

# Implementing The Blue Midnight Wish Hash Function on Xilinx Virtex-5 FPGA Platform

Mohamed El-Hadedy\*<sup>†</sup>, Martin Margala<sup>†</sup>, Danilo Gligoroski \* and Svein J. Knapskog \*

\* Norwegian University of Science and Technology (NTNU), Trondheim, Norway

<sup>†</sup> University of Massachusetts (UMASS), Lowell, MA, USA

\* Email: mohamed.elhadedy@q2s.ntnu.no, danilog@item.ntnu.no, Svein J. Knapskog@q2s.ntnu.no

<sup>†</sup> Email: Martin\_Margala@uml.edu, Mohamed\_Aly@uml.edu

**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. The core functionality with finalization implementation without padding stage 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 56 Mbps.

**Keywords**-SHA-3; Blue Midnight Wish; NIST;

## I. INTRODUCTION

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]. Although there is another family of standardized hash function called SHA-2, ready to replace SHA-1, consequently, the discovered mathematical weakness which might exist indicates the need for using stronger hash functions [2].

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, they are algorithmically similar to SHA-1, 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 start a 4 year world-wide development process, including a competition for the superior algorithm design, for choosing the next cryptographic hash standard SHA-3.

The new hash standard SHA-3 is currently under development - the function will be selected via an open competition running between 2008 and 2012. The Blue Midnight Wish (BMW) hash function is one of the candidates from Second Round of the competition and it is one of the fastest

proposed new designs in the SHA-3 competition in software [4]. In this paper, we show the proposed architecture design is simple, area efficient and provides significant throughput improvements over previous works. The proposed BMW-256 hash function core is evaluated in FPGA using Virtex5 XC5VLX110 Xilinx device [5, 6].

The rest of the paper is organized as follows. In Section 2, we describe briefly the compression function of the new BMW-256 algorithm version.

Section 3, highlights of the architecture and FPGA implementation of the proposed BLUE MIDNIGHT WISH hash function core sub-system. In Section 4, the synthesis results of the FPGA implementation are given with comparisons with other related works. Finally, in section 5, conclusions, observations and future work are discussed.

## II. ALGORITHM SPECIFICATION

Blue Midnight wish hash function is a one of the 14 candidates in second round of the NIST's SHA-3 competition. It was tweaked in order to resist attacks by Thomsen [7]. BMW is a family of hash functions, containing four major instances, e.g., BMW-n, for n= 224, 256, 384, 512, where n is the size of hash output. BMW is a wide-pipe Merkle-Damgrd hash construction with an unconventional compression function, where the nonlinearity is derived from the overlap of modular addition and XOR operations. The most innovative parts of the design are the compression function construction and the design of the permutations; much of the design is novel and unique amongst the second-round candidates. BMW has very good performance and appears to be suitable for a wide range of platforms. It has modest memory requirements. The BMW has four different operations in the hash computation stage: bit-wise 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. In the double pipe design, the sizes of the inputs to the compression functions are twice the message digest size.

As shown in Fig.1, the compression function of BMW takes the chaining of 16 words  $H_0^{(i-1)}, H_1^{(i-1)}, \dots, H_{15}^{(i-1)}$ ,

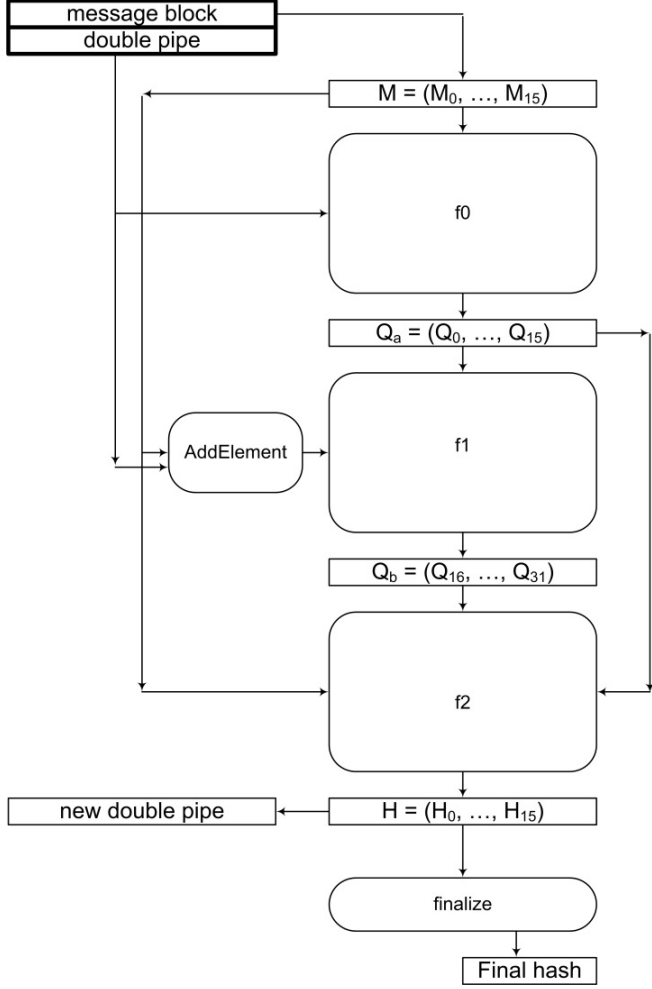


Figure 1. Representation of the compression function in Blue Midnight Wish

and a message 16 words  $M_0^{(i)}, M_1^{(i)}, \dots, M_{15}^{(i)}$  as input, and produces the updated chaining  $H_0^{(i)}, H_1^{(i)}, \dots, H_{15}^{(i)}$ . The size of a word is 32 bits for BMW-224/256 and 64 bits for BMW-384/512. The compression function after tweaks [5,6] uses 2 main parts. The first one comprises three functions, called  $f_0$ ,  $f_1$ , and  $f_2$ , in sequence to generate  $H^{(i)}$ . As shown in the following. Inputs for the function  $f_0$  are two arguments as shown in Fig.1: The first argument consists of sixteen 32-bit words, which are working as initial double pipe values  $H_0^{(i-1)}, H_1^{(i-1)}, \dots, H_{15}^{(i-1)}$ , as shown in Table.I, for BMW-256. The second argument consists of sixteen 32-bit words, which represent the input message block:  $M_0^{(i)}, M_1^{(i)}, \dots, M_{15}^{(i)}$ . As shown in Table.II, the function  $f_0(M^{(i)}, H^{(i-1)})$  computes  $M^{(i)} \oplus H^{(i-1)}$ , produces a temporary  $W_j^{(i)}, j = 1, 2, 3, \dots, 15$ , and  $Q_a^{(i)} = (Q_0^{(i)}, Q_1^{(i)}, \dots, Q_{15}^{(i)})$  from  $W_j^{(i)}$ . The second function  $f_1$  takes the same message block  $M^{(i)}$ , and  $Q_a^{(i)}$  (the output from  $f_0$ ) as input, and generates the second part of the quadruple

pipe  $Q_b^{(i)} = (Q_0^{(i)}, Q_1^{(i)}, \dots, Q_{15}^{(i)})$ , through ADD\_Element, EXPAND\_1 and EXPAND\_2 expression functions as shown in Eq.(1) and Table.III. Expansion functions make use of the s-transform along with the r-transforms as shown in Eq. (2), and Eq. (3).

Table I  
INITIAL DOUBLE PIPE  $H^{(i-1)}$  FOR BMW-256 (HEXADECIMAL VALUES)

BLUE MIDNIGHT WISH-256			
$H_0^{(i-1)}$	0x40414243	$H_8^{(i-1)}$	0x60616263
$H_1^{(i-1)}$	0x44454647	$H_9^{(i-1)}$	0x64656667
$H_2^{(i-1)}$	0x48494A4B	$H_{10}^{(i-1)}$	0x68696A6B
$H_3^{(i-1)}$	0x4C4D4E4F	$H_{11}^{(i-1)}$	0x6C6D6E6F
$H_4^{(i-1)}$	0x50515253	$H_{12}^{(i-1)}$	0x70717273
$H_5^{(i-1)}$	0x54555657	$H_{13}^{(i-1)}$	0x74757677
$H_6^{(i-1)}$	0x58595A5B	$H_{14}^{(i-1)}$	0x88898A8B
$H_7^{(i-1)}$	0x5C5D5E5F	$H_{15}^{(i-1)}$	0x8C8D8E8F

For  $ii = 0, 1 : Q_{(ii+16)}^{(i)} = Expand_1(ii + 16)$

$$Expand_1(i) = S_1(Q_{(j-16)}^i) + S_2(Q_{(j-15)}^i) + S_3(Q_{(j-14)}^i) + S_0(Q_{(j-13)}^i) + S_1(Q_{(j-12)}^i) + S_2(Q_{(j-11)}^i) + S_3(Q_{(j-10)}^i) + S_0(Q_{(j-9)}^i) + S_1(Q_{(j-8)}^i) + S_2(Q_{(j-7)}^i) + S_3(Q_{(j-6)}^i) + S_0(Q_{(j-5)}^i) + S_1(Q_{(j-4)}^i) + S_2(Q_{(j-3)}^i) + S_3(Q_{(j-2)}^i) + S_0(Q_{(j-1)}^i) + ADD\_Element(j - 16) \quad (1)$$

For  $ii=2,3,4,5,\dots,15 : Q_{(ii+16)}^{(i)} = Expand_2(ii + 16)$

$$Expand_2(i) = (Q_{(j-16)}^i) + r_1(Q_{(j-15)}^i) + (Q_{(j-14)}^i) + r_2(Q_{(j-13)}^i) + (Q_{(j-12)}^i) + r_3(Q_{(j-11)}^i) + (Q_{(j-10)}^i) + r_4(Q_{(j-9)}^i) + (Q_{(j-8)}^i) + r_5(Q_{(j-7)}^i) + (Q_{(j-6)}^i) + r_6(Q_{(j-5)}^i) + (Q_{(j-4)}^i) + r_7(Q_{(j-3)}^i) + S_4(Q_{(j-2)}^i) + S_5(Q_{(j-1)}^i) + ADD\_Element(j - 16)$$

Note that ADD\_Element(j) index expressions involving the variable j for left rotations, M and H are computed modulo(16).

$$ADD\_Element(j) = (ROTL^{(j+1)}(M_{(j)}^{(i)})) + ROTL^{(j+4)}(M_{(j+3)}^{(i)}) + ROTL^{(j+11)}(M_{(j+10)}^{(i)}) + K_{j+16} \oplus H_{j+7}^{(i)} \quad (2)$$

$$r_1(x) = ROTL^3(x), r_2(x) = ROTL_7(x), r_3(x) = ROTL^{13}(x), r_4(x) = ROTL^{16}(x), r_5(x) = ROTL^{19}(x), r_6(x) = ROTL^{23}(x), r_7(x) = ROTL^{27}(x), S_4(x) = SHR^1(x) \oplus x, S_5(x) = SHR^2(x) \oplus x \quad (3)$$

Table II  
DEFINITION OF THE FUNCTION  $f_0$  OF BLUE MIDNIGHT WISH

1. Bijective Transform of $M^{(i)} \oplus H^{(i-1)}$	
$W_0^{(i-1)} = (M_5^{(i)} \oplus H_5^{(i-1)}) - (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)}) + (M_{14}^{(i)} \oplus H_{14}^{(i-1)})$	
$W_1^{(i-1)} = (M_6^{(i)} \oplus H_6^{(i-1)}) - (M_8^{(i)} \oplus H_8^{(i-1)}) + (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) + (M_{14}^{(i)} \oplus H_{14}^{(i-1)}) - (M_{15}^{(i)} \oplus H_{15}^{(i-1)})$	
$W_2^{(i-1)} = (M_0^{(i)} \oplus H_0^{(i-1)}) + (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_9^{(i)} \oplus H_9^{(i-1)}) - (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) + (M_{15}^{(i)} \oplus H_{15}^{(i-1)})$	
$W_3^{(i-1)} = (M_0^{(i)} \oplus H_0^{(i-1)}) - (M_1^{(i)} \oplus H_1^{(i-1)}) + (M_8^{(i)} \oplus H_8^{(i-1)}) - (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)})$	
$W_4^{(i-1)} = (M_1^{(i)} \oplus H_1^{(i-1)}) + (M_2^{(i)} \oplus H_2^{(i-1)}) + (M_9^{(i)} \oplus H_9^{(i-1)}) - (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) - (M_{14}^{(i)} \oplus H_{14}^{(i-1)})$	
$W_5^{(i-1)} = (M_3^{(i)} \oplus H_3^{(i-1)}) - (M_2^{(i)} \oplus H_2^{(i-1)}) + (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) - (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) + (M_{15}^{(i)} \oplus H_{15}^{(i-1)})$	
$W_6^{(i-1)} = (M_4^{(i)} \oplus H_4^{(i-1)}) - (M_0^{(i)} \oplus H_0^{(i-1)}) - (M_3^{(i)} \oplus H_3^{(i-1)}) - (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)})$	
$W_7^{(i-1)} = (M_1^{(i)} \oplus H_1^{(i-1)}) - (M_4^{(i)} \oplus H_4^{(i-1)}) - (M_5^{(i)} \oplus H_5^{(i-1)}) - (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) - (M_{14}^{(i)} \oplus H_{14}^{(i-1)})$	
$W_8^{(i-1)} = (M_2^{(i)} \oplus H_2^{(i-1)}) - (M_5^{(i)} \oplus H_5^{(i-1)}) - (M_6^{(i)} \oplus H_6^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)}) - (M_{15}^{(i)} \oplus H_{15}^{(i-1)})$	
$W_9^{(i-1)} = (M_0^{(i)} \oplus H_0^{(i-1)}) - (M_3^{(i)} \oplus H_3^{(i-1)}) + (M_6^{(i)} \oplus H_6^{(i-1)}) - (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_{14}^{(i)} \oplus H_{14}^{(i-1)})$	
$W_{10}^{(i-1)} = (M_8^{(i)} \oplus H_8^{(i-1)}) - (M_1^{(i)} \oplus H_1^{(i-1)}) - (M_4^{(i)} \oplus H_4^{(i-1)}) - (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_{15}^{(i)} \oplus H_{15}^{(i-1)})$	
$W_{11}^{(i-1)} = (M_8^{(i)} \oplus H_8^{(i-1)}) - (M_0^{(i)} \oplus H_0^{(i-1)}) - (M_2^{(i)} \oplus H_2^{(i-1)}) - (M_5^{(i)} \oplus H_5^{(i-1)}) + (M_9^{(i)} \oplus H_9^{(i-1)})$	
$W_{12}^{(i-1)} = (M_1^{(i)} \oplus H_1^{(i-1)}) + (M_3^{(i)} \oplus H_3^{(i-1)}) - (M_6^{(i)} \oplus H_6^{(i-1)}) - (M_9^{(i)} \oplus H_9^{(i-1)}) + (M_{10}^{(i)} \oplus H_{10}^{(i-1)})$	
$W_{13}^{(i-1)} = (M_2^{(i)} \oplus H_2^{(i-1)}) + (M_4^{(i)} \oplus H_4^{(i-1)}) + (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) + (M_{11}^{(i)} \oplus H_{11}^{(i-1)})$	
$W_{14}^{(i-1)} = (M_3^{(i)} \oplus H_3^{(i-1)}) - (M_5^{(i)} \oplus H_5^{(i-1)}) + (M_8^{(i)} \oplus H_8^{(i-1)}) - (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) - (M_{12}^{(i)} \oplus H_{12}^{(i-1)})$	
$W_{15}^{(i-1)} = (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) - (M_4^{(i)} \oplus H_4^{(i-1)}) - (M_6^{(i)} \oplus H_6^{(i-1)}) - (M_9^{(i)} \oplus H_9^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)})$	
2. Further bijective transform of $W_j^{(i)}, j = 1, 2, 3, \dots, 15$	
$Q_0^{(i)} = S_0(W_0^{(i)}) + H_1^{(i-1)}; Q_1^{(i)} = S_1(W_1^{(i)}) + H_2^{(i-1)}; Q_2^{(i)} = S_2(W_2^{(i)}) + H_3^{(i-1)}; Q_3^{(i)} = S_3(W_3^{(i)}) + H_4^{(i-1)};$	
$Q_4^{(i)} = S_4(W_4^{(i)}) + H_5^{(i-1)}; Q_5^{(i)} = S_0(W_1^{(i)}) + H_6^{(i-1)}; Q_6^{(i)} = S_1(W_6^{(i)}) + H_7^{(i-1)}; Q_7^{(i)} = S_2(W_7^{(i)}) + H_8^{(i-1)};$	
$Q_8^{(i)} = S_3(W_8^{(i)}) + H_9^{(i-1)}; Q_9^{(i)} = S_4(W_9^{(i)}) + H_{10}^{(i-1)}; Q_{10}^{(i)} = S_0(W_{10}^{(i)}) + H_{11}^{(i-1)}; Q_{11}^{(i)} = S_1(W_{11}^{(i)}) + H_{12}^{(i-1)};$	
$Q_{12}^{(i)} = S_2(W_{12}^{(i)}) + H_{13}^{(i-1)}; Q_{13}^{(i)} = S_3(W_{13}^{(i)}) + H_{14}^{(i-1)}; Q_{14}^{(i)} = S_4(W_{14}^{(i)}) + H_{15}^{(i-1)}; Q_{15}^{(i)} = S_3(W_{15}^{(i)}) + H_0^{(i-1)};$	
3. S-transform used in $f_0$ Function	
$S_0(x) = SHR^1(x) \oplus SHL^3(x) \oplus ROTL^4(x) \oplus ROTL^{19}(x)$	
$S_1(x) = SHR^1(x) \oplus SHL^2(x) \oplus ROTL^8(x) \oplus ROTL^{23}(x)$	
$S_2(x) = SHR^2(x) \oplus SHL^1(x) \oplus ROTL^{12}(x) \oplus ROTL^{25}(x)$	
$S_3(x) = SHR^2(x) \oplus SHL^2(x) \oplus ROTL^{15}(x) \oplus ROTL^{29}(x)$	

Table III  
 $K_j$  FOR BLUE MIDNIGHT WISH (HEXADECIMAL VALUES)

BLUE MIDNIGHT WISH-256			
$K_0$	0x55555550	$K_8$	0x 7fffff8
$K_1$	0x5aaaaaa5	$K_9$	0x 8555554d
$K_2$	0x5ffffffa	$K_{10}$	0x 8aaaaaa2
$K_3$	0x6555554f	$K_{11}$	0x 8fffff7
$K_4$	0x6aaaaaa4	$K_{12}$	0x 9555554c
$K_5$	0x6fffff9	$K_{13}$	0x 9aaaaaa1
$K_6$	0x7555554e	$K_{14}$	0x 9fffff6
$K_7$	0x7aaaaaa3	$K_{15}$	0x a555554b

As shown in Table.IV, the third function  $f_2$  takes three arguments: Message block  $M(i)$  and the quadruple pipes  $Q_a(i)$  and  $Q_b(i)$  and generates the value of the double pipe  $H^i$ .

The second part contains the same functions but instead of initial double pipe values  $H_0^{(i-1)}, H_1^{(i-1)}, H_2^{(i-1)}, \dots, H_{15}^{(i-1)}$ , it will use constant values  $CONST_j^{final} = (CONST_0^{final}, CONST_1^{final}, CONST_2^{final}, \dots,$

$CONST_{15}^{final}$ ) as shown in Table.V and the input message block will be the new double pipe  $H^{(i)} = (H_0^{(i)}, H_1^{(i)}, H_2^{(i)}, \dots, H_{15}^{(i)})$ . The reason to use  $CONST_j^{final}$  values is to remove one degree of freedom to the attackers who try to find pseudo collisions and pseudo-preimages. Additionally, the final invocation of the compression function is a measure for any attack whereby an attacker can find near collisions or near-pseudo-collisions of the compression function of BMW [5,6].

### 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 as shown in Fig.3. 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) as shown in Fig.4. 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

Table IV  
DEFINITION OF THE FOLDING FUNCTION  $f_2$  OF BLUE MIDNIGHT WISH

1. Cumulative temporary variables XL and XH	
$XL = Q_{16}^{(i)} \oplus Q_{17}^{(i)} \oplus Q_{18}^{(i)} \oplus Q_{19}^{(i)} \oplus Q_{20}^{(i)} \oplus Q_{21}^{(i)} \oplus Q_{22}^{(i)} \oplus Q_{23}^{(i)}$	
$XH = XL \oplus Q_{24}^{(i)} \oplus Q_{25}^{(i)} \oplus Q_{26}^{(i)} \oplus Q_{27}^{(i)} \oplus Q_{28}^{(i)} \oplus Q_{29}^{(i)} \oplus Q_{30}^{(i)} \oplus Q_{31}^{(i)}$	
2. The new double pipe $H^{(i)}$	
$H_0^{(i)} = (SHL^5(XH) \oplus SHR^5(Q_{16}^{(i)}) \oplus M_0^{(i)}) + (XL \oplus Q_{24}^{(i)} \oplus Q_0^{(i)})$	
$H_1^{(i)} = (SHL^7(XH) \oplus SHL^8(Q_{17}^{(i)}) \oplus M_1^{(i)}) + (XL \oplus Q_{25}^{(i)} \oplus Q_1^{(i)})$	
$H_2^{(i)} = (SHL^5(XH) \oplus SHL^5(Q_{18}^{(i)}) \oplus M_2^{(i)}) + (XL \oplus Q_{26}^{(i)} \oplus Q_2^{(i)})$	
$H_3^{(i)} = (SHR^1(XH) \oplus SHL^5(Q_{19}^{(i)}) \oplus M_3^{(i)}) + (XL \oplus Q_{27}^{(i)} \oplus Q_3^{(i)})$	
$H_4^{(i)} = (SHR^3(XH) \oplus (Q_{20}^{(i)} \oplus M_4^{(i)}) + (XL \oplus Q_{28}^{(i)} \oplus Q_4^{(i)})$	
$H_5^{(i)} = (SHL^6(XH) \oplus SHR^5(Q_{21}^{(i)}) \oplus M_5^{(i)}) + (XL \oplus Q_{29}^{(i)} \oplus Q_5^{(i)})$	
$H_6^{(i)} = (SHR^4(XH) \oplus SHL^6(Q_{22}^{(i)}) \oplus M_6^{(i)}) + (XL \oplus Q_{30}^{(i)} \oplus Q_6^{(i)})$	
$H_7^{(i)} = (SHR^1(XH) \oplus SHL^2(Q_{23}^{(i)}) \oplus M_7^{(i)}) + (XL \oplus Q_{31}^{(i)} \oplus Q_7^{(i)})$	
$H_8^{(i)} = ROTL^9(H_4^{(i)} + (XH \oplus Q_{24}^{(i)} \oplus M_8^{(i)})) + (SHL^8(XL) \oplus Q_{23}^{(i)} \oplus Q_8^{(i)})$	
$H_9^{(i)} = ROTL^{10}(H_5^{(i)} + (XH \oplus Q_{25}^{(i)} \oplus M_9^{(i)})) + (SHR^6(XL) \oplus Q_{16}^{(i)} \oplus Q_9^{(i)})$	
$H_{10}^{(i)} = ROTL^{11}(H_6^{(i)} + (XH \oplus Q_{26}^{(i)} \oplus M_{10}^{(i)})) + (SHL^6(XL) \oplus Q_{17}^{(i)} \oplus Q_{10}^{(i)})$	
$H_{11}^{(i)} = ROTL^{12}(H_7^{(i)} + (XH \oplus Q_{27}^{(i)} \oplus M_{11}^{(i)})) + (SHL^4(XL) \oplus Q_{18}^{(i)} \oplus Q_{11}^{(i)})$	
$H_{12}^{(i)} = ROTL^{13}(H_8^{(i)} + (XH \oplus Q_{28}^{(i)} \oplus M_{12}^{(i)})) + (SHR^3(XL) \oplus Q_{19}^{(i)} \oplus Q_{12}^{(i)})$	
$H_{13}^{(i)} = ROTL^{14}(H_9^{(i)} + (XH \oplus Q_{29}^{(i)} \oplus M_{13}^{(i)})) + (SHR^4(XL) \oplus Q_{20}^{(i)} \oplus Q_{13}^{(i)})$	
$H_{14}^{(i)} = ROTL^{15}(H_{10}^{(i)} + (XH \oplus Q_{30}^{(i)} \oplus M_{14}^{(i)})) + (SHR^7(XL) \oplus Q_{21}^{(i)} \oplus Q_{14}^{(i)})$	
$H_{15}^{(i)} = ROTL^{16}(H_{11}^{(i)} + (XH \oplus Q_{31}^{(i)} \oplus M_{15}^{(i)})) + (SHR^2(XL) \oplus Q_{22}^{(i)} \oplus Q_{15}^{(i)})$	

Table V  
 $CONST_j^{final}$  FOR BMW-256 (HEXADECIMAL VALUES)

BLUE MIDNIGHT WISH-256			
$CONST_0^{final}$	0Xaaaaaa0	$CONST_8^{final}$	0Xaaaaaa8
$CONST_1^{final}$	0Xaaaaaa1	$CONST_9^{final}$	0Xaaaaaa9
$CONST_2^{final}$	0Xaaaaaa2	$CONST_{10}^{final}$	0Xaaaaaaa
$CONST_3^{final}$	0Xaaaaaa3	$CONST_{11}^{final}$	0Xaaaaaaab
$CONST_4^{final}$	0Xaaaaaa4	$CONST_{12}^{final}$	0Xaaaaaaac
$CONST_5^{final}$	0Xaaaaaa5	$CONST_{13}^{final}$	0Xaaaaaaad
$CONST_6^{final}$	0Xaaaaaa6	$CONST_{14}^{final}$	0Xaaaaaaae
$CONST_7^{final}$	0Xaaaaaa7	$CONST_{15}^{final}$	0Xaaaaaaaf

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  $2 \times 1$  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

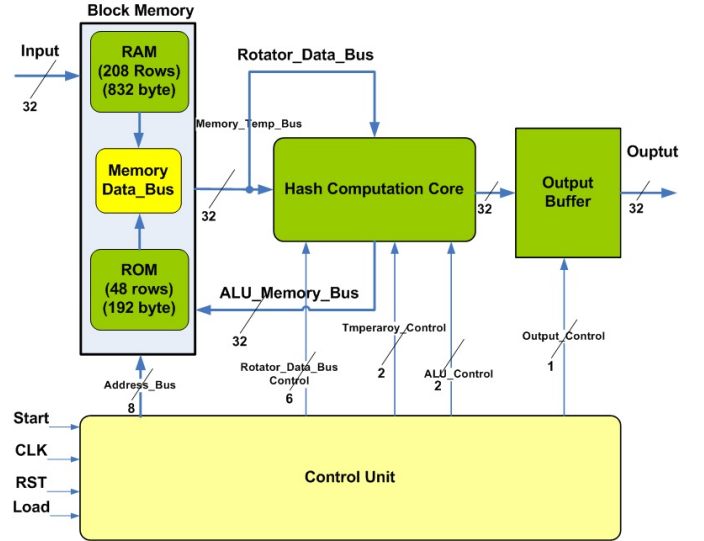


Figure 2. BLUE MIDNIGHT WISH-256 Core Architecture

output stage.

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

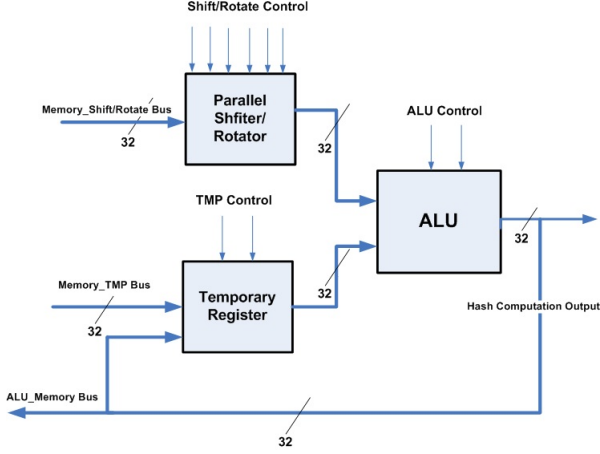


Figure 3. Hash Computation Core

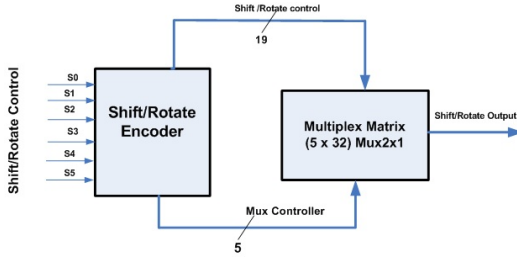


Figure 4. Parallel Shifter/Rotator Block

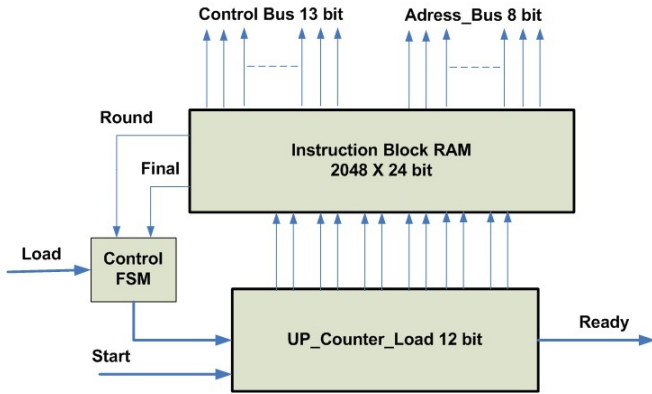


Figure 5. BMW-256 Control Unit

values  $H^{(i)} = (H_0^{(i)}, H_1^{(i)}, H_2^{(i)}, \dots, H_{15}^{(i)})$ .

**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 subsystems. Instructions Block RAM translated after placement and routing to one 36K Block RAM and one 18K block RAM. The Control Unit produces the 13 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  $H_0^{(i)}, H_1^{(i)}, \dots, H_{15}^{(i)}$  values in the message locations and transfer  $Const^{final}$  values in the  $H_0^{(i)}, H_1^{(i)}, \dots, H_{15}^{(i)}$  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 VII (page 6). Using the BMW-256 operations in Table I, we see that we can execute the function  $f_0$  in 426 cycles, function  $f_1$  in 452 cycles and finally function  $f_3$  in 170 cycles. The total cycles for a round which we use to calculate the throughput is 1048 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 [8] in VIRTEX 5 XC5VLX110 Xilinx devices. In Table VI, we compare this implementation for small FPGAs with the previous BMW implementation [9] and two others SHA-3 candidates (ECHO [10] and Keccak [11]), which are using block RAMs in their implementations. 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 51 times.

Table VI  
COMPACT IMPLEMENTATIONS OF SHA-3 CANDIDATES ON VIRTEX-5 FPGA

Algorithm Name	FPGA Type	Area(Slice)	Frequency [MHZ]	Throughput	Memory Blocks
<b>Proposed</b>	Virtex5 XC5VLX110	84	116	56 Mbps	3
<b>BMW-256 [9]</b>	Virtex5 XC5VLX110	1980	264	5Mbps	—
<b>ECHO-224/256 [10]</b>	Virtex5 XC5V1x50-2	127	352	72 Mbps	1
<b>Keccak [11]</b>	Virtex5 XC5V1x50-3	444	265	70 Mbps	1

Table VII  
BLUE MIDNIGHT WISH-256 HASHING CORE OPERATIONS (EXECUTION TIMES)

Operation	Proposed	BMW-256[9]
Load	1	1
XOR	1	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

## 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 without a large increase in area, and a reduced number of Block RAMs. It will certainly be beneficial in some future usage scenarios to do a full implementation in ASIC.

## REFERENCES

- [1] National Institute of Standards and Technology, "Secure Hash Standard (SHS), FIPS PUB 180-3", Federal Information Processing Standards Publication, October 2008,
- [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 modifications made on the Blue Midnight Wish cryptographic hash function before entering the Second Round of SHA-3 hash competition", [http://people.item.ntnu.no/~daniilog/Hash/BMW-SecondRound/Supporting\\_Documentation/Round2Mods.pdf](http://people.item.ntnu.no/~daniilog/Hash/BMW-SecondRound/Supporting_Documentation/Round2Mods.pdf)
- [7] S. S. Thomsen. "Pseudo-cryptanalysis of the Original Blue Midnight Wish". In S. Hong and T. Iwata, editors, Fast Software Encryption, LNCS, Seoul, South Korea, 2010. Springer. To appear.
- [8] Xilinx, "Device Package User Guide", 2010 [http://www.xilinx.com/support/documentation/user\\_guides/ug112.pdf](http://www.xilinx.com/support/documentation/user_guides/ug112.pdf)
- [9] 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.
- [10] J. L. Beuchat, E. Okamoto, Teppi Yamazaki, "A compact FPGA Implementation of the SHA-3 Candidate ECHO ". technical report, Eprint, 2010
- [11] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "Keccak sponge function family main document (version 2.0)", 2009, available online at. <http://keccak.noekoeon.org>