Resource-Efficient Implementation of BLUE MIDNIGHT WISH-256 Hash Function on Xilinx FPGA Platform

Mohamed El-Hadedy*, Martin Margala †, Danilo Gligoroski * and Svein J. Knapskog *
* *Norwegian University of Science and Technology (NTNU), Trondheim, Norway*
† *University of Massachusetts(UMASS), Lowell,MA,USA*
* *Email: mohamed.elhadedy@q2s.ntnu.no, danilog@item.ntnu.no, Svein J. Knapskog@q2s.ntnu.no*
† *Email: Martin_Margala@uml.edu*

Abstract—**This paper presents the design and analysis of an area efficient Blue Midnight Wish compression function with digest size of 256 bits (BMW-256) on FPGA platforms. The proposed architecture achieves significant improvements in system throughput with reduced area. We demonstrate the performance of the proposed BMW hash function core using VIRTEX 5 FPGA implementation. The new BMW hash function design allows for 16X speed up in performance while consuming significantly lower area than previously reported (i.e. just 445 slices).**

Keywords-**Hash Function Standard; SHA-2; Blue Midnight Wish; BMW-256;**

I. INTRODUCTION

In cryptography and information security, hash functions are considered the "Swiss army knife"- they are used in countless protocols and algorithms. Recently, there have been two SHA algorithms introduced. SHA-1, and SHA-2, and although they have some similarities, they have also significant differences [1,2]. SHA-1 is the most 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[1]. the discovered mathematical weaknesses which might exist indicates the urgent need for using stronger hash functions [2]. Already, there exist another family of standardized hash function called SHA-2 ready to replace SHA-1.

The SHA-2 family is a family of four algorithms that differs 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 less than desirable, and the National Institute of Standards and Technology (NIST) have felt the need for and made efforts to develop an improved new family of hash functions [2, 3]. 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 the superior 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 four or 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 [4]. In this paper, the proposed hardware design of BMW is simple, area efficient and provides significant throughput improvements over previous works. The proposed BMW hash function core -256 is evaluated in FPGA using VIRTEX II XCV300-6PQ240 Xilinx device [5,6].

The rest of the paper is organized as follows. In Section 2, we describe briefly the compression function of of the second round version of the BMW-256 algorithm. of the second round version of the BMW-256 algorithm. In Section 4,BMW Hashing operations. In section 5, the synthesis results of the FPGA implementation are given with comparisons with other related works. Finally, in section 6, conclusions are presented, and observations and future work are discussed.

II. THE COMPRESSION FUNCTION OF BLUE MIDNIGHT WISH – 256

The BMW-256 compression function is shown in Fig.1. We refer to the variant that creates the 256 bit message digest as BMW-256. The basic data block used is called a word which is 32 bits long. BMW has four different operations in the hash computation stage: bitwise logical word XOR operation, word addition and subtraction, shift operations (left or right), and rotate left operation. BMW uses a double pipe design to increase the resistance against generic multi-collision attacks and length extension attacks [7,8]. 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

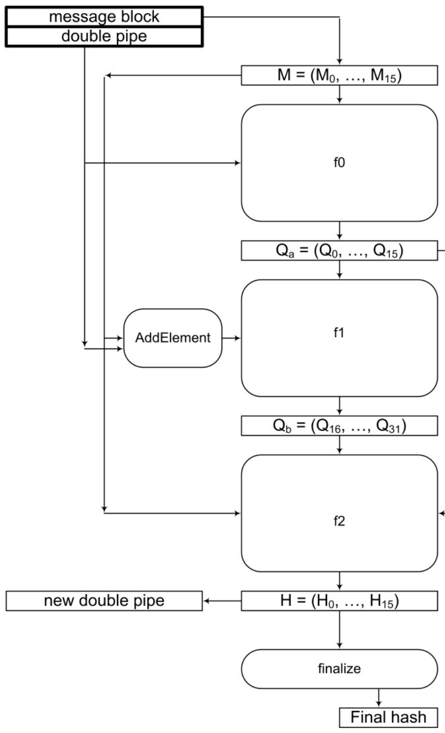

Figure 1: Graphical representation of the compression function in Blue Midnight Wish

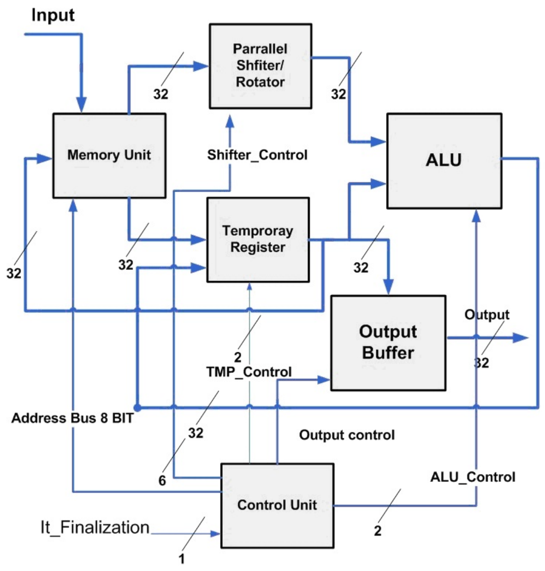

Figure 2: Graphical representation of the compression function in Blue Midnight Wish

(previous output of the BMW compression) $H^{(i-1)}$ of 512 bits size of the current double pipe $H^{(i)}$

The compression function [5,6] uses 2 main parts as shown in Fig.1. The first one uses three separate functions f_0, f_1, f_2 to generate $H^{(i)} = (H_0^{(i)}$, $H_1^{(i)}$,..., $H_{15}^{(i)})$. Inputs for the function F_0 are two arguments: 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 F1 are also three arguments, Message block $M^{(i)}$, AddElement and the value of $Q_a^{(i)}$. The function $F_1(M^{(i)}$, Add_Element, $Q_a^{(i)}$) 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 takes three arguments; 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 $H^{(i)}$, hence $H^{(i)} = (H_0^{(i)}$, $H_1^{(i)}$,..., $H_{15}^{(i)}$). The second part contains the same functions but instead of initial values $H_0^{(i-1)}$, $H_1^{(i-1)}$,..., $H_{15}^{(i-1)}$, it will use $Constant_j^{final} = (Constant_0^{final}$, $Constant_1^{final}$,... , $Constant_{15}^{final}$) values and the input message block will be the new double pipe $H^{(i)} = (H_0^{(i)}$, $H_1^{(i)}$,..., $H_{15}^{(i)}$). The reason to use the final Constants is to remove one degree of freedom to the attackers who try to find pseudo collisions and pseudo-pre-images.

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 units, Parallel shifter/Rotator, ALU (Arithmetic logic unit), Temporary Register, Output stage and Control Unit. Their operations are as follows:

Parallel Shifter/Rotator: It contains a 5 x 32 Mux (Multiplex) matrix each one is Mux 2x1 with big Encoder (5 X 11). This component is responsible for the shift and rotation operations of the 32 bit word. It receives 32 bit parallel data from the Memory Block and transmits 32 bit parallel data to the ALU. That happens decided by the value of shifter control word. Because we have 46 operations in BMW hash Core, the width of shifter control word is 6 control bits as shown in Fig.3.

ALU component: The ALU component offers four different operations in the hash computation stage: bit-wise logical word XOR operation, 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 trans-

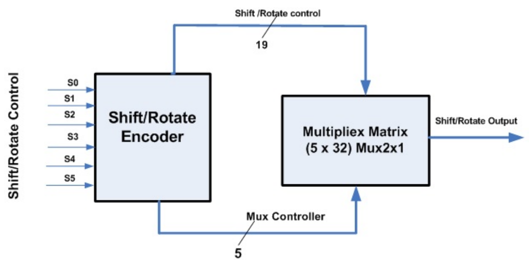

Figure 3: Parallel Shifter/Rotator Block

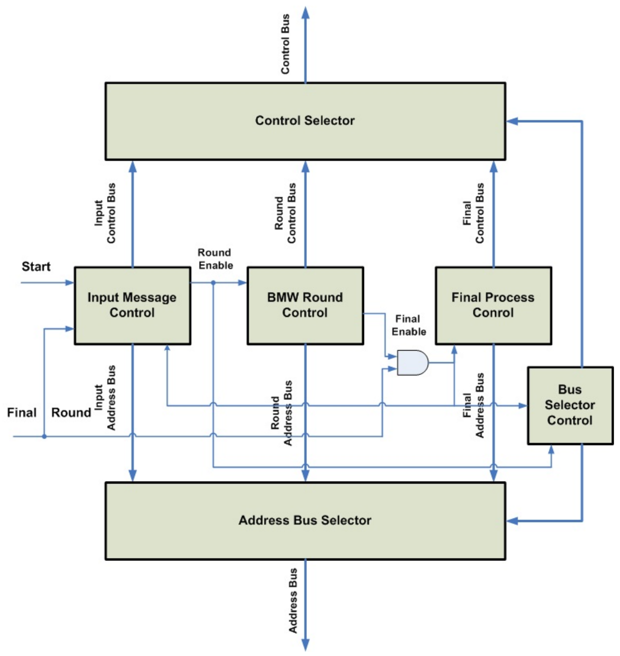

Figure 4: BMW-256 Controller

mit the output to the Temporary Register to work as a parallel accumulator.

Temporary Register: It contains 32 Mux 2x1 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 Block: 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 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 RAM to store the BMW-256 input block Message ($M_0^{(i)}$, $M_1^{(i)}$,..., $M_{15}^{(i)}$), the intermediate values of the BMW hash function, and the final double pipe values $H^{(i)} = (H_0^{(i)}, H_1^{(i)}, H_2^{(i)},...,H_{15}^{(i)})$.

Controller: It has been designed as a Moore FSM as shown in Fig. 4 and Fig. 5. it contains six operative

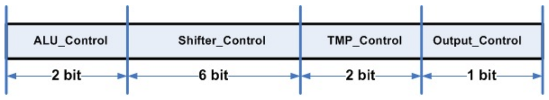

Figure 5: Control Bus

parts, all of them working together to produce Memory Address words to control the Memory Block traffic with the other BMW-256 sub-systems . The controller produces the control word to control the data flow between the BMW-256 Core sub-systems. The controller subsystems are working as the following:

The Input Message Control, once the start signal becomes high, starts to organize the sixteen input messages inside RAM locations. Subsequently, the Round Enable signal becomes high, that will make the BMW Round Control starts to execute the f_0, f_1, and f_2 according to the BMW-256 Algorithm operations which was described in section 2. Finally, in the final message round when the Final Round signal becomes high, the final Process Control Block starts to transfer $Const^{final}$ in messages locations and starts the Round Control Block to work to produce the Final Hash Output. Both Control Selector and Address Bus Selector are multiplexers controlled by the combinational circuit block called Bus Selector Control.

IV. Blue Midnight Wish256 hashing operations

In this section we describe how the Computation Hash Core works to execute the internal functions in BMW-256. For example, if we would XOR two pieces of data in locations number 4 and 5 in the memory unit, and write the result in location number 7. First, the Controller gives order to the Memory Block to choose location number 4. Then the Controller asks the Temporary Register to pick up the data from data bus and subsequently the same operation happens with location number 5. But instead of using the Temporary Register, the Parallel Shifter/Rotator picks up the data. Now, the Controller asks the Operation Encoder unit to give order to the ALU unit to add these data and save them in the Temporary Register. Finally, the Controller gives order to the Memory Block 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 number of cycles for each operation in Table I. Using the BMW-256 operations in Table I helps to 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 [9] and it was synthesized (synthesis, placement and

Table I: BLUE MIDNIGHT WISH-256 Hashing Core Operations (execution times)

Operation	Proposed	BMW-256[11]
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)	Throughput
Proposed	Virtex XCV300	1314	6 Mbps
	Virtex5 XC5VLX110	445	21 Mbps
BMW-256 [11]	Virtex XCV300	2147	1.07 Mbps
	Virtex5 XC5VLX110	1980	5Mbps
SHA-256[12]	Virtex XCV200	4768	291Mbps
SHA-256 [13]	VirtexE XCV600	5828	———-

routing) using ISE foundation 10.1 [10] in VIRTEX XCV300-6PQ240 and Virtex5 XC5VLX110 Xilinx devices. In Table II, we compare the area size for different designs for SHA-2 [12,13] in VIRTEX and VIRTEX 5 Xilinx devices. By using the proposed BMW-256 Core, we have achieved around 38% lower area compared to previous designs for BMW-256 while increasing throughput around 6 times compared to the previous values measured, on the same FPGA Virtex XCV300 device and around 77 % lower area compared to the previous BMW-256 while increasing the throughput 16 times compared to the previous design on the same FPGA Virtex5 XC5VLX110.

VI. Conclusion and Future Work

In This paper we 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 messages words of 32 bits and processes them. The goal was to use as small area as possible in order to minimize the hardware cost. We have achieved around 72% lower area compared to SHA-256 on the same FPGA device. For the future work, it will be a challenge to improve this design, to improve the throughput while keeping the optimized the area usage and implement it on ASIC.

References

[1] 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.

[2] NIST (2006). *"NIST Comments on Cryptanalytic Attacks on SHA-1"*.

[3] 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

[4] eBACS (2010). *"ECRYPT Benchmarking of Cryptographic Systems"*.

[5] D. Gligoroski, V. Klima, S. J. Knapskog, M. El-Hadedy, J.Amundsen ,*"Blue Midnight Wish"*, In proceeding of The First SHA-3 Candidate Conference, February 2009, Belgium- Leuven

[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"*,

[7] Joux, A., *"Multicollisions in iterated hash functions. Application to cascaded constructions"*.In Proceedings of CRYPTO 2004. LNCS, vol. 3152, pp. 430440, 2004.

[8] Lucks, S., *A failure-friendly design principle for hash functions*, In proceeding of ASIACRYPT, 2005.

[9] *"Model Sim PE/PLUS User's Manual*, Model technology, 2008

[10] Xilinx, *"Device Package User Guide"*, 2010

[11] 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.

[12] N. Sklavos, O. Koufopavlou, *"Implementation of the SHA-2 Hash Family Standard Using FPGAs"*, The Journal of Supercomputing, 31(3), pp.227-248, 2005.

[13] M. McLoone, J. V. McCanny, *"Efficient single-chip implementation of SHA-384 & SHA-512"*. In Proceedings of the International Conference on Field-Programmable Technology (FTP), pp. 311-314, 2002