Four-stage Pipelining for Two Messages in MD5 Implementation with Data Forwarding

Hoang Anh Tuan†, Katsuhiro Yamazaki † and Shigeru Oyanagi †

Abstract Hash algorithms or message digest algorithms such as SHA and MD5 are used to generate a unique message digest for an arbitrary message. They contain many internal loops that can be used for pipelining implementation. This paper describes an improvement of 3-stage pipeline implementation of MD5 algorithm. Data forwarding together with alternately message digests computation are used to remove data dependency among steps in the main loop, helps to break the main computation of the MD5 algorithm into 4-stage pipeline for implementation on FPGA. The implementation achieves a throughput of 1.04 Gbps with 1064 hardware slices and 1 BRAM on Xilinx Vertex-II XC2V4000-6 FPGA.

1. Introduction

Hash algorithms are used to generate a unique message digest for an arbitrary message. It contains many internal loops in the main computation module. Several step/round-based pipelining MD5 implementations were given [2]. Besides, the internal loop of the main computation was also utilized for 3 stage pipeline MD5 implementations with data forwarding technique [3-4]. However, the unbalance in computation time among stages, which reduces the general speed and so throughput of the system, requires an improvement to 4-stage pipelining implementation with alternately message digests computation in combination with data forwarding methodology.

2. Main computation and data movements in one message MD5

The main module of the MD5 algorithm is used to solve the equation

$$A = B + ((A+T[i]+X[k]+Func(B,C,D))<<< s) \quad [1]$$

$$A \leftarrow D; B \leftarrow A; C \leftarrow B; D \leftarrow C$$

in which A, B, C, D are variables that contain digest results, initiated by constants. Func is a combination of other four functions (F, G, H, I), which are used in four rounds respectively. X is the input message, which is considered as 16 32-bit keys. T is a constant table which contains 64 values, and <<< represents a circle shift left operation.

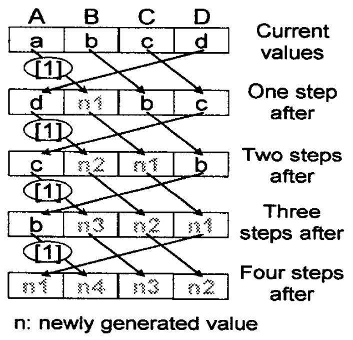


Figure 1. Trace of A within 4 steps

Figure 1 shows data movements of the message digest data through 4 steps. Assume that current values of the operands A, B, C, and D are a, b, c, and d, the values of the variable A will be a, d, c, and b depend on the step number. This means that those values can be forwarded to A for advanced use.

3. Data dependency, data forwarding, and pipeline stages

Equation [1] can be re-written into equation [2] for 4-stage pipelining model.

tempB = B	[2a]	
AT = A+T[i]	[2b] AT stage	
ATX = AT + X[k]	[2c] ATX stage	
S = (AT+Func(B,C,D))<<< s	[2d] Shift stage	[2]
B = B + S	[2e] Final stage	
A ← D; C ← tempB; D ← C	[2f]	

Equation [2b] and [2c] have very light data dependency on the previous values of the operands A, B, C, and D because we can forward values of D, C, and B into A for advanced use as can be recognized in Figure 1. However, equations [2d] and [2e], which are used to compute new value of operand B, are tightly dependent on the previous value of B. In other words, [2d] has data dependency on [2e] and vice versa. Therefore, no pre-computation can be done for these two equations.

Figure 2 shows data dependency among the operands and the equations. It clearly shows that AT and ATX can be pre-computed any time but S and B ([2d] and [2e]) must be computed one after the other.

From Figures 1, 2 and equation [2], the 4 stages of pipeline for one message is given in Figure 3. The data dependency of S into B ([2d] depends on [2e]) requires one more clock to complete, so, generates a spare time of all stages in every 2 clocks. Hence, this spare time can be used to calculate the message digest of the other message.

Figure 4 shows the 4 stages computation with 2 messages. The 2 messages are computed alternatively for each step, in which, numbers x.y show the message number (x) and the step number (y).

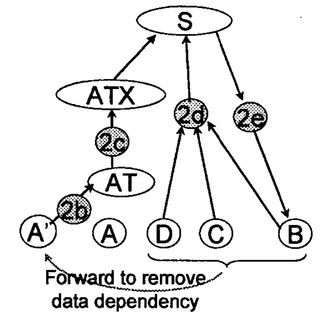


Figure 2. Data dependencies among operands and equations

† Ritsumeikan University, Graduate School of Science and Engineering

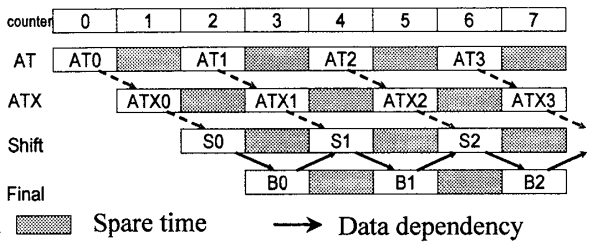


Figure 3. 4-stage pipelining for one message

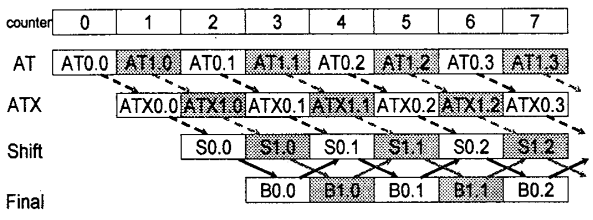
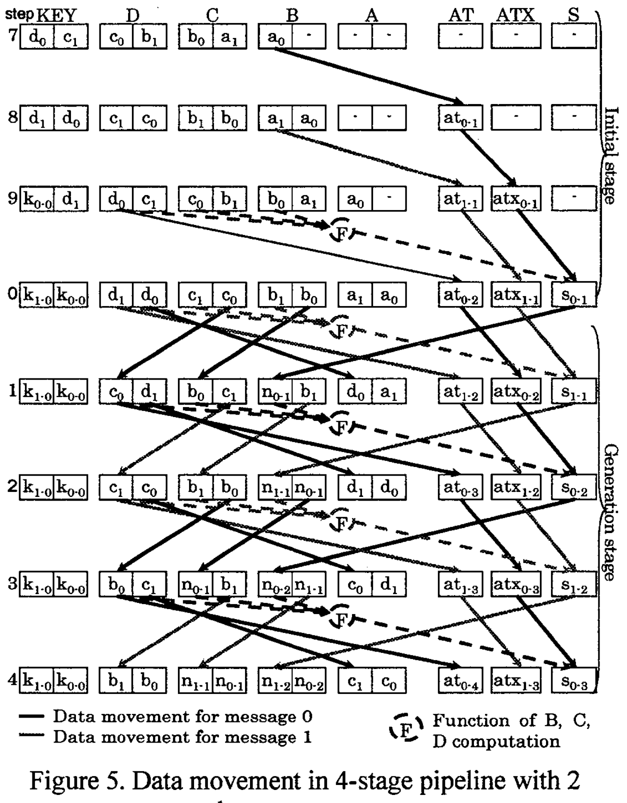


Figure 4. 4-stage pipelining with 2 alternate messages

Therefore, 2 messages can be computed after 128 clocks (64 steps for each).

4. Four-stage pipelining design and implementation

Figure 5 shows the data movement in 4-stage pipeline with 2 messages, message 0 and message 1. Supporting for the 2 message digest, the message digest operands A, B, C, and D are extended into 64 bits each. In general, the high 32 bits are used for message digest of message 1 while the remaining low bits are used for the message 0.



step
KEY
D
C
B
A
AT
ATX
S
Initial stage
Generation stage
Data movement for message 0
Data movement for message 1
F
Function of B, C, D computation

Figure 5. Data movement in 4-stage pipeline with 2 alternate messages

Two stages are shown in Figure 5. The initial stage is used to read the initial values of operands A, B, C, and D into their locations. The generation stage is used to create the message digest of the message.

The initiation stage will take around 9 clocks to read data into its location as can be seen in step 2 (left hand side) by shifting data from left to right.

After the initial stage, the generation stage start. Supporting for the computation of equation [2e] at the first step of generation stage, the computation of [2b] must start 3 steps before, at step 7 of initial stage by adding a_0 (message digest A of message 0) with the constant. Hence, Figure 5 shows from step 7 of the initial stage. During this stage, the 2 data parts of one operand are swapped during the computation to keep the locations fixed to the computing modules (AT, ATX, Shift, and Final). Then, the generation stage will starts from 0, counts up and ends at 127 (128 clocks) before generating the final digest result.

The same operation with same input data for all modules allows us to implement this design efficiently. The keys (input message) are designed to locate in the BRAM in order to save the valuable registers.

5. Implementation results and discussions

The 4-stage pipelining MD5 design was implemented on the Virtex-II XC2V4000-6 devices and compiled by Xilinx ISE 8.1 version. The throughput of the design achieves 1.04 Gbps, equals to 137.6 MHz of frequency, and utilizes 1,064 hardware slices and 1 BRAM. The hardware size increases 1.2 times while the throughput increases 1.4 times in comparison with the 3-stage pipeline implementation [4]. The hardware size can be reduced if the BRAM is utilized to implement the constant table.

6. Conclusion and Future work

This paper described the implementation of the core of MD5 algorithm into 4-stage pipeline. The results show that this architecture achieves throughput of 1.04 Gbps while requiring 1,064 hardware slices and 1 BRAM on the XC2V4000 device, which shows a good tradeoff between hardware and throughput. The methodology should be applied into other hashing algorithms implementations.

References

[1] RFC 1321 – The MD5 Message-Digest Algorithm

[2] K.Jarvinen et al.: Hardware Implementation Analysis of the MD5 Hash Algorithm, Proc 38th IEEE International Conference on System Sciences-2005.

[3] Hoang Anh Tuan et al.: Three stages pipelined MD5 implementation on FPGA, FIT2007, LC-008, pp. 61-64, 2007.

[4] Hoang Anh Tuan et al.: Pipeline MD5 Implementations on FPGA with Data Forwarding, IEICE Technical Report, RICONF2007-27, pp. 71-76, 2007.