Resource-Efficient Implementation of Blue Midnight Wish-256 Hash Function on Xilinx FPGA Platform

Mohamed El Hadedy[1,2], Martin Margala [2], Danilo Gligoroski[3] and Svein J. Knapskog [1]

[1]The Norwegian Center of Excellence for Quantifiable Quality of Service in Communication Systems(Q2S),
Norwegian University of Science and Technology (NTNU),
O.S.Bragstads plass 2E, N-7491 Trondheim, Norway
mohamed.elhadedy@q2s.ntnu.no, Knapskog@q2s.ntnu.no

[2] Department of Electrical and Computer Engineering, University of Massachusetts Lowell,
Ball 301, One University Ave, Lowell, MA 01854, USA
Mohamed_Aly@uml.edu, Martin_Margala@uml.edu

[3]Department of Telematics, Faculty of Information Technology, Mathematics and Electrical Engineering,
The Norwegian University of Science and Technology (NTNU),
O.S.Bragstads plass 2E, N-7491 Trondheim, Norway
danilog@item.ntnu.no

Abstract

This paper presents the design and analysis of an area efficient implementation of the SHA-3 candidate Blue Midnight Wish (BMW-256) hash function with digest size of 256 bits on an FPGA platform. Our architecture is based on a 32 bit data-path. A fully autonomous implementation of BMW on Xilinx Virtex-5 FPGA requires 84 slices and two blocks of memory: one memory block to store the intermediate values and hash constants and the other memory block to store the instruction controls. The proposed implementation achieves a throughput of 28 Mpbs.

I. Introduction

To obtain efficient and secure computerized information handling, hash functions are used in countless protocols and algorithms. Until now, two generations of SHA algorithms have been standardized and widely deployed - SHA-1, and SHA-2, and although they have some similarities, they have also significant differences [1]. SHA-1 is the most frequently used member of the SHA hash family, employed in hundreds of different applications and protocols. However, in 2005, we witnessed a significant theoretical breakthrough in breaking the current cryptographic standard SHA-1 [2]. The discovered mathematical weaknesses which were shown to exist indicated the need for replacement with a stronger hash function [3], although there exist another family of standardized hash function called SHA-2 which officially replaced SHA-1 in 2010.

The SHA-2 family is a family of four algorithms that differ from each other by different digest size, different initial values and different word size. The digest sizes are: 224, 256, 384 and 512 bits. Although no attacks have yet been reported on the SHA-2 variants, their operational performance is in many settings less than desirable, and the National Institute of Standards and Technology (NIST) have felt the need for an improved new family of hash functions [4]. At the end of 2007, NIST decided to invite cryptographic algorithms designers and developers to participate in an open competition running between 2008 and 2012 for choosing a new candidate for the next cryptographic hash standard SHA-3. This work is now well underway, as the competition is about to enter into its third phase, in which five of the strongest candidates will be singled out for the final testing until a winner may be declared in 2012. The Blue Midnight Wish (BMW) hash function is one of the candidates promoted to the second round of the SHA-3 competition and implemented in software, it is one of the fastest proposed new designs running in the competition [5]. In this paper, we proposed a hardware design of BMW-256 which is simple, area efficient and provides significant throughput improvements over previous work. The proposed BMW-256 hash function core is implemented in FPGA using Virtex XCV300 and Virtex 5 XC5VLX110 devices.

The rest of the paper is organized as follows. In Section 2, we describe briefly the compression function of the second round version of the BMW-256 algorithm, while Section 3 contains the architectural description of the

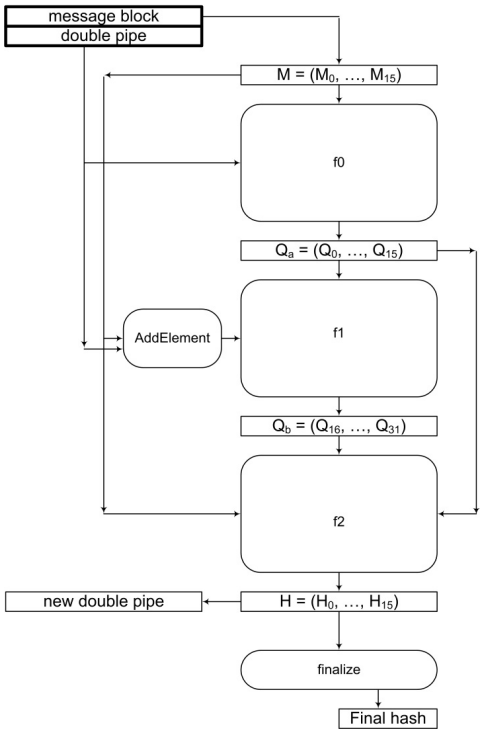

Fig. 1: Graphical representation of the hash function Blue Midnight Wish

design. In Section 4, the BMW hashing operations are detailed out. In section 5, the synthesis results of the FPGA implementation are given and comparisons with other related works are shown. Finally, in section 6, our conclusions are presented, and some observations and possibilities for future work are discussed.

II. The Hash Function of BLUE MIDNIGHT WISH – 256

The BMW-256 has function is shown in Fig. 1. We refer to the variant that creates a 256 bit message digest as BMW-256. The basic data block which is used is 32 bits long. The algorithm has four different operations in the hash computation stage: bit-wise logical word XOR, word addition and subtraction, shifts (left or right), and rotate left. The BMW uses a double pipe design to increase the resistance against generic multi-collision attacks and length extension attacks. In the double pipe design, the sizes of the inputs to the compression function are twice the message digest size. The inputs to the compression function are the message blocks $M^{(i)}$ of size 512 bits, along with the initialization vector $H^{(i-1)}$ of 512 bits (previous double pipe) and the output is the current double pipe $H^{(i)}$.

The hash function has two main parts: 1. Message digesting part and 2. Finalization part as it is shown in Fig. 1. The first part uses three separate functions f_0, f_1 and f_2 to define the so called "compression function" of Blue Midnight Wish. The output of the compression function is $H^{(i)} = (H_0^{(i)}, H_1^{(i)}, ..., H_{15}^{(i)})$. There are two inputs for the function f_0: The first argument consists of sixteen 32-bit words, which are working as initial values $H_0^{(i-1)}$, $H_1^{(i-1)}$,..., $H_{15}^{(i-1)}$. The second argument consists of sixteen 32-bit words, which represent the input message block: $M_0^{(i)}$, $M_1^{(i)}$,..., $M_{15}^{(i)}$.

The function $f_0(M^{(i)}, H^{(i-1)})$ computes $M^{(i)} \oplus H^{(i-1)}$ and produces $Q_a^{(i)}$ as the first part of the extended (quadrupled) pipe, hence $Q_a^{(i)} = (Q_0^{(i)}, Q_1^{(i)}, ..., Q_{15}^{(i)})$. The inputs for the function f_1 are three different arguments, the message block $M^{(i)}$, the previous double-pipe $H^{(i-1)}$ and the value of $Q_a^{(i)}$. The function $f_1(M^{(i)}, H^{(i-1)}, Q_a^{(i)})$ computes the second part of the extended (quadrupled) pipe $Q_b^{(i)}$, hence $Q_b^{(i)} = (Q_{16}^{(i)}, Q_{17}^{(i)}, ..., Q_{31}^{(i)})$.

The third function f_2 also takes three arguments; the message block $M_{(i)}$ and the values of both $Q_a^{(i)}$ and $Q_b^{(i)}$. The function $f_2(M^{(i)}, Q_a^{(i)}, Q_b^{(i)})$ computes the new double-pipe value $H^{(i)}$, i.e. $H^{(i)} = (H_0^{(i)}, H_1^{(i)}, ..., H_{15}^{(i)})$.

The second part (finalization) contains of the same compression function defined in the message digesting part (so it uses the same functions f_0, f_1 and f_2), but instead of initial values $H_0^{(i-1)}$, $H_1^{(i-1)}$,..., $H_{15}^{(i-1)}$, it use $Constant_j^{final}$ = ($Constant_0^{final}$, $Constant_1^{final}$,... , $Constant_{15}^{final}$) values and the role that was played by the input message block

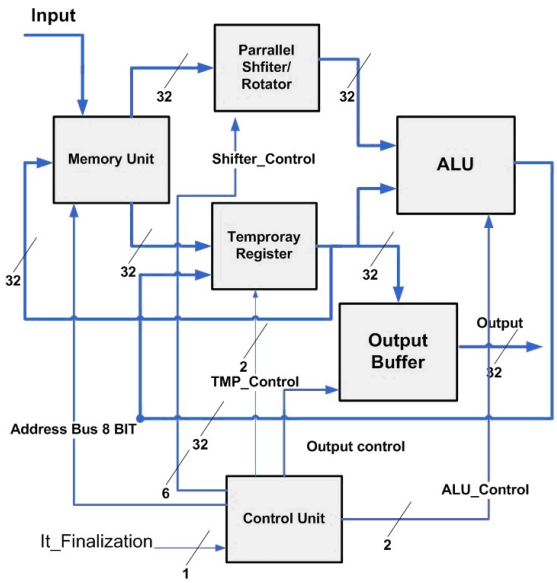

Fig. 2: BLUE MIDNIGHT WISH-256 Core Architecture

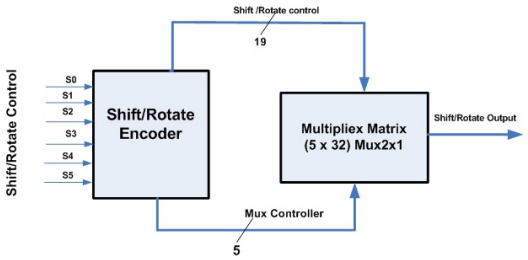

Fig. 3: Parallel Shifter/Rotator Block

in the previous message digesting part, now will be played by the last obtained double-pipe $H^{(i)} = (H_0^{(i)}, H_1^{(i)}, ..., H_{15}^{(i)})$.

III. Blue Midnight Wish256 Core Architecture

Fig. 2 shows the complete architecture of the entire BMW core process, which includes six main hardware operative parts, Memory unit, Parallel Shifter/Rotator, ALU (Arithmetic Logic Unit), Temporary Register, Output Buffer and Control Unit. Their operations are as follows:

Parallel Shifter/Rotator: It contains a 5 x 32 Mux matrix each one is a 2 x 1 multiplex with a large encoder (5 X 11). This component is responsible for the shift and rotation operations of the 32 bit words. It receives 32 bit parallel data from the memory Block and transmits 32 bit parallel data to the ALU. That happens dependent on the value of the shifter control word. Because we have 46 operations in the BMW hash core, the width of shifter control word is 6 control bits as shown in Fig. 3.

ALU : The ALU component offers three different operations in the hash computation stage: bit-wise logical word XOR, word addition and subtraction (modulo 2^{32}). The ALU component receives 32 bit data words from the Parallel Shifter/Rotator and the Temporary Register and transmit the output to the Temporary Register to work as a parallel accumulator.

Temporary Register: It contains a 32 Mux 2 x 1 and a shift register. The Temporary Register works as an accumulator. It receives 32 bit words from The Memory Unit and The ALU and transmits data 32 bit words to the ALU and the output stage.

Memory unit: To implement the BMW-256 core memory block, we used an FPGA block RAM of size 256 x 32 bits. As we mentioned in section 3.1, the memory block contains a ROM to store the BMW-256 constants K_j, J=0,1,..., 15 , $H^{(i-1)}$ and the $Constant_j^{final}$. In addition, the memory block contains sufficient RAM to store the BMW-256 input message blocks ($M_0^{(i)}$, $M_1^{(i)}$,..., $M_{15}^{(i)}$), the intermediate values of the BMW hash function, and the final double pipe values $H^{(i)} = (H_0^{(i)}, H_1^{(i)}, H_2^{(i)}, ..., H_{15}^{(i)})$.

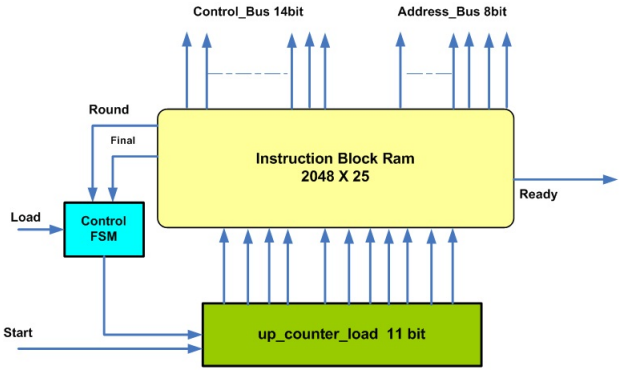

Fig. 4: BMW-256 Control Unit

Control Unit: It has been designed as a 2048 x 25 bit Instruction Block RAM, an 11 bit up_counter_load bit and a Control FSM (Finite State Machine) as shown in Fig. 4. it contains three operative parts, all of them working together to produce 8 bit memory address words to control the memory block traffic with the other BMW-256 sub-systems. The Control Unit produces the 14 bit control word to control the data flow between the BMW-256 core sub-systems. The Control Unit subsystems are working as follows:

once the Start and Load signals becomes high, the organization of the sixteen input messages inside RAM location is started. Subsequently, the Load signal becomes low and the Instruction Block RAM starts to control the BMW hashing core to execute the f_0, f_1, and f_2 according to the BMW-256 algorithm operations which was described in section 2. Finally, the Round signal becomes high, and BMW hashing core starts to transfer the $H_0^{(i)}$,$H_1^{(i)}$,...,$H_{15}^{(i)}$ values in the message locations and and transfer $Const^{final}$ vaules in the $H_0^{(i)}$,$H_1^{(i)}$,...,$H_{15}^{(i)}$ locations. After that the Final signal becomes high and the final hash output. The Control FSM is used to organize the movement of instructions from the up_counter_load according to the value of each of the signals Load, Round and Final.

IV. Blue Midnight Wish-256 hashing operations

In this section we describe how the computation hash core works to execute the internal functions in BMW-256. As an example, we will explain how to XOR two blocks of data present in locations number 4 and 5 in the Memory Unit, and write the result in location number 7. First, the Control Unit gives order to the Memory Unit to choose location number 4. Then the Control Unit asks the Temporary Register to pick up the data from the data bus and subsequently the same operation happens with location number 5. However, instead of using the Temporary Register, the Parallel Shifter/Rotator picks up the data. Now, the Control Unit asks the Shift/Rotate Encoder to give order to the ALU to add these data and store them in the Temporary Register. Finally, the Control Unit gives order to the Memory Unit to pick up the data and place them in location number 7. Because we used the Parallel Shift/Rotate, and the parallel Arithmetic Logic Unit which has an output size of 32 bit, we succeeded to reduce the number of cycles for each operation shown in Table I (page 5). Using the BMW-256 operations in Table I, we see that we can execute the function f_0 in 413 cycles, function f_1 in 476 cycles and finally function f_3 in 171 cycles.

V. Performance Evaluation

The BMW-256 core has been designed in VHDL and it was synthesized (synthesis, placement and routing) using ISE foundation 10.1 [7] in VIRTEX XCV300-6PQ240 and VIRTEX 5 XC5VLX110 Xilinx devices. In Table II, we compare this implementation optimized for small FPGAs with the previous similar implementation. By using the proposed structure we have spent around 96% less area compared to previous design for BMW-256 on the same FPGA VIRTEX 5 XC5VLX110 device while increasing the measured throughput around 27 times.

VI. Conclusion and Future Work

In this paper we have presented an FPGA implementation of a new BMW-256 hashing core structure with 256 bits of message digest using a parallel shifter/rotator and a parallel 32 bit word arithmetic logic unit (ALU). The BMW-256 core receives 16 message words of 32 bits and processes them. The goal was to use as small area as possible in order to minimize the hardware cost. For the future work, we will take on the challenge to improve this design. The goal is to improve the throughput while keeping the optimized the area usage. It will certainly be beneficial in some future usage scenarios to do a full implementation in ASIC.

TABLE I: BLUE MIDNIGHT WISH-256 hashing core operations (execution times)

Operation	Proposed	BMW-256[8]
Load	1	1
XOR	3	32
ADD	1	32
SUB	1	32
S_0	4	127
S_1	4	128
S_2	4	129
S_3	4	132
S_4	4	34
S_5	2	34
R_1	1	3
R_2	1	7
R_3	1	13
R_4	1	16
R_5	1	19
R_6	1	23
R_7	1	27

TABLE II: BLUE MIDNIGHT WISH-256 performance results

Algorithm Name	FPGA Type	Area(Slice)	Frequency [MHZ]	Throughput	Memory Blocks
Proposed	Virtex XCV300	895	38	9 Mbps	1
	Virtex5 XC5VLX110	84	116	28 Mbps	2
BMW-256 [8]	Virtex XCV300	2147	60	1.07 Mbps	——
	Virtex5 XC5VLX110	1980	264	5Mbps	——

REFERENCES

[1] National Institute of Standards and Technology, *"Secure Hash Standard (SHS), FIPS PUB 180-3"*, Federal Information Processing Standards Publication, October 2008, http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf

[2] X. Wang, A. C. Yao, and F. Yao. *"Cryptanalysis on SHA-1 hash function"*. In proceeding of The Cryptographic hash workshop. National Institute of Standards and Technology, November 2005.

[3] NIST (2006). *"NIST Comments on Cryptanalytic Attacks on SHA-1"*. http://csrc.nist.gov/groups/ST/hash/statement.html

[4] William E. Burr, *"Cryptographic Hash Standards: Where Do We Go from Here?"*, IEEE Security and Privacy, Vol. 4, No. 2, pp. 88-91, Mar./Apr. 2006, doi:10.1109/MSP.2006.37

[5] D. Gligoroski, V. Klima, S. J. Knapskog, M. El-Hadedy, Jorn Amundsen and S. F. Mjolsnes, *"Cryptographic Hash Function BLUE MIDNIGHT WISH"*, Submission to NIST (Round 2) of SHA-3 Competition, September 2009

[6] D. Gligoroski, V. Klima, *"A Document describing all modications made on the Blue Midnight Wish cryptographic hash function before entering the Second Round of SHA-3 hash competition"*, http://people.item.ntnu.no/~danilog/Hash/BMW-SecondRound/Supporting_Documentation/Round2Mods.pdf

[7] Xilinx, *"Device Package User Guide"*, 2010 http://www.xilinx.com/support/documentation/user_guides/ug112.pdf"

[8] M. El Hadedy, D. Gligoroski, S. J. Knapskog, *"Low Area Implementation of the Hash Function "Blue Midnight Wish - 256" for FPGA platforms"*. In Proceedings of The International Conference on Intelligent Networking and Collaborative Systems. IEEE Computer Society 2009 ISBN 978-0-7695-3858-7.