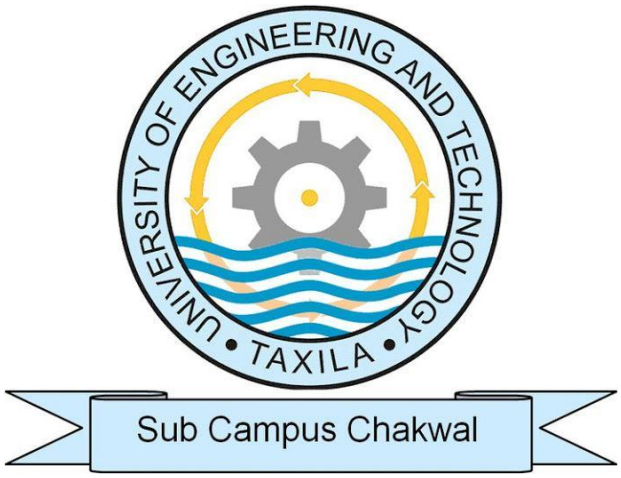
SHA-256 Cryptographic Processor based on FPGA

Shafiq Ahmad, Amjad Hussain

University of Engineering and Technology Taxila sub-campus Chakwal



Project Supervisor

Prof. Kamran Javed

Abstract

The Field Programmable Gate Array (FPGA) has a great advantage in use for Cryptography. We implemented a Cryptographic Processor using Xilinx Spartan III XSA-S1000 FPGA device. Cryptographic Processor design is able to keep up hardware's batter speed in computations the algorithm this processor required. Our developed Cryptographic Processor is approximately 20 folds faster than the dual core processer by Intel. This Cryptographic Processor can be used as Data Authenticator and many other software related security applications.

Keywords: 1_Block: 512 bits, Padding: Including extra data after original one, Parsing: Dividing data into 1_Block

FPGA sixth semester BSc Electronics Project

SHA-256 Cryptographic Processor based on FPGA

Introduction

The use of a FPGA has a vital advantages in the performance of Cryptographic Processor. As compared to Application Specific Integrated Circuits (ASIC), FPGA offer way more flexibility in usage of Cryptography including the following reasons:

1) FPGA can be reconfigured on site, so it is far more effortless to work with than ASIC.

2) After the release if new updates come by vendor, use can update his device to meet the latest requirements.

3) Cryptographic Processor can be implemented on FPGA with better optimization of size and time scales with the help of software like Xilinx.

4) Cryptographic Processor implemented on FPGA can be less complicated in Algorithm.

5) FPGA based Cryptographic Processor have sequential type hardware providing it way better speed performance than any software based application.

6) FPGA based Cryptographic Processor have much lower cost and easy availability than ASIC.

System Architecture

We used Verilog to Implement SHA-256 Hardware Algorithm on Xilinx 14.01 with the Simulation help of ModelSim 6.7. System takes input from a UART 8-bits in one cycle and then stores it in 1 Byte Addressable Memory. SHA-256 Processor waits for the "byte_stop" Signal from the UART to confirm that all the input data from the user has been received by UART and stored in Memory of SHA-256.

1. **Padding and Parsing:**

   First Module in SHA-256 is Padding and Parser Module. This module includes the extra data after the original data has stopped to make total data for SHA-256 's next modules a 1_Block (512bits Block) and then the Parser in this module stores all of this data after adding a 64-bit chunk which represents the the Size of the Input data. This module also issues the "padding_done" to signal the other modules that padding and parsing has been done.

2. **Message Scheduler**

   In message scheduler we implement the 0-15 iterations for 32bit 'w' words which is all the 1_Block data from padding (32*16=512 bits). After 16 'w' words we perform iterations from already received 16 words to create further 48 'w' words.

For total 64 iterations:

Message schedule, {Wt}:

for $0 \leqslant t \leqslant 15$ $\quad Wt = Mt$

for $16 \leqslant t \leqslant 63$ $\quad Wt = \sigma (Wt-2) + Wt-7 + \sigma (Wt-15) + Wt-16$

3. Iterative Processing

In iterative processing we create 8 registers each 32bit

a, b, c, d, e, f, g, h

with initial values equal to 8 intermediate initial values

H0 = 32'h6a09e667

H1 = 32'hbb67ae85

H2 = 32'b3c6ef372

H3 = 32'ha54ff53a

H4 = 32'h510e527f

H5 = 32'h9b05688c

H6 = 32'h1f83d9ab

H7 = 32'h5be0cd19

and perform following compression functions:

for all 64 iterations:

semation_1 <= (e ROTR 6) ^ (e ROTR 11) ^ (e ROTR 25)

ch <= (e and f) ^ ((~ e) and g)

T1 <= h +semation_1 + ch + k[n] + w[n]

Semation_0 = (a ROTR 2) ^ (a ROTR 13) ^ (a ROTR 22)

maj = (a and b) ^ (a and c) ^ (b and c)

T2 = semation_0 + maj

h <= g g <= f f <= e e <= d + T1 d <= c c <= b b <= a a <= T1 + T2

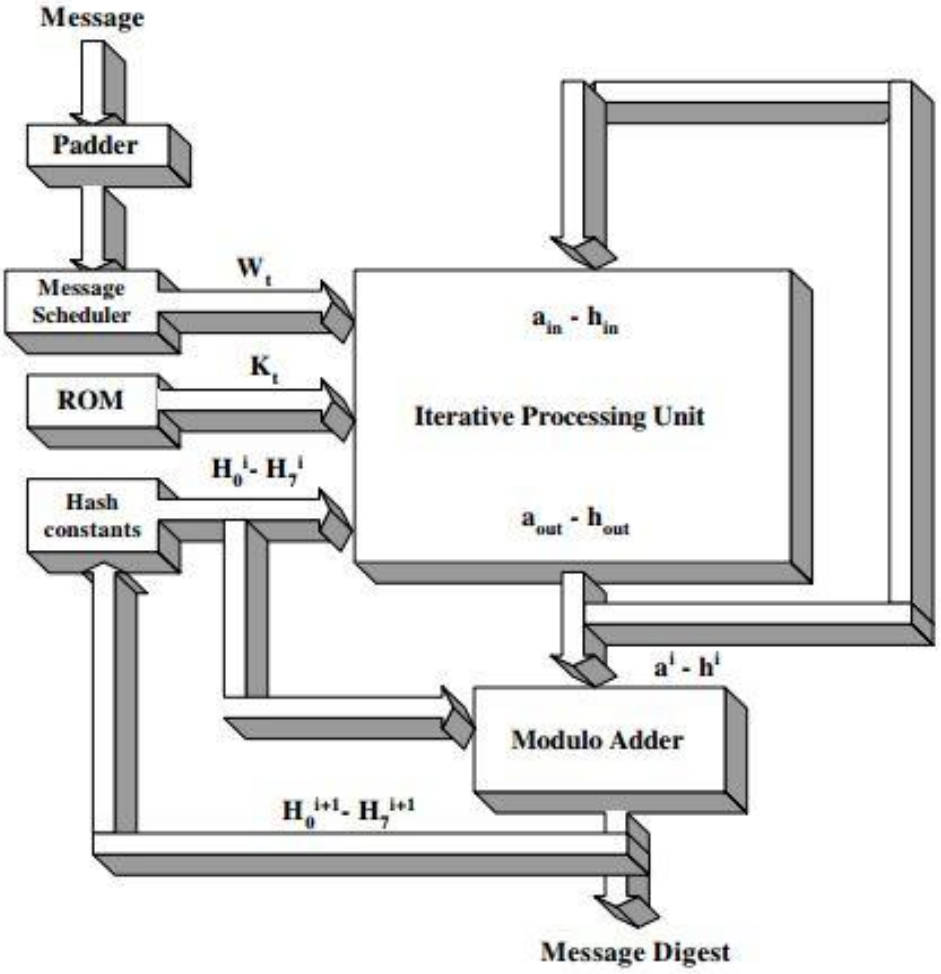
Where K's values are pre-defined by FIPS (Federal Information & Processing Standards)

4. Message Digest

In message digest we perform addition of Initial values of "H" and values of 8 iteration registers for 1_Block.

After addition we concatenate values of 8 'H' each 32bit giving the out of 256 bits Hash

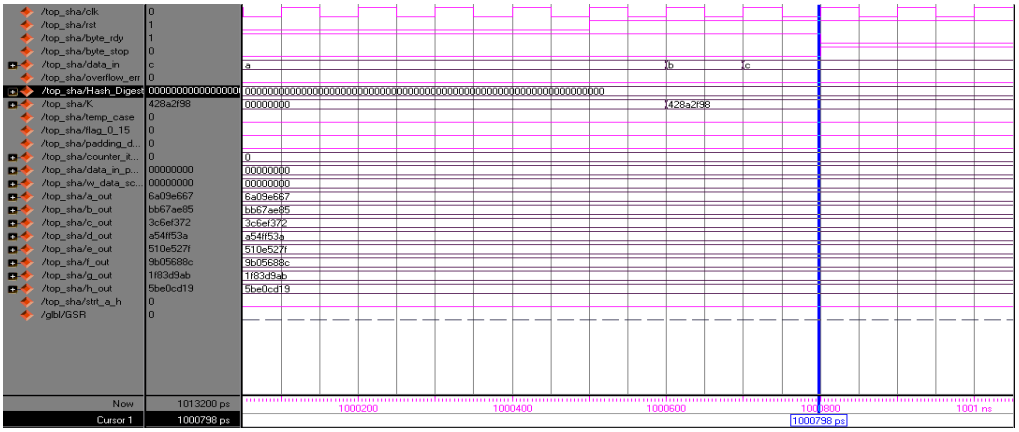
Block Diagram

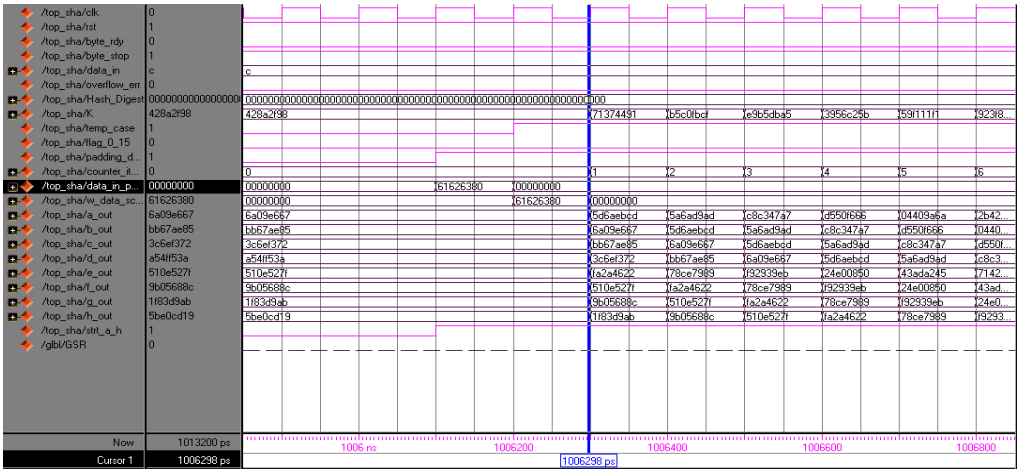


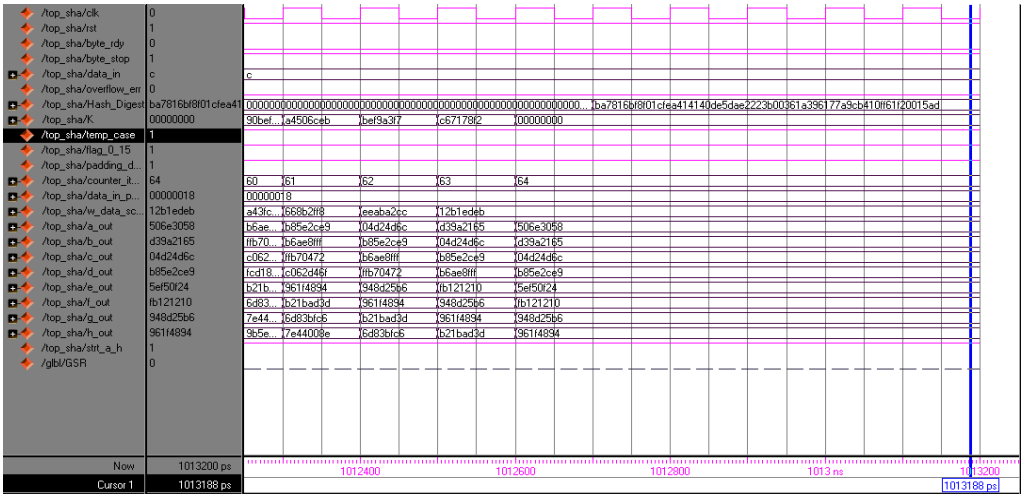
Simulation Results

Simulation results of input “abc” on ModelSim with output hash

“ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad”







Tables

Comparison of Different FPGA Devices

Column Head	Spartan III	Virtex 4	Virtex 5
Max Frequency	81.693Mhz	156.519Mhz	184.468Mhz
Cycles	68	68	68
Throughput	615Mbps	1178Mbps	1388Mbps
Efficiency	0.6997	1.3899	1.3995

Note: This comparison was based on the original information provided by the vendor of these devices

Applications

In Applications of SHA-256 there is a wide range of fields. Mostly it is used in following Digital Security Applications:

1. Website Credential Security
2. Software File Authentication
3. Data Transmission Authentication
4. Secure Code Transmission
5. Proof of Real Work in Digital Currencies

Conclusion

The Implementation of FPGA based Cryptographer is much faster than the Software based Cryptographer which works on general Processor. Now a day when there is a big race of fast cryptographers due to the Cryptography of Digital Currency like **Bitcoin** which is based on SHA-256. FPGA based Cryptographers and Design provides an upper edge. Our design ensures that user gets cheap and fast hardware for this kind of applications. The Importance of SHA-256 can be determined by the fact that it is required to use by Law of United States in some Government Secure Applications.

National Institute of Standards and Technology (NIST) states:

"Federal agencies should stop using SHA-1 for...applications that require collision resistance as soon as practical, and must use the SHA-2 family of hash functions for these applications after 2010"

Acknowledgment

Our work on this Cryptographer was supported by the resources of University of Engineering and Technology Taxila and personal supervision on Prof. Kamran Javed.

References

[1] FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION (FIPS PUB 180-4) http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[2] National Institute of Standards and Technology (NIST) http://www.nic.funet.fi/pub/crypt/cryptography/symmetric/aes/nist/Rijndael.pdf

[3] Chanjuan Li, Qingguo Zhou, Yuli Liu, Qi Yao (Cost-efficient Data Cryptographic Engine Based on FPGA) http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5992044

[4] Wikipedia SHA-2 sub-topic SHA256 https://en.wikipedia.org/wiki/SHA-2