Low Area FPGA and ASIC Implementations of the Hash Function "Blue Midnight Wish-256"

Mohamed El-Hadedy*, Danilo Gligoroski† , Svein J. Knapskog* and Einar Johan Aas‡

* Q2S/ NTNU, Trondheim, Norway, e-mail: hadedy@q2s.ntnu.no

† Telematics /NTNU, Trondheim, Norway, e-mail: danilog@item.ntnu.no

* Q2S/NTNU, Trondheim, Norway, e-mail: Svein.J.Knapskog@q2s.ntnu.no

‡Electronic and Telecommunications/NTNU, Trondheim, Norway, e-mail: einar.j.aas@iet.ntnu.no

Abstract— Hash functions are widely used in information security and cryptography. They are used in countless applications such as message authentication codes (MAC), Digital Signatures (DS) and mobile trusted modules (MTM). Serious attacks have been reported against cryptographic hash algorithms, including SHA-1. Because the SHA-1 and SHA-2 families share a similar design, the National Institute of Standards and Technology (NIST) in 2007 decided to start a world-wide development process for choosing the next secure hash standard SHA-3. A pivotal part of the process is an open competition for bringing forward new and secure cryptographic hash functions. The Blue Midnight Wish hash function is one of the second round candidates for the SHA-3 competition. In this paper, we describe low area FPGA and ASIC implementations for the Blue Midnight Wish compression function with digest size of 256 bits (BMW-256). Using Xilinx FPGA platform Virtex 5 "XC5VLX30", we implemented BMW-256 using 1986 slices (including the internal memory), and using only 122 slices for an implementation that uses external memory. By using 0.8 µm CMOS standard cell library the ASIC implementation of BMW-256 takes approximately 13.5 Kgates (including the internal memory), and only 4 Kgates for an implementation that uses external memory.

***Keywords*— Hash Function Standard, SHA-3, Blue Midnight Wish, BMW-256.**

I. INTRODUCTION

The primary application of hash functions in cryptography is message integrity. The hash value provides a digital fingerprint of a message's contents, which ensures that the message has not been altered intentionally or non-intentionally. There are several well-known hash functions in use today such as MD5 and Secure Hash Algorithm (SHA). The SHA hash functions are a set of cryptographic hash functions designed by the National Security Agency (NSA) and published by the NIST as a U.S. Federal Information Processing Standard. The acronym SHA stands for Secure Hash Algorithm. There are two SHA algorithms: SHA-1 and SHA-2, and although they have some similarities, they have also significant differences. SHA-1 is the most used member of the SHA hash family, employed in hundreds of different applications and protocols. However, in 2005, a very significant theoretical development in detecting some security flaws in SHA-1 has been made by Wang et. al.[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 NIST have felt the need for and made efforts to develop an improved new family of hash functions [2, 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 hash function (BMW) is one of the fastest (in software) proposed new designs in the SHA-3 competition in software [4]. In this paper, low area FPGA and ASIC implementations for Blue Midnight Wish compression function with digest size of 256 bits (BMW-256) [5] and comparison with others [6] are introduced.

Two ASIC implementations found in the literature used CMOS libraries with 0.18 and 0.13 micrometer, respectively. We had only access to 0.8 micrometer technology. Thus, instead of comparing real silicon area, we compare equivalent gate count, which is a somewhat inaccurate metric for area. But it is safe to assume that lower equivalent gate count will lead to lower silicon area.

This work is organized as follows: In Section 2, BMW-256 compression function is described briefly. In Section 3, the proposed system architecture for the BMW-256 compression function is presented in details. In Section 4, the hardware implementation synthesis results are described. In Section 5, comparisons with other related works are given. Finally, in section 6, conclusions, observations and future work are discussed.

II. SECURE BMW-256 HASH FUNCTION

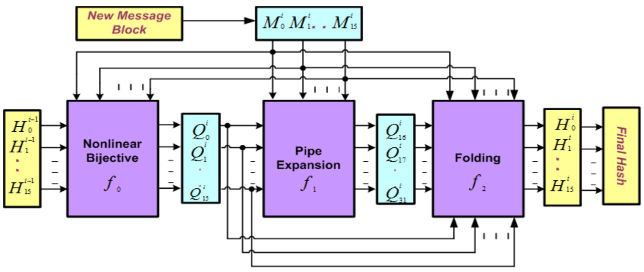

Fig.1 Graphical presentation of the compression function in Blue Midnight Wish

BMW-256 algorithm belongs to BMW family, which contains four different algorithms with different digest size, initial values and word size [5]. As shown in Fig.1, input message M is first padded to make its length 512 bit and output is divided to sixteen message blocks $M_0^{(i)}$, $M_1^{(i)}$,…., $M_{15}^{(i)}$. These messages with sixteen initial values $H_0^{(i-1)}$, $H_1^{(i-1)}$,..., $H_{15}^{(i-1)}$ is combined together by using nonlinear Bijective function f_0 to produce first part of the extended double pipe $Q_0^{(i)}$, $Q_1^{(i)}$, ..., $Q_{15}^{(i)}$. This result is combined with $M^{(i)}$ message blocks through pipe expansion function f_1 to produce second part of the extended double pipe $Q_{16}^{(i)}$, $Q_{17}^{(i)}$, ..., $Q_{31}^{(i)}$. Finally, first and second parts of the extended double pipe are combined together with $M^{(i)}$ message blocks through folding function f_2 to produce the final hash value $H_0^{(i)}$, $H_1^{(i)}$,..., $H_{15}^{(i)}$. BMW using double pipe guarantees a resistance against generic multi-collision attach and against length extension attacks [5].

III. BMW-256 CORE ARCHITECTURE

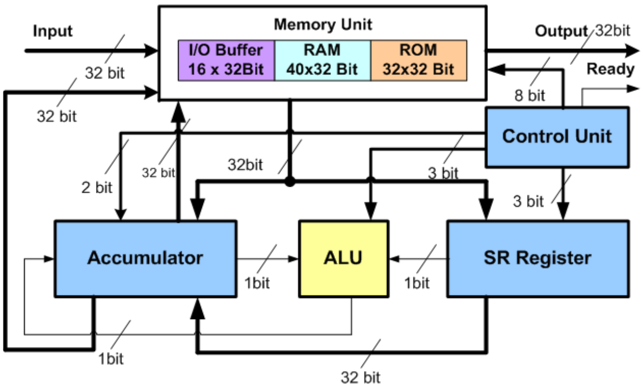

Fig.2 Architecture of BMW 256 Hashing Core

Fig.2 shows the complete architecture to the entire BMW core process, which includes five main hardware operative parts, named: "SR Register", "Accumulator Register", "ALU", "Memory Blocks", and "Control Unit". Their operation is as follows:

SR Register: This component is responsible for the shifting and rotation operations for 32 bit words according to the specification of the BMW 256 hashing core [5]. It receives 32 bit parallel data from the Memory Blocks and transmits two kinds of data. One of them is 32 bit parallel data to the Accumulator Register and another one is serial data transferred bit by bit to the ALU unit. This will happen decided by the value of the three control bits SR, S1 and S0 as shown in Fig.3 and Table I.

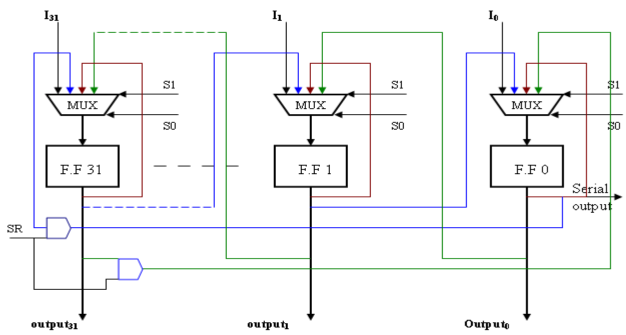

Fig.3 SR Register (Shift Register)

TABLE I SR REGISTER OPERATIONS

SR	S1	S0	Operation
x	1	1	LOAD
x	0	0	HOLD
0	0	1	SHR (Shift Right)
0	1	0	SHL (Shift Left)
1	0	1	ROR (Rotate Right)
1	1	0	ROL (Rotate Right)

x (Don't care sign)

Accumulator Register: this component receives parallel data 32 bit from SR register and Memory Blocks and receives data serial from ALU unit to update the result which are stored according two 2 control bits A1 and A0 as shown in Fig.4 and Table II.

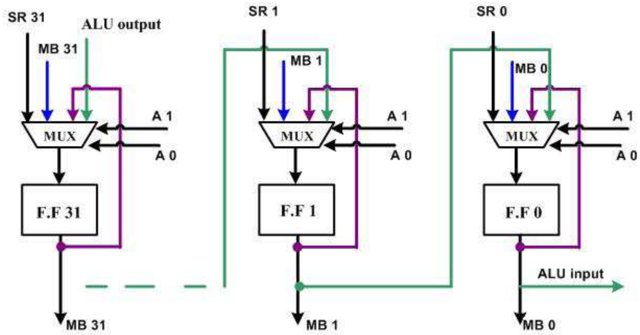

Fig.4 Accumulator Register

TABLE II ACCUMULATOR REGISTER OPERATIONS

A1	A0	Operation
1	0	LOAD (SR Register)
1	1	LOAD (Memory Blocks)
0	0	HOLD
0	1	SHR (Shift Right)

ALU unit: this component receives serial data bit by bit from both registers Accumulator and SR; then transmits serial data to accumulator for updating the data inside according to three bit control word as shown in Fig.5 and Table III. In fact this ALU unit contains one bit Full adder combined with 2 Mux 2x1 and D- Flip-Flop. First of Mux is used to choose which operation will happen, Arithmetic or logic (XOR operation), and another one for initializing the D-Flip-Flop for subtraction operation.

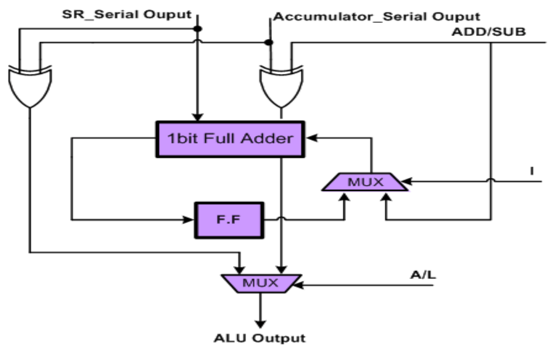

Fig.5 one bit Arithmetic and Logic Unit (ALU)

TABLE III ONE BIT ALU OPERATIONS

L	I	ADD/SUB	Operation
0	0	0	ADD
0	0,1	1	SUB
1	x	x	XOR

x (Don't care sign)
I = 0,1 (that means initializing Full adder to work as subtractor)

Memory Blocks: this component for some applications may be external, and for other applications internal. It contains three operative parts, all of them are working according to 8 bit control word generated from Control unit. First part, I/O buffer 16 x 32 bit component, which is responsible to receive 16 messages M_0, M_1,…, M_{15} and send them to Accumulator and SR registers for processing according to the mathematical algorithm for BMW 256 hashing core [5]. After finishing calculation, it receives the Hashed messages inside it. The second part is RAM component, it has locations for forty words and each one is 32 bit. 32 locations are specified for the data which are coming from the calculations in Bijective transform and expanded terms in BMW 256 hashing core [5]. The other 8 locations are used for temporary calculations. The last part contains ROM component divided in two parts. The first part contains 16 words. Each one is 32 bit, it represents the initial values $H_0^{(i-1)}$, $H_1^{(i-1)}$,..., $H_{15}^{(i-1)}$. The second part contains 16 words each is 32 bit; it represents the constants which are used to produce the second quadrupled pipe $Q_b^{(i)} = (Q_{16}^{(i)}, Q_{17}^{(i)}, \ldots, Q_{31}^{(i)})$ [5].

Control Unit: this component is designed to control the data flow in the design as well as the movement of data between Hash computation components. As shown in Fig.6, Control Unit contains four operative parts, first and second parts are 12 bit up counter and instruction encoder. These are working to generate 7 bit control for controlling the Operation Encoder. This encoder contains 38 operations. Four of them are basic operations as Add, Subtraction, XOR, Hold, and other operations are combinations between the basic operations according to the mathematical model [5].

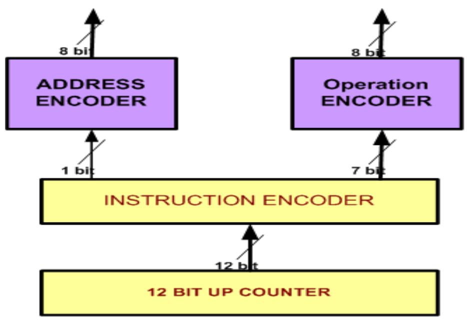

Fig.6 Control Unit

The Control Unit works according to 12 bit up counter and Instruction Encoder components. First, instruction Encoder gives order to Address Encoder to produce the data locations in Memory unit and that will translate to data, which SR, Accumulator and ALU Units will use it according to Operation Encoder. For example, if we would add two pieces of data in locations number 4 and 5 in memory unit, and write the result in location number 7. First, Instruction Encoder unit gives order to address encoder unit, which gives order to memory unit to choose locations number 4. Then Instruction Encoder unit asks Accumulator register to pick up the data from data bus and then the same operation happens with location number 5. But instead of Accumulator register, the SR register picks up data. Now, Instruction Encoder unit asks operation encoder unit to give order to one bit ALU unit to add these data and save it in Accumulator register. Finally, Instruction Encoder gives order to Address Encoder to pick up data and located in location number 7.

IV. BMW-256 HASHING CORE HARDWARE IMPLEMENTATIONS

This section presents the BMW-256 hashing core with (internal and external memory) implementations in several Xilinx devices such as Spartan, Virtex FPGA families and ASIC using 0.8 µm CMOS standard cell library as shown in Table IV and Table V. We used VHDL Model Sim SE 6.3 [7] and thereafter synthesizing the design using ISE 10.1 for FPGA implementation [8] and Synopsys (Design Vision) for ASIC implementation [9]. Our goal is to build BMW-256 hashing core with as low area as possible for applications that need low area implementations such as Mobile Trusted Module (MTM).

TABLE IV SYNTHESIS RESULTS FOR BMW 256 HASHING CORE WITH AND WITHOUT MEMORY BLOCKS IN DIFFERENT XILINX FPGA DEVICES

Xilinx device	Area (slices)		Estimated Clock Frequency	Estimated Throughput
	Internal Memory	External Memory		
Spartan 2E "Xcs400efg6 76-7"	2136	1369	56 MHz	1.05 Mbit/sec
Spartan 3A " XC3S400A-5FT256"	2092	1440	100 MHz	2 Mbit/sec
Virtex "XCV300-6PQ240"	2139	1347	60 MHz	1.14 Mbit/sec
Virtex II "XC2V500-6FG456"	2090	1359	125 MHz	1.1 Mbit/sec
Virtex-5 "5vlx30ff676 -3"	1980	122	264 MHz	5 Mbit/sec

TABLE V SYNTHESIS RESULTS FOR BMW 256 HASHING CORE WITH AND WITHOUT MEMORY BLOCKS IN ASIC

	Internal Memory	External Memory
Equivalent Gate Count (~Kgate)	13.5	4
Total Dynamic Power	22.6041 mW	9.0675 mW

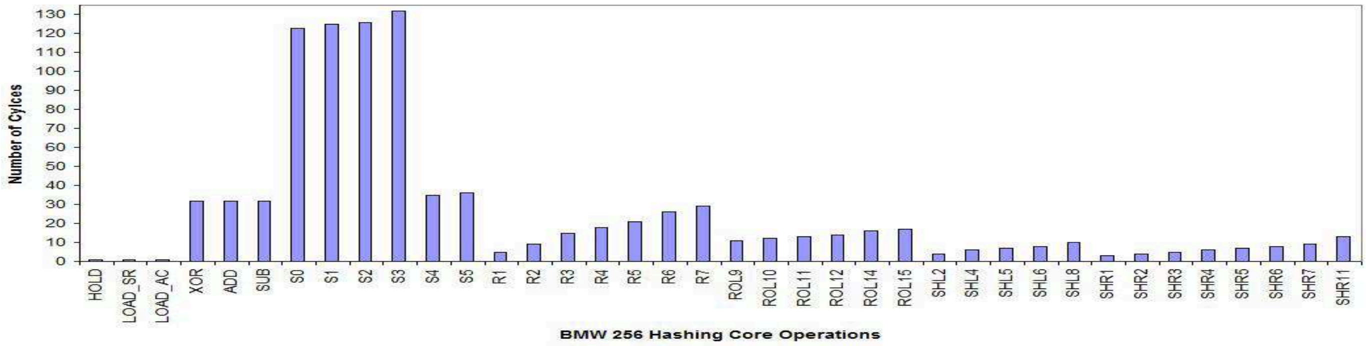

Fig.7 BMW-256 Hashing core Operations

TABLE VI IMPLEMENTATIONS COMPARISONS

Algorithm Name	**FPGA** Implementation			**ASIC** Implementation	
	FPGA Type	Area (Slices)	Estimated Throughput	Technology	Equivalent gate Count (K gate)
SHA-2 256 [10]	Virtex	* 2384 CLBs = 4768	291 Mbps	------	-------
SHA-2 256 [11]	Virtex E	5828	-------	-------	-------
Grøstl- 256 [12]	Spartan3	2486	404 Mbps	UMC 0.18 μm	17
Keccak- 256 [13]	Virtex 5	**444	70 Mbps	ST 0.13 μm	**5
Shabal- 256 [14]	Virtex 5	2307	1.33 Gbps	-------	------
BMW-256 (Proposed)	Virtex 5	**122	5 Mbps	CYB 0.8 μm	**4
		1980			13.5

* In Virtex each CLB is equal to two slices [15]
** Using External Memory

V. PERFORMANCE EVALUATION

This section presents comparisons between BMW-256 hashing core implementations and two different designs for SHA-2 [10,11] and also with candidates from SHA-3 competition [12,13,14].

From Table IV, Table V and Table VI, it's clear that BMW-256 hashing Core achieves the lowest area compared to other hash functions.

As shown in Table IV, our throughput is quite low. That happens because; some BMW-256 hashing core operations take much time in hardware as shown in Fig.7. These operations take much time because we are using one bit arithmetic logic unit and small RAM size (160 byte) according to the mathematical model for BMW-256 [5] in order to save area. This performance is sufficient for the relevant applications.

VI. CONCLUSION AND FUTURE WORK

In this paper, we introduced new FPGA implementations in different Xilinx devices, and ASIC implementation for the new hash function, Blue Midnight Wish – with 256 bits of message digest (BMW-256). The BMW-256 hashing core receives 16 message words, each one 32 bits, and it processes them. The goal is to use as small area as possible in order to minimize the hardware cost. In FPGA implementation, we have achieved around 35% lower area compared to SHA-256 on the same (or similar FPGA device), around 16% lower area compared to Grøstl-224/256 (one of the other SHA-3 candidates), around 72.5% lower area compared to Keccak-256 and around 15% lower area compared to Shabal-256. For ASIC implementation, we have achieved 4 Kgate using external memory and 13.5 Kgate using internal memory which is also lower compared to others. Due to the small area, our throughput is low compared to others.

For future work, it would be a challenge to improve this design, with optimized area but with high throughput. It can be done for example by repeating four times SR and Accumulator registers. That will reduce the number of cycles for computing the whole compression function, and will increase the throughput.

As a final note in this conclusion, we would like to emphasize that this work was performed on the version of Blue Midnight Wish that was initially proposed for the

SHA-3 competition. If any tweak is proposed for Blue Midnight Wish – this work will have to be updated accordingly.

References

[1] Wang, X., Yao, A. C., Yao, F. "Cryptanalysis on SHA-1", http://csrc.nist.gov/groups/ST/hash/documents/WangSHA1-New-Result.pdf.

[2] NIST Comments on *Cryptanalytic* Attacks on *SHA-1*, " *csrc.nist.gov/groups/ST/toolkit/documents/shs/NISTHashComments-final.pdf*"

[3] Cryptographic Hash Standards,"www.csee.wvu.edu/~katerina/Teaching/CS-465-Fall-2008/HashStandards.pdf"

[4] ECRYPT Benchmarking of Cryptographic Systems," http://bench.cr.yp.to/results-hash.html"

[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] M.Zeghid, B.Bouallegue, A.Baganne, M.Machhout, R.Tourki. "Reconfigurable Implementation of the New Secure Hash Algorithm". *In proceedings of the International Conference on Avability, Reliability and Security 2007 (ARES 2007)*, pp. 281-285, 2007.

[7] Model Sim PE/PLUS User's Manual. Model technology*, 2008*

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

*[9] Synopsys, "*Design Vision *User Guide Tutorial ," www2.informatik.uni-jena.de/~ct_mgr/a1/pdf/dvug.pdf*

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

[11] 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.*

[12] B.Jungk, S.Reith, J.u. Apfelbeck, "On Optimized FPGA Implementations of the SHA-3 Candidate Grøst"," http://eprint.iacr.org/2009/206.pdf"

[13] G.Berton, J. Daemen, M. Peeters, G. V. Assche, "Keccak sponge function family main document", http://keccak.noekeon.org/keccak-main-1.2.pdf"

[14] B. Baldwin, A.Byrne, M. Hamilton, N. Hanley, R. P. McEvoy, W. Pan , W. P. Marnane," FPGA implementations of SHA-3 Candidates: Cube Hash, Grøstl, Lan, Shabal and Spectral Hash", "http://eprint.iacr.org/2009/342.pdf"

[15] Virtex Architecture Guide, http://www.cse.unsw.edu.au/~cs4211/seminars/va/VirtexArchitecture.html#CLB_Overview

Mohamed El-Hadedy is PhD student at the Centre for **Q**uantifiable **Q**uality of **S**ervice in Communication Systems (Q2S) at Norwegian University of Science and Technology- Trondheim, Norway. He received the M.Sc Degree in electronic and communication engineering from Electronic and Communications department, Faculty of engineering, Mansoura University- Egypt in 2006. His research interests are Cryptography, Computer Security, Computer Architecture Design, FPGA and ASIC Implementations.

Danilo Gligoroski is professor at the Department of Telematics at Norwegian University of Science and Technology - Trondheim, Norway. He received the PhD degree in Computer Science from Institute of Informatics, Faculty of Natural Sciences and Mathematics, at University of Skopje – Macedonia in 1997. His research interests are Cryptography, Computer Security, Discrete algorithms and Information Theory and Coding.

Svein Johan Knapskog received his MS in Electrical Engineering from Norwegian University of Science and Technology – Trondheim in 1972. His research interests are Network Security, Cryptography, and Security Standards. Currently he is a head of the "Centre for Quantifiable Quality of Service in Communication Systems – Q2S", at the Norwegian University of Science and Technology, Trondheim, Norway.

Einar Johan Aas received his Ph.D. degree from the Norwegian Institute of Technology in 1972. In 1972, he joined SINTEF (The Foundation for Scientific and Industrial Research). Since 1981 he has been a full professor in Electronic Design Methodology at NTH, now the Norwegian University of Science and Technology. Dr. Aas is author and co-author of more than 400 technical and scientific publications. His current areas of research include VLSI design, verification and testing.