Penerbit Akademia Baru

Journal of Advanced Research Design
ISSN (online): 2289-7984 | Vol. 10, No.1. Pages 14-24, 2015

Performance Evaluation of Cryptographic Algorithms on Reconfigurable Hardware: MD5 based on Timing and Area Implementation

S. Suhaili[*,1,a] *and T. Watanabe*[2,b]

[1]Faculty of Engineering, Universiti Malaysia Sarawak, 94300 Kota Samarahan, Sarawak, Malaysia
[2]Graduate School of Information, Production and Systems, Waseda University, 2-7 Hibikino, Wakamatsu-Ku, Fukuoka 808-0135, Japan
[*,a]*sushamsiah@feng.unimas.my,* [b]*watt@waseda.my,*

Abstract – *Cryptographic algorithm has become one of the most important aspects in hardware implementation of embedded security system design. Message Digest (MD) is one of the cryptographic algorithms that can be used in any security design application. Nowadays, designing the high speed, low power, and small area implementation of cryptographic algorithms on reconfigurable hardware is one of the critical subjects for hardware application. The purpose of this paper had been to analyse the structure of MD5 hash function for high performance implementation in order to obtain small area implementation, as well as to increase the speed of the design on FPGA. In this paper, the frequency maximum for MD5 design with both grey and binary signal encoding is discussed. The results retrieved from the analysis showed that the differing results were caused by the switching bit of the signal input. Besides, the frequency maximum of MD5 that employed binary signal encoding provided the highest frequency maximum with smaller area implementation. By using grey signal encoding, the frequency maximum was almost similar to the MD5 binary signal encoding, but it suffered larger area implementation. On top of that, this research focused on timing and area implementation of the design, where TimeQuest timing analyser was applied to optimize the output of the design.*

Keywords: FPGA, Frequency Maximum, Grey, HDL, MD5

1.0 INTRODUCTION

Cryptographic algorithms on reconfigurable hardware have been widely used for data protection in automotive, multimedia content, or any cryptographic material. Therefore, studies and analyses pertaining to cryptographic algorithms, which involve the encryption technique, have become an important aspect in embedded digital security system design for the next generation. Moreover, cryptographic hash function, or better known as message digest algorithms, is an algorithm that translates a random string of characters into hash code output. Hash function has been widely used for cryptographic application and it is useful for message authentication because it is strong enough against collision. One of the most famous hash functions is the MD5 message digest, which was developed by Ronald Rivest [1]. Moreover,

Penerbit Akademia Baru
Journal of Advanced Research Design
ISSN (online): 2289-7984 | Vol. 10, No.1. Pages 14-24, 2015

MD5 algorithm is intended for digital signature applications, where a large file must be compressed in a secure manner before being encrypted [2].

In addition, high performance of the hash function design is important to improve the throughput of the design since nowadays all systems require fast implementation. Besides, efficient hardware implementation is one of the solutions of cryptographic algorithms on reconfigurable hardware. Therefore, modification of the algorithms with specific target device, such as structure and resources, needs to be taken into account. Apart from that, design techniques and design tools also play important roles in the design process. Hardware Description Language (HDL) coding styles can enhance the speed of the design with some modification on the placement of the register. This is one of the methods employed to increase the frequency maximum of the design. The motivation of this research was to analyse the structure of MD5 hash function as it is important for message authentication code application. For some security application reasons, it is good to have a short signature. Therefore, signing off long messages to the hash function is the solution to this problem.

2.0 METHODOLOGY

In this research, the project began with pre-processing the message of MD5 by calculating the bit length of arbitrary input. It was a transformation that involved variable size input, *m*, and returned a fixed-size string called hash code. The modern hash function tries to improve the internal compression function and the sequence of the processing message [2]. Algorithm 1 shows the pseudo code for MD5 compression function.

Algorithm 1: MD5 Compression Function

```
In    clk, rst, message [31:0]
Out   Hash_MD5Output [127:0]
1  Message[31:0]
2       {
3        Message Padding
4        Append Message
5        Initial input A[31:0],B[31:0],C[31:0],D[31:0]
6         {
7          Round 1 → F: 16 Steps
8          Round 2 → G: 16 Steps
9          Round 3 → H: 16 Steps
10         Round 4 →  I: 16 Steps
11        }
12      Hash output + initial input A[31:0],B[31:0],C[31:0],D[31:0]
13    }
14 Hash_MD5Output[127:0]
```

The process of message padding for *n*th-bit message was padded with a single 1-bit at the end of the message bit, while the rest of the bit had the value of 0-bit until the length of the message was congruent to 448 mod 512. After padding the bit of the message input, the remaining 64 bits were reserved for appending the message length with all zeroes, except for the last bytes, which was the length of the message counter. Therefore, the overall message length, M, was equal to 512 bits. Figure 1 illustrates the process of the MD5 compression function. The MD5 message algorithm started processing with four-word constant initial value input of 32-bit register (A,B,C, and D) buffer initialization, as shown in Table 1. The process was executed

Penerbit Akademia Baru

Journal of Advanced Research Design
ISSN (online): 2289-7984 | Vol. 10, No.1. Pages 14-24, 2015

until four rounds with each round consisted of 16 steps. The iterative process for each round was continued until round 64 in order to obtain the hash code output of MD5 algorithm.

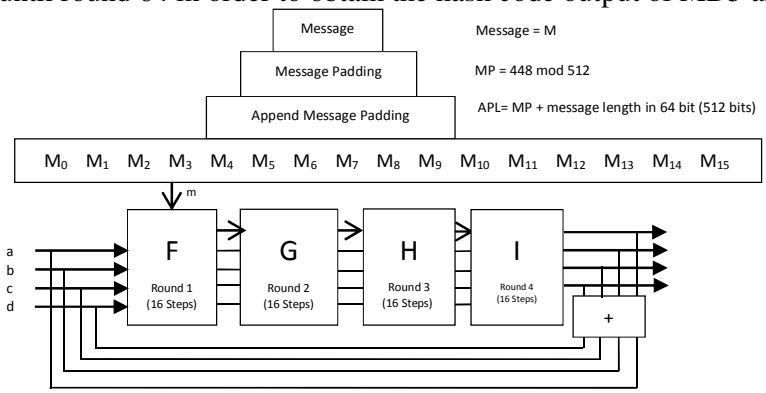

Figure 1: MD5 Compression Function

Table 1: Buffer Initialization

	Normal Value	Little-Endian Format
A	32'h01234567	32'h67452301
B	32'h89abcdef	32'hefcdab89
C	32'hfedcba98	32'h98badcfe
D	32'h76543210	32'h10325476

The security compression comes from the original input message bit-length in order to obtain the shorter bit length of the output hash code. In this design, there were four non-linear functions; F, G, H, and I. Hence, 64 steps were performed to complete the overall algorithms. Besides, equations 1, 2, 3, and 4 show the auxiliary function of four rounds in MD5 algorithms, where $\wedge$, $\vee$, $\neg$, and $\oplus$ represent logical AND, OR, NOT, and XOR operations respectively. All the operations were executed in the little-endian format. Moreover, Rivest chose the little-endian architecture for interpreting the message of a sequence of 32-bit because based on his observation on several processors; little-endian format offered fast processing [1] of hash function that strongly depended on the security of the compression function [1]. The following are the related terms:

$$F(B,C,D) = (B \wedge \mathrm{C}) \vee (\neg B \wedge \mathrm{D}) \quad (1)$$

$$G(B,C,D) = (B \wedge \mathrm{D}) \vee (\mathrm{C} \wedge \neg D) \quad (2)$$

$$H(B,C,D) = B \oplus C \oplus D \quad (3)$$

$$I(B,C,D) = C \oplus (B \vee \neg D) \quad (4)$$

Besides, i was considered as a step index of the MD5 algorithms. For each step of the algorithm, 64-element of the 32-bit constant table T[i], which was constructed from sine function, had been applied. Let << S denote a left circular shift by S bits. Then, let $M_i[k]$ denote the k-th 32-bit word of M_i. The operations of the MD5 function were executed from i = 0 until i= 63. Equation 5 shows the operation for MD5 rounds with *Func*(B,C,D), which represented function F(B,C,D) for the first round, G(B,C,D) for the second round, H(B,C,D) for the third round, and I(B,C,D) for the fourth round.

Penerbit Akademia Baru
Journal of Advanced Research Design
ISSN (online): 2289-7984 | Vol. 10, No.1. Pages 14-24, 2015

$$(A, B, C, D, M_i[k], S, T[i])$$
$$A = B + ((A + Func(B, C, D) + M_i[k] + T[i]) \ll S)$$
$$A \leftarrow D, B \leftarrow A, C \leftarrow B, D \leftarrow C \quad (5)$$

The final step of MD5 algorithms was to obtain the output of the hash function. By adding the initial value (IV) with the last output from the compression function *I(B,C,D),* the hash code was produced. Then, the output hash code had to be converted to its normal value in order to obtain the final hash output.

2.1 MD5 Step Function and Frequency Maximum

The architecture of MD5 step function was implemented based on basic architecture, which is known as iterative looping. Iterative looping is looping that is based on the same block of step function module with a number of iteration round. There were 64 rounds as the output was fed back to the input of the design. This type of design utilized a small logic area, but it consumed more clock cycle. Moreover, the structure of the MD5 step function architecture highly affected the throughput of the output hash. In this paper, Verilog code was used to construct the MD5 logic design. Figure 2 shows the structure of the MD5 step function. It had been the main operation for the implementation of MD5 algorithms, where 64 steps were run iteratively. Based on MD5 step function, four operation functions, such as F, G, H, and I, were executed continuously during the process.

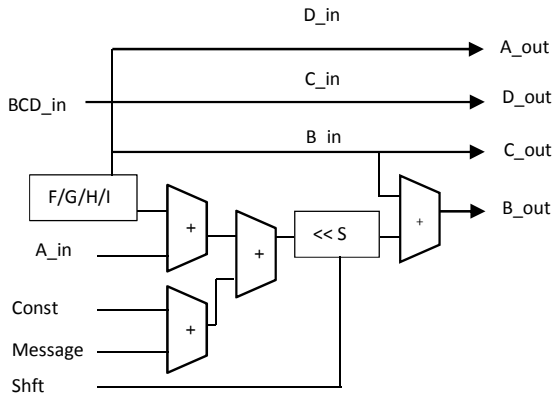

Figure 2: MD5 Step Function Architecture

It had been the most important step in the MD5 hash function where the process of four functions; F, G, H, and I, had to be carried out iteratively to complete the overall hash processing. The architecture of MD5 algorithms was successfully tested. Modification on the structure for functions F, G, H, and I improved the frequency maximum of the design. Besides, CAD tools played an important role in producing better results. In this paper, two types of signal encodings were employed to design the MD5 algorithm in order to evaluate the effects of MD5 hash function during the synthesis and the implementation processes on reconfigurable hardware. There are three main inputs to the step function module, such as Message value, Constant, and Shift. Message value, Constant, Shift are represented by M_value, Const, and Shft, respectively, as shown in equation 6.

Penerbit Akademia Baru
Journal of Advanced Research Design
ISSN (online): 2289-7984 | Vol. 10, No.1. Pages 14-24, 2015

TConst = {Const[43:12], Shft[11:4], M_value[3:0]} (6)

Meanwhile, the register block for inputs A, B, C, and D only occurred in the multiplexer module, where all the inputs were processed before the step function module. Therefore, there was no register block in this step function module. Moreover, in order to improve the performance of the design, a small modification on the HDL code style, especially in the step function module, could be done. The main focus of this project was to increase the speed of the MD5 design. Thus, the frequency maximum of the design had to be improved. Step function module is one of the main target modules that can increase the speed of the design. Furthermore, moving the placement of register could also increase the performance of the design. Besides, adding output registers can maximize the clock frequency, while adding input registers can minimize both the setup and the hold window.

The frequency maximum of the design can be created by using the sequential system delay. In this project, Altera Quartus II Arria II GX was used as the CAD tools to implement the design. Hence, the frequency maximum was calculated during the execution of the synthesis process based on board level of family device. The delay of the design came from the gate propagation delay where the digital logic was constructed from the transistors because the transistors functioned as on/off switches. Thus, a smaller transistor has the ability to switch faster than a large transistor. Therefore, the switching delay of the transistor created delay in the logic gate. This propagation delay (t_{pd}) contributed to pin delays where pin delays are based on cell and net delays. Hence, propagation delays for modern integrated circuit were considered where the setup(t_{su}) and the hold(t_{hd}) time had been taken into consideration in order to ensure the stability of the input. Figure 3 illustrates the condition of a stable input via setup and hold time.

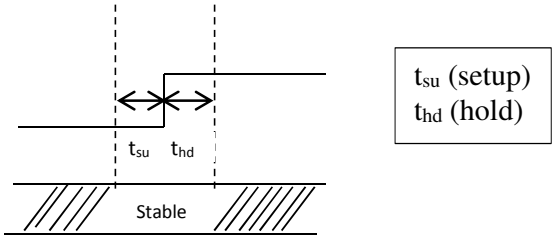

Figure 3: Setup and Hold Time

In addition, there are three types of sequential system delays, such as pin-to-pin propagation delay (t_{p2p}), clock-to-output propagation delay (t_{c2Q}), and register-to-register delay (t_{R2R}). In the modern circuit designs, t_{R2R} is the largest delay among all these three types of delays. This delay starts at output of the register to function as the input of another register where the route always involves at least two registers. For example, if the sequential system design consists of two registers and there are combinational logic gate, the overall delay for t_{R2R} is shown in equation 7. Similarly for board level delay, if there are two blocks of sequential system design, the overall t_{R2R} can be calculated with equation 8. By considering U_1 and U_2 as two different blocks of sequential system design, the combination started at output U_1until input U_1.

$$t_{c2Q_{FF}} + t_{comb\_R2R} + t_{su\_FF} = t_{R2R} \quad (7)$$

$$U_{1\_tc2Q} + U_{2\_tpd} + U_{1\_tsu} = t_{c2Q\_sys} \quad (8)$$

Penerbit Akademia Baru
Journal of Advanced Research Design
ISSN (online): 2289-7984 | Vol. 10, No.1. Pages 14-24, 2015

On the other hand, the maximum clock frequency was calculated from sequential delay until board level delay where the longest delay within each of these types of delay was obtained. Therefore, the longest delay amongst these three types of sequential system delay had been the worst case of the design. This delay was the minimum delay for a circuit to function properly. Thus, this minimum clock period allowed for gate output to reach a stable value. Besides, the setup and hold adjustment in the board level could also improve the frequency maximum of the system design. Hence, external signal to the circuit must not violate t_{su} before the clock and t_{hd} after the clock at the input to the internal register. Nevertheless, the adjustment in the board level can be executed between data and clock to input register where the value for two different paths can give the total t_{su} and t_{hd}.
Figure 4 illustrates the adjusted setup and hold time. As mentioned earlier, the worst case for the setup time was the longest delay from data input to register and the shortest delay from clock to input register, whereas the hold time setup was the opposite from the worst case for time setup. Equations 9 and 10 represent the total time for setup and hold after adjustment. The negative value of t_{hd} was specified as zero. Hence, the total setup and hold time affected the improvement of the maximum clock frequency.

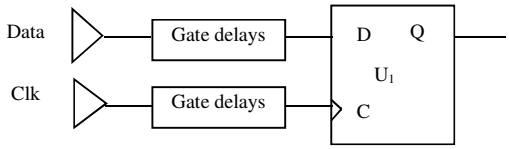

Figure 4: Adjusted setup and hold time

$$(T_{pd\_data(MAX)} - t_{pd\_clk(MIN)}) + t_{su\_ff} = T_{su\_total} \quad (9)$$

$$(T_{pd\_clk\ (MAX)} - t_{pd\_data(MIN)}) + t_{hd\_ff} = T_{hd\_total} \quad (10)$$

2.2 MD5 Architecture Design

After designing the MD5 step function, the next stage was to construct the architecture of the MD5 design. The architecture of the MD5 design began with four initial inputs and four different non-linear functions, such as F, G, H, and I. The 32-bit input of the MD5 algorithm used in_mem[511:0] memory to keep the 448-bit data where the data for the last 64-bit had been calculated by checking each byte of the message input. In this project, the input was padded with a padding unit until 448-bit because the message length could only be calculated after 448-bit. Besides, the counter message calculated the length of the message in the last 64-bit, and the overall result of the message input was 512 bits. With that, the architecture of the MD5 design had been successfully tested.

During the synthesis and the implementation processes of the design, the frequency maximum was obtained and the design must meet the requirement of timing analysis. Thus, the analysis of the clock constraint had to be considered. Based on the Altera Quartus II as the CAD tools of the design, the TimeQuest timing analyzer was considered as the best solution to offer the clock constraint to the input of the design. The timing must be met to avoid violation in the design in order to obtain a positive slack, which is the difference between data arrival and data required. Figure 5 shows the top level of the MD5 design, which is the block diagram of the design after synthesis and implementation processes. It was developed by using the Altera Quartus II CAD tool when no syntax error occurred during the synthesis and the

Journal of Advanced Research Design
ISSN (online): 2289-7984 | Vol. 10, No.1. Pages 14-24, 2015

implementation processes. Moreover, there were four inputs; clk, rst, load, and message [31:0]. If there was a message, the load input was logic 1 in order to receive the 32-bit input message to the Message_input_Opt module. The process of the message continuously looped until 64 rounds. Finally, the 128-bit hash_MD5Output output was obtained.

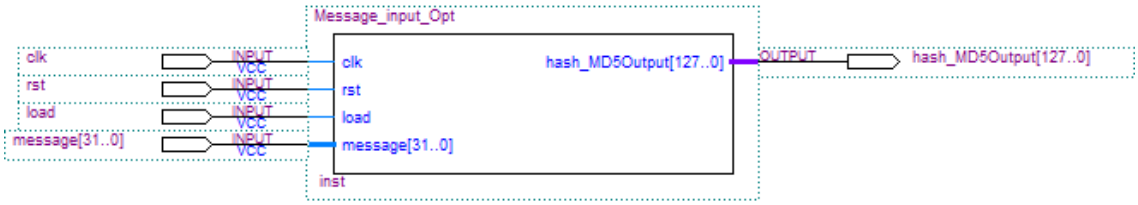

Figure 5: Top Level MD5 Algorithm

In addition, various types of signal encodings can be designed to encode the individual states, such as binary encoding, one-hot encoding, and grey encoding. Signal encoding is one way to reduce power consumption of the design that includes the finite state machine. In this project, only two types of signal encodings were used, namely binary and grey encodings. Power consumption can be reduced by using the grey coding if compared to binary encoding because there is only 1-bit of the state switching during the transition process. Figure 6 illustrates the state encoding for both binary and grey encodings. By considering A as input and E as output of the system, the state binary encoding of the finite state machine starts from 0,1,2,3, and 4, whereas for grey encoding, the state of the sequence starts from 0,1,3,4, and 6. Besides, Figure 6 clearly shows that the state of the grey signal encoding switches 1-bit for each transition.

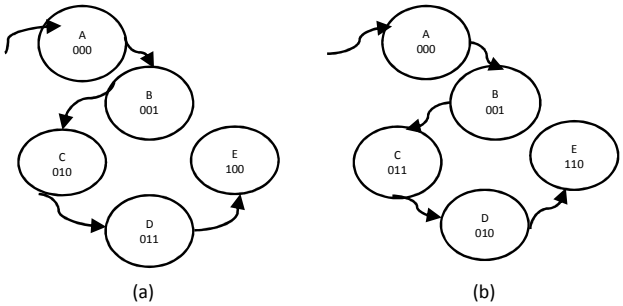

Figure 6: State Encoding: (a) Binary Encoding (b) Gary Encoding

In this project, the top levels of MD5 algorithms consisted of various types of modules. These modules were connected from the input module until the output module of the design. The process of the message was mainly operated by counter MD5 module, as illustrated in Fig. 7. This counter module controlled the input message of MD5 where the 32-bit message was divided into four 8-bit messages to calculate the length of the message. The last 64-bit was the bit length of the message. Therefore, the counter counted the message at round 14 of round 4, where the entire bit message was calculated byte by byte.

Journal of Advanced Research Design
ISSN (online): 2289-7984 | Vol. 10, No.1. Pages 14-24, 2015

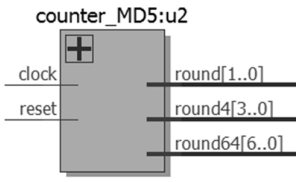

Figure 7: Counter MD5 module

Table 2 shows the first 16 rounds of round bit 0 that had been executed during the hash processing for both binary and grey encodings. 1-bit of round output consisted of 16 clock cycles, whereas 1-bit of round 4 output was equal to 4 clock cycles. In this project, the round number for the input to be padded with the padding unit was round 56 of round 64, where each round consisted of one clock cycle. The processing of the message was completed until round bit 4. Thus, the calculation for the message length could be calculated at round 14 of round 4.

Table 2: Counter Round Output

Round	Binary	Gary
round	0	0
round4	14	9
round64	56	36

Furthermore, gating the clock at the input of Step Function module is one of the techniques to save dynamic power consumed within the System on Chip (SoC). In this project, MD5 clock gating was designed for both binary and grey encodings. Block enable was inserted together with the clock to convert them into the clock-gated implementation, as shown in Figure 8. Moreover, the evaluation of the performance had been based on timing and area implementation, which determined the speed of the design.

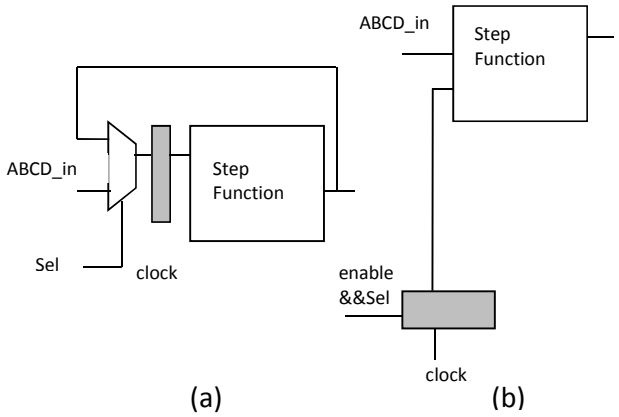

Figure 8: Gating the clock (a) Normal Implementation (b) Clock-Gated Implementation

Journal of Advanced Research Design
ISSN (online): 2289-7984 | Vol. 10, No.1. Pages 14-24, 2015

3.0 RESULTS AND DISCUSSION

The comparisons of frequency maximum for different types of MD5 designs were analysed. In this analysis, the input message, m, was tested. Table 3 illustrates the synthesis and the implementation results for MD5, MD5_Gray, MD5_Gating, and MD5_Gray_Gating. Moreover, two types of signal encoding designs, as depicted in Table 3, namely MD5 and MD5_Gray, represented MD5 binary and MD5 Gary encodings respectively. Meanwhile, the other two designs of MD5 were for clock-gated implementation. MD5_Gating was clock-gated for binary encoding, whereas MD5_Gray_Gating illustrated clock-gated for Gary encoding. Simulation was carried out by using ModelSim Altera to verify the correctness of the Verilog coding output. From this table, MD5 gave the highest frequency maximum with clock constraint 7. As for the 900mV 100C model, the frequency maximum for MD5 was 145.94 MHz with a slack for the setup time at 0.148 ns, while for the 900 mV -40C model, the frequency maximum for MD5 was about 152.32 MHz with a slack for the setup time at 0.435 ns. The performance of the overall design was almost similar where the frequency maximum for four different types of the design had been at the same range. From Table 3, the frequency maximum for MD5_Gray had been the lowest among all these designs. Moreover, by gating the clock into the MD5 design, the speed of the design could be improved. From this analysis, the switching bit of the state did not affect the performance in terms of speed of the design. Furthermore, these designs reduced the power of consumption because of the switching bit of the state during the transition process.

Table 3: Synthesis and Implementation of MD5 Design

MD5 Design	Device	Clock constraint	Slow 900 mV 100C Model		Slow 900 mV -40C Model	
			slack(ns)	*fMax* (MHz)	slack (ns)	*fMax* (MHz)
MD5	Arria II GX	7	0.148	145.94	0.435	152.32
MD5_Gray	Arria II GX	7.5	0.344	139.74	0.657	146.13
MD5_Gating	Arria II GX	7	0.113	145.2	0.37	150.83
MD5_Gray_Gating	Arria II GX	7	0.093	144.78	0.429	152.18

The results of MD5 design implementation on reconfigurable hardware relied on the choice of FPGA family devices and the structure of HDL code chosen. Therefore, after designing the MD5 architecture, the processes of synthesis and implementation of the design had to be executed in order to obtain the critical path of the design. In this project, the TimeQuest timing analyzer was used to optimize and to produce better *fMax* with clock constraint input. Each design had to specify the accurate value for clock constraint. Therefore, the iteration process was executed several times for all designs. Moreover, there had been a number of reasons for applying the TimeQuest in designing the MD5 for this project. It was easier to use since it provided simple GUI and interactive reporting for analysing timing, industry standard, and more powerful with SDC format that allowed for faster, easier description, and analysis of advanced design constructs. The main purpose of designing the different types of MD5 designs for this project was to evaluate the performance of the design in terms of speed and area based on Arria II GX family devices from Altera Quartus II. Besides, the clock constraint of the

design could increase the performance of the design, as well as meet the timing requirement with positive slack. In other words, the critical path of the design could be reduced.

Table 4 shows the area implementation of MD5 designs. From this table, the smallest area for implementation was MD5. The usage of combinational ALUT was 1989, while the total register was 1713. From this analysis, MD5 binary encoding offered high speed and small area implementation. However, MD5_Gray_Gating used large area implementation, which was about 2221 for combinational ALUT and 1936 total register even though the frequency maximum for MD5_Gray_Gating was more or less similar to the MD5 design, but it suffered from large area implementation. Based on these results, the area implementation of combinational ALUT for four different MD5 designs had been in the range of 1989 to 2221, whereas for the total register, the range was from 1713 to 1936. In fact, there were some differences in terms of area implementation for different types of MD5 designs. In order to reduce the area implementation, the register implementation had to be reduced.

Table 4: Area Implementation of MD5

MD5 Design	Combinational ALUT	Total Register
MD5	1989	1713
MD5_Gray	2140	1935
MD5_Gating	2080	1713
MD5_Gray_Gating	2221	1936

3.1 Performance Evaluation

In fact, there are several other researchers who have implemented the MD5 designs based on both Altera and Xilinx CAD tools. Table 5 shows the comparison results for synthesis and implementation of MD5 design. The output results for MD5[3], which was based on iterative technique improved the frequency maximum of the design by using Carry Save Adder (CSA) with 102.7 MHz of frequency maximum. On the other hand, MD5[4] provided fast performance and a large area of logic resources for pipelining design, whereas the results based on iterative implementation with Xilinx provided a small logic area for implementation with 78.3 MHz of frequency maximum. Furthermore, the iterative looping for MD5[5] with Xilinx, which was iterated several times to perform the complete loop, contributed to 60.20 MHz frequency maximum. While the implementation of MD5[6] showed that the device utilization of iterative design was significantly small, 880 slices with 21 MHz of frequency maximum had been the result. Based on these results, it also showed that the implementation of MD5 on Stratix II GX offered 103.32 MHz with clock constraint 11, whereas MD5[3] contributed to only 102.7 MHz. However, this design employed a large area for implementation. Besides, it was clearly observed that Arria II GX family device provided high frequency maximum, which was 152.32 MHz, with combinational ALUT at about 1989 and the usage of the total register was 1713. Hence, Arria II GX could contribute to high performance of the design, as well as increase the throughput of the design.

Table 5: Synthesis and Implementation Comparison of MD5 design

Design	Device	fMax (MHz)	Combinational ALUT (Altera)	Total Register (Altera)	Slices (Xilinx)
MD5	Arria II	152.32	1989	1713	-
MD5	Stratix II GX	103.32	1847	1633	-

MD5[3]	Stratix II GX	102.7	1352	462	-
MD5[4]	Virtex II	78.30	-	-	1325
MD5[5]	Virtex II	60.20	-	-	1369
MD5[6]	Virtex V	21	-	-	880

4.0 CONCLUSION

The architecture of MD5 design for both grey and binary signal encodings had been successfully synthesized and analysed with Altera Quartus II Arria II GX, as well as Stratix II GX by using Verilog code. By giving different values of constraint to the clock of MD5 design, the frequency maximum of the design increased significantly. Therefore, the improvement of frequency maximum for the design could enhance the throughput of the MD5 design. This leads to the high performance of the MD5 design. Furthermore, various other methodologies or techniques can be implemented to increase the frequency maximum in order to obtain high throughput of the implementation. HDL coding styles also could affect the implementation results of the design. One of the key technologies to increase the performance of the design is pipelining, but this structure had the tendency to increase resource utilization. In addition, a circuit with many XOR gates would produce many glitches that could consume more power due to fast switching activity. Besides, FPGA offers higher performance and low cost design with the development tool of HDL. Hence, it is the best choice when dealing with algorithm, but it suffers from high power consumption. Thus, the usage of different types of techniques in reducing the SoC power consumption without high-level circuit design tools can produce low power circuit design. In the near future, FPGA may be expected to be cost-effective, even for small-end applications.

REFERENCES

[1] F.R. Henriquez, N.A. Saqib, A.D. Perez, C.K. Koc, Cryptographic algorithms on reconfigurable hardware, Springer series on Signal and Communication (2006) 189-201.

[2] R.L. Rivest, The MD5 Message-Digest Algorithm, RFC 1321, MIT Laboratory for Computer Science and RSA Data Security Inc., April (1992).

[3] Y. Wang, Q. Zhao, L. Jiang, Y. Shao, Ultra high throughput implementations for MD5 hash algorithm on FPGA, high performance computing and applications, Lecture Notes in in Computer Science 5938 (2010) 433-441.

[4] K. Jarvinen, M. Tommiska, J. Skytta, Hardware implementation analysis of the MD5 hash algorithm, In Proceedings of the 38th Hawaii International Conference on System Sciences, (2005).

[5] J.M. Diez, S. Bojanic, Lj. Stanimirovic, C. Carreras, O. Nieto-Taladriz, Hash algorithms for cryptographic protocols: FPGA implementations, 10th Telecommunications forum TELFOR'2002, Belgrade, Yugoslavia, Nov. 26-28 (2002).

[6] J. Deepakumara, H.M. Heys, R. Venkatesan, FPGA implementation of MD5 hash algorithm, Proceedings of the Canadian Conference on Electrical and Computer Engineering, CCECE 2001, Toronto, Canada, 2 (2001) 919-924.