifically, an FPGA contains programmable logic components called logic elements (LEs) and a hierarchy of reconfigurable interconnects that allow the LEs to be physically connected. You can configure LEs to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, the logic blocks also include memory elements, which may be simple flipflops or more complete blocks of memory.

1.1 Stream Ciphers

Stream ciphers are cryptographic algorithms that transform a stream of plaintext messages of varying bit-length into cipher text of the same length, usually by generating a key stream that is then XORed with the plaintext. Using a shared secret key, stream ciphers can be used to provide confidentiality, i.e., restrict access to secret data to the parties in possession of the key by encrypting the plaintext secret data. In general, stream ciphers have very strong security properties, use few resources and high throughput thus making them ideal for mobile applications; well-known examples of stream ciphers include the RC4 cipher used in 802.11 Wireless Encryption Protocol, E0 cipher used in Bluetooth protocol, and the SNOW 3G cipher used by the 3GPP group in the new mobile cellular standard.

1.2 Hash Functions

Like hash functions, stream ciphers are important cryptographic primitives. However, hash functions transform arbitrary-length input messages into fixed-length message digests. They are used in many applications in commitment schemes, digital signatures and message authentication codes. To this end they are required to satisfy different security properties. These security properties include.

i) Preimage resistance, i.e., given f (x) it is infeasible to find x,
ii) Second preimage resistance, i.e., given x it is infeasible to find x1 _ x : f (x) = f (x1), and
iii) collision resistance, i.e., it is infeasible to find x, x1 : x1 _ x and f (x) = f (x1).Informally, a hash function is collision resistant if it is practically infeasible to find two distinct messages m1 and m2
iv) That produces the same message digest.

2. Objective

In previous design BLAKE hash function, JH hash function, Skein Hash function and Ghrostl hash function is implemented. All these hash function required more rounding for encryption [4].Our objective is to provide more encryption using less number of rounding i.e more permutation has to done, in 5 candidates of SHA-3 finalist. Keccak hash function provides more encryption in less rounding i.e for 512 bit encryption 10 rounds will be required. To design and implement SHA-3 algorithm using keccak hash function. Cryptography has to be done for 512 bits. Design has to be done using HDL coding which has to support high through put core.

Main objective is to design SHA-3 algorithm done by candidate keccak for 512 bit cryptography. Verilog coding for proposed design is done in Xilinx ISE tool. Optimization Target is one of the most important decisions to make in order to develop a fair comparison. The possible choices include Maximum Throughput, Minimum Area, Maximum Throughput to Area Ratio, Minimum Latency, etc. All of the aforementioned targets can be used to make a comparison. Out of them, we have selected Maximum throughput to Area Ratio as our criteria of choice.

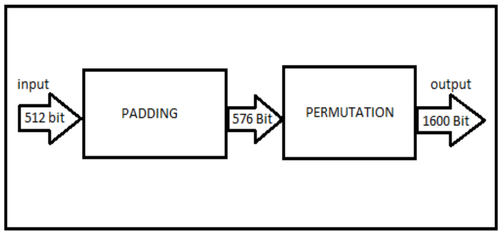


Fig 1.Whole core diagram of the system

The winning algorithm, Keccak (pronounced “catch-ack”), was created by Guido Bertoni, Joan Daemen and Gilles Van Assche of STMicroelectronics and MichaëlPeeters of NXP Semiconductors. The team’s entry beat out 63 other submissions that NIST received after its open call for candidate algorithms in 2007, when it was thought that SHA-2, the standard secure hash algorithm, might be threatened. Keccak will now become NIST’s SHA-3 hash algorithm.

3. Methodology

3.1 Architecture of Core

Hash algorithms are used widely for cryptographic applications that ensure the authenticity of digital documents, such as digital signatures and message authentication codes. These algorithms take an electronic file and generate a short "digest," a sort of digital fingerprint of the content In this selection has advantage over other possible choices. First, it is practical, as hardware cores are typically applied in situations. Where the size of the processed data is significant and the speed of processing is essential. Secondly, throughout the entire design process this optimization criterion is a very reliable guide.[7] At every junction where the decisions must be made, it starting from the choice of high-level hardware architecture and down to the choice of the particular FPGA tool options, this criterion facilitates the decision process, leaving very few possible paths for further investigation. Each submitted algorithm as compared to other submission (of the same hash length), including first and second preimage resistance to generic attacks.

Also, if other security factors raised by the public comments during the evaluation process, including attacks which demonstrate that the actual security of the algorithm is less than the strength claimed by the submitter. This key note talk deal with the evaluation of computation efficiency. Computational efficiency essentially refers to the throughput of an implementation of the software. The memory required for hardware and software implementations will be considered during the evaluation process. Memory requirements will include factor such as gate count for hardware implementation and RAM requirement and code size for software implantation. Algorithms with greater flexibility that meet the need of more users are preferable. For example "flexibility" includes the algorithm in order to achieve and efficiency algorithm implementation for wide variety of platform. Including constrained environment such as smart cards. The algorithms will be judged according to relative simplicity of design.

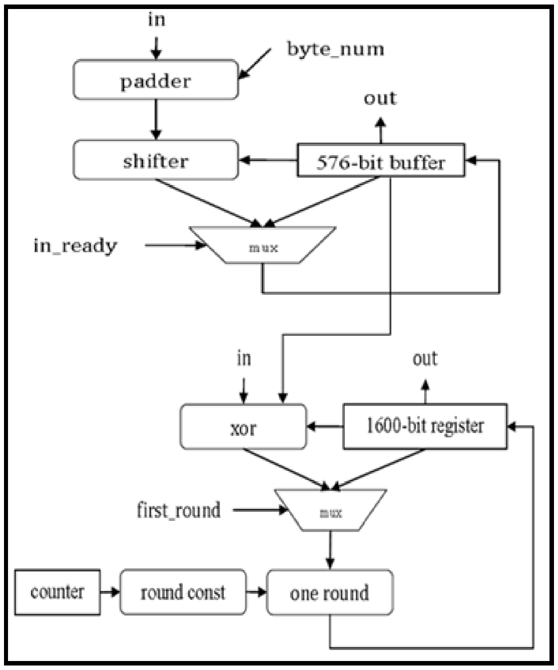


Fig 2 Architecture of the padding and permutation module

The width of the user input is far less than 576 bit. So the padding module uses a buffer to assemble the user input. If the buffer grows full, the padding module notices the permutation module its output is valid. Then the permutation module begins calculation, the buffer cleared, the padding module waiting for input simultaneously. In the high throughput core, two rounds are done per clock cycle. The round constant module is implemented by combinational logic, saving resource than block RAM, because most bits of the round constant is zero.

- **FPGAs** are selected as primary implementation platform.
- **Uniform input/output** interface and protocol is used in implementations of all of the SHA-3 candidates.
- **Optimization** target is throughput to area ratio.
- **The same basic building blocks** are used in implementations of all candidates, by reusing the same source codes for low level operations. This approach rule out the possible inconsistencies in optimizations of basic logic operations, possible if these operations were implemented separately for each candidate.
- **Hardware description** language is verilog. Different languages may have different level of optimization capability. Using the same language ensures that a design will not get a better result simply because of the different treatment of divergent languages by the current generation of CAD tool.
- **CAD tools** selected are tools developed by the FPGA vendors :{ Xilinx: Xilinx ISE Design Suite v. 13.1}

4. Simulation

These are the final result of keccak hash function. RTL view of keccak hash function and waveform are given below.

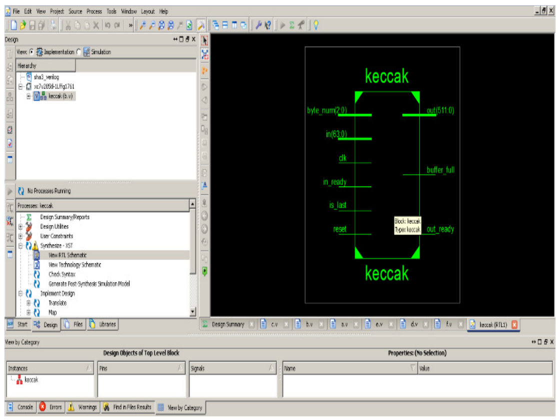
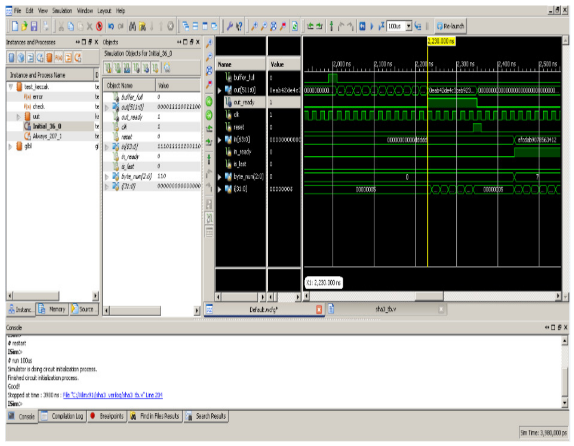


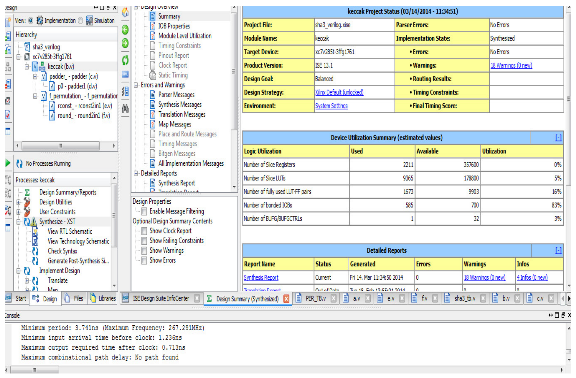
Fig 3 RTL View of Keccak Hash Function



```
# restart
ISim>
# run 100us
Simulator is doing circuit initialization process.
Finished circuit initialization process.
Good!
Stopped at time : 3980 ns : File "C:/Xilinx91i/sha3_verilog/sha3_tb.v" Line 204
ISim>
```

Fig 4 Waveform of Keccak Hash Function

The give table shows that how many registers LUT'S and memory are required for design of architecture. For the 512 bit encryption process 2236 registers and 9365 LUT'S are used.



keccak Project Status (03/14/2014 - 11:34:51)

Project File:	sha3_verilog.xise	Parser Errors:	No Errors
Module Name:	keccak	Implementation State:	Synthesized
Target Device:	xc7v285t-3ffg1761	• Errors:	No Errors
Product Version:	ISE 13.1	• Warnings:	18 Warnings (0 new)
Design Goal:	Balanced	• Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	
Environment:	System Settings	• Final Timing Score:	

Device Utilization Summary (estimated values)

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	2211	357600	0%
Number of Slice LUTs	9365	178800	5%
Number of fully used LUT-FF pairs	1673	9903	16%
Number of bonded IOBs	585	700	83%
Number of BUFG/BUFGCTRLs	1	32	3%

Detailed Reports

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Fri 14. Mar 11:34:50 2014	0	18 Warnings (0 new)	4 Infos (0 new)

```
Minimum period: 3.741ns (Maximum Frequency: 267.291MHz)
Minimum input arrival time before clock: 1.236ns
Maximum output required time after clock: 0.713ns
Maximum combinational path delay: No path found
```

Fig 5 Synthesis Result

5. Conclusion

Keccak hash function hardware implementations are described in this paper. For implementation of this function use virtex-7 FPGA device in this paper provide more encryption using less no of rounding i.e more permutation has to done, in 5 candidates of SHA-3 finalist. Keccak hash function provides more encryption in less rounding i.e for 512 bit encryption 10 rounds will be required to design and implement SHA-3 algorithm using keccak hash function.

References

[1] National Institute of Standard and Technology (NIST), "Cryptographic hash algorithm competition", 2007, available on lineathttp://www.nist.gov/itl/csd/ct/hash_competition.cfm

[2] FatmaKahri, BelgacemBouallegue, Mohsen Machhout and RachedTourki Electronics and Micro-Electronics Laboratory "An FPGA implementation of the SHA-3: The BLAKE Hash Function"(2012) (E. μ. E. L) Faculty of Sciences of Monastir, Tunisia Kahri-fatma@gmail.com

[3] A. H. Namin& M. A. Hasan (2010 a), Implementation of the Compression Function for Selected SHA-3 Candidates on FPGA. Retrieved Feb. 25th, 2010, from University ofWaterloo, Department of Electrical and Computer Engineering.

[4] Schorr (2010),Performance Analysis of a Scalable Hardware FPGA Skein Implementation. Retrieved February 2010, from Kate Gleason College of Engineering Department of Computer Engineering Rochester ,New York.

[5] J. Elbirt (2009), Understanding and Applying Cryptograph and Data Security. Book ISBN 978-1-4200-6160-4 (alk.paper).

[6] Regenscheid, R. Perlner, S. Chang, J. Kelsey, M. Nandi, & S. Paul (2009), Status Report on the First Round of the SHA-3 Cryptographic Hash Algorithm Competition. Retrieved September 2009, from National Institute of Standards and Technology, U.S. Department of Commerce.

[7] C. Rechberger (2010), Second-Preimage Analysis of Reduced SHA-1, from KatholiekeUniversiteit Leuven, Department of Electrical Engineering.

[8] ImadFakhriAlshaikhli, Mohammad A. Ahmad, Hanady Mohammad Ahmad (2012). "Protection of the Texts Using Base64 and MD5."JACSTRVol 2, No 1 (2012)(1): 12.

[9] E. Andreeva, B. Mennink, B. Preneel& M. Skrobot (2012), Security Analysis and Comparison of the SHA-3 Finalists BLAKE, Grostl, JH, Keccak, and Skein. fromKatholiekeUniversiteit Leuven.

[10] Belgium. E. B. Kavun& T. Yalcin (2012), On the Suitability of SHA-3 Finalists for Lightweight Applications.from Horst Görtz Institute, Ruhr University, Chair of Embedded Security, Germany

[11] Brian Philofsky, "HDL Coding and design practices for improving Virtex-5 utilization, performance, and power", XcellJounal, Issue 59, Four Quarter 2006, Xilinx.

[12] Guido Bertoni, Joan Daemen, MichaëlPeeters and Gilles Van Assche."Keccak sponge function family main document", version 1.2, April 2009, available on line at http://keccak.noekeon.org/.