JTE HCMUTE
ISSN: 1859-1272

TẠP CHÍ KHOA HỌC GIÁO DỤC KỸ THUẬT
Trường Đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh
Website: https://jte.edu.vn
Email: jte@hcmute.edu.vn

Design and Evaluation Secure Hash Algorithm SHA-256 On ZynQ-702 Hardware Platform

Thi Hong Hieu Nguyen[1], Thi Anh Duong Tran[1], Anh Duong Van[1], Nhut Minh Ho[2], Ngo Lam Nguyen[1], Quang Phuc Truong[*1]

[1]Ho Chi Minh City University of Technology and Education, Vietnam
[2]Posts and Telecommunications Institute of Technology, Vietnam

[*]Corresponding author. *Email: phuctq@hcmute.edu.vn*

ARTICLE INFO

Received: 06/06/2023
Revised: 12/07/2023
Accepted: 04/08/2023
Published: 28/08/2024

KEYWORDS

Secure hash algorithm;
Blockchain;
IoT;
Co-Processor;
FPGA.

ABSTRACT

Hash function plays an important role in security, widely used in privacy authentication. Secure Hash Algorithm (SHA) is a cryptographic algorithm with internal complex structure using many logical and mathematical transformations to generate 256-bits Message digest. It is almost impossible to implement a hash algorithm by software because it takes CPU a long time to fetch instructions, which can cause low speed and consume computer resources. On the other hand, hash algorithms work more effectively when implemented in hardware because hardware uses bit-level and instruction-level parallel processing. The use of Application-Specific Integrated Circuits (ASICs) can help to accelerate SHA, better performance and lower power consumption. However, FPGA-based systems are much more flexible and reprogrammable, design flow is also cheaper while FPGA is able to optimize speed and power consumption. In this paper, we propose a design that implements the secure hash algorithm SHA-256 and evaluate performance of the hashing system. The results are verified based-on simulation using Xilinx Vivado 2019.1 software.

Thiết Kế Và Đánh Giá Thuật Toán Băm Bảo Mật SHA-256 Trên Nền Tảng Phần Cứng ZynQ-702

Nguyễn Thị Hồng Hiếu[1], Trần Thị Ánh Dương[1], Văn Ánh Dương[1], Hồ Nhựt Minh[2], Nguyễn Ngô Lâm[1], Trương Quang Phúc[1*]

[1]Trường Đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh, Việt Nam
[2]Học viện Công nghệ Bưu chính Viễn Thông, Việt Nam

[*]Tác giả liên hệ. *Email: phuctq@hcmute.edu.vn*

THÔNG TIN BÀI BÁO

Ngày nhận bài: 06/06/2023
Ngày hoàn thiện: 12/07/2023
Ngày chấp nhận đăng: 04/08/2023
Ngày đăng: 28/08/2024

KEYWORDS

Thuật toán băm bảo mật;
Chuỗi khối;
IoT;
Bộ đồng xử lý;
FPGA.

TÓM TẮT

Hàm băm đóng vai trò quan trọng trong các hệ thống bảo mật, được sử dụng rộng rãi trong xác thực quyền riêng tư. Thuật toán băm là một thuật toán mã hóa có cấu trúc phức tạp, sử dụng nhiều phép biến đổi logic, biến đổi toán học bên trong. Do vậy khi triển khai thuật toán băm bằng phần mềm, CPU sẽ mất nhiều thời gian để tìm nạp lệnh dẫn tới tốc độ chậm và tiêu tốn tài nguyên máy tính. Hướng tiếp cận thực thi thuật toán băm trên phần cứng sẽ giúp tối ưu về tốc độ bởi phần cứng hoạt động xử lý song song. Có thể thiết kế hệ thống thuật toán băm như một chip ASIC với chức năng xác định tích hợp bên trong máy tính, song hướng thiết kế FPGA đem lại lợi ích về tính linh hoạt, có thể tái lập trình được. Trong bài báo này, chúng tôi thiết kế một hệ thống thực thi thuật toán băm bảo mật SHA-256 và đánh giá các thông số tài nguyên sử dụng, công suất hoạt động trên nền tảng phần cứng ZynQ-702. Chức năng của thuật toán được kiểm chứng thông qua mô phỏng dạng sóng sử dụng phần mềm Xilinx Vivado 2019.1.

Doi: https://doi.org/10.54644/jte.2024.1421

JTE
HCMUTE
ISSN: 1859-1272

TẠP CHÍ KHOA HỌC GIÁO DỤC KỸ THUẬT
Trường Đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh
Website: https://jte.edu.vn
Email: jte@hcmute.edu.vn

1. Giới thiệu

Hàm băm mật mã (Cryptographic Hash Function) là một mã bảo mật được sử dụng trong công nghệ blockchain (chuỗi khối), có vai trò bảo mật khối tin và kết nối các khối trong chuỗi lại với nhau. Hàm băm tạo ra bản tóm tắt các thông tin chứa trong khối, đặc trưng cho từng khối và đảm bảo không có hai khối nào mang hai bản tóm tắt giống nhau. Do vậy, có thể dựa trên mã băm để tìm kiếm giao dịch trong chuỗi. Hàm băm được sử dụng phổ biến trong blockchain hiện nay là SHA-256 vì nó có khả năng xử lý được trên dữ liệu có kích thước lớn giới hạn trong 2^{64} bit, bản tóm tắt do SHA-256 tạo ra có độ rộng 256-bit bao quát được nhiều dữ liệu vào hơn SHA-1 và thuật toán của mã băm này cũng phức tạp hơn, ngăn chặn được sự tấn công trái phép đoạt lấy thông tin trong giao dịch.

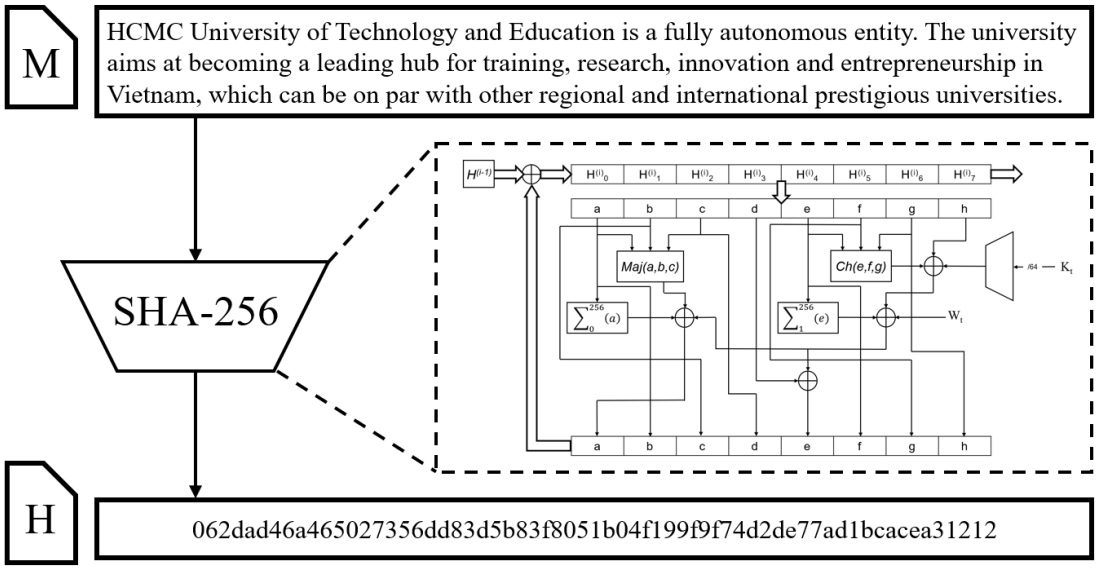
Hơn nữa, blockchain cung cấp giải pháp bảo mật thông tin cho hệ thống IoT nhưng để vận hành được công nghệ này là một bài toán lớn. Các thiết bị trong hệ thống IoT sử dụng nguồn chủ yếu là pin, thiết bị nhỏ gọn, tiêu thụ ít điện năng. Song công nghệ chuỗi khối cần một máy tính lớn mới có thể vận hành được các thuật toán mã hóa bởi các thuật toán này rất phức tạp, sử dụng rất nhiều các phép biến đổi tính toán số học, biến đổi logic, chạy trên nhiều vòng lặp và xử lý trên dữ liệu kích thước lớn. Ngoài ra, yêu cầu về tốc độ cũng được đề cao phải xử lý nhanh mới đáp ứng được tiêu chí thời gian thực trong IoT. Việc tăng tốc độ xử lý các thuật toán mã hóa bảo mật gần như không phù hợp để triển khai bằng phần mềm, hướng tiếp cận thực thi trên phần cứng có thể mang lại hiệu quả đáng kể hơn.

Các cách triển khai thuật toán băm như trong bài báo [1], [2] và đặc biệt là [3] hướng tới việc tái cấu trúc lại kiến trúc phần cứng bên trong bộ xử lý với mục đích là tăng tốc độ tính toán mà vẫn đảm đảo tài nguyên sử dụng ở mức chấp nhận được. Mặt khác, cách triển khai như trong [4] đó là xây dựng thành phần tạo mã băm như một khối chức năng đặc biệt, xem nó như một bộ đồng xử lý cho CPU đem lại hiệu quả đáng kể và thích hợp cho hệ thống IoT. Ngoài ra trong [5], tác giả đã xây dựng một bộ đồng xử lý SHA-3, chuẩn hàm băm mới nhất trong họ các thuật toán băm, nhằm giải quyết vấn đề bảo mật cho IoT. Có thể thấy, các thiết kế này có được tính linh loạt nhờ vào thiết kế trên FPGA mà vẫn đảm bảo tối ưu về tài nguyên và mức tiêu thụ điện năng, giảm bớt áp lực tính toán cho bộ xử lý trung tâm.

Mã hóa là một phần thiết yếu trong thương mại điện tử, giao dịch qua Internet, IoT blockchain [6]. Khi quá trình xử lý các thuật toán mã hóa này tiêu tốn quá nhiều năng lượng của máy chủ, sự có mặt của bộ đồng xử lý mã hóa là cần thiết. Hiện nay đã có phần cứng chuyên dụng cho bộ đồng xử lý mã hóa được sản xuất bởi NXP Semiconductors, là họ các bộ đồng xử lý C29x như C291, C292, C293 [7]. Bên cạnh đó còn có bộ đồng xử lý mã hóa IBM PCIe 4770, 4769, 4768, 4767 là các module phần cứng xử lý mã hóa được triển khai trên bo mạch PCIe có thể lập trình được [8].

2. Bộ đồng xử lý mã băm

2.1. Hàm băm (Hash function) trong mật mã (Cryptography)



Hình 1. *Tính chất một chiều của hàm băm [5]*

Hàm băm tạo ra bản tóm tắt (Message digest) cho bất kì tệp tin dữ liệu nào cần được xác thực. Việc xác thực phải đảm bảo xác minh được danh tính của tệp tin được gửi đi và đảm bảo sự toàn vẹn của nó khi đến phía người nhận, mọi sửa đổi khác với tệp tin gốc đều phải được phát hiện ra. Thuật toán băm tập trung vào hai tính chất: tính chất một chiều và tính chất phân biệt. Tính chất một chiều không cho phép từ mã băm hay bản tóm tắt của tệp tin có thể truy ngược được về tin nhắn gốc, như hình 1.

Tính chất phân biệt cho thấy rõ sự khác nhau giữa hai bản tóm tắt được tạo ra từ hai tệp tin khác nhau, như hình 2.

Hình 2. *Tính chất phân biệt của hàm băm*

Thuật toán băm phải đảm bảo được hai tính chất trên mới được xem là đạt chuẩn bảo mật cho thông tin dữ liệu. Ứng dụng nổi bật của hàm băm chính là dùng trên chữ ký số, khi mà chi phí để tạo chữ ký số lên toàn bộ văn bản trở nên quá tốn kém, ta có thể tạo chữ ký số chỉ trên bản tóm tắt của dữ liệu. Hàm băm cũng được dùng trong hệ thống lưu trữ mật khẩu, giúp xác thực người dùng nào đang đăng nhập chỉ bằng cách so sánh mã băm của mật khẩu vừa nhập với mã băm mật khẩu đã lưu trữ trong hệ thống. Ngoài ra hệ thống cũng có thể gửi mã băm tới người dùng thực để xác minh rằng có phải chính họ đang đăng nhập hay không. Yêu cầu của thuật toán băm trong trường hợp này phải xử lý nhanh để phù hợp với nhu cầu người dùng.

2.2. Thuật toán băm bảo mật SHA-256

Thuật toán băm bảo mật biểu diễn quá trình tính toán một dữ liệu vào thành dạng tóm gọn của dữ liệu đó. Thuật toán băm SHA-256 xử lý trên dữ liệu có độ rộng giới hạn trong 2^{64} bit, bản tóm tắt được tạo ra từ SHA-256 luôn có kích thước gói gọn trong 256 bit, độ bảo mật cao phù hợp ứng dụng trong nhiều hệ thống. Hoạt động của thuật toán băm bao gồm hai bước: tiền xử lý và tính toán. Bước tiền xử lý tiếp nhận dữ liệu vào và đệm, sao cho độ dài của dữ liệu được đệm là bội số của 512. Giả sử một tin nhắn M có độ dài l bit, quá trình tiền xử lý sẽ đệm bit "1" vào cuối M cùng với k bit "0" tiếp sau đó. Theo tiêu chuẩn được đề ra bởi [9], giá trị k là nghiệm nhỏ nhất của phương trình (1).

$$l + 1 + k \equiv 448 \bmod 512 \quad (1)$$

Sau khi đệm k bit "0", tiền xử lý sẽ thêm 64 bit độ dài l tính theo hệ nhị phân vào cuối để M đạt độ dài là bội số của 512. Mỗi đoạn 512-bit của M được xem là một block, ngõ ra của bước tiền xử lý là các block 512-bit sau khi đệm: $M^{(1)}$, $M^{(2)}$, $M^{(3)}$,..., $M^{(i)}$.

Ở bước tính toán, các block $M^{(1)}$, $M^{(2)}$, $M^{(3)}$,..., $M^{(i)}$ được đưa vào lần lượt để xử lý. Mỗi block 512-bit $M^{(i)}$ sẽ được phân tách thành 16 word 32-bit $M_1^{(i)}$, $M_2^{(i)}$,..., $M_{15}^{(i)}$, đồng thời mở rộng số lượng 32-bit word này lên 64 word theo công thức sau:

$$W_t = M_t^{(i)} \qquad (0 \le t \le 15) \quad (2)$$

$$W_t = \sigma_1^{256}(W_{t-2}) + W_{t-7} + \sigma_0^{256}(W_{t-15}) + W_{t-16} \qquad (16 \le t \le 63) \quad (3)$$

Theo đó, hàm $\sigma(x)$ là một hàm logic được quy định trong [9] như sau:

$$\sigma_0^{256}(x) = \mathrm{ROTR}^7(x) \oplus \mathrm{ROTR}^{18}(x) \oplus \mathrm{SHR}^3(x) \quad (4)$$

$$\sigma_1^{256}(x) = \mathrm{ROTR}^{17}(x) \oplus \mathrm{ROTR}^{19}(x) \oplus \mathrm{SHR}^{10}(x) \quad (5)$$

Các word W_t này sẽ được đưa vào vòng lặp để tính toán ra các giá trị trung gian a, b, c, d, e, f, g, h. Các giá trị này sẽ lưu giá trị băm cần để tính toán cho block kế tiếp. Đối với block đầu tiên, các giá trị trung gian sẽ được gán bằng giá trị băm khởi tạo $H^{(0)}_0$, $H^{(0)}_1$,..., $H^{(0)}_7$. Trong SHA-256, các giá trị băm khởi tạo được chọn bằng cách lấy 32 bit đầu của phần thập phân căn bậc 2 tám số nguyên tố đầu tiên [9].

$$H^{(0)}_0 = 6a09e667 \qquad H^{(0)}_1 = bb67ae85 \qquad H^{(0)}_2 = 3c6ef372 \qquad H^{(0)}_3 = a54ff53a \quad (6)$$

$H^{(0)}_4$ = 510e527f $H^{(0)}_5$ = 9b05688c $H^{(0)}_6$ = 1f83d9ab $H^{(0)}_7$ = 5be0cd19

Có 64 vòng lặp tương ứng với việc tính toán trên 64 word của mỗi block. Mỗi vòng lặp thực thi một loạt các phép tính toán phức tạp trong bảng 1.

Bảng 1. *Thuật toán vòng lặp xử lý trên từng word*

Input: W_t

for (t = 0, t < 64, t++)

{

$T_1 = h + \sum_1^{256}(e) + Ch(e, f, g) + K_t^{256} + W_t$; (7)

$T_2 = \sum_0^{256}(a) + Maj(a, b, c)$; (8)

h = g; g = f; f =e; e = d + T_1; d = c; c = b; b = a; a = T_1 + T_2; (9)

}

Output: a, b, c, d, e, f, g, h

Trong đó các hàm ∑(x), Ch(x,y,z) và Maj(x,y,z) là các hàm tính toán logic được quy định trong [9].

$$\sum_0^{256}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x) \quad (10)$$

$$\sum_1^{256}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x) \quad (11)$$

$$Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z) \quad (12)$$

$$Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \quad (13)$$

Ngoài ra còn có hằng số K_t tham gia vào quá trình tính toán ra giá trị băm kế tiếp, với K_t là 64 giá trị trong hệ Hexa được quy định theo chuẩn [9].

428a2f98	71374491	b5c0fbcf	e9b5dba5	3956c25b	59f111f1	923f82a4	ab1c5ed5
d807aa98	12835b01	243185be	550c7dc3	72be5d74	80deb1fe	9bdc06a7	c19bf174
e49b69c1	efbe4786	0fc19dc6	240ca1cc	2de92c6f	4a7484aa	5cb0a9dc	76f988da
983e5152	a831c66d	b00327c8	bf597fc7	c6e00bf3	d5a79147	06ca6351	14292967
27b70a85	2e1b2138	4d2c6dfc	53380d13	650a7354	766a0abb	81c2c92e	92722c85
a2bfe8a1	a81a664b	c24b8b70	c76c51a3	d192e819	d6990624	f40e3585	106aa070
19a4c116	1e376c08	2748774c	34b0bcb5	391c0cb3	4ed8aa4a	5b9cca4f	682e6ff3
748f82ee	78a5636f	84c87814	8cc70208	90befffa	a4506ceb	bef9a3f7	c67178f2

Như vậy, với N block 512-bit sau khi đệm, bước tính toán sẽ làm các công việc trong bảng 2 để đưa ra giá trị băm cuối cùng.

Bảng 2. *Thuật toán vòng lặp xử lý trên từng block*

Input: 512-bit block $M^{(i)}$

for (i = 1, i <= N, i++)

{

a = $H^{(i-1)}_0$; b = $H^{(i-1)}_1$; c = $H^{(i-1)}_2$; d = $H^{(i-1)}_3$; (14)

e = $H^{(i-1)}_4$; f = $H^{(i-1)}_5$; g = $H^{(i-1)}_6$; h = $H^{(i-1)}_7$; (15)

64 vòng lặp W_t;

$H^{(i)}_0$ = a + $H^{(i-1)}_0$; $H^{(i)}_1$ = b + $H^{(i-1)}_1$; $H^{(i)}_2$ = c + $H^{(i-1)}_2$; $H^{(i)}_3$ = d + $H^{(i-1)}_3$; (16)

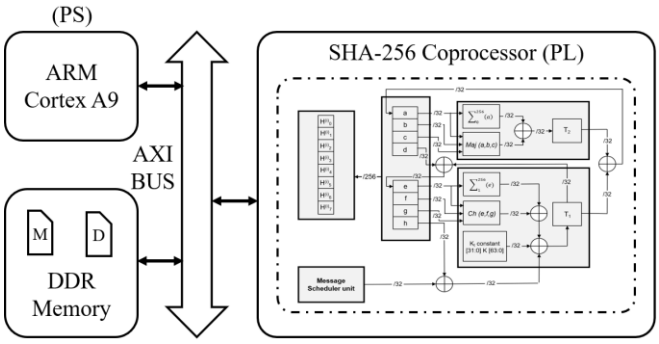
$H^{(i)}_4$ = e + $H^{(i-1)}_4$; $H^{(i)}_5$ = f + $H^{(i-1)}_5$; $H^{(i)}_6$ = g + $H^{(i-1)}_6$; $H^{(i)}_7$ = h + $H^{(i-1)}_7$; (17)

}

Output: $H = H^{(N)}_0 \| H^{(N)}_1 \| H^{(N)}_2 \| H^{(N)}_3 \| H^{(N)}_4 \| H^{(N)}_5 \| H^{(N)}_6 \| H^{(N)}_7$ (18)

2.3. Bộ đồng xử lý mã băm

Bộ đồng xử lý là một bộ xử lý bổ sung được thiết kế để hỗ trợ cho CPU trong máy tính. Bộ đồng xử lý nói đơn giản là một mạch thực hiện các tác vụ riêng biệt như mã hóa, xử lý số học dấu chấm động, xử lý đồ họa, xử lý tín hiệu, giao tiếp I/O với ngoại vi, mục đích là để giảm tải hoạt động cho CPU, tăng hiệu năng hệ thống, cho phép CPU chỉ tập trung vào các tác vụ cần thiết. Thuật toán băm có cấu trúc thuật toán phức tạp, sử dụng nhiều phép biến đổi logic, phép biến đổi toán học, lặp trong nhiều vòng lặp để cho ra mã băm nên không thể xem việc tạo mã băm như một phần thuộc kiến trúc tập lệnh thông thường. Thêm vào đó khi triển khai thuật toán băm bằng phần mềm sẽ tiêu tốn rất nhiều thời gian để CPU tìm nạp lệnh tương ứng, dẫn đến tốc độ chậm và giảm tuổi thọ CPU. Do vậy bộ đồng xử lý mã băm là giải pháp giúp tối ưu tốc độ và tài nguyên cho máy tính. Một bộ đồng xử lý mã băm triển khai trên phần cứng FPGA sẽ có kiến trúc như mô tả trong hình 3.



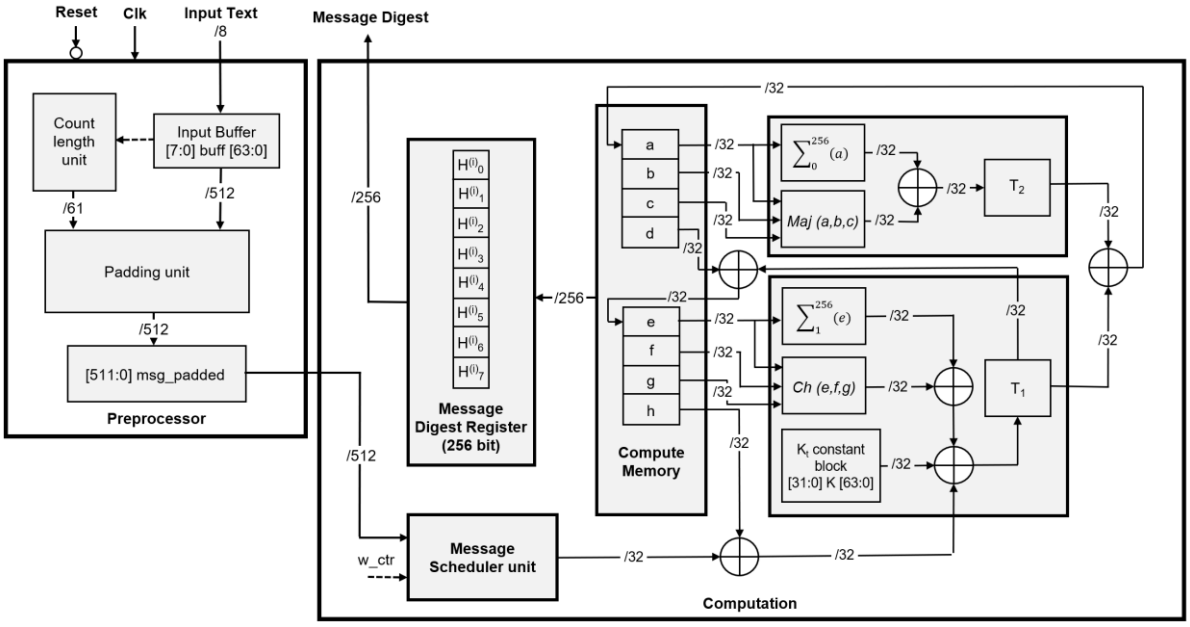
Hình 3. *Kiến trúc bộ đồng xử lý mã băm triển khai trên FPGA [5]*

Trong hình 3, bộ đồng xử lý mã băm nằm trong thành phần logic lập trình được (Programmable Logic - PL) của FPGA, kết nối với hệ thống xử lý (Processing System - PS) là một chip ARM Cortex A9 và bộ nhớ DDR thông qua bus AXI (Advanced eXtensible Interface). Bộ đồng xử lý mã băm trên thực hiện trên PL, phù hợp cho việc thiết kế phần cứng linh hoạt hơn là triển khai thành chip ASIC.

3. Thiết kế hệ thống

3.1. Sơ đồ khối toàn hệ thống

Hệ thống thực hiện thuật toán băm bảo mật SHA-256 có sơ đồ khối tổng quát được mô tả trong hình 4 dưới đây.

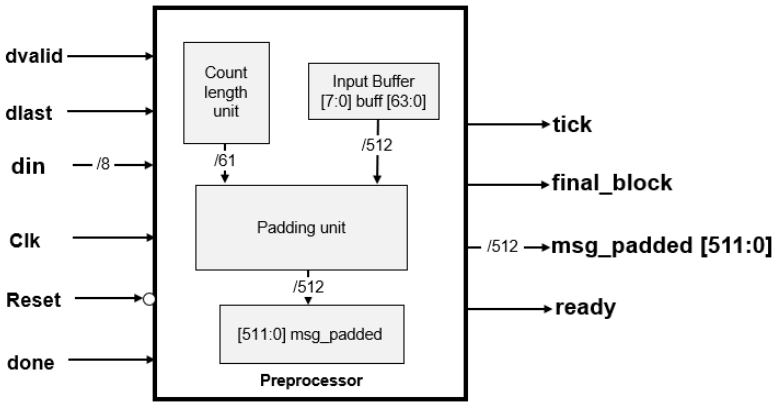


Hình 4. *Sơ đồ khối tổng quát hệ thống thực thi thuật toán băm SHA-256*

Theo hình 4, hệ thống bao gồm khối Preprocessor có vai trò đệm dữ liệu và khối Computation thực hiện việc mở rộng số lượng word cũng như chạy vòng lặp để tính toán ra các giá trị băm. Đối với khối Preprocessor, do bước tiền xử lý cần 64 bit độ dài *l* của dữ liệu để đệm nên khối này có một bộ đếm 61 bit để đếm số lượng ký tự được nhập vào. Dữ liệu vào là ký tự ASCII 8 bit, mỗi khi có một ký tự được nhập, hệ thống sẽ lưu nó vào bộ nhớ [7:0]buff[63:0] và đồng thời cũng kích hoạt bộ đếm lên. Khi bộ nhớ buff đầy, dữ liệu lưu bên trong sẽ được đẩy ra ngõ ra của khối Preprocessor và đưa vào khối Computation, tương ứng với một block được đưa vào để tính toán. Khối Computation có thành phần Message Scheduler đảm nhận việc mở rộng số lượng 32-bit word lên thành 64 word và sẽ đưa các word này vào vòng lặp để tính toán. Các khối bên trong hệ thống phải hoạt động đồng bộ với nhau. Cứ mỗi khi một block được tạo thành bởi khối Preprocessor sẽ lập tức được đưa vào khối Computation để tính toán ra các giá trị băm, các giá trị băm sẽ được lưu trong một thanh ghi 256-bit sử dụng cho block tiếp theo. Quá trình này cứ lặp lại cho tới khi block cuối cùng được đưa vào tính toán, hệ thống sẽ xuất các giá trị băm cuối cùng tại ngõ ra.

3.2. Thiết kế các khối bên trong

3.2.1. Khối Preprocessor



Hình 5. *Thiết kế top module của khối Preprocessor*

Trong hình 5, khối Preprocessor gồm một bộ đếm dùng để đếm số lượng ký tự của dữ liệu vào, một bộ nhớ để lưu dữ liệu có kích thước 512bit cũng là kích thước của một block, một bộ đệm bit "1" và "0" thực chất là một máy trạng thái sẽ chuyển đổi trạng thái để thực hiện đệm cho phù hợp với dữ liệu được nhập vào. Mô tả chức năng của các chân ngõ vào ngõ ra được thể hiện trong bảng 3.

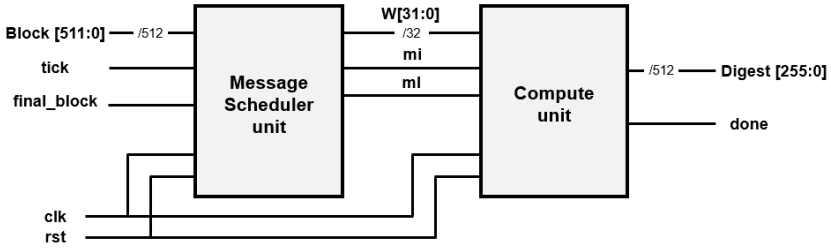
Bảng 3. *Chức năng của các chân tín hiệu trong khối Preprocessor*

Tên tín hiệu	I/O	Độ rộng (bit)	Chức năng
clk	Input	1	Tín hiệu xung clock
rst	Input	1	Tín hiệu reset
din	Input	8	Tín hiệu dữ liệu vào là các ký tự ASCII có độ dài 8 bit.
dvalid	Input	1	Tín hiệu báo hiệu có dữ liệu được nhập vào
dlast	Input	1	Tín hiệu báo hiệu ký tự cuối cùng của dữ liệu được nhập vào
done	Input	1	Tín hiệu báo quá trình tạo mã băm đã kết thúc
tick	Output	1	Tín hiệu cho biết đã đệm xong 1 block
final_block	Output	1	Tín hiệu cho biết block cuối cùng đã được đệm
msg_padded	Output	512	Thanh ghi lưu trữ 1 block 512-bit
ready	Output	1	Tín hiệu thông báo khối đang rảnh, sẵn sàng nhận dữ liệu vào

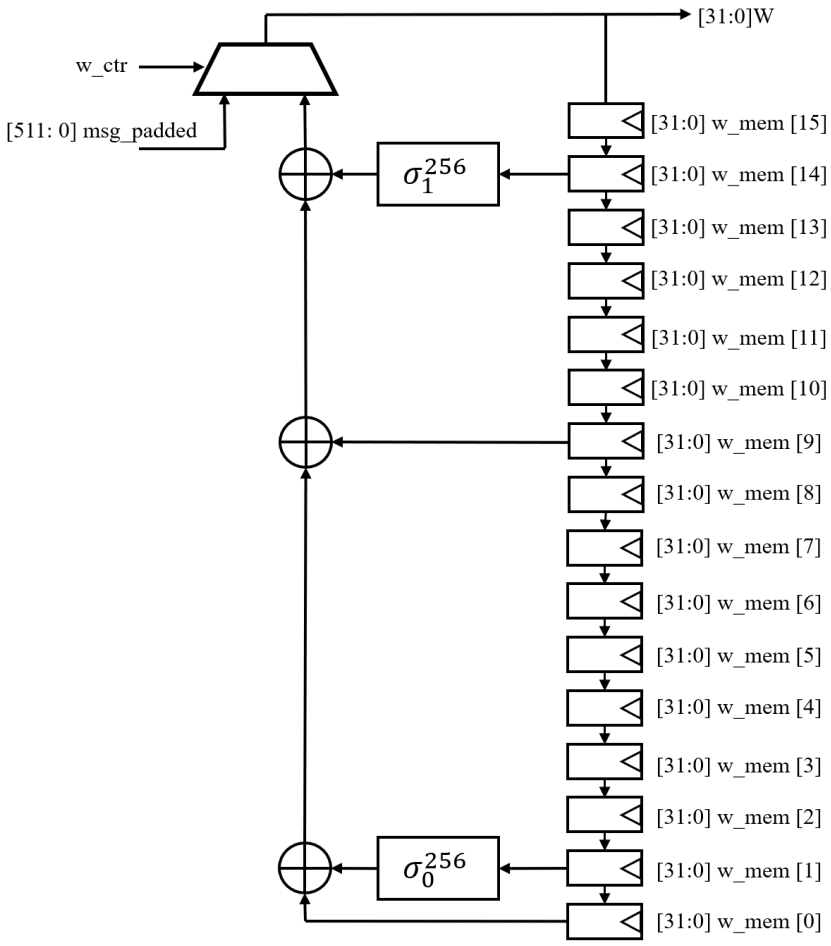
Tín hiệu dvalid có vai trò kích hoạt bộ đếm bên trong khối Preprocessor, tín hiệu dlast tích cực để bắt đầu quá trình đệm. Khi một chuỗi ký tự được nhập vào có kích thước lớn hơn một block (512 bit), bộ đếm vẫn sẽ tiếp tục đếm cho đến khi dlast tích cực, lúc này quá trình đệm vẫn chưa diễn ra, bộ nhớ buff khi đã đầy sẽ tự động đẩy ra ngõ ra để đưa vào khối Computation và reset lại buff để nhận ký tự tràn tiếp theo.

3.2.2. Khối Computation

Khối Computation bao gồm hai thành phần con hoạt động đồng bộ với nhau: Message Scheduler và Compute. Sơ đồ kết nối giữa hai thành phần con này được mô tả trong Hình 6 dưới đây cùng với mô tả chức năng các đường tín hiệu trong bảng 4.



Hình 6. *Sơ đồ kết nối giữa hai thành phần con trong khối Computation*



Hình 7. *Sơ đồ thanh ghi dịch trong thành phần Message Scheduler*

Thành phần Message Scheduler có vai trò phân tách block 512 bit thành các word 32 bit và mở rộng số lượng word này lên 64 word cho thành phần Compute tính toán trong 64 vòng lặp. Trong hệ thống này, thay vì sử dụng 64 thanh ghi 32-bit để lưu trữ các word ta có thể giảm bớt số lượng thanh ghi xuống còn 16 thanh ghi và dịch lần lượt từng word sang Compute để tính toán như hình 7.

Bảng 4. *Chức năng các đường tín hiệu kết nối trong khối Computation*

Tên tín hiệu	I/O	Độ rộng (bit)	Chức năng
clk	Input	1	Tín hiệu xung clock
rst	Input	1	Tín hiệu reset
block	Input	512	Ngõ vào là các block 512 bit lấy từ khối Preprocessor
tick	Input	1	Nhận tín hiệu tick của khối Preprocessor để tích cực
final_block	Input	1	Nhận tín hiệu final_block của khối Preprocessor để tích cực, thông báo block cuối cùng đang được đưa vào
mi	Inout	1	Tín hiệu tích cực trong suốt quá trình dịch 64 word vào Compute
ml	Inout	1	Tín hiệu tích cực khi block cuối cùng được đưa vào Message Scheduler
W	Inout	32	word 32-bit
done	Output	1	Tín hiệu thông báo quá trình tạo mã băm kết thúc
digest	Output	256	Mã băm ngõ ra

Dữ liệu w[31:0] đưa vào thành phần Compute được chọn bởi một bộ mux điều khiển bằng tín hiệu w_ctr. Tín hiệu w_ctr thực chất là một biến đếm 6-bit đếm từ 0 đến 63 để chọn word tương ứng đưa vào thành phần Compute. Các tín hiệu mi và ml cùng phối hợp để báo hiệu cho thành phần Compute hoạt động và đưa giá trị băm tính toán được ra ngõ ra khi block cuối cùng đã được xử lý xong.

4. Kết quả và đánh giá

4.1. Kết quả mô phỏng

Kiểm tra chức năng của hệ thống vừa thiết kế thông qua hai testcase được mô tả trong bảng 5.

Bảng 5. *Xây dựng các testcase để kiểm tra chức năng hệ thống*

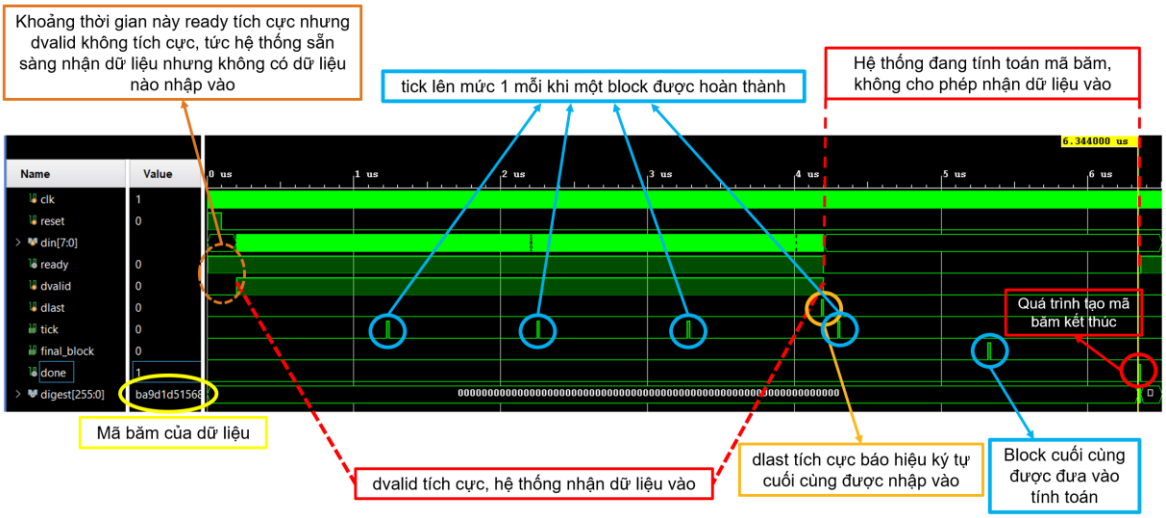
Testcase	Nội dung kiểm tra
Testcase 1	**Mục tiêu**: Kiểm tra tính chính xác về chức năng hoạt động của hệ thống. **Thông số đầu vào:** Tần số xung clock: 62.5MHz. **Thực hiện:** + Truyền dữ liệu vào với kích thước bất kì. + Xem xét và kiểm tra hoạt động của hệ thống, tiến hành đánh giá kết quả.
Testcase 2	**Mục tiêu:** Kiểm tra độ nhạy của hệ thống với các dữ liệu đầu vào có sự khác biệt rất nhỏ. **Thông số đầu vào**: Tần số xung clock: 62.5MHz. **Thực hiện:** + Truyền hai dữ liệu vào riêng biệt chỉ có 1 ký tự khác biệt. + Xem xét và kiểm tra hoạt động của hệ thống, tiến hành đánh giá kết quả.

4.1.1. Mô phỏng testcase 1

Chuỗi ký tự như hình 8 được đưa vào hệ thống, ta được dạng sóng ngõ ra trong hình 9.

```
Ho Chi Minh City University of Technology and
Education is a distinct one. This is the first
university to train technical teachers, the leading
university in the national Technical Education system,
with a long history and formation and development.
```

Hình 8. *Dữ liệu ngõ vào của testcase 1*

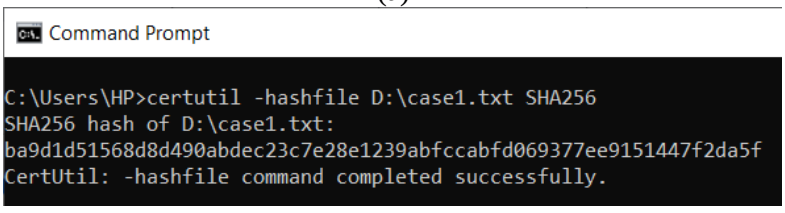


Hình 9. *Dạng sóng của testcase 1*

Ban đầu khi chưa có dữ liệu vào, hệ thống ở trong trạng thái chờ với tín hiệu ready mức 1 cho đến khi dvalid tích cực thông báo có ký tự được nhập vào. Lúc này bộ đệm tiếp nhận và lưu trữ dữ liệu vào, khi bộ đệm đầy, dữ liệu sẽ được dịch sang khối Computation để tính toán, lúc này tín hiệu tick tác động thông báo có một block được đưa vào xử lý. Quá trình tiếp nhận và dịch diễn ra tiếp tục cho tới khi dlast tích cực, lúc này ký tự cuối cùng đã được nhập vào và hệ thống bắt đầu quá trình Tiền xử lý. Quá trình tạo mã băm kết thúc khi tín hiệu done lên mức 1, tín hiệu ready ở mức 0 trong suốt thời gian hệ thống tính toán mã băm để không cho phép dữ liệu mới nào được nhập vào. Ready lên mức 1 trở lại sau khi quá trình tính toán kết thúc. Để kiểm chứng giá trị băm được tạo ra có chính xác với chuẩn thuật toán băm bảo mật SHA-256 hay không, ta so sánh kết quả ở ngõ ra với giá trị băm SHA-256 được tạo từ cửa sổ Command Prompt hệ điều hành Windows 10 như trong hình 10.

```
------------------------------------------------------
               Module SHA-256
------------------------------------------------------
                 0 - Machine is ready
            200000 - Start receiving data
relaunch_sim: Time (s): cpu = 00:00:02 ; elapsed = 00:00:06 . Memory (MB): peak = 853.023 ; gain = 0.000
run 3 us
           1224000 - Completed                    1 block
           2248000 - Completed                    2 block
           3272000 - Completed                    3 block
run 3 us
           4184000 - Received final data
           4200000 - Machine is busy
           4296000 - Completed                    4 block
           5320000 - Completed                    5 block - The last block
           6344000 - Completed generation SHA-256
Total length:                250
Digest value SHA-256: ba9d1d51568d8d490abdec23c7e28e1239abfccabfd069377ee9151447f2da5f
           6360000 - Machine is ready
------------------------------------------------------
```

(a)



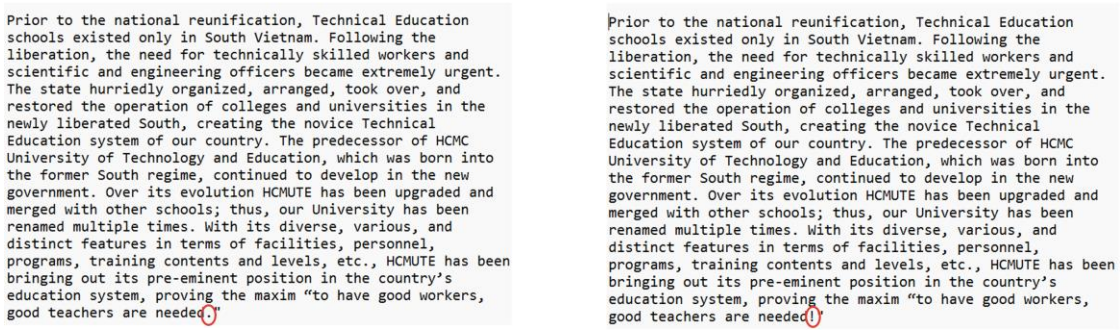
(b)

Hình 10. *(a) Mã băm ngõ ra của hệ thống và (b) Giá trị băm chuẩn của SHA-256*

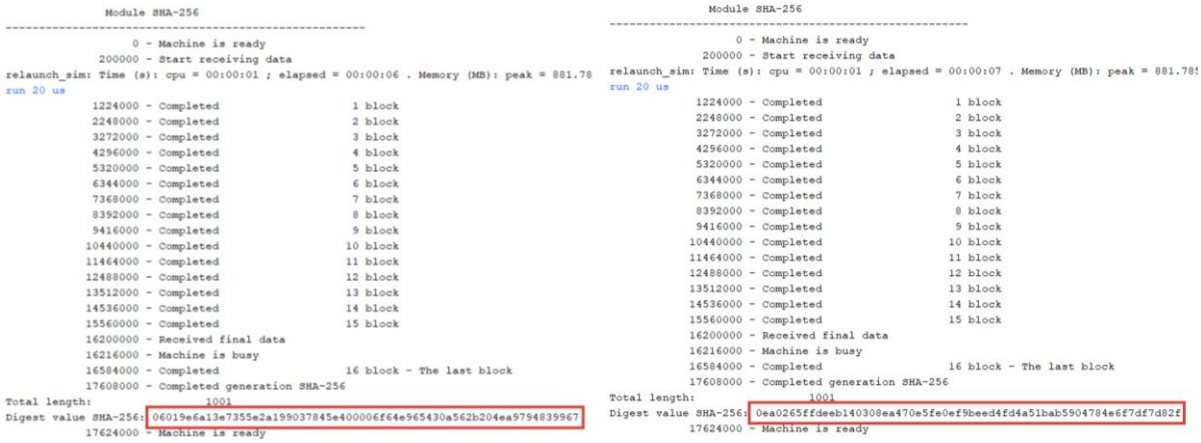
Từ hai kết quả trên có thể thấy hệ thống đã tạo ra mã băm đúng với chuẩn của thuật toán băm bảo mật SHA-256.

4.1.2. Mô phỏng testcase 2

Đưa vào hệ thống 2 chuỗi ký tự như hình 11, chỉ khác nhau ký tự cuối cùng để kiểm tra tính chất phân biệt của mã băm do hệ thống tạo ra, xem xét và đánh giá hai mã băm trong hình 12.



Hình 11. *Dữ liệu (a) và (b) khác nhau ở ký tự cuối cùng*



Hình 12. *Mã băm (a) của dữ liệu thứ nhất và mã băm (b) của dữ liệu thứ 2*

So sánh hai mã băm có thể dễ dàng nhận thấy chúng có sự khác biệt rõ rệt mặc dù dữ liệu gốc có tỉ lệ giống nhau là 99%.

4.2. Tài nguyên sử dụng của hệ thống

Các thông số tài nguyên sử dụng và công suất tiêu thụ trong hệ thống thực thi thuật toán băm SHA-256 thiết lập của bộ FPGA Board Zybo Z7-02 (xc7z020clg400-1) được tổng hợp lần lượt trong bảng 6, và bảng 7.

Bảng 6. *Thông số tài nguyên của hệ thống thực thi thuật toán băm SHA-256*

Site type	Used	Available	Utilization
Number of LUTs	2568	53200	4.83%
Number of Slice Logics	2504	53200	4.71%
Number of Registers	1347	136400	1.27%
Number of slice Logic	1345	136400	1.26%

Rõ ràng, công suất của hệ thống khi thực thi được đo bởi phần mềm là tương đối thấp như trong bảng 7. Hệ thống hoạt động tốt trong điều kiện nhiệt độ phòng bình thường. Theo lý thuyết, hệ thống xử lý một block 512-bit trong 64 chu kỳ xung clock. Đối với hệ thống được thiết kế ở đây cần 65 chu kỳ xung để xử lý, trong đó bộ tính toán cần 64 chu kỳ xung để tính toán ra các giá trị băm H_0, H_1,…, từ khi bắt

đầu nhận block dữ liệu 512-bit từ bộ tiền xử lý, và cần thêm chu kỳ 1 xung để dịch các giá trị này vào cho việc tính toán trong vòng lặp tiếp theo. Từ đó, số liệu về Throughput được tính toán dưới đây:

$$Throughput = \frac{Block\ size\ x\ Clock\ frequency}{Clock\ cycles\ per\ block} = \frac{512\ x\ 62.5}{65} = 492.3\ Mbps \quad (19)$$

Bảng 7. *Thông số công suất tiêu thụ và điều kiện hoạt động*

Total On-chip Power (W)	0.027
Dynamic Power (W)	0.020
Static Power (W)	0.007
Max Ambient (°C)	83.5
Junction Temperature (°C)	26.5

Bảng 8 so sánh chi tiết các thông số tài nguyên của hệ thống đã thiết kế với một số nghiên cứu SHA-256 đã được công bố.

Bảng 8. *So sánh thông số thiết kế với các nghiên cứu khác*

Design	Device	Clock frequency (MHz)	Clock cycles per block	Throughput (Mbps)	Slice
SHA-256	Zybo Z7-02	62.5	65	492.3	762
SHA-256 [10]	Virtex XCV300E-8	88	73	617	1261
SHA-256 [11]	Virtex-II	73.975	38	996.7	2032
SHA-256 [2]	Virtex-5	64.45	280	117.8	139
SHA-256 [12]	Virtex-4	170.75	65	1344.98	610

Dựa theo bảng 8 so sánh thông số thiết kế với các nghiên cứu khác có thể thấy:

- Nghiên cứu [10], [11], [12] số liệu throughput lớn hơn nhiều so với thiết kế của chúng tôi vì các nghiên cứu này tập trung vào xử lý trên tần số cao, tốc độ nhanh. Tuy nhiên đi kèm theo đó là tài nguyên về slice lớn hơn, tốn nhiều diện tích hơn.
- Nghiên cứu [2] có số lượng slice nhỏ cho thấy tối ưu về mặt diện tích, song so sánh số liệu throughput không mang lại ưu điểm về tốc độ như thiết kế của chúng tôi đã đạt được.

Như vậy, thiết kế đạt yêu cầu về tốc độ và diện tích, có thể phù hợp với hệ thống IoT như đã đề cập trong phần giới thiệu. Hệ thống thiết kế vẫn còn tiêu tốn nhiều tài nguyên hơn so với các nghiên cứu SHA-256 khác đã công bố. Mặt bằng chung, hệ thống hoạt động được với tần số clock trung bình 62.5MHz, xử lý trên một block cần 65 chu kỳ xung nhịp, throughput khoảng 492.3 Mbps. Mặt khác, các nghiên cứu được đề cập trong bảng 8 sử dụng nhiều kỹ thuật mới để tối ưu tốt thiết kế, giảm thiểu tiêu tốn tài nguyên, tần số hoạt động cao và thông lượng nhiều hơn.

5. Kết luận

Tóm lại, thông qua đánh giá kết quả mô phỏng và đánh giá dựa trên các thông số tài nguyên, công suất, cho thấy hệ thống thực thi thuật toán băm bảo mật SHA-256 của chúng tôi hoạt động đúng chức năng tạo ra mã băm theo tiêu chuẩn NIST đề ra cho SHA-256, bản tóm tắt được tạo ra có khả năng phân biệt giữa hai dữ liệu vào khác nhau. Hệ thống hoạt động được tại tần số 62.5MHz với tài nguyên sử dụng ở mức hợp lý. Công suất hệ thống thấp, phù hợp để tích hợp trên một bộ đồng xử lý mã hóa. Tuy nhiên, một hệ thống thực thi thuật toán băm, yêu cầu về tốc độ và tài nguyên sử dụng luôn được đề cao. Do đó thiết kế cần áp dụng thêm các kỹ thuật đặc biệt như sử dụng kiến trúc pipeline hoặc kỹ thuật cân bằng độ trễ để tối ưu về tốc độ và tài nguyên hệ thống.

Xung đột lợi ích

Các tác giả tuyên bố không có bất kỳ xung đột lợi ích nào trong bài báo này.

TÀI LIỆU THAM KHẢO

[1] H. E. Michail, G. S. Athanasiou, V. Kelefouras, G. Theodoridis, and a. C. E. Goutis, "On the exploitation of a high-throughput SHA-256 FPGA design for HMAC," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 5, no. 1, pp. 1-28, 2012.
[2] R. García, I. A. Badillo, M. M. Sandoval, C. F. Uribe, and R. Cumplido, "A compact FPGA-based processor for the Secure Hash Algorithm SHA-256," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 194-202, 2014.
[3] Y. Chen and S. Li, "A High-Throughput Hardware Implementation of SHA-256 Algorithm," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, Seville, Spain, 2020.
[4] A. Fairouz and S. P. Khatri, "An FPGA-Based Coprocessor for Hash Unit Acceleration," in *2017 IEEE International Conference on Computer Design (ICCD)*, Boston, MA, USA, 2017.
[5] I. L. R. Azevedo, A. S. Nery, and A. da C. Sena, "A SHA-3 Co-Processor for IoT Applications," in *2020 Workshop on Communication Networks and Power Systems (WCNPS)*, Brasilia, Brazil, 2020.
[6] S. Ni, Y. Dou, K. Chen, and L. Deng, "A Novel Design of Flexible Crypto Coprocessor and Its Application," in *Advanced Computer Architecture - 10th Annual Conference, ACA 2014*, Shenyang, China, 2014.
[7] NXP Semiconductors, "Crypto Coprocessor C29x", 2014 [Online]. Available: https://www.nxp.com/products/processors-and-microcontrollers/legacy-mpu-mcus/crypto-coprocessors/crypto-coprocessor:C29x.
[8] IBM, "IBM PCIe Cryptographic Coprocessor", 2023 [Online]. Available: https://www.ibm.com/products/pcie-cryptographic-coprocessor.
[9] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", *Federal Information Processing Standards Publication*, United States, 2015.
[10] K. K. Ting, S. C. L. Yuen, K. H. Lee, and P. H. W. Leong, "An FPGA Based SHA-256 Processor," in *Field-Programmable Logic and Applications, Reconfigurable Computing Is Going Mainstream*, Montpellier, France, 2002.
[11] R. McEvoy, F. Crowe, C. Murphy, and W. Marnane, "Optimisation of the SHA-2 family of hash functions on FPGAs," in *IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures (ISVLSI'06)*, Karlsruhe, Germany, 2006.
[12] M. Padhi and R. Chaudhari, "An optimized pipelined architecture of SHA-256 hash function," in *2017 7th International Symposium on Embedded Computing and System Design (ISED)*, Durgapur, India, 2017.

Nguyen Thi Hong Hieu is currently a student at the Ho Chi Minh City University of Technology and Education (HCMUTE), Vietnam. Her main research interests include semiconductor technology, responsible for full flow implementation of VLSI design. Email: 19119177@student.hcmute.edu.vn.
ORCID: https://orcid.org/0009-0002-3502-7767

Tran Thi Anh Duong is currently a student at the Ho Chi Minh City University of Technology and Education (HCMUTE), Vietnam. Her major is Computer Engineering and interested in front-end of VLSI design.
Email: 19119017@student.hcmute.edu.vn.
ORCID: https://orcid.org/0009-0002-6298-5784

Van Anh Duong received engineer's degree in Automotive engineering, in 2013, and Master's degree, in 2015 from Ho Chi Minh city, University of Technology and Education. From 2014-2021, he is a Lecturer at Cao Thang Technical College. He had instructed his students when they competed in Minicar Racing Contest, Eco Mileage Challenge in Ha Noi. From 2022, he is a Lecturer at Ho Chi Minh City, University of Technology and Education (HCMUTE). In 2023, his students participated in Robocon 2023 with him. Besides, He still guides graduation thesis for his students. His research includes Automotive Powertrains system, Automotive Chassis System, Vehicle Stability Control and material for batteries for EV.
Email: duongva@hcmute.edu.vn. ORCID: https://orcid.org/0000-0003-1572-4266

Ho Nhut Minh is currently a lecturer at the Posts and Telecommunications Institute of Technology. He earned his Bachelor's degree in Electronics and Telecommunications Engineering from Ho Chi Minh City University of Technology and Education in 2010. In 2014, Minh completed his Master's degree in Telecommunication Engineering at the Ho Chi Minh City campus of the Posts and Telecommunications Institute of Technology. His research interests encompass a wide range of fields, including power converters, machine drives, wind power generation, power quality, power systems, Internet of Things (IoT), embedded systems, and machine learning. Email: minhhn@ptit.edu.vn.
ORCID: https://orcid.org/0009-0003-2204-6990

Nguyen Ngo Lam is currently a lecturer at the Faculty For High Quality Training, Ho Chi Minh City University of Technology and Education. He received his Bachelor and Master degree in radio and electronics engineering from the Ho Chi Minh City University of Technology, Vietnam in 2000 and 2004 respectively. His research interests include wireless communication, data communication, digital signal processing, computer. Email: lamnn@hcmute.edu.vn.
ORCID: https://orcid.org/0009-0002-6580-0175

Truong Quang Phuc was born in Can Tho City, Vietnam. He received the B.Eng degree in Electronics and Telecommunication engineering and the M.Eng degree in Electronics engineering from the Ho Chi Minh City University of Technology and Education, Vietnam, in 2011 and 2014, respectively. Currently, He is with the Faculty of Electrical and Electronics Engineering, Ho ChiMinh City University of Technology and Education, Vietnam as a Senior Lecturer and Ph.D student as well. His research interests include convex optimization techniques, heterogeneous networks, Internet of Things, and Intelligent Reflecting Surfaces (IRS). Email: phuctq@hcmute.edu.vn.

ORCID: https://orcid.org/0000-0003-2344-9436