

**ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ**

-----o0o-----



LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC

**BỘ MÃ HÓA ĐƯỜNG CONG ELLIPTIC
256 BIT SỬ DỤNG FPGA**

GVHD: TS. Trần Hoàng Linh

SVTH: Nguyễn Tuấn Hùng

Vương Đình Hưng

MSSV: 1511359

1511422

TP. HỒ CHÍ MINH, THÁNG 06 NĂM 2019

----- ☆ -----

----- ☆ -----

Số: _____/BKĐT
Khoa: **Điện – Điện tử**
Bộ Môn: **Điện Tử**

NHIỆM VỤ LUẬN VĂN TỐT NGHIỆP

1. HỌ VÀ TÊN: Nguyễn Tuấn Hùng MSSV: 1511359
Vương Đình Hưng 1511422
2. NGÀNH: **ĐIỆN TỬ - VIỄN THÔNG** LỚP: DD15KSVT
3. Đề tài: Bộ mã hóa đường cong elliptic 256 bit sử dụng FPGA
4. Nhiệm vụ (Yêu cầu về nội dung và số liệu ban đầu):
 - Tạo chữ kí số ECDSA theo chuẩn FIPS 186-4, không bao gồm kiểm tra chữ kí.
 - Tạo khóa công khai cho ECDHE và tính toán khóa chia sẻ dựa theo chuẩn ANSI X9.63/IEEE 1363.
 - Hỗ trợ cho 2 đường cong elliptic 256-bit : NIST P-256 (Weierstrass curves), X25519 (Montgomery curves).
5. Ngày giao nhiệm vụ luận văn: 06/09/2019
6. Ngày hoàn thành nhiệm vụ: 31/05/2019
7. Họ và tên người hướng dẫn: Ts. Trần Hoàng Linh

Nội dung và yêu cầu LVTN đã được thông qua Bộ Môn.

Tp.HCM, ngày 19 tháng 06 năm 2019
CHỦ NHIỆM BỘ MÔN

NGƯỜI HƯỚNG DẪN CHÍNH

PHẦN DÀNH CHO KHOA, BỘ MÔN:

Người duyệt (chấm sơ bộ):.....
Đơn vị:.....
Ngày bảo vệ:.....
Điểm tổng kết:
Nơi lưu trữ luận văn:

LỜI CẢM ƠN

Nhóm chúng em xin gửi lời cảm ơn sâu sắc đến thầy Trần Hoàng Linh và các quý thầy cô đã tận tình hướng dẫn, cùng các bạn, các anh chị sinh viên đã giúp đỡ tận tình trong quá trình thực hiện luận văn tốt nghiệp này.

Cảm ơn các thầy cô vì đã cho chúng em được sống một tuổi trẻ Bách Khoa đáng nhớ. Cảm ơn thầy cô vì đã đánh giá cao sự tiến bộ hơn điểm trung bình môn, đồng hành bên từng bước trưởng thành trên con đường học tập của chúng em.

Cảm ơn ba mẹ đã nuôi nấng và tạo điều kiện cho chúng con học tập trong suốt thời gian qua, chăm lo để chúng con có thể cất cặp sách vào học ngôi trường Bách Khoa.

Nhờ sự hướng dẫn, kinh nghiệm của các thầy cô và các bạn, nhóm chúng em có thể thực hiện được đề tài tương ứng với yêu cầu đặt ra. Tuy còn một số khuyết điểm, nhưng nhóm chúng em cũng đã học được nhiều kiến thức giá trị, có ích cho sự nghiệp điện tử trong tương lai.

Xin cảm ơn Arrive Technologies Vietnam đã hỗ trợ phòng nghiên cứu cũng như chia sẻ kinh nghiệm để nhóm chúng em có thể hoàn thành luận văn này.

Tp. Hồ Chí Minh, ngày 19 tháng 06 năm 2019
Sinh viên

TÓM TẮT LUẬN VĂN

Luận văn này trình bày về thiết kế bộ mã hóa đường cong elliptic 256 bit trên ngôn ngữ verilog, phục vụ trong mục đích bảo mật thông tin. Luận văn trình bày đầy đủ các mặt về lý thuyết toán học của mã hóa đường cong elliptic, chức năng và ứng dụng. Thiết kế của luận văn được miêu tả với đầy đủ các sơ đồ khối và cách thức hoạt động. Thiết kế được mô phỏng để kiểm tra kết quả bằng các bài kiểm tra được quy định trong các chuẩn về mã hóa. Thiết kế được tổng hợp bằng Quartus trên nền tảng FPGA Cyclone V. Luận văn cũng trình bày về các ưu, khuyết điểm còn tồn tại của thiết kế và hướng phát triển tương lai cho đề tài này.

MỤC LỤC

1. GIỚI THIỆU	1
1.1. Tổng quan	1
1.2. Tình hình nghiên cứu trong và ngoài nước	1
1.3. Nhiệm vụ luận văn.....	2
2. LÝ THUYẾT VỀ MÃ HÓA ĐƯỜNG CONG ELLIPTIC	3
2.1. Giao thức SSL/TLS, IPsec.....	3
2.1.1. SSL/TLS	3
2.1.2. IPsec	5
2.2. Giao tiếp mã hóa khóa bất đối xứng.....	6
2.3. Mã hóa đường cong elliptic	7
2.3.1. Cơ sở toán học	7
2.3.2. Đường cong elliptic	8
2.3.3. Đường cong Weierstrass rút gọn	8
2.3.4. Đường cong Montgomery	10
2.3.5. Elliptic curve Diffie – Hellman Ephemeral (ECDHE)	10
2.3.6. Elliptic curve digital signature algorithm (ECDSA)	11
2.3.7. Thông số của các đường cong elliptic sử dụng trong luận văn	12
3. THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG	12
3.1. Yêu cầu thiết kế	12
3.2. Phân tích	13
3.3. Sơ đồ khối tổng quát:.....	13
3.4. Sơ đồ khối chi tiết và nhiệm vụ, chức năng từng khối:	14
3.4.1. Bộ điều khiển chính (Main controller):	14
3.4.2. Bộ điều khiển số học (Arithmetic Unit Controller).....	18
3.4.3. Bộ tính toán số học (Arithmetic Unit).....	30
4. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM	37
4.1. Các giải thuật cho đường cong Montgomery	39
4.2. Các giải thuật cho đường cong Weierstrass	41
5. KẾT QUẢ THỰC HIỆN.....	42
5.1. Cách thức đo đạc, thử nghiệm	42
5.2. Số liệu đo đạc	43
5.3. Giải thích và phân tích về kết quả thu được	45

6.	KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	46
6.1.	Kết luận	46
6.2.	Hướng phát triển.....	47
7.	TÀI LIỆU THAM KHẢO	48
8.	PHỤ LỤC.....	52

DANH SÁCH HÌNH MINH HỌA

Hình 2-1 Quá trình bắt tay SSL/TLS (TLS 1.2)	3
Hình 2-2 Sơ đồ hoạt động của IPsec	5
Hình 2-3 Cộng và nhân điểm trên đường cong elliptic	9
Hình 3-1 IP Core ECC	13
Hình 3-2 Sơ đồ khối tổng quát ECC Core	13
Hình 3-3 Bộ điều khiển chính	14
Hình 3-4 Máy trạng thái MC	16
Hình 3-5 Sơ đồ giải thuật ECDSA tại MC	16
Hình 3-6 Sơ đồ giải thuật tạo chìa khóa ECDHE tại MC	17
Hình 3-7 Sơ đồ giải thuật tính chìa khóa ECDHE tại MC	17
Hình 3-8 Mô hình bộ điều khiển số học	18
Hình 3-9 Máy trạng thái của AUC	20
Hình 3-10 Sơ đồ khối bộ điều khiển số học (AUC)	21
Hình 3-11 Sơ đồ khối bộ nhân vô hướng cho đường cong Weierstrass	21
Hình 3-12 Sơ đồ khối bộ nhân vô hướng cho đường cong Montgomery	26
Hình 3-13 IP RAM tinh chỉnh của FPGA dùng M10K	27
Hình 3-14 8-bit LFSR feedback ở 8, 6, 5, 4	27
Hình 3-15 256-bit LFSR feedback ở 256, 254, 251, 246	28
Hình 3-16 Giải thuật chọn k	28
Hình 3-17 Sơ đồ khối bộ tính R / S / INV	29
Hình 3-18 Tính toán r	29
Hình 3-19 Tính toán s	30
Hình 3-20 Mô hình bộ tính toán số học	30
Hình 3-21 Sơ đồ khối AU	32
Hình 3-22 Máy trạng thái của AU	32
Hình 3-23 Bộ cộng modulo	33
Hình 3-24 Altera IP LPM cho ADD/SUB	34
Hình 3-25 Bộ nhân Montgomery	34
Hình 3-27 Cấu trúc bộ nhân Montgomery	35
Hình 3-28 Bộ nghịch đảo Montgomery	36
Hình 3-29 Bộ hoán đổi	37
Hình 4-1 Ngôn ngữ Python	38

Hình 4-2 Google Colab	38
Hình 4-3 Ví dụ về sử dụng Google Colab và một đoạn mã Python đơn giản	38
Hình 5-1 Kết quả timing của model 2_CLA_PL.....	44
Hình 5-2 Kết quả timing của model NO_CLA_PL.....	44
Hình 5-3 Kết quả tổng hợp của model 2_CLA_PL.....	45
Hình 5-4 Kết quả tổng hợp của model NO_CLA_PL.....	45
Hình 6-1 Intel FPGA Acceleration Hub PCIe card for servers	47
Hình 6-2 Sức mạnh bảo mật của các đường cong elliptic [4]	48
Hình 8-1 Kết quả waveform ECDHE GEN X25519 thiết kế NO_CLA_PL	52
Hình 8-2 Kết quả bộ nhớ ECDHE GEN X25519 thiết kế NO_CLA_PL	52
Hình 8-3 Kết quả waveform ECDHE COMP X25519 thiết kế NO_CLA_PL	52
Hình 8-4 Kết quả bộ nhớ ECDHE GEN X25519 thiết kế NO_CLA_PL	52
Hình 8-5 Kết quả waveform ECDHE GEN NIST-P256 thiết kế NO_CLA_PL.....	52
Hình 8-6 Kết quả bộ nhớ ECDHE GEN NIST-P256 thiết kế NO_CLA_PL.....	52
Hình 8-7 Kết quả waveform ECDSA NIST-P256 thiết kế NO_CLA_PL	53
Hình 8-8 Kết quả bộ nhớ ECDSA NIST-P256 thiết kế NO_CLA_PL	53
Hình 8-9 Kết quả waveform ECDHE GEN X25519 thiết kế 2_CLA_PL	54
Hình 8-10 Kết quả bộ nhớ ECDHE GEN X25519 thiết kế 2_CLA_PL	55
Hình 8-11 Kết quả waveform ECDHE COMP X25519 thiết kế 2_CLA_PL.....	55
Hình 8-12 Kết quả bộ nhớ ECDHE COMP X25519 thiết kế 2_CLA_PL	55
Hình 8-13 Kết quả waveform ECDHE GEN NIST-P256 thiết kế 2_CLA_PL.....	55
Hình 8-14 Kết quả bộ nhớ ECDHE GEN NIST-P256 thiết kế 2_CLA_PL.....	55
Hình 8-15 Kết quả waveform ECDSA NIST-P256 thiết kế 2_CLA_PL	56
Hình 8-16 Kết quả bộ nhớ ECDSA NIST-P256 thiết kế 2_CLA_PL	56

DANH SÁCH BẢNG SỐ LIỆU

Bảng 1 So sánh kích thước chìa khóa giữa RSA, DH, DSA và ECDHE, ECDSA.....	7
Bảng 2 Thông số đường cong elliptic secp256r1	12
Bảng 3 Thông số đường cong elliptic X25519.....	12
Bảng 4 Mô tả chân MC	14
Bảng 5 MC Mode[1:0]	15
Bảng 6 MC Mode[4:2]	15
Bảng 7 MC status	15
Bảng 8 MC Ngõ vào dữ liệu.....	15
Bảng 9 MC Ngõ ra dữ liệu	15
Bảng 10 Mô tả chân AUC	18
Bảng 11 AUC mode[2:0].....	19
Bảng 12 AUC mode[4:3].....	19
Bảng 13 AUC status	19
Bảng 14 AUC Ngõ vào dữ liệu	19
Bảng 15 Ngõ ra dữ liệu.....	20
Bảng 16 Số lượng phép cộng khi w tăng.....	22
Bảng 17 Mô tả chân AU	30
Bảng 18 AU Opcode [1:0].....	31
Bảng 19 AU Opcode [2].....	31
Bảng 20 AU Opcode [3].....	31
Bảng 21 AU status.....	31
Bảng 22 Kết quả timing và tài nguyên của thiết kế.....	44

1. GIỚI THIỆU

1.1. Tổng quan

Việc trao đổi thông tin, dữ liệu giữa các cá nhân, tập thể, doanh nghiệp thông qua internet đóng vai trò rất quan trọng trong thời đại hiện nay. Nhưng thực tế internet tồn tại nhiều lỗ hổng có thể bị kẻ xấu lợi dụng để đánh cắp thông tin đã đặt ra yêu cầu và thách thức cho lĩnh vực an ninh mạng. Hiện nay đã có nhiều cách để đảm bảo tính bảo mật thông tin, điển hình như IP security thực hiện ở lớp mạng trong mô hình OSI. Ngoài ra còn một cách tiếp cận khác được thực hiện phía trên lớp TCP: TLS/SSL.

TLS (Transport Layer Security) cùng với SSL (Secure Sockets Layer) là các giao thức mật mã nhằm mục đích bảo mật sự vận chuyển trên internet. Các giao thức này sử dụng mã hóa khóa bất đối xứng bằng các chứng thực X.509 để xác thực bên kia và trao đổi cho nhau một khóa đối xứng. Sau đó, khóa đối xứng này sẽ được dùng để mã hóa thông tin liên lạc giữa hai bên. Giao tiếp mã hóa TLS/SSL chủ yếu được thiết lập giữa một server và nhiều clients, từ đó, thông tin sẽ được mã hóa bởi CPU để tránh bị đánh cắp hay thay thế trong quá trình vận chuyển. Khi số lượng client kết nối tới server tăng lên, khối lượng dữ liệu mà server cần phải xử lý tăng lên khiến tiêu tốn công suất lớn. Vấn đề đặt ra lúc này là cần phải xây dựng một giải pháp giúp server có thể mã hóa dữ liệu với thông lượng lớn nhưng tiêu thụ ít năng lượng hơn.

Từ tháng 8/2018, TLS 1.3 đã được thông qua và sẽ dần thay thế TLS 1.2 hiện đang rất phổ biến. Trong đó, sự thay đổi chủ yếu là loại bỏ các giải thuật mã hóa bất đối xứng cũ, nặng nề và mở đường để mã hóa đường cong elliptic trở thành tiêu chuẩn trong tương lai. Nhiệm vụ của luận văn này là xây dựng một hệ thống thực hiện tính toán khóa bất đối xứng cho ECDHE và chữ ký số ECDSA.

1.2. Tình hình nghiên cứu trong và ngoài nước

Việc xây dựng một vi xử lý chuyên dụng, tối ưu hóa trong việc tính toán các phép mã hóa đường cong elliptic đã diễn ra từ rất lâu. Nhưng vì sự thống trị của mã hóa RSA cho đến thời gian gần đây, khi các thiết bị di động gọn nhẹ và Internet of things lên ngôi thì các dạng mã hóa đường cong elliptic mới được sử dụng nhiều vì tính chất của nó: tiêu tốn ít năng lượng hơn và ít bộ nhớ hơn.

Có rất nhiều phương pháp khác nhau để tính toán và xử lý mã hóa đường cong elliptic, các phương pháp khác nhau về hiệu suất, diện tích, bộ nhớ chiếm dụng, mức năng

lượng tiêu tốn cũng như mức độ bảo mật. Ở đây nhóm chúng em tập trung vào những nghiên cứu trên phần cứng ASIC và FPGA. Nghiên cứu của Izu và những người khác [21] cho thấy một phương pháp áp dụng với những thuật toán đơn giản và lợi dụng cấu trúc FPGA để đạt hiệu quả cao, đánh đổi với diện tích. Nghiên cứu trên cũng dựa vào gốc nghiên cứu của Aoki và những người khác [22]. Một nghiên cứu sử dụng nhân DSP đáng lưu ý là của Itoh [23], tuy vậy việc sử dụng nhân DSP đòi hỏi tùy chỉnh cụ thể về phần cứng không thích hợp để xây dựng một nhân mã hóa cho nhiều loại ứng dụng trên các phần cứng FPGA khác nhau.

C.Kocher chỉ ra khả năng bị tấn công bằng các phương pháp phân tích năng lượng đáng lưu ý trong việc xây dựng phần cứng [24] khi ứng dụng mã hóa đường cong trong các thiết bị như thẻ thông minh (smart card). Fischer và những người khác cũng đưa ra một cấu trúc ví dụ về việc chống phân tích năng lượng [25].

Nghiên cứu [26] của công ty IP Cores rất đáng để tham khảo, một IP Core ECC hỗ trợ nhiều đường cong và dễ dàng ứng dụng trong các FPGA, ASIC khác nhau. Tuy vậy lại không hỗ trợ những đường cong ngoài hệ nhị phân và chỉ sử dụng cho các ứng dụng đơn giản (không tối ưu cho các đường cong đặc biệt) và đòi hỏi bộ nhớ ngoài.

Nghiên cứu của nhóm chúng em được thực hiện với mong muốn xây dựng một IP Core xử lý mã hóa nhị phân vừa dễ ứng dụng và vừa tối ưu cho các đường cong SSL/TLS [26,21], có thể ứng dụng vào các hệ thống SSL/TLS Offload cho các máy chủ như [27] tại Việt Nam.

1.3. Nhiệm vụ luận văn

Nội dung 1: Tìm hiểu lý thuyết về giao thức TLS/SSL, IPSEC (IKE)

Nội dung 2: Tìm hiểu lý thuyết về giao tiếp mã hóa khóa bất đối xứng.

Nội dung 3: Tìm hiểu lý thuyết về mã hóa đường cong elliptic.

Nội dung 4: Thiết kế hệ thống tính toán khóa bất đối xứng cho ECDHE và chữ ký số ECDSA theo mục tiêu đã đặt ra như sau:

- Tạo chữ ký số ECDSA theo chuẩn FIPS 186-4, không bao gồm kiểm tra chữ ký.
- Tạo khóa công khai cho ECDHE và tính toán khóa chia sẻ dựa theo chuẩn ANSI X9.63/IEEE 1363.
- Hỗ trợ cho 2 đường cong elliptic 256-bit phổ biến hiện nay [14] như: NIST P-256 (Weierstrass curves), X25519 (Montgomery curves).

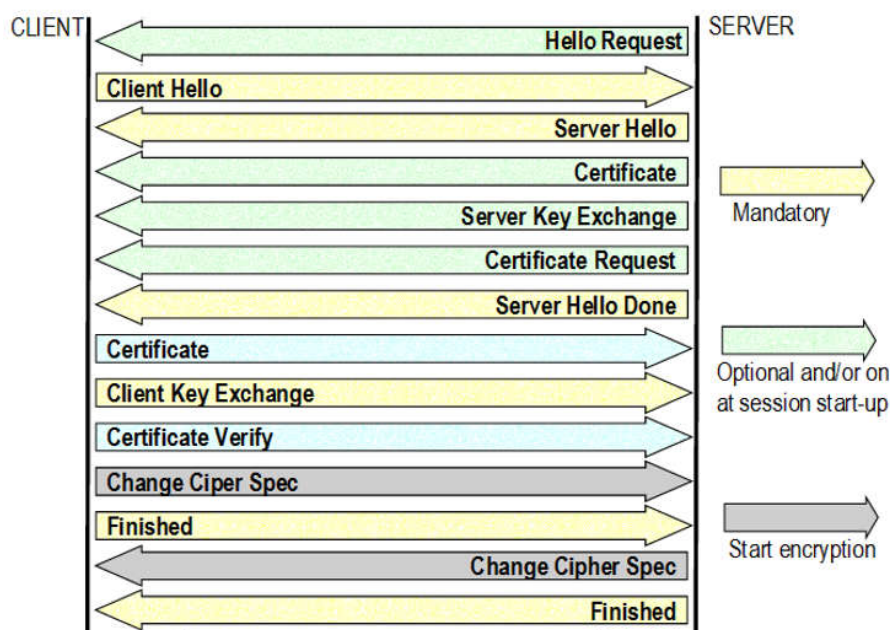
2. LÝ THUYẾT VỀ MÃ HÓA ĐƯỜNG CONG ELLIPTIC

2.1. Giao thức SSL/TLS, IPsec

2.1.1. SSL/TLS

TLS/SSL là các giao thức mã hóa nhằm cung cấp sự bảo mật trong mạng lưới máy tính và đã được ứng dụng rộng rãi vào các trình duyệt web, email, VoIP, ... Mục tiêu cơ bản của giao thức TLS/SSL là đảm bảo sự riêng tư và toàn vẹn dữ liệu giữa hai hay nhiều ứng dụng với nhau. Kết nối giữa một client và một server có bảo mật bởi giao thức TLS/SSL bao gồm các đặc điểm sau:

- Kết nối được bảo vệ nhờ việc mã hóa đối xứng được sử dụng để mã hóa các dữ liệu gửi đi. Chìa khóa để mã hóa đối xứng được tạo độc lập ứng với mỗi kết nối, dựa trên một thông tin chung đã được bí mật thương lượng ở đầu phiên làm việc. Server và client trao đổi chi tiết với nhau về loại giải thuật và chìa khóa mã hóa nào sẽ được dùng trước khi bất kì byte dữ liệu nào được truyền đi. Quá trình thương lượng này được đảm bảo an toàn và đáng tin cậy (bên thứ ba không thể lấy cắp, giải mã hay thay thế thông tin được gửi đi mà không bị phát hiện).
- Danh tính giữa những client và server có thể xác thực bằng mã hóa khóa bất đối xứng. Quá trình xác thực này không bắt buộc nhưng ít nhất một trong hai phải thực hiện (thường là server).
- Đảm bảo kết nối đáng tin cậy vì mỗi tin nhắn truyền đi bao gồm một dòng mã ứng với tin nhắn đó, đề phòng trường hợp tin nhắn bị mất hay thay thế khi vận chuyển.



Hình 2-1 Quá trình bắt tay SSL/TLS (TLS 1.2)

Quá trình bắt tay giữa server và client sẽ diễn ra tuần tự như sau:

- Hello Request: cho phép server khởi động lại quá trình bắt tay nhưng ít khi được dùng. Nếu kết nối đã được duy trì trong thời gian dài, server có thể dùng gói tin này để bắt client thương lượng lại một khóa phiên mới, bảo mật hơn.
- Client Hello: nếu client muốn kết nối tới server thì tin nhắn đầu tiên phải là Client Hello. Gói tin bao gồm các thông tin quan trọng như mã phiên, cipher suite (tập hợp các thuật toán dùng để mã hóa) và một số thông tin mở rộng.
- Server Hello: server sẽ gửi tin nhắn này để đáp lại Client Hello sau khi nó đã chọn được một tập thuật toán mã hóa thích hợp.
- Server Certificate: server bắt buộc phải gửi gói tin này để xác minh danh tính của nó.
- Server Key Exchange: gói tin này sẽ được gửi ngay lập tức sau gói Server Certificate (hay Server Hello nếu đây là một kết nối ẩn danh). Server Key Exchange sẽ được gửi nếu Server Certificate không có đủ dữ liệu để client trao đổi pre-master secret.
- Certificate Request: một server ẩn danh có thể yêu cầu xác thực từ client, nếu phù hợp với cipher suite đã chọn.
- Server Hello Done: tin nhắn được gửi để thông báo kết thúc quá trình chào hỏi của server và đợi hồi đáp từ client.
- Client Certificate: đây là tin nhắn đầu tiên mà client có thể gửi sau khi nhận được Server Hello Done và chỉ gửi nếu server yêu cầu xác thực.
- Client Key Exchange: gói tin này luôn luôn được truyền đi bởi client và nó phải theo sau Client Certificate, nếu gói tin này được gửi. Với tin nhắn này, pre-master secret sẽ được thiết lập.
- Certificate Verify: đây là tin nhắn gửi kèm với Client Certificate để giúp server kiểm tra danh tính client.
- Finished: thông báo quá trình xác thực và trao đổi khóa đối xứng thành công

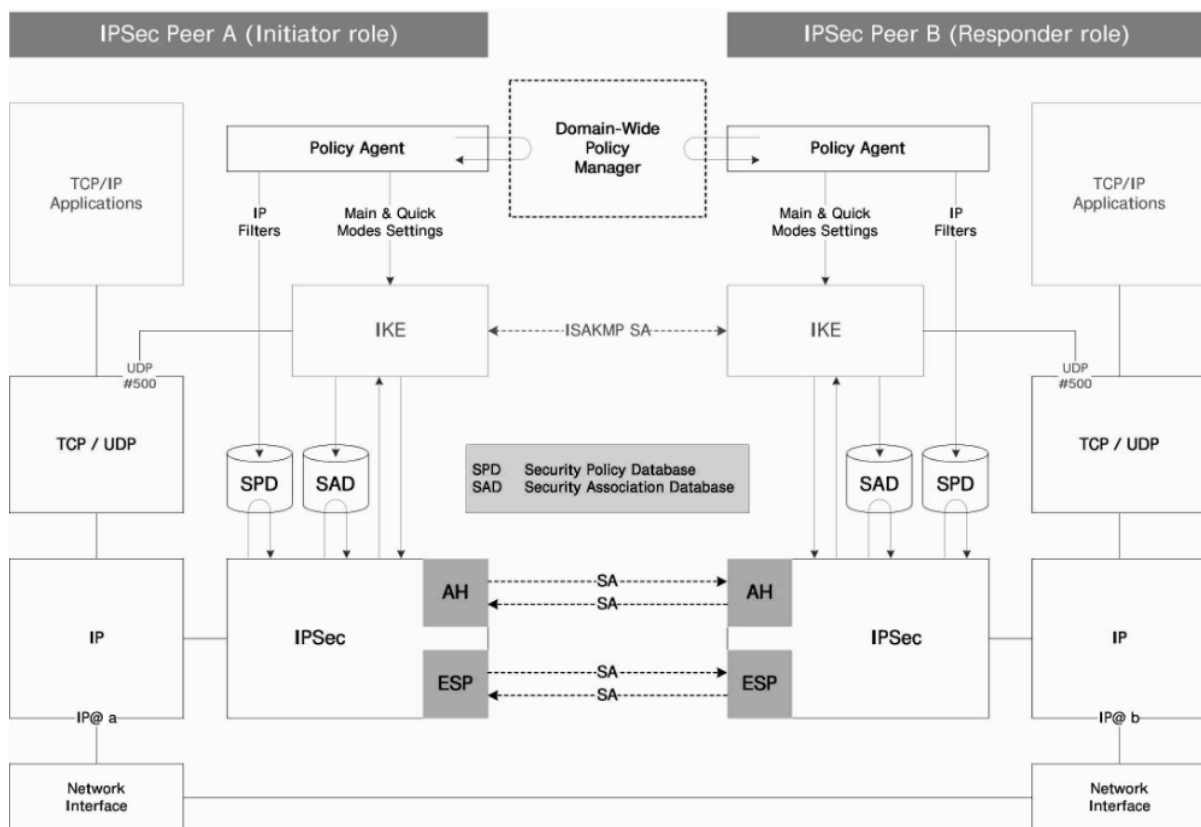
Bên cạnh đó, các giao thức TLS sau này còn bổ sung thêm tính năng forward secrecy nhằm đảm bảo rằng bất kì chìa khóa mã hóa nào bị lộ ra trong tương lai cũng không thể sử dụng để giải mã các tin nhắn đã gửi đi trong quá khứ.

Trong khung nhìn mô hình TCP/IP, TLS và SSL đều mã hóa dữ liệu của các kết nối mạng trên một tầng phụ thấp của tầng ứng dụng. Theo hệ thống tầng cấp của mô hình OSI, TLS/SSL được khởi chạy ở tầng 5 (tầng phiên) rồi hoạt động trên tầng 6 (tầng trình diễn): trước tiên tầng phiên bắt tay dùng mật mã bất đối xứng để đặt cấu hình mật mã và chìa khóa chia sẻ dành cho phiên đó; sau đó, tầng trình diễn mã hóa phần còn lại của thông điệp dùng mật mã đối xứng và khóa của phiên đó. Trong cả hai mô hình, TLS và SSL phục vụ tầng giao vận bên dưới, các đoạn trong tầng này chứa dữ liệu mật mã hóa.

Quá trình chia sẻ chìa khóa chung thông qua kênh không bảo mật của SSL/TLS được thực hiện bằng giao tiếp mã hóa khóa bất đối xứng được làm rõ ở các phần sau.

2.1.2. IPsec

IPSEC là nền tảng mã hóa ở Layer 3, ở mức IP. IPsec tạo các tunnel để các IP Packet có thể lưu thông một cách bảo mật. IPSEC có thể bảo mật cả địa chỉ IP nếu lựa chọn. Hình vẽ sau tương đối cho thấy rõ toàn bộ hoạt động của IPsec:



Hình 2-2 Sơ đồ hoạt động của IPsec

IPSEC chứa hai database ở mỗi host, Security Policy Database (SPD) và Security Association Database (SAD). Sử dụng hai database này và dựa trên việc phân loại packet đầu

vào và packet đầu ra, các packet hoạt động trong các kênh bảo mật của IPsec và được mã hóa bằng một chìa khóa chung thông qua các giải thuật mật mã đối xứng.

Để chia sẻ chìa khóa chung cho các kênh bảo mật, IPsec sử dụng một giao thức gọi là Internet Key Exchange (hiện có hai phiên bản IKEv1 và IKEv2). Internet Key Exchange giống như quá trình bắt tay SSL/TLS tạo ra một SA gọi là ISAKMP SA (IKEv1) hay IKE SA (IKEv2). IKE cho phép hai đầu host IPSEC xác thực lẫn nhau và tạo một kênh bảo mật được bảo mật bằng hai SA trên, qua kênh bảo mật này thì hai host sẽ chia sẻ các IPSEC SA hay còn được gọi là các Child SA (IKEv2) để tạo cơ chế bảo mật cho các kênh bảo mật IPSEC.

Quá trình tạo ra ISAKMP SA hay IKE SA đòi hỏi IPsec thực hiện giao tiếp mã hóa bất đối xứng nhằm xác thực danh tính hai bên và chia sẻ chìa khóa chung trên mạng không bảo mật. Quá trình này được làm rõ ở phần tiếp theo.

2.2. Giao tiếp mã hóa khóa bất đối xứng

Mã hóa khóa công khai, hay còn gọi là mã hóa khóa bất đối xứng, sử dụng một cặp chìa gồm: public key và private key. Trong đó, public key có thể được công bố rộng rãi cho bất kì người dùng nào, nhưng private key chỉ được giữ riêng bởi chủ sở hữu. Hai chức năng phổ biến nhất của mã hóa khóa công khai gồm:

- Mã hóa dữ liệu: dữ liệu được mã hóa bằng public key chỉ có thể giải mã bằng private key tương ứng.
- Chữ ký số: người gửi kết hợp private key và tin nhắn cần gửi để tạo ra một chữ ký số, sau đó tin nhắn này có thể kiểm tra bởi bất kì người dùng nào có public key của người gửi.

Trong giao thức TLS/SSL, giao tiếp mã hóa bất đối xứng được ứng dụng vào quá trình trao đổi pre-master secret, cơ sở để tạo khóa đối xứng. Các phương pháp được ứng dụng nhiều gồm có: RSA, DSA, DHE, ECDSA, ECDHE. Thực tế, với yêu cầu bảo mật ngày càng cao, kích thước chìa khóa của giải thuật RSA và DHE ngày càng lớn, gây nặng nề trong việc tính toán và truyền gửi. Thay vào đó, các giải thuật sử dụng tính chất của đường cong elliptic cho ra mức độ bảo mật tương đương nhưng kích thước chìa khóa lại giảm đi đáng kể.

Bảng 1 So sánh kích thước chìa khóa giữa RSA, DH, DSA và ECDHE, ECDSA

Kích thước chìa khóa của RSA, DSA và Diffie-Hellman (bits)	Kích thước chìa khóa của ECDHE và ECDSA (bits)
1024	160
2048	224
3072	256
7680	384
15360	521

2.3. Mã hóa đường cong elliptic

2.3.1. Cơ sở toán học

Nhóm: là cấu trúc bao gồm một tập G và một toán tử nhị nguyên trên G (toán tử đó có thể là phép cộng, phép nhân hoặc là một phép tính toán học khác). Với mỗi cặp số (a, b) , các tiên đề sau được thỏa mãn:

- $a, b \in G \Rightarrow a \cdot b \in G$
- $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ với mọi $a, b, c \in G$
- Tồn tại $e \in G$ thỏa mãn $a \cdot e = e \cdot a = a$ với mọi $a \in G$
- Với mỗi $a \in G$, tồn tại một phần tử $a' \in G$ sao cho $a \cdot a' = a' \cdot a = e$ (a' được gọi là phần tử nghịch đảo của a).

Chúng ta sử dụng kí hiệu $\{G, \cdot\}$ cho nhóm nhân và $\{G, +\}$ cho nhóm cộng. Trong nhóm cộng, phần tử trung hòa là 0 và phần tử trung hòa của a là $-a$; còn trong nhóm nhân, phần tử trung hòa là 1 và phần tử nghịch đảo của a là a^{-1} . Nếu một nhóm G có hữu hạn phần tử thì nó là nhóm hữu hạn và bậc của nhóm G là số phần tử có trong nhóm, nếu không thì G là nhóm vô hạn.

Một nhóm được gọi là nhóm Abelian nếu thỏa: $a \cdot b = b \cdot a$ với mọi $a, b \in G$.

Nhóm cyclic: một nhóm G được gọi là cyclic nếu mỗi phần tử trong G là lũy thừa a^k (k là số nguyên) của một phần tử cố định $a \in G$. Lúc này phần tử a gọi là phần tử khởi tạo của nhóm G . Một nhóm cyclic luôn luôn là nhóm Abelian.

Vành: là một tập R với hai toán tử $+$ và \cdot , thỏa các điều kiện sau:

- $\{R, +\}$ là một nhóm Abelian..
- $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ với mọi $a, b, c \in R$
- $a \cdot (b+c) = a \cdot b + a \cdot c$ và $(a+b) \cdot c = a \cdot c + b \cdot c$ với mọi $a, b, c \in R$

Trường: một trường F là tập hợp các phần tử với hai toán tử cộng và nhân. Trường bao gồm hết các tính chất của vành và thỏa thêm điều kiện:

- Với mỗi phần tử a trong trường F , ngoại trừ 0 , tồn tại phần tử $a^{-1} \in F$ để $a \cdot a^{-1} = 1$

Trường hữu hạn bậc p (prime field): cho một số nguyên tố p , trường hữu hạn bậc p , $GF(p)$, được định nghĩa là tập $Z_p = \{0, 1, \dots, p-1\}$, cũng với các phép toán đại số modulo p . Tất cả các phần tử trong Z_p (trừ 0) đều tồn tại giá trị nghịch đảo modulo (tức là tồn tại $z, w \in Z_p$ sao cho $wz = 1 \pmod{p}$ hay $z = w^{-1} \pmod{p}$)

Trường hữu hạn bậc 2^m (binary field): cách để tạo nên một trường hữu hạn bậc 2^m (F_{2^m}) là thông qua biểu diễn đa thức cơ bản. Ở đây, các phần tử trong F_{2^m} là các đa thức nhị phân (đa thức có hệ số thuộc trường $F_2 = \{0, 1\}$) với bậc lớn nhất là $m-1$:

$$F_{2^m} = a_{m-1}z^{m-1} + a_{m-2}z^{m-2} + \dots + a_1z + a_0 \quad ; a_i \in \{0, 1\}$$

2.3.2. Đường cong elliptic

Một đường cong elliptic E trên trường K được định nghĩa bởi phương trình sau cùng với một điểm O tại vô cùng:

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

Trong đó, $a_1, a_2, a_3, a_4, a_6 \in K$ và $\Delta \neq 0$

$$\Delta = -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6$$

$$d_2 = a_1^2 + 4a_2$$

$$d_4 = 2a_4 + a_1a_3$$

$$d_6 = a_3^2 + 4a_6$$

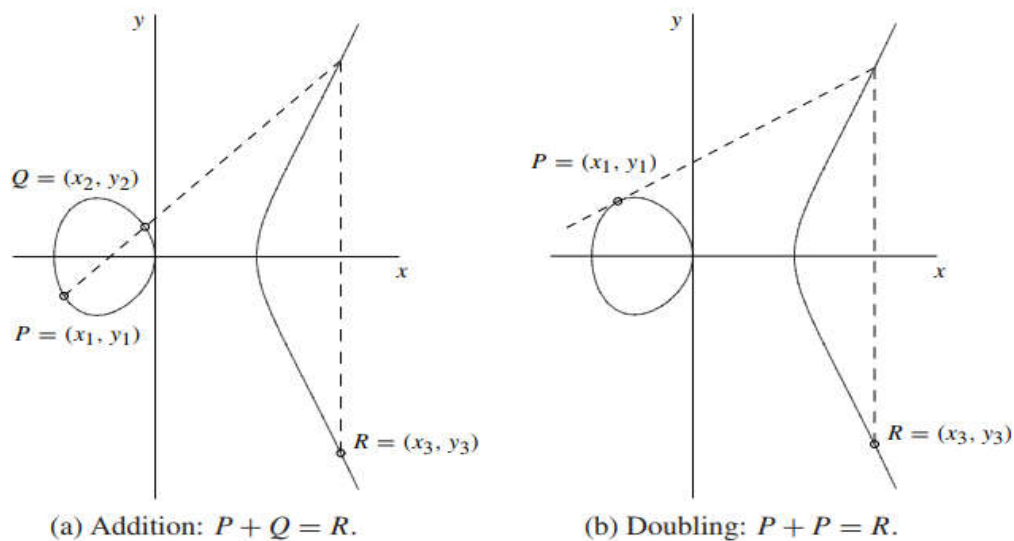
$$d_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2$$

2.3.3. Đường cong Weierstrass rút gọn

Ứng dụng thực tế trong mã hóa đường cong elliptic, người ta thường sử dụng dạng rút gọn của đường cong Weierstrass:

$$y^2 = x^3 + ax + b \quad (a, b \in K \text{ và } \Delta = -(4a^3 + 27b^2) \neq 0)$$

Về khía cạnh hình học, để xác định điểm $R = P + Q$ ($P, Q \neq O, P \neq -Q$), đường thẳng đi qua hai điểm P và Q sẽ cắt đường cong tại điểm thứ ba R' , lấy đối xứng điểm thứ ba qua tung độ thu được điểm R . Tương tự, để tìm điểm $R = 2P = P + P$, ta vẽ tiếp tuyến đi qua P và cắt đường cong tại điểm R' , lấy đối xứng qua tung độ ta thu được điểm R .



Hình 2-3 Cộng và nhân điểm trên đường cong elliptic

Về mặt đại số, điểm P, Q biểu diễn dưới tọa độ Affine có dạng $P(x_1:y_1)$ và $Q(x_2:y_2)$. Công thức tổng quát cho phép tính $R(x_3:y_3) = P + Q$ ($P, Q \neq O, P \neq -Q$):

$$x_3 = \lambda^2 - x_1 - x_2; \quad y_3 = \lambda(x_1 - x_3) - y_1$$

$$\text{với } \lambda = \frac{y_1 - y_2}{x_1 - x_2} \text{ nếu } P \neq Q \text{ và } \lambda = \frac{3x_1^2 + a_4}{2y_1} \text{ nếu } P = Q$$

Để hạn chế việc tính toán nghịch đảo modulo khi cộng hai điểm, các điểm trên đường cong elliptic thường được chuyển sang hệ tọa độ xạ ảnh, cụ thể là tọa độ Jacobian. Điểm $P(x_1:y_1)$ lúc này sẽ được biểu diễn thành $P(X_1:Y_1:Z_1)$ với $x_1 = \frac{X_1}{Z_1}$ và $y_1 = \frac{Y_1}{Z_1}$. (Lưu ý rằng nếu $Z_1 = 0$ thì $P = O$). Phương trình đường cong trở thành:

$$Y^2 = X^3 + a_4XZ^4 + a_6Z^6$$

Công thức tính cộng hai điểm $P(X_1:Y_1:Z_1)$ và $Q(X_2:Y_2:Z_2)$ với điều kiện $Q \neq \pm P$ và $P, Q \neq O$ cho ra điểm $R(X_3:Y_3:Z_3)$:

$$X_3 = R^2 + G - 2V; \quad Y_3 = R(V - X_3) - S_1G; \quad Z_3 = Z_1Z_2H$$

$$R = S_1 - S_2; \quad G = H^3; \quad V = U_1H^2; \quad S_1 = Y_1Z_2^3; \quad S_2 = Y_2Z_1^3$$

$$H = U_1 - U_2; \quad U_1 = X_1Z_2^2; \quad U_2 = X_2Z_1^2$$

Nhân đôi điểm $P(X_1:Y_1:Z_1)$ để thu được điểm $R(X_3:Y_3:Z_3)$, ta có công thức sau:

$$X_3 = M^2 - 2S; \quad Y_3 = M(S - X_3) - 8T; \quad Z_3 = 2Y_1Z_1$$

$$M = 3X_1^2 + a_4Z_1^4; \quad T = Y_1^4; \quad S = 4X_1Y_1^2$$

2.3.4. Đường cong Montgomery

Một đường cong Montgomery trên trường F_p được định nghĩa bởi phương trình:

$$By^2 = x^3 + Ax^2 + x \quad (A, B \in F_p, B \neq 0, A^2 \neq 4)$$

Chuyển sang hệ tọa độ xạ ảnh với $x = \frac{X}{Z}$ và $y = \frac{Y}{Z}$ ta có phương trình:

$$BY^2Z = X^3 + AX^2Z + XZ^2$$

Trong tọa độ Affine, công thức tính $R(x_3, y_3) = P + Q$ ($P, Q \neq O, P \neq -Q$) có dạng:

$$x_3 = B\lambda^2 - A - x_1 - x_2; \quad y_3 = \lambda(x_1 - x_3) - y_1$$

$$\text{với } \lambda = \frac{y_1 - y_2}{x_1 - x_2} \text{ nếu } P \neq Q \text{ và } \lambda = \frac{3x_1^2 + 2Ax_1 + 1}{2By_1} \text{ nếu } P = Q$$

Tiếp theo, ta chuyển sang hệ tọa độ xạ ảnh đối với đường cong Montgomery. Lợi thế của đường cong này trong tọa độ xạ ảnh đó là có thể tính được điểm R mà không cần quan tâm đến tọa độ Y trong suốt quá trình tính toán. Như vậy, ta chỉ tập trung vào tọa độ X và Z . Điểm $P(x:y)$ sẽ chuyển thành $P(X:Y:Z)$ hay $P(X:Z)$ với $Z \neq 0$, riêng với hai điểm đặc biệt $O=(0:1:0)$ và $T=(0:0:1)$ khi sang tọa độ xạ ảnh sẽ là $O=(1:0)$ và $T=(0:0)$.

Giả sử có tọa độ của điểm $P-Q \neq \{O, T\}$. Công thức tính cộng hai điểm $P(X_1:Z_1)$ và $Q(X_2:Z_2)$ với điều kiện $Q \neq \pm P$ và $P, Q \neq \{O, T\}$ để thu được $R(X_3:Z_3)$:

$$X_3 = Z_{P-Q}[(X_1 - Z_1)(X_2 + Z_2) + (X_1 + Z_1)(X_2 - Z_2)]^2$$

$$Z_3 = X_{P-Q}[(X_1 - Z_1)(X_2 + Z_2) - (X_1 + Z_1)(X_2 - Z_2)]^2$$

Nhân đôi điểm $P(X_1:Z_1)$ để thu được điểm $R(X_3:Z_3)$, ta có công thức sau:

$$X_3 = (X_1 + Z_1)^2(X_1 - Z_1)^2$$

$$Z_3 = 4X_1Z_1[(X_1 - Z_1)^2 + \frac{A+2}{4}(4X_1Z_1)]$$

2.3.5. Elliptic curve Diffie – Hellman Ephemeral (ECDHE)

ECDHE là phương pháp được dùng trong quá trình trao đổi khóa giữa sever và client, để kết quả cuối cùng là một pre-master secret mà hai bên đều biết. Trong quá trình bắt tay, sau khi server và client đã thống nhất được cipher suite và đường cong sẽ sử dụng, từ đó cả hai biết được miền tham số gồm các giá trị:

- p : số nguyên tố p định nghĩa cho trường F_p
- a, b : tham số của phương trình đường cong

- $G(x, y)$: tọa độ điểm bắt đầu
- n : bậc của G , được định nghĩa là số nguyên nhỏ nhất mà $nG = O$
- h : tỉ số $\frac{|E(F_p)|}{n}$, các đường cong được chuẩn hóa thường có $h=1$

Quá trình thỏa thuận pre-master secret sẽ diễn ra như sau:

- Server chọn một số ngẫu nhiên k ($0 < k < n$), đây chính là server private key. Sau đó, server tạo public key kG và gửi đi cho client.
- Client chọn một số ngẫu nhiên k' ($0 < k' < n$), đây chính là client private key. Sau đó, client tạo public key $k'G$ và gửi cho server.
- Server và client sau khi có được public key của đầu còn lại có thể tính ra được pre-master secret $P = kk'G = k'kG$

2.3.6. Elliptic curve digital signature algorithm (ECDSA)

Mã hóa đường cong elliptic còn ứng dụng được để tạo chữ ký số. Tương tự như ECDHE, quá trình bắt tay cũng sẽ quyết định đường cong cho ECDSA với miền tham số trên trường F_p gồm:

- p : kích thước của trường hữu hạn F_p , $p > 3$ và là một số nguyên tố
- SEED (không bắt buộc): một chuỗi ký tự dài ít nhất 160 bits
- a, b : tham số định nghĩa phương trình đường cong
- $G(x, y)$: tọa độ điểm bắt đầu, $G \neq O$
- n : bậc của G
- h : tỉ số $\frac{|E(F_p)|}{n}$

Mục tiêu luận văn chỉ quan tâm đến quá trình tạo chữ ký, trong đó cần một private key d trước tiên, rồi đến quá trình tạo chữ ký số sẽ diễn ra như sau:

- Bước 1: chọn ngẫu nhiên một số k , $0 < k < n$
- Bước 2: tính $kG = (x_1 : y_1)$
- Bước 3: tính $r = x_1 \bmod(n)$. Nếu $r = 0$, quay lại bước 1
- Bước 4: tính $e = H(m)$ với $H(m)$ là hàm mã hóa một chiều với tin nhắn m . Lưu ý rằng chỉ lấy số bit cao nhất ứng với chiều dài yêu cầu của đường cong.
- Bước 5: tính $s = k^{-1}(e + dr) \bmod(n)$. Nếu $s = 0$, quay lại bước 1
- Bước 6: Trả về chữ ký số (r, s) .

2.3.7. Thông số của các đường cong elliptic sử dụng trong luận văn

Bao gồm các thông số những đường cong elliptic được hỗ trợ: secp256r1 và X25519.

[11] [13] [6]

Bảng 2 Thông số đường cong elliptic secp256r1

Thông số	Giá trị
p	FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFF
a	FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFC
b	5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E 27D2604B
G	04 6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296 4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE3357 6B315ECE CBB64068 37BF51F5
n	FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2 FC632551
h	01

Về điểm G của các đường cong elliptic thuộc SECP (NIST – Curves), bắt đầu bằng 0x04 chỉ dạng không nén, đầy đủ tọa độ (x, y) của thông số này.

Bảng 3 Thông số đường cong elliptic X25519

Thông số	Giá trị
p	$2^{255} - 19$
a	486662
U(P)	09
V(P)	147816194475895447910205935684099868872646061346164752889648818 37755586237401
n	$2^{252} + 0x14def9dea2f79cd65812631a5cf5d3ed$
h	08

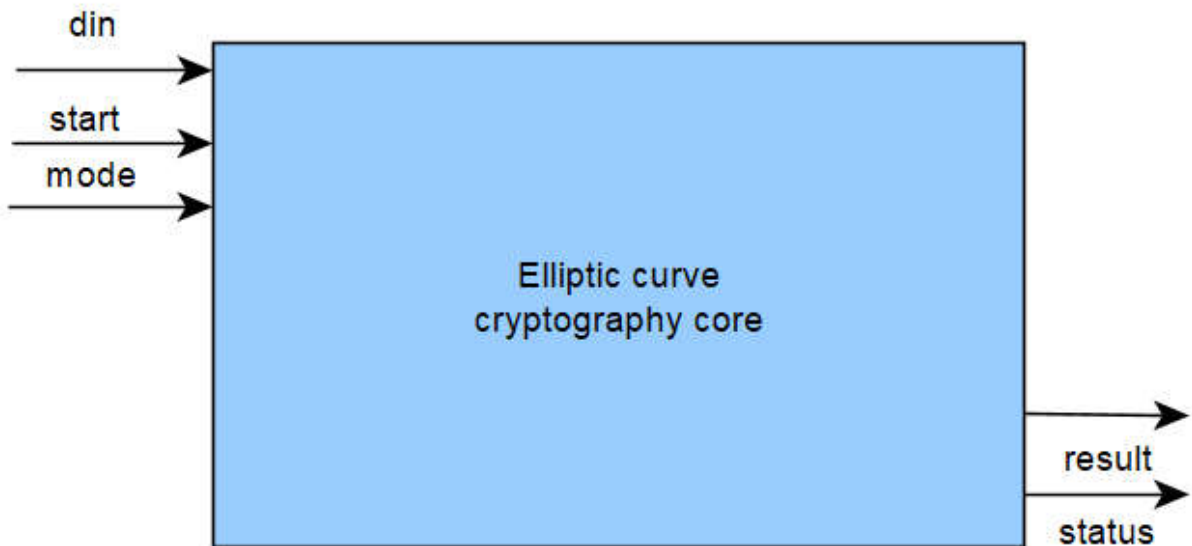
3. THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG

3.1. Yêu cầu thiết kế

- Thiết kế bộ mã hóa elliptic curve bằng ngôn ngữ phần cứng Verilog dưới dạng một Intellectual Property (IP) core để có thể sử dụng trong nhiều dự án khác nhau, dễ dàng thay đổi, phù hợp cho ASIC và FPGA.
- Thiết kế đảm bảo tiết kiệm tài nguyên (đủ sử dụng cho FPGA Altera Cyclone V)
- Thiết kế đạt được tốc độ xử lý trong khoảng 100ms cho 1 tác vụ.

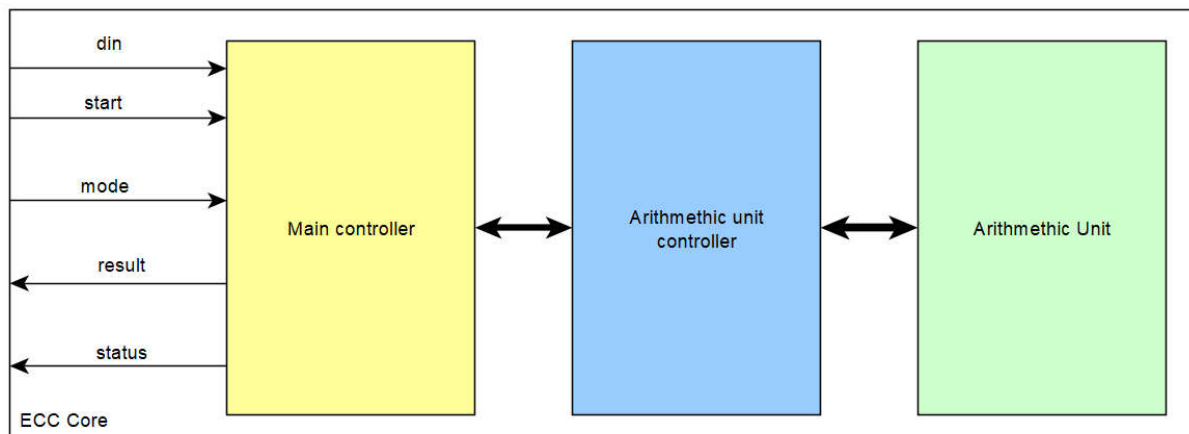
3.2. Phân tích

Phương pháp thiết kế được thực hiện dưới dạng các mô đun phần cứng trực quan, dễ thay đổi sửa chữa với mỗi mô đun là một giải thuật toán học.



Hình 3-1 IP Core ECC

3.3. Sơ đồ khối tổng quát:

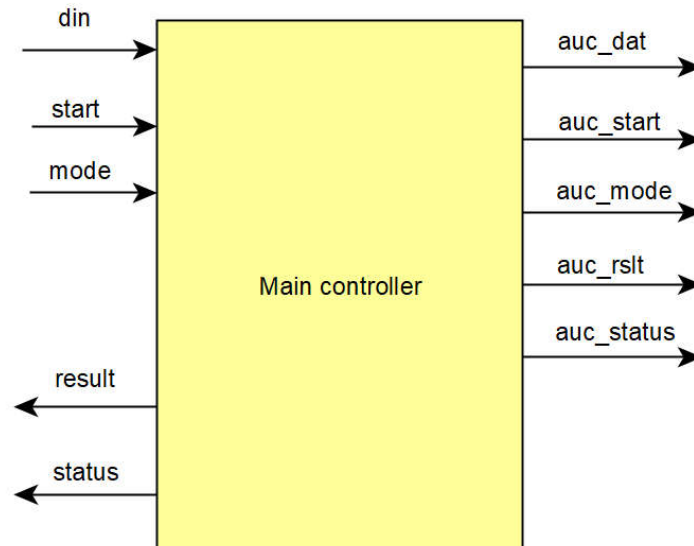


Hình 3-2 Sơ đồ khối tổng quát ECC Core

3.4. Sơ đồ khối chi tiết và nhiệm vụ, chức năng từng khối:

3.4.1. Bộ điều khiển chính (Main controller):

3.4.1.1. Mô hình chi tiết



Hình 3-3 Bộ điều khiển chính

3.4.1.2. Mô tả chân kết nối

Bảng 4 Mô tả chân MC

Chân	Số bit	Loại	Mô tả
start	1	Ngõ vào	MC FSM bắt đầu
mode	3	Ngõ vào	MC FSM chế độ
status	2	Ngõ ra	MC trạng thái
Data in	768	Ngõ vào	ECC Ngõ vào dữ liệu
Data out	256	Ngõ ra	ECC Ngõ ra dữ liệu
AUC start	1	Ngõ ra	AUC bắt đầu
AUC mode	2	Ngõ ra	AUC chế độ
AUC status	2	Ngõ vào	AUC trạng thái
AUC data in	256	Ngõ vào	AUC Ngõ vào dữ liệu
AUC data out	256	Ngõ ra	AUC Ngõ ra dữ liệu

Bảng 5 MC Mode[1:0]

Mode[1:0]	Mô tả (tình chỉnh chế độ)
00	ECDSA Sign
01	ECDHE Gen
1x	ECDHE Comp

Bảng 6 MC Mode[4:2]

Mode[2]	Mô tả (tùy chọn đường cong elliptic)
0	secp256r1(NIST P-256)
1	X25519

Bảng 7 MC status

Status	Mô tả
00	idle
01	computing
10	done
11	error

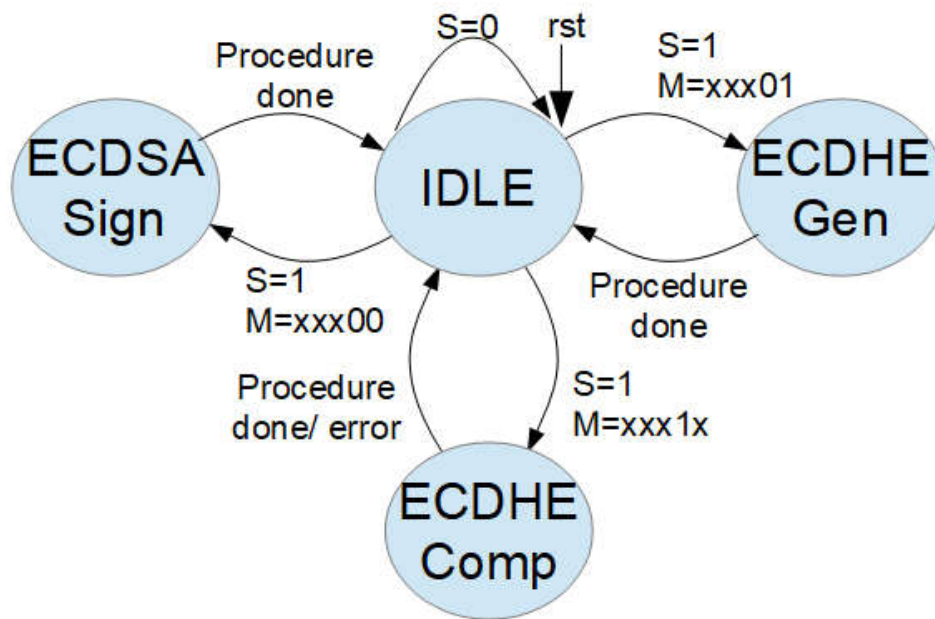
Bảng 8 MC Ngõ vào dữ liệu

Data in	Mô tả	
ECDSA sign	{hashed_message,private_key}	
ECDHE Gen	Không sử dụng	
ECDHE Comp	Weierstrass curves	{ X_coordinate_Q,Y_coordinate_Q, k}
	Montgomery curves	{X_coordinate_Q, k}

Bảng 9 MC Ngõ ra dữ liệu

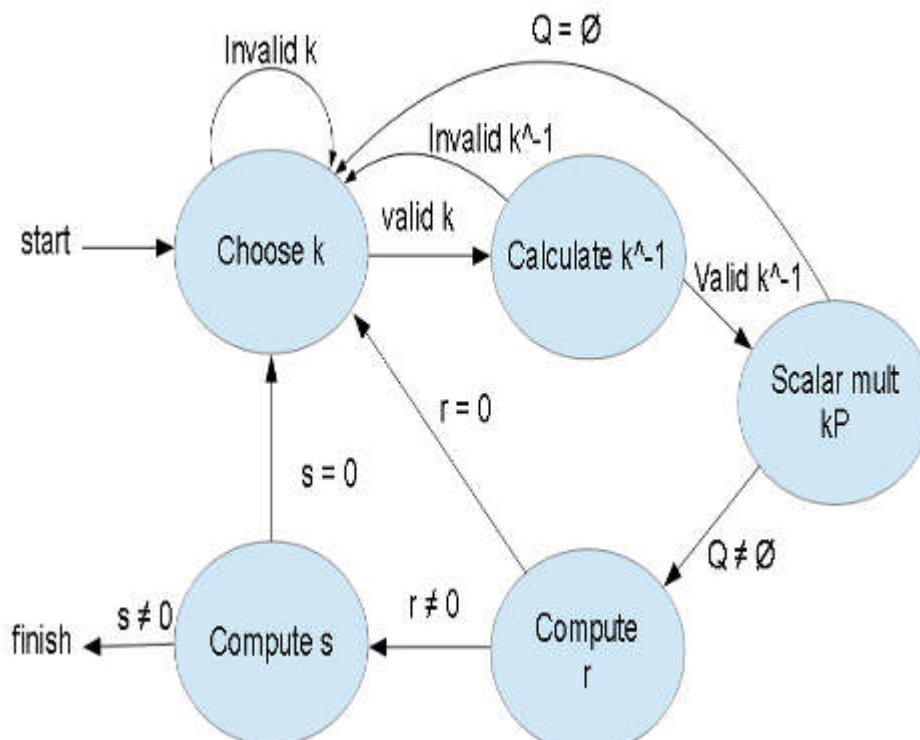
Data out	Mô tả
ECDSA sign	{r,s} (signature)
ECDHE Gen	{X_coordinate_kG,k}
ECDHE Comp	{Premaster_secret}

3.4.1.3. Các máy trạng thái (FSM)



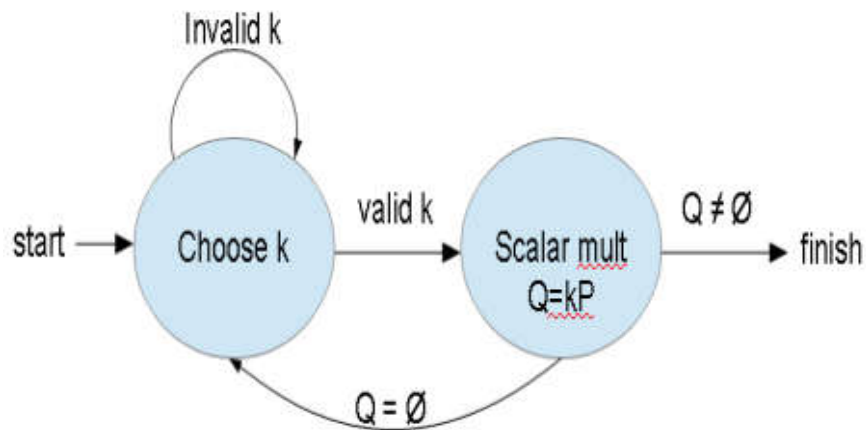
Hình 3-4 Máy trạng thái MC

3.4.1.4. Giải thuật tạo chữ ký ECDSA



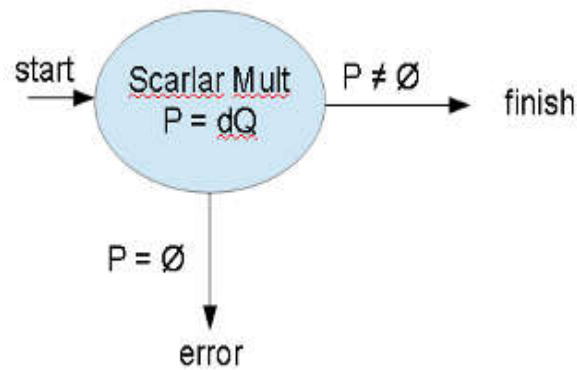
Hình 3-5 Sơ đồ giải thuật ECDSA tại MC

3.4.1.5. Giải thuật tạo chìa khóa công khai ECDHE



Hình 3-6 Sơ đồ giải thuật tạo chìa khóa ECDHE tại MC

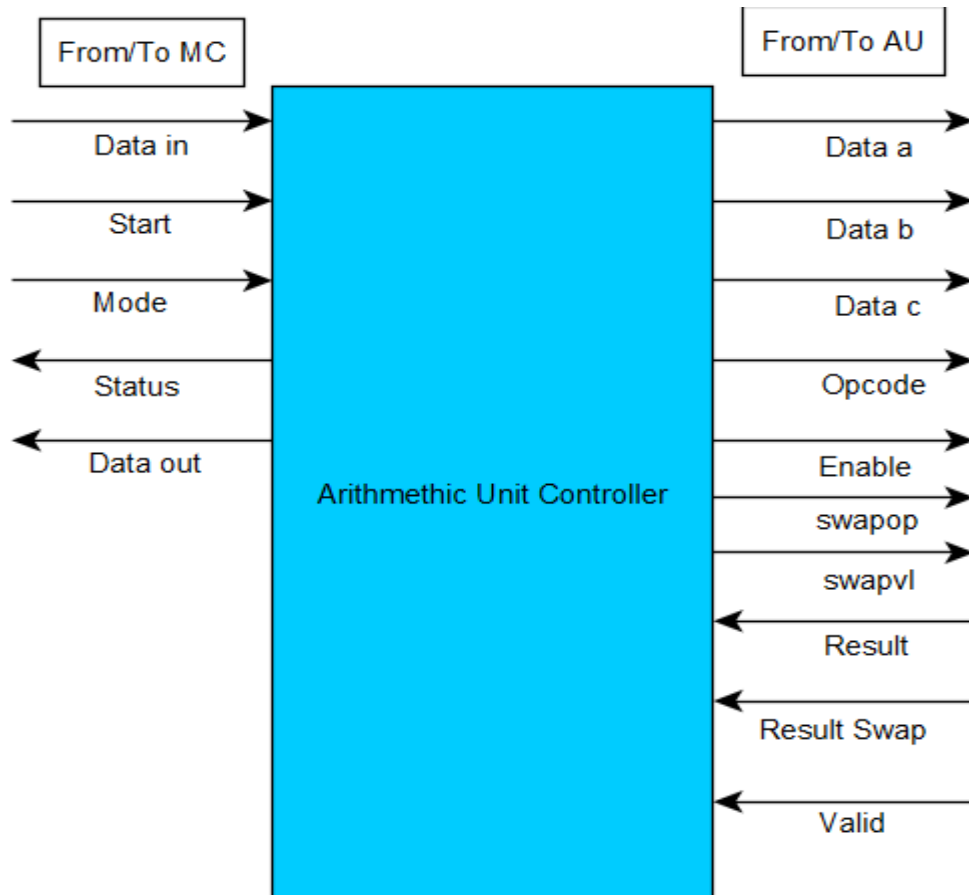
3.4.1.6. Giải thuật tính chìa khóa đối xứng ECDHE



Hình 3-7 Sơ đồ giải thuật tính chìa khóa ECDHE tại MC

3.4.2. Bộ điều khiển số học (Arithmetic Unit Controller)

3.4.2.1. Mô hình chi tiết



Hình 3-8 Mô hình bộ điều khiển số học

3.4.2.2. Mô tả chân kết nối

Bảng 10 Mô tả chân AUC

Chân	Loại	Mô tả
AUC data in	Ngõ vào	Ngõ vào dữ liệu
AUC data out	Ngõ ra	Ngõ ra dữ liệu
AUC start	Ngõ vào	AUC bắt đầu FSM
AUC mode	Ngõ vào	AUC chế độ FSM
AUC status	Ngõ ra	AUC status indicator
Data a	Ngõ ra	Toán tử 1 st AU
Data b	Ngõ ra	Toán tử 2 nd AU
Data c	Ngõ ra	Toán tử 3 rd AU
Results	Ngõ vào	Kết quả từ AU

AU en	Ngõ ra	AU Chân cho phép
AU opcode	Ngõ ra	AUC opcode
AU vld	Ngõ vào	valid kết quả từ AU
AU swapop	Ngõ ra	AU dùng phép hoán đổi
AU swapvl	Ngõ ra	AU giá trị hoán đổi

Bảng 11 AUC mode[2:0]

AUC mode[2:0]	Mô tả
000	Choose k
001	Calculate k^{-1}
010	Compute r
011	Compute s
100	Weierstrass scalar multiplication
101	Montgomery scalar multiplication
110-111	Reserved

Bảng 12 AUC mode[4:3]

AUC mode[3]	Mô tả
0	Secp256r1
1	X25519

Bảng 13 AUC status

AUC status	Mô tả
00	idle
01	computing
10	done
11	error

Bảng 14 AUC Ngõ vào dữ liệu

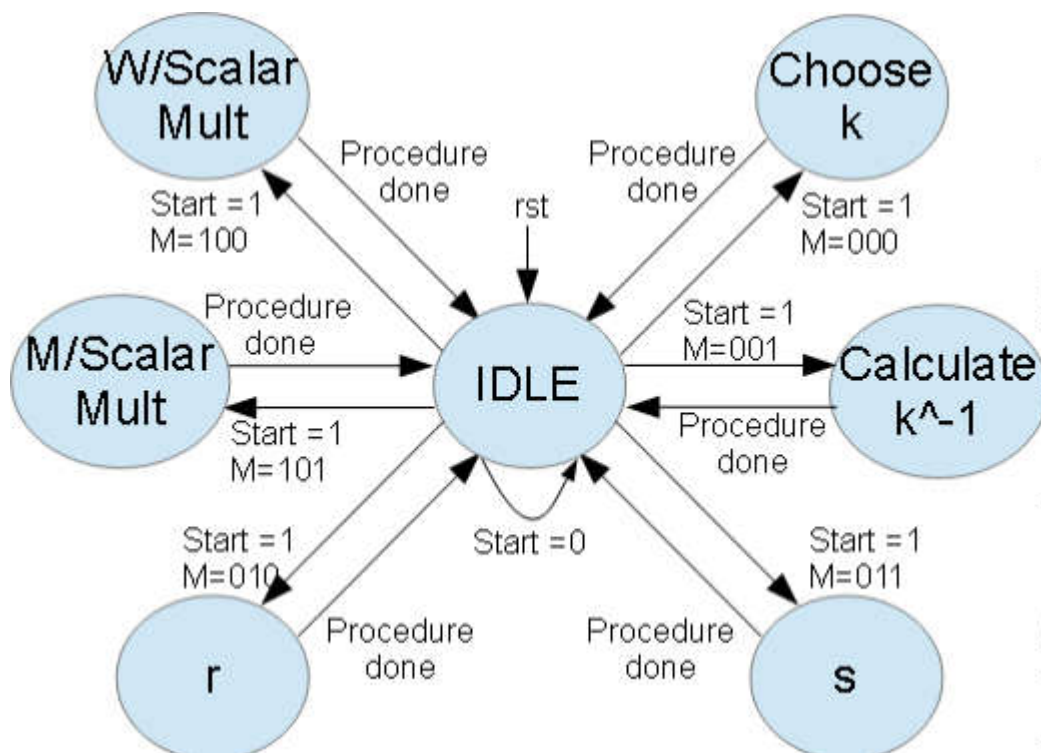
AUC data in	Mô tả
Choose k	Không sử dụng
Calculate k^{-1}	Không sử dụng
Compute r	Không sử dụng
Compute s	{hashed_message_e,private_key_d}
Weierstrass scalar multiplication	ECDHE Comp {X_coordinate_Q,Y_coordinate_Q,k}

	Còn lại	{X_coordinate_Q,Y_coordinate_Q}
Montgomery scalar multiplication	ECDHE Comp	{X_coordinate_Q, k}
	Còn lại	{X_coordinate_Q}

Bảng 15 Ngõ ra dữ liệu

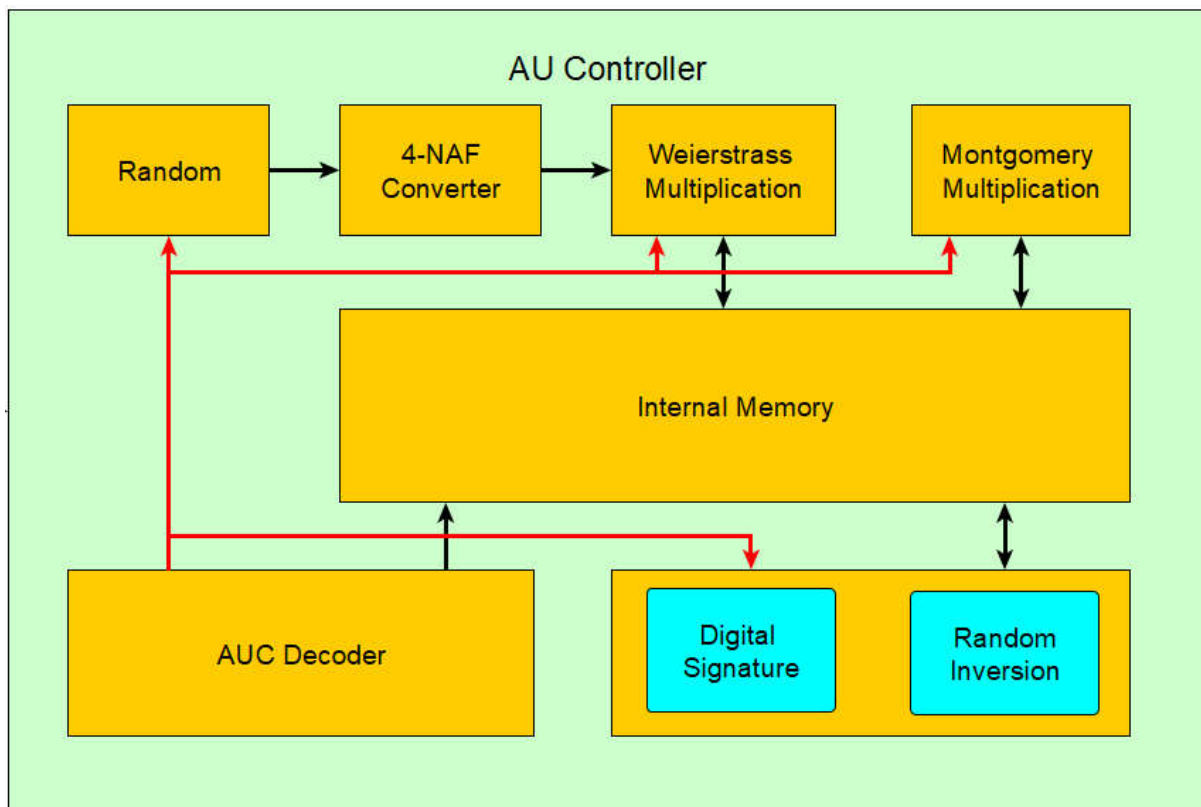
AUC data out	Mô tả
Choose k	{number_k}
Calculate k^{-1}	{k_invert}
Compute r	{r}
Compute s	{s}
Weierstrass scalar multiplication	{X_coordinate_Q}
Montgomery scalar multiplication	{X_coordinate_Q}

3.4.2.3. Máy trạng thái (FSM)



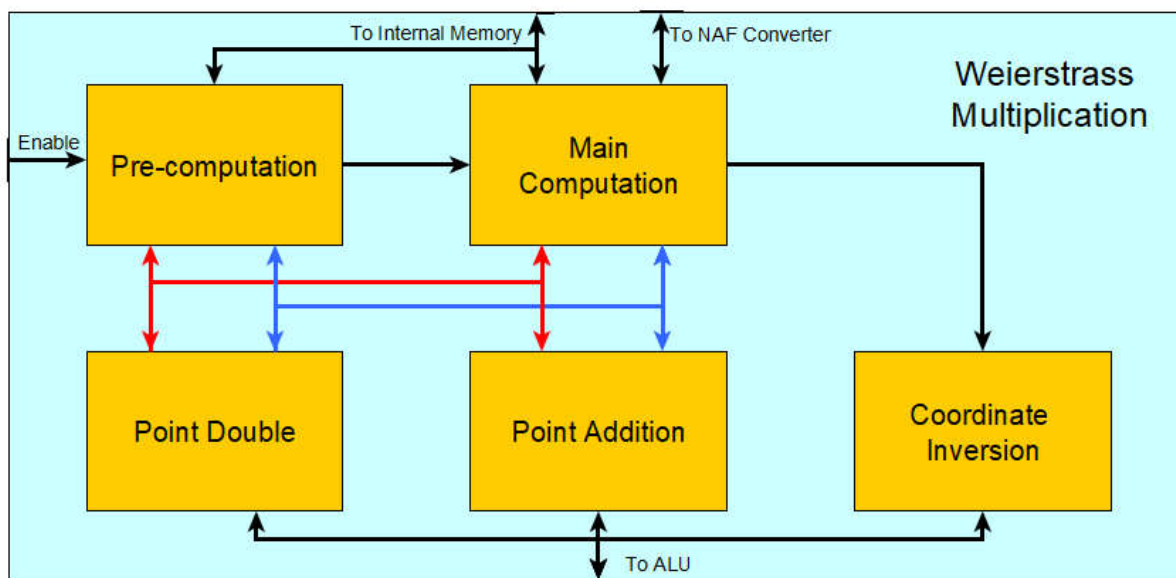
Hình 3-9 Máy trạng thái của AUC

3.4.2.4. Sơ đồ khối bộ AUC



Hình 3-10 Sơ đồ khối bộ điều khiển số học (AUC)

3.4.2.5. Bộ nhân vô hướng cho đường cong Weierstrass



Hình 3-11 Sơ đồ khối bộ nhân vô hướng cho đường cong Weierstrass

Sử dụng phương pháp của số w-NAF (non-adjacent form) là cách tối ưu để tính scalar multiplication. Phương pháp này đòi hỏi $2^{w-2} - 1$ phép cộng (addition) và 1 phép nhân đôi

(double) cho bước chuẩn bị tính toán và $\frac{L}{w+1}$ phép cộng và L phép nhân đôi cho phần tính toán (L là độ dài bit của k) [30]. Số lượng phép nhân đôi không phụ thuộc vào w nhưng số lượng phép cộng phụ thuộc vào w trong phần tính chính. Dựa vào đó, chúng em lập bảng thống kê số phép cộng cần dùng ứng với các giá trị khác nhau.

Bảng 16 Số lượng phép cộng khi w tăng

Độ dài bit của k	w = 2	3	4	5	6
160	53.33	41	35	33.67	51
192	64	49	41.4	39	42.42
224	74.67	57	47.8	44.3	47
256	85.33	65	54.2	49.6	51.57
384	128	97	79.8	71	69.85
512	170.67	129	105.4	92.3	88.14

Nhận xét rằng tổng số phép cộng cần dùng đạt cực tiểu khi kích thước của sổ là 5 hoặc 6, tùy vào độ dài bit của số ngẫu nhiên k. Với đường cong elliptic 256 bit, w = 5 cho tốc độ xử lý tốt nhất và cần 5 điểm tọa độ phải được tính toán trước. Tuy nhiên, w = 4 cũng là một đánh đổi đáng cân nhắc và hợp lý hơn giữa tốc độ xử lý (performance) và diện tích (area) (chỉ cần tính trước 3 điểm). Một lưu ý nữa là điểm trên đường cong elliptic thường được biểu diễn dưới một hệ tọa độ tham chiếu, cụ thể là hệ tọa độ Jacobian, nhằm tối ưu việc tính toán.

Giải thuật tính w-NAF(k)

Input: an integer k

Output: w-NAF(k) = $(u_{l-1} \dots u_1 u_0)$

Operation:

Set $c = k$, $l = 0$

While $c > 0$ do

If c is odd then

Set $u_l = c \bmod 2^w$

If $u_l > 2^{w-1}$ then set $u_l = u_l - 2^w$

Set $c = c - u_l$

Else set $u_l = 0$

Set $c = c/2$, $l = l + 1$

Return w-NAF(k)

Giải thuật của số w-NAF cho đường cong Weierstrass

Input: integer k and w, and point $P = (x, y)$

Output: the point $Q = kP$

```

Precomputation:
// Compute uP for u odd and  $2 < u < 2^{w-1}$ 
    Set  $P_0 = P, T = 2P$ 
    For i from 1 to  $2^{w-1} - 1$  do
        Set  $P_i = P_{i-1} + T$ 
Main computation:
    Compute w-NAF(k) =  $(u_{l-1} \dots u_1 u_0)$ 
    Set  $Q = O$ , where the point of infinity is  $O$ 
    For j from l-1 downto 0 do
        Set  $Q = 2Q$ 
        If  $u_j \neq 0$  then set  $i = \frac{|u_j|-1}{2}$ 
        If  $u_j > 0$  then set  $Q = Q + P_i$ 
        Else set  $Q = Q - P_i$ 
    Return Q

```

3.4.2.6. Giải thuật bậc thang cho đường cong Montgomery

Đầu tiên, giải thuật Montgomery ladder được đề xuất với ý định tăng tốc độ phép nhân vô hướng trên đường cong elliptic dạng Montgomery. Sau đó, giải thuật này đã được Brier và Joye tổng quát hóa trên F_p . Theo đó, đối với đường cong elliptic, tung độ y là không cần thiết trong phép cộng và nhân điểm (công thức tính đã được trình bày ở phần lý thuyết). Nhận xét rằng việc tính toán chỉ sử dụng hoành độ giúp tiết kiệm nhiều phép nhân dẫn tới thuật toán nhanh hơn, đồng thời yêu cầu bộ nhớ cũng ít hơn.

Giải thuật Montgomery ladder

```

Input:  P, d
Output: x_dP
Operation:
    R[0] = P, R[1] = 2P
    For i = l-2 downto 0 do
        R[1-d_i] = R[0] + R[1]
        R[d_i] = 2R[d_i]
    Return R[0]

```

Giải thuật được sử dụng trong luận văn để thực hiện nhân vô hướng cho đường cong elliptic Montgomery (ở đây là đường cong X25519) là giải thuật của S.Turner, viết năm 2016 trong chuẩn IETF RFC 7748 [32]. Giải thuật này đảm bảo thời gian xử lý là bằng nhau cho tất cả giá trị nhập vào, vậy nên khi thay đổi hoặc cải thiện giải thuật cần phải đáp ứng được điều này. Ngoài ra giải thuật chính xác với tất cả mọi trường hợp, được khẳng định bởi thư S.Turner gửi Microsoft và trình bày tại buổi thuyết trình của ông [36]

Giải thuật trong RFC được viết dưới dạng Python Code như sau:

```
x_1 = u
x_2 = 1
z_2 = 0
x_3 = u
z_3 = 1
swap = 0

For t = bits-1 down to 0:
    k_t = (k >> t) & 1
    swap ^= k_t
    // Conditional swap; see text below.
    (x_2, x_3) = cswap(swap, x_2, x_3)
    (z_2, z_3) = cswap(swap, z_2, z_3)
    swap = k_t

A = x_2 + z_2
AA = A^2
B = x_2 - z_2
BB = B^2
E = AA - BB
C = x_3 + z_3
D = x_3 - z_3
DA = D * A
CB = C * B
x_3 = (DA + CB)^2
z_3 = x_1 * (DA - CB)^2
x_2 = AA * BB
z_2 = E * (AA + a24 * E)

// Conditional swap; see text below.
(x_2, x_3) = cswap(swap, x_2, x_3)
(z_2, z_3) = cswap(swap, z_2, z_3)
Return x_2 * (z_2^(p - 2))
```

Qua dạng Python Code của giải thuật, chúng em thực hiện một số thay đổi phù hợp với phân cứng và cho ra giải thuật sau:

```
x_1 = u
x_2 = 1
z_2 = 0
x_3 = u
```

```

z_3 = 1
swap = 0

For t = bits-1 down to 0:
    swap = swap ^ k[t]
    // Conditional swap; see text below.
    (x_2, x_3) = SWAP(swap, x_2, x_3)
    (z_2, z_3) = SWAP(swap, z_2, z_3)
    swap = k[t]

A = x_2 + z_2
AA = A^2
B = x_2 - z_2
BB = B^2
E = AA - BB
C = x_3 + z_3
D = x_3 - z_3
DA = D * A
CB = C * B
x_3 = (DA + CB)^2
z_3 = x_1 * (DA - CB)^2
x_2 = AA * BB
z_2 = E * (AA + a24 * E)

// Conditional swap; see text below.
#(x_2, x_3) = SWAP(0, x_2, x_3)
#(z_2, z_3) = SWAP(0, z_2, z_3)
Return x_2 * (MONTINV(z_2))

```

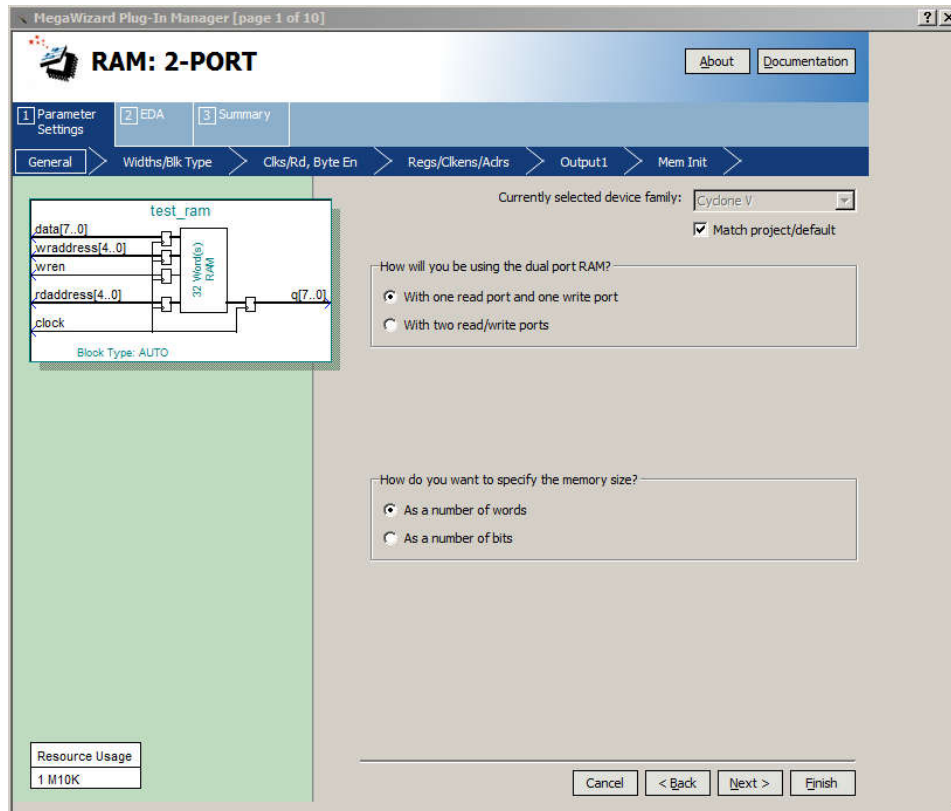
Một số thay đổi của giải thuật phù hợp cho việc tính toán trên phần cứng ở thời điểm hiện tại:

- Bỏ phép dịch k, chuyển sang dùng index.
- Bỏ giá trị hoán đổi ở hai lần hoán đổi cuối cùng, vì 3 bit cuối của k luôn bằng 0 sau khi decode đúng dạng ở RFC 7748 [32].
- Bỏ phép mũ cuối cùng, phép mũ Montgomery có thể được thực hiện trên phần cứng nhưng tốn quá nhiều thời gian, thay vào đó là phép nghịch đảo Montgomery cho ra kết quả tương tự, có thể được chứng minh như sau:

Given $t = a^{-1} \bmod p$ và $t = a^{p-2} \bmod p$
 $a * t \bmod p = 1 \bmod p$

3.4.2.7. Bộ nhớ Random Access Memory (RAM)

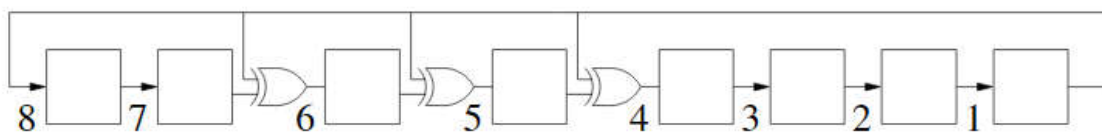
RAM sử dụng trong thiết kế là 1 block RAM M10K trên Cyclone V FPGA 256-bit x 32. Có sử dụng cho phép ghi và đọc dữ liệu cùng địa chỉ cùng 1 clock (giá trị mới). Độ trễ đọc RAM là 3 clock. Chúng em sử dụng sample code của Intel để gọi block ram và pipeline output để đạt độ trễ đọc mong muốn.



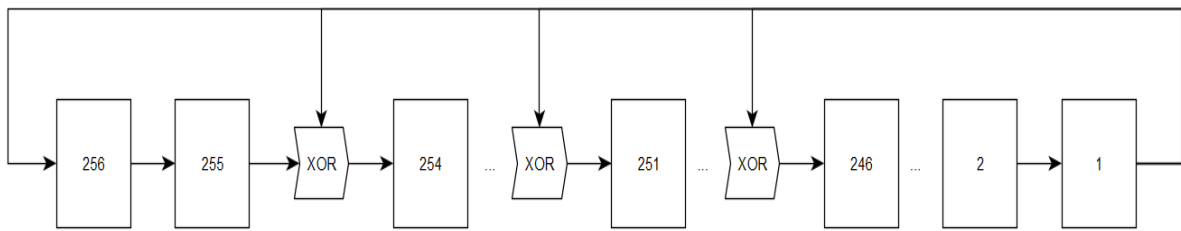
Hình 3-13 IP RAM tinh chỉnh của FPGA dùng M10K

3.4.2.8. Bộ chọn số ngẫu nhiên

Bộ tạo số ngẫu nhiên (RNG) cần một nguồn Entropi đủ tốt để có tính bảo mật cao. Với thiết kế do chưa dùng nguồn Entropi từ bên ngoài, chúng em sử dụng Linear Feedback Shift Register [36] với các feedback ở vị trí 256,254,251,246 để đạt maximum-cycle và dùng một hạt giống để khởi tạo quá trình hoạt động của LFSR. LFSR được thiết kế theo dạng Galois như hình vẽ.



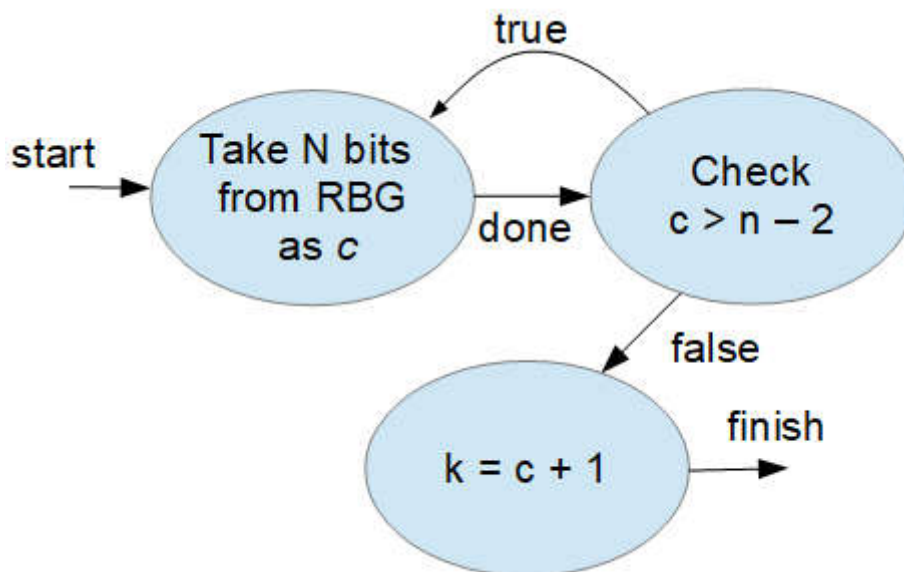
Hình 3-14 8-bit LFSR feedback ở 8, 6, 5, 4



Hình 3-15 256-bit LFSR feedback ở 256, 254, 251, 246

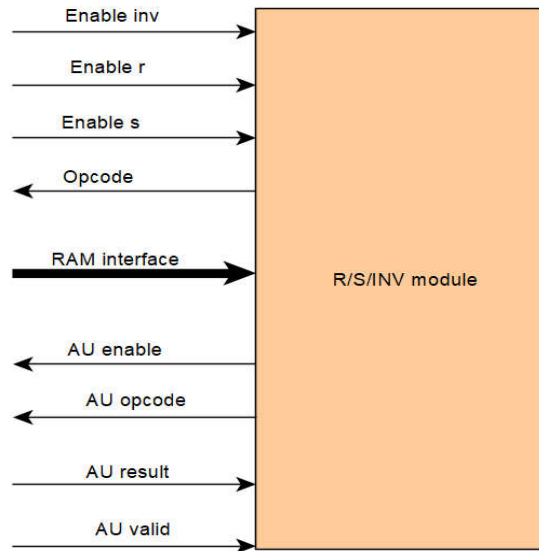
Bộ tạo số ngẫu nhiên thực hiện trong thiết kế có hai chế độ, sử dụng ngõ ra của LFSR hoặc đưa vào một số ngẫu nhiên tùy chọn, với chế độ 2 chủ yếu có các mục đích thử nghiệm và mô phỏng. Để chọn một số ngẫu nhiên phù hợp cho đường cong elliptic cần tuân theo phương pháp kiểm tra tính hợp lệ của chuẩn FIPS 186-4. Có hai phương pháp như sau:

- [3] Appendix B.5.2 Per-Message Secret Number Generation by Testing Candidates.
- Phương pháp thay thế, dùng [3] Appendix B.5.1.



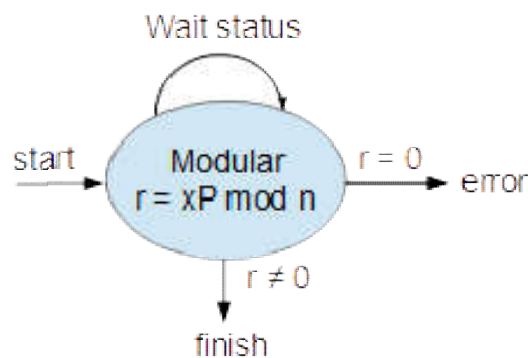
Hình 3-16 Giải thuật chọn k

3.4.2.9. Bộ tính chữ kí số (r, s) và tính nghịch đảo modulo số k

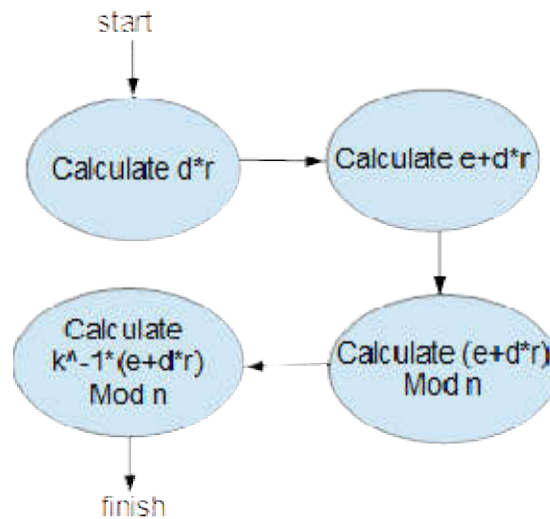


Hình 3-17 Sơ đồ khối bộ tính R / S / INV

Cả ba quá trình tính toán chữ kí số (r, s) và tính nghịch đảo modulo được tích hợp vào một module vì độ phức tạp không lớn và có chung interface với các khối ngoài. Ở đây, quá trình tính nghịch đảo modulo chỉ đơn giản là lấy dữ liệu ra từ RAM, đưa tới bộ AU và kết quả đưa lại vào RAM, giải thuật chi tiết để tính nghịch đảo modulo sẽ trình bày ở phần sau của luận văn này. Quy trình tính chữ kí số (r, s) như sau:



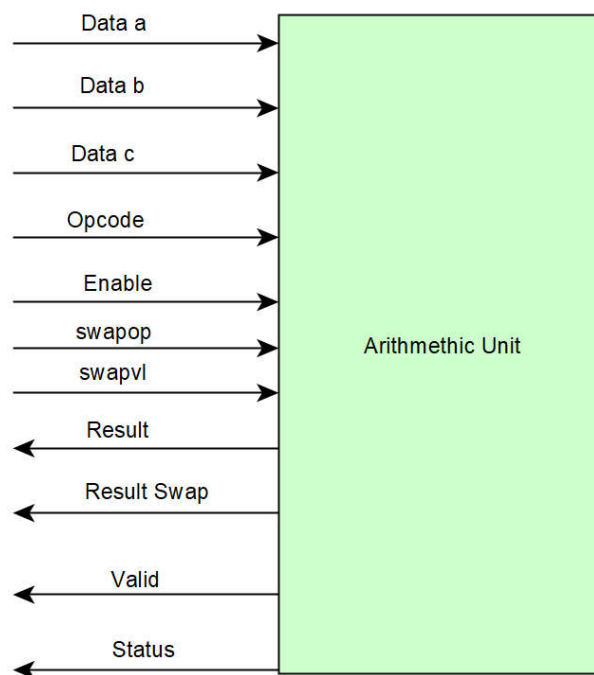
Hình 3-18 Tính toán r



Hình 3-19 Tính toán s

3.4.3. Bộ tính toán số học (Arithmetic Unit)

3.4.3.1. Mô hình chi tiết



Hình 3-20 Mô hình bộ tính toán số học

3.4.3.2. Mô tả chân kết nối

Bảng 17 Mô tả chân AU

Chân	Số Bit	Loại	Mô tả
Data a	256	Ngõ vào	toán tử 1 st
Data b	256	Ngõ vào	toán tử 2 nd

Data c	1	Ngõ vào	Nhớ (cho phép cộng)
Results	256	Ngõ ra	Kết quả tính toán từ AU
AU en	1	Ngõ vào	AU Chân cho phép
AU opcode	4	Ngõ vào	AUopcode
AU status	2	Ngõ ra	AU trạng thái
AU Swapop	1	Ngõ vào	Chọn phép hoán đổi (ưu tiên)
AU Swapvl	1	Ngõ vào	Giá trị hoán đổi (0 : không hoán đổi; 1 : hoán đổi)
AU Results Swap	256	Ngõ ra	Kết quả 2 của phép biến đổi

Bảng 18 AU Opcode [1:0]

AU Opcode [1:0]	Mô tả
00	Modular Addition
01	Montgomery Multiplication
10	Montgomery Inversion
11	Montgomery Exponent (OFF)

Bảng 19 AU Opcode [2]

AU Opcode [2]	Mô tả
0	Secp256r1 (NIST P-256)
1	X25519

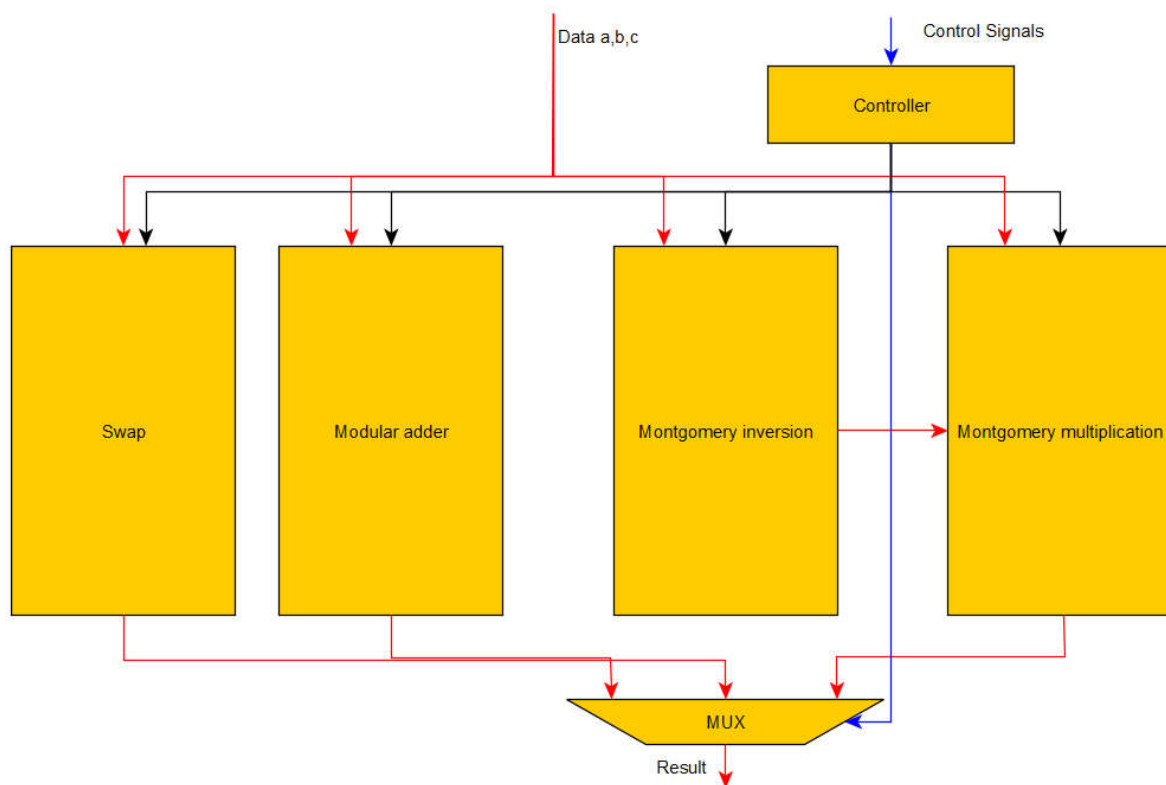
Bảng 20 AU Opcode [3]

AU Opcode [3]	Mô tả
0	Modulo P
1	Modulo N

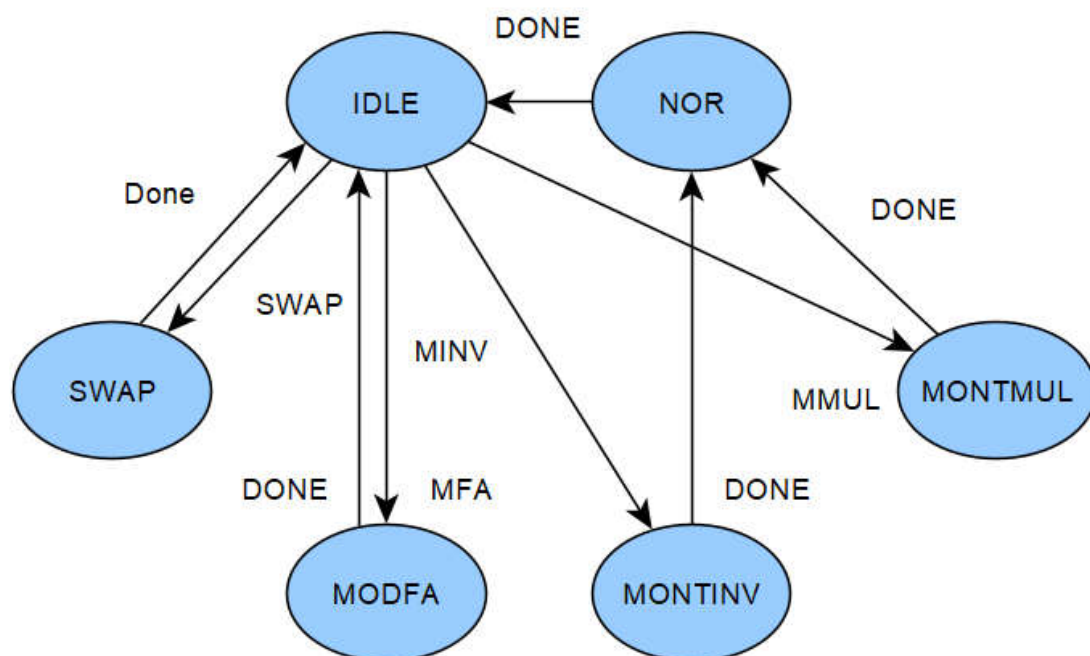
Bảng 21 AU status

AU status [1:0]	Mô tả
00	idle
01	computing
10	done

3.4.3.3. Sơ đồ khối tổng quát



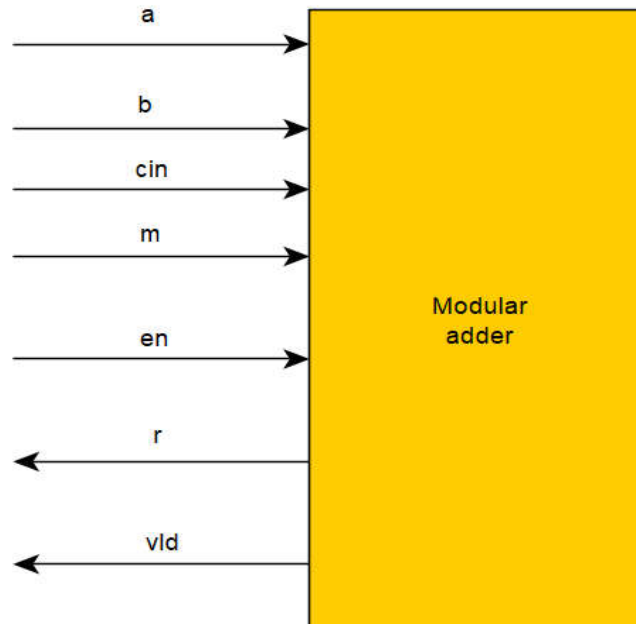
Hình 3-21 Sơ đồ khối AU



Hình 3-22 Máy trạng thái của AU

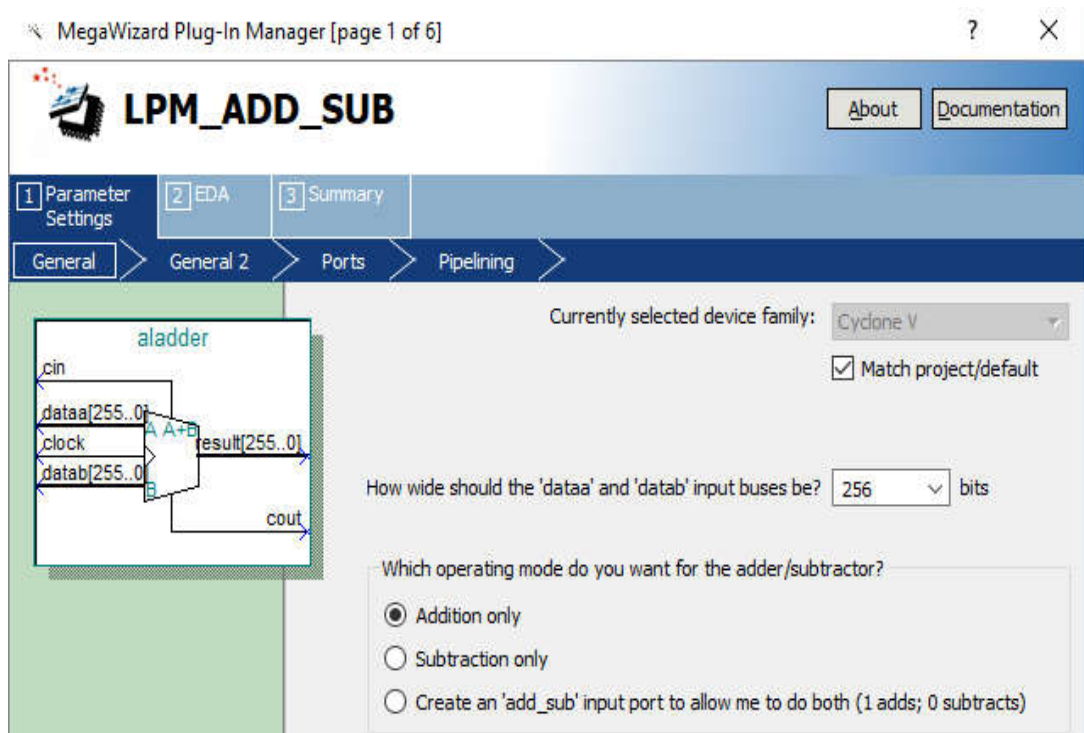
Máy trạng thái AU cho thấy kết quả từ hai bộ nghịch đảo Montgomery và bộ nhân Montgomery cần được thay đổi lại (NOR) bằng một phép nhân Montgomery nữa trước khi trả về kết quả chính xác.

3.4.3.4. Phép cộng Modulo (Modular adder)



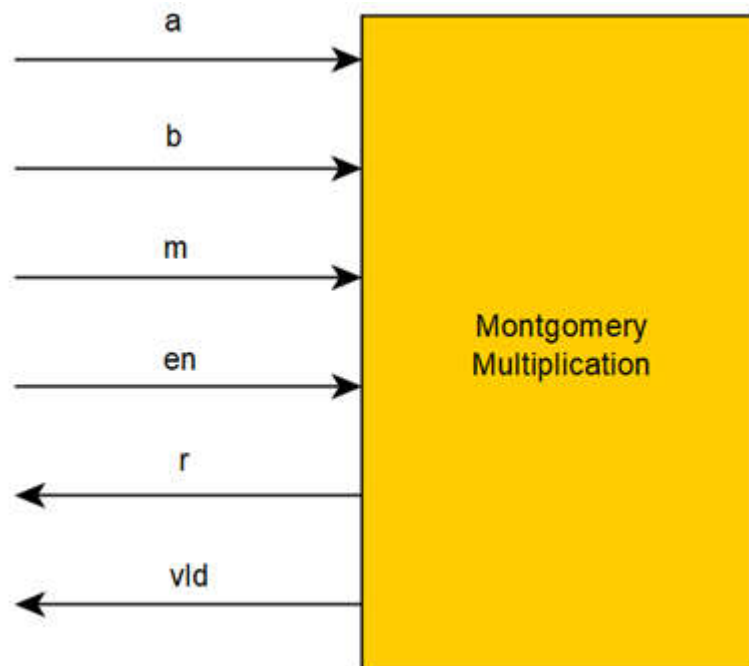
Hình 3-23 Bộ cộng modulo

Bộ cộng modulo dùng bộ cộng carry load adder và Intel's Arithmetic IP LPM dùng thực hiện các phép tính cộng và trừ. Bộ cộng Intel's Arithmetic IP LPM, cụ thể là LPM_ADD_SUB dễ sử dụng và được dùng để thay thế CLA trong các vị trí của bộ cộng trong thiết kế chính. Đây cũng là nơi cần thay đổi nhiều để đạt được thời gian giới hạn mong muốn [35]



Hình 3-24 Altera IP LPM cho ADD/SUB

3.4.3.5. Phép nhân Modulo



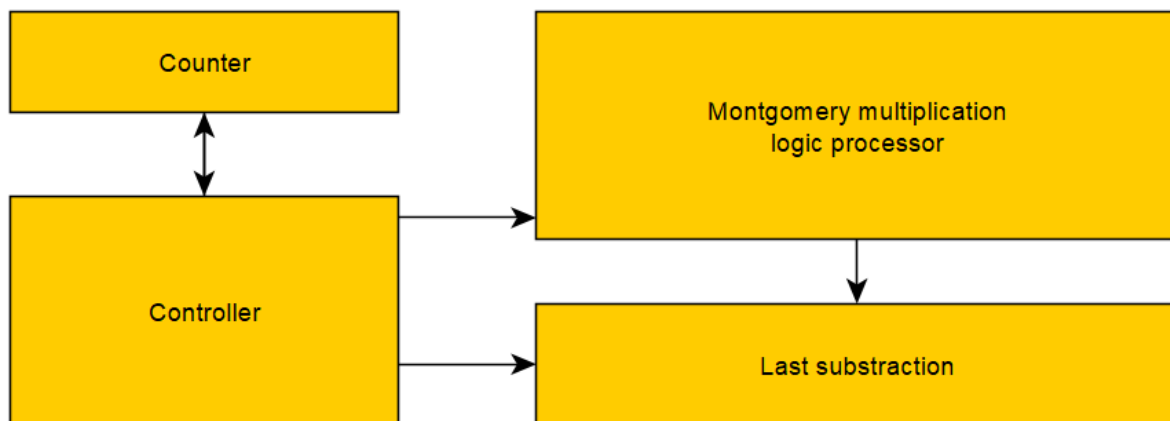
Hình 3-25 Bộ nhân Montgomery

Phép nhân Montgomery sử dụng bit nhỏ nhất (LSB) của kết quả để cộng thay vì trừ và shift phải cho mỗi lần tính. Hàm tính tích Montgomery được nêu bên dưới với các số hạng (dạng nhị phân):

Số chia lấy dư M có thể là một số nguyên k bit (i.e $0 < M < 2^k$), và $A, B < M$, khi đó độ dài bit của bộ nhân, n , là $k+2$.

Ngõ vào	a	Thừa số $A = \sum_{i=0}^{k-1} a_i 2^i$
	b	Thừa số $B = \sum_{i=0}^{k-1} b_i 2^i$
	m	Số modulo $M = \sum_{i=0}^{k-1} m_i 2^i$
Ngõ ra	r	$ABR^{-1} \bmod(M)$
Tính toán	<ol style="list-style-type: none"> 1. $S = 0$ 2. For $i = 0$ to $n-1$ { 3. $S = S + a[i]*B$; 4. $S = (S + S[0]*M)/2$ } 5. If $S > M$ 6. $S = S - M$; 7. Return S; 	

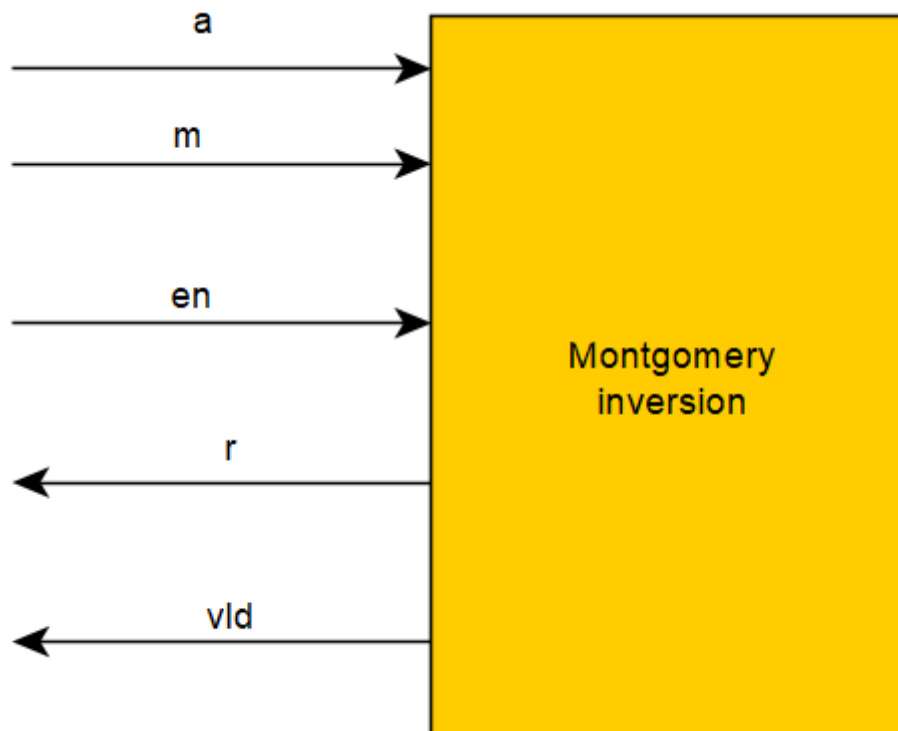
Cấu trúc của bộ nhân Montgomery để thực hiện giải thuật trên như sau:



Hình 3-26 Cấu trúc bộ nhân Montgomery

Bộ logic processor cho cấu trúc Montgomery bao gồm phần giải thuật trong vòng lặp, sử dụng mux và carry load adder, thanh ghi dịch để tính các giá trị của S qua từng vòng lặp.

3.4.3.6. Phép lấy nghịch đảo Modulo



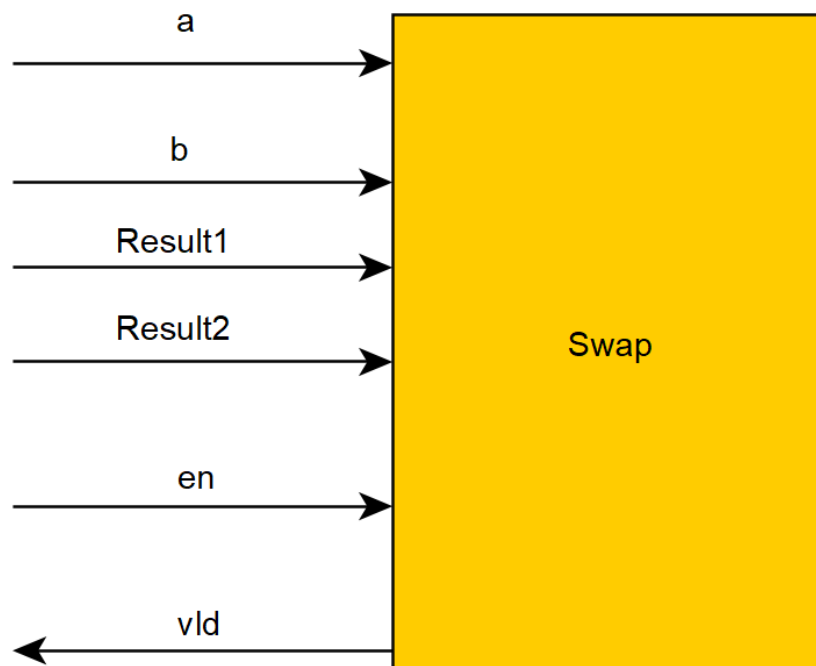
Hình 3-27 Bộ nghịch đảo Montgomery

Bộ nghịch đảo Montgomery thực hiện bằng giải thuật nghịch đảo Montgomery và có 2 phần, chia làm hai module tương ứng phase 1 và phase 2. Kết quả của bộ nghịch đảo Montgomery ở dạng $a^{-1} \cdot 2^n \text{ mod}(p)$, để thu được $a^{-1} \text{ mod}(p)$, chúng em thực hiện phép nhân Montgomery cho kết quả và 1 mod(p).

Ngõ vào	a	Giá trị cần tính nghịch đảo a
	p	Số modulo p
Ngõ ra	r	Tích nghịch đảo modulo $r = a^{-1} 2^n \text{ mod}(p)$
Tính toán	Phase I: <ol style="list-style-type: none"> 1. $u = p, v = a, r = 0, s = 1$ 2. $k = 0$ 3. While ($v > 0$) { 4. If u is even then $u = u/2, s = 2s$ 5. Else if v is even then $v = v/2, r = 2r$ 6. Else $u > v$ then $u = (u - v)/2, r = r + s, s = 2s$ 7. Else $v > u$ then $v = (v - u)/2, r = 2r, s = r + s$ 8. $k = k + 1$ } 	

	9. If $r \geq p$ then $r = r - p$ 10. Return $r = p - r$ and k Phase II: 1. For $i = 1$ to $k - n$ { 2. If r is even then $r = r/2$ 3. Else then $r = (r + p)/2$ } 4. Return $x = r$
--	---

3.4.3.7. Phép hoán đổi (swap)



Hình 3-28 Bộ hoán đổi

Phép hoán đổi được đề ra cho giải thuật bậc thang Montgomery từ RFC 7748 [32], giải thuật trong RFC 7748 đòi hỏi giữ thời gian xử lý là hằng số và dùng các phép XOR cho vi xử lý. Với phần cứng chỉ cần dùng các thanh ghi và các cổng mux để hoán đổi.

4. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM

Yêu cầu đặt ra cho phần mềm: Phần mềm thể hiện được các giải thuật sẽ được thực hiện trên phần cứng, hỗ trợ trong việc tính toán và sửa lỗi phần cứng

Để phần mềm đơn giản dễ sử dụng, chúng em sử dụng ngôn ngữ Python trên nền tảng Google Colab.

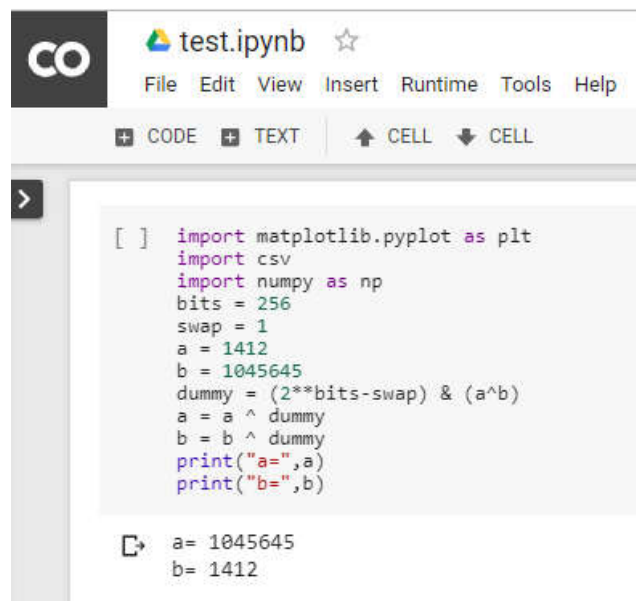


Hình 4-1 Ngôn ngữ Python



Hình 4-2 Google Colab

Các giải thuật sử dụng chia làm các đoạn code nhỏ, sử dụng để kiểm tra các bước tính toán riêng lẻ và các phương trình toán học. Google Colab là một nền tảng mở cho phép sử dụng máy chủ Google để chạy các đoạn mã Python trên một tập tin ghi chú (ipynb).

A screenshot of the Google Colab web interface. At the top, there's a header with the Google Colab logo and the text "test.ipynb" with a star icon. Below this is a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Underneath the menu bar are tabs for "+ CODE", "+ TEXT", and buttons for "↑ CELL" and "↓ CELL". The main area shows a code cell with the following Python code:

```
[ ] import matplotlib.pyplot as plt
import csv
import numpy as np
bits = 256
swap = 1
a = 1412
b = 1045645
dummy = (2**bits-swap) & (a^b)
a = a ^ dummy
b = b ^ dummy
print("a=",a)
print("b=",b)
```

Below the code cell, the output is displayed: "a= 1045645" and "b= 1412".

Hình 4-3 Ví dụ về sử dụng Google Colab và một đoạn mã Python đơn giản

4.1. Các giải thuật cho đường cong Montgomery

Dựa trên RFC 7748 [32] và thư viện giải thuật HACL [33], chúng em sử dụng các đoạn mã nhỏ cho các chức năng yêu cầu trên RFC 7748 bao gồm 4 chức năng:

Giải mã tọa độ điểm vào U:

- Tọa độ điểm vào U được truyền đi dưới dạng Little-Endian, cần được bỏ MSB của byte cuối cùng.
- Chương trình gồm 2 đoạn thực hiện 2 chức năng decode Little-Endian sang dạng integer và mask đi MSB thứ 255.

```
def decodeLittleEndian(b, bits):
    return sum([b[i] << 8*i for i in range((bits+7)/8)])

def decodeUCoordinate(u, bits):
    u_list = [ord(b) for b in u]
    # Ignore any unused bits.
    if bits % 8:
        u_list[-1] &= (1<<(bits%8))-1
    return decodeLittleEndian(u_list, bits)
```

Giải mã số vô hướng K:

- Số vô hướng K cũng truyền dưới dạng Little-Endian, cần bỏ MSB của byte cuối cùng. Set MSB thứ 2 của byte cuối cùng. Bỏ 3 LSB của byte đầu tiên và đổi thành dạng integer.

```
def decodeScalar25519(k):
    k_list = [ord(b) for b in k]
    k_list[0] &= 248
    k_list[31] &= 127
    k_list[31] |= 64
    return decodeLittleEndian(k_list, 255)
```

Thực hiện phép tích vô hướng trên đường tròn Montgomery với giải thuật bậc thang Montgomery:

- Giải thuật bậc thang cung cấp trong RFC 7748 cho hai đường cong elliptic X448 và X25519 là giải thuật có thể sử dụng cho mọi trường hợp (không có trường hợp đặc biệt) và thời gian xử lý của giải thuật là hằng số (tránh bị tấn công phân tích thời gian xử lý máy chủ)

```
x_1 = u
x_2 = 1
```



```

z_2 = 0
x_3 = u
z_3 = 1
swap = 0

For t = bits-1 down to 0:
    k_t = (k >> t) & 1
    swap ^= k_t
    // Conditional swap; see text below.
    (x_2, x_3) = cswap(swap, x_2, x_3)
    (z_2, z_3) = cswap(swap, z_2, z_3)
    swap = k_t

A = x_2 + z_2
AA = A^2
B = x_2 - z_2
BB = B^2
E = AA - BB
C = x_3 + z_3
D = x_3 - z_3
DA = D * A
CB = C * B
x_3 = (DA + CB)^2
z_3 = x_1 * (DA - CB)^2
x_2 = AA * BB
z_2 = E * (AA + a24 * E)

// Conditional swap; see text below.
(x_2, x_3) = cswap(swap, x_2, x_3)
(z_2, z_3) = cswap(swap, z_2, z_3)
Return x_2 * (z_2^(p - 2))
cswap(swap, x_2, x_3):
    dummy = mask(swap) AND (x_2 XOR x_3)
    x_2 = x_2 XOR dummy
    x_3 = x_3 XOR dummy
    Return (x_2, x_3)

```

Mã hóa kết quả của phép tích vô hướng

- Kết quả tích vô hướng cũng được dịch sang dạng Little-Endian để truyền đi.

```

def encodeUCoordinate(u, bits):
    u = u % p

```

```
return ''.join([chr((u >> 8*i) & 0xff)
                for i in range((bits+7)/8)])
```

Ngoài ra còn có các tập tin Python dùng để nhập liệu, với các con số trong ví dụ cần kiểm tra đề ra bởi RFC 7748 [32]:

- Dưới đây là một ví dụ của số vô hướng đầu tiên cần kiểm tra tại RFC 7748:

```
s1bI= 0xa546e36bf0527c9d3b16154b82465edd62144c0ac1fc5a18506a2244ba449ac4
s1b = bytes([0xa5, 0x46, 0xe3, 0x6b, 0xf0, 0x52, 0x7c, 0x9d, 0x3b, 0x16,
0x15,0x4b, 0x82, 0x46, 0x5e, 0xdd, 0x62, 0x14, 0x4c, 0x0a, 0xc1, 0xfc,
0x5a, 0x18, 0x50, 0x6a, 0x22, 0x44, 0xba, 0x44, 0x9a, 0xc4])
s1 = 0x449a44ba44226a50185afcc10a4c1462dd5e46824b15163b9d7c52f06be346a0
u1bI= 0xe6db6867583030db3594c1a424b15f7c726624ec26b3353b10a903a6d0ab1c4c
u1b = bytes([0xe6, 0xdb, 0x68, 0x67, 0x58, 0x30, 0x30, 0xdb, 0x35, 0x94,
0xc1,0xa4, 0x24, 0xb1, 0x5f, 0x7c, 0x72, 0x66, 0x24, 0xec, 0x26, 0xb3,
0x35, 0x3b, 0x10, 0xa9, 0x03, 0xa6, 0xd0, 0xab, 0x1c, 0x4c])
u1 = 0x4c1cabd0a603a9103b35b326ec2466727c5fb124a4c19435db3030586768dbe6
r1bI= 0xc3da55379de9c6908e94ea4df28d084f32eccf03491c71f754b4075577a28552
```

4.2. Các giải thuật cho đường cong Weierstrass

Với đường cong Weierstrass, chỉ sử dụng một số phép tính để kiểm tra các giá trị trung gian trong giải thuật, đơn giản hóa quá trình sửa lỗi và kiểm tra giải thuật.

```
P256 =
115792089210356248762697446949407573530086143415290314195533631308867097853
951
p = P256
x2 = 0x6e319884fc076ff55690ffa6ee558de6e15b418099ec7f63004ceb895710c6b7
y2 = 0x17f9b03a0c19672bc220e1f7be5c83801e83b1b56dadd75afd36d96800b520db
z2 = 0xebd3a29702dd99d22206a04bfe8ecff34939f5d69d9dbb599da37c91c4017fc09

x1 = 0x18df5d56d2c7bd61605973520491fee8a42b1971ecdebd22d50710aff1d0dc02
y1 = 0xed0a0366dff582ea02c9fec790f1d4ed9d2481e2a8cde32d8d15b76e70a4ca8
z1 = 0xb496b743ceaa78e7140aae52202ebc6c18ebe5f0463003def3adab398a3ad3d8

s1 = (y1*pow(z2,3,p)) %p
s2 = (y2*pow(z1,3,p)) %p
u1 = (x1*pow(z2,2,p)) %p
u2 = (x2*pow(z1,2,p)) %p
```

```

h = (u1 - u2) %p
r = (s1-s2) %p
g = pow(h,3,p)
v = (u1*pow(h,2,p)) %p
x3 = (pow(r,2,p) - 2*v + g) %p
y3 = (r*(v-x3) - s1*g) %p
z3 = z1*z2*h %p

e= 0xA41A41A12A799548211C410C65D8133AFDE34D28BDD542E4B680CF2899C8A8C4
d= 0xC477F9F65C22CCE20657FAA5B2D1D8122336F851A508A1ED04E479C34985BF96
x = 0x2B42F576D07F4165FF65D1F3B1500F81E44C316F1F0B3EF57325B69ACA46104F
n =
115792089210356248762697446949407573529996955224135760342422259061068512044
369
k = 0x7A1A7E52797FC8CAA435D2A4DACE39158504BF204FBE19F14DBB427FAEE50AE
signr = x %n
signs = (pow(k, n-2, n)*(e+d*signr)) %n
t1 = d*signr %n
t2 = (e+d*signr) %n
t3 = pow(k, n-2, n)

```

5. KẾT QUẢ THỰC HIỆN

5.1. Cách thức đo đạc, thử nghiệm

Thiết kế phần cứng sau khi hoàn tất được thử nghiệm bằng các chương trình mô phỏng phần cứng. Nhóm chúng em sử dụng hai chương trình mô phỏng và một chương trình phân tích dạng sóng (ModelSim, NCVerilog và Simvision) để thử nghiệm thiết kế phần cứng cần thiết kế một Testbench chạy các giá trị mô phỏng cũng như kiểm tra các giá trị đầu ra của thiết kế.

Để đo đạc các thông số về tài nguyên và phân tích thời gian của thiết kế, nhóm chúng em sử dụng Intel Quartus 18.1 Lite và dùng thông số của FPGA 5CSXFC6D6F31C6 dòng Cyclone V của Intel, phiên bản nhỏ hơn của FPGA 5CSXFC6D6F31C6N trên DE10-Standard kit.

Hai thiết kế được đo đạc thử nghiệm là NO_CLA_PL và 2_CLA_PL. NO_CLA_PL là thiết kế không có pipeline cho các bộ cộng sử dụng trong thiết kế và 2_CLA_PL là thiết kế có pipeline 2 tầng cho các bộ cộng dùng trong thiết kế (trừ các bộ đếm).

5.2. Số liệu đo đạc

Kết quả mô phỏng cho thấy thiết kế chạy chính xác các trường hợp đề ra trong các thông số đo kiểm của các chuẩn về ECDHE, ECDSA, đường cong Montgomery và đường cong Weierstrass. Dưới đây là kết quả mô phỏng bằng nverilog cho thiết kế NO_CLA_PL.

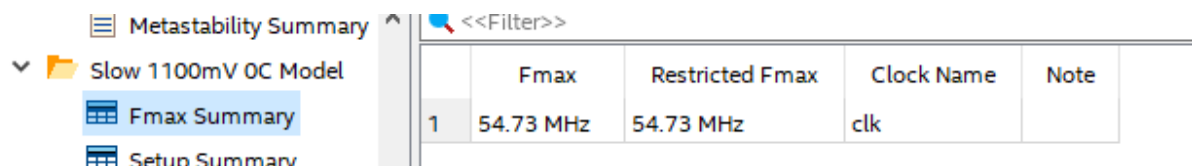
[illegible]

Kết quả mô phỏng cho thiết kế 2_CLA_PL và các hình vẽ dạng sóng, bộ nhớ được ghi ở phần phụ lục.

Các kết quả tài nguyên và thời gian giới hạn (timing) của thiết kế sau khi tổng hợp trên phần mềm Quartus:

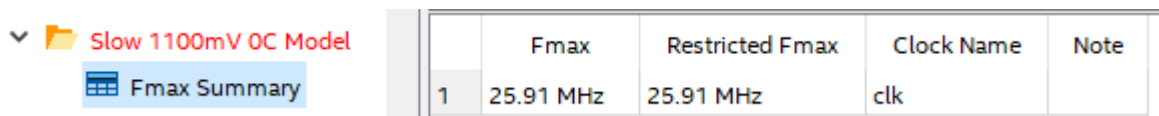
Bảng 22 Kết quả timing và tài nguyên của thiết kế

Thiết kế	ALM	Block Memory Bit	Timing	Trường hợp	Latency (clocks) Worst-case	Latency (ms) Worst-case
NO_CLA_PL	10587	8192	25.91 MHz	Weierstrass	2294835	88
				Motgomery	1359360	53
2_CLA_PL	10534	8192	54.73 MHz	Weierstrass	11287536	205
				Montgomery	6597392	120



Metastability Summary				
Slow 1100mV OC Model				
Fmax Summary				
Setup Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	54.73 MHz	54.73 MHz	clk	

Hình 5-1 Kết quả timing của model 2_CLA_PL



Slow 1100mV OC Model				
Fmax Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	25.91 MHz	25.91 MHz	clk	

Hình 5-2 Kết quả timing của model NO_CLA_PL

The screenshot shows the Quartus II interface with the following components:

- Project Navigator:** Displays the design hierarchy starting with 'Cyclone V: 5CSXFC6D6F31C6', followed by 'ecc_top', 'ecc_core:iecc_core', 'aluwrap:aluwrap', 'cswap:cswap', 'fflop:fflop1', 'modfa:modfa', 'montinv:montinv', 'montprowrap:montprowrap', and 'fflop:iff1'.
- Tasks:** Shows the compilation tasks: Compile Design, Analysis & Synthesis, Fitter (Place & Route), Assembler (Generate program), Timing Analysis, and EDA Netlist Writer. All tasks are marked as successful.
- Flow Summary:** Displays the compilation results for 'ecc_top'.

Flow Summary	
Flow Status	Successful - Sat May 25 03:12:50 2019
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	ecc_top
Top-level Entity Name	ecc_top
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	10,534 / 41,910 (25 %)
Total registers	19915
Total pins	11 / 499 (2 %)
Total virtual pins	0
Total block memory bits	8,192 / 5,662,720 (< 1 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

Hình 5-3 Kết quả tổng hợp của model 2_CLA_PL

The screenshot shows the Quartus II interface with the following components:

- Project Navigator:** Displays the design hierarchy starting with 'Cyclone V: 5CSXFC6D6F31C6', followed by 'ecc_top', 'ecc_core:iecc_core', 'aluwrap:aluwrap', 'cswap:cswap', 'fflop:fflop1', 'modfa:modfa', 'montinv:montinv', 'montprowrap:montprowrap', and 'montpro:imontpro'.
- Tasks:** Shows the compilation tasks: Compile Design, Analysis & Synthesis, Fitter (Place & Route), Assembler (Generate program), Timing Analysis, and EDA Netlist Writer. All tasks are marked as successful.
- Flow Summary:** Displays the compilation results for 'ecc_top'.

Flow Summary	
Flow Status	Successful - Sat May 25 11:36:29 2019
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	ecc_top
Top-level Entity Name	ecc_top
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	10,587 / 41,910 (25 %)
Total registers	14976
Total pins	11 / 499 (2 %)
Total virtual pins	0
Total block memory bits	8,192 / 5,662,720 (< 1 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

Hình 5-4 Kết quả tổng hợp của model NO_CLA_PL

5.3. Giải thích và phân tích về kết quả thu được

Kết quả mô phỏng cho thấy thiết kế chạy chính xác các trường hợp mã hóa được đề ra để thử nghiệm trong các chuẩn được nhắc đến.

Kết quả tổng hợp cho thấy thiết kế sử dụng nguồn tài nguyên nhỏ, đúng với yêu cầu đặt ra. Độ trễ của thiết kế vẫn còn lớn và không đạt được mức tốc độ cao (300 MHz) để sử dụng trong các ứng dụng mạng máy chủ có đòi hỏi cao về tốc độ.

6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. Kết luận

Sau toàn bộ quá trình thực hiện và thiết kế bộ mã hóa đường cong elliptic 256 bit, nhóm chúng em rút ra được nhiều kinh nghiệm, trong đó có những kinh nghiệm về thời gian giới hạn và tài nguyên của một thiết kế, cũng như kinh nghiệm trong việc tối ưu hóa các giải thuật từ phần mềm và một số kinh nghiệm về các giải thuật mã hóa, có thể kể đến các điểm chính sau:

- Cần phải lưu ý về thời gian giới hạn khi thiết kế bộ cộng trừ, trong mã hóa, với yêu cầu về số bit lớn đòi hỏi các phép tính đơn giản cũng phải được pipeline ra nhiều tầng để không tạo ra đường dẫn giới hạn (critical path) lớn.
- Với một module phần cứng, nên pipeline tất cả ngõ vào và ngõ ra (từ 1 cho đến 3 tầng), tài nguyên register trong một thiết kế FPGA thường không hạn hẹp so với các khối RAM và nên được sử dụng tối đa. Qua đó sẽ tăng tốc độ của thiết kế cũng như cho phép chương trình tổng hợp dễ thực hiện sắp xếp và đi dây cho các module, tránh bị lỗi thời gian.
- Với các thiết kế đổi từ giải thuật phần mềm sang phần cứng nên thực hiện song song hóa các phép tính toán nhằm tối ưu sức mạnh xử lý song song của phần cứng so với vi xử lý, cũng như song song hóa các bus data để đơn giản hóa.
- Khi thiết kế cho một FPGA cụ thể, nên sử dụng cái IP được cung cấp bởi nhà sản xuất và thay đổi các thông số nếu cần thiết, các IP này thường tổng hợp tốt hơn. Điển hình với Intel là sử dụng các IP trong IP catalog.
- Lưu ý giữ các tính chất của giải thuật từ phần mềm sang phần cứng, ví dụ đòi hỏi của giải thuật phải được xử lý trong thời gian cố định khi xử lý bằng vi xử lý thì phần cứng cũng phải thỏa mãn tính chất này (chống Side-channel attack).
- Mã hóa là một đề tài đòi hỏi nhiều tính toán phức tạp và các bài kiểm tra tính an toàn của một ứng dụng cho mã hóa không dễ để vượt qua. Trong đó có các bài kiểm tra về tính an toàn với tấn công trên bộ tạo số ngẫu nhiên và tấn công về thời gian xử lý.
- **Ưu điểm của thiết kế:**
 - Tiêu thụ ít tài nguyên.
 - Đơn giản để sử dụng, nhiều tính năng.
 - Thiết kế module dễ thay đổi sửa chữa.
- **Khuyết điểm của thiết kế:**

- Giải thuật sử dụng chưa được tối ưu hết mức.
- Tần số hoạt động tối đa của thiết kế chưa cao.
- Có thể bị tấn công thời gian xử lý (với đường cong Weierstrass).

6.2. Hướng phát triển

Đề tài thiết kế bộ mã hóa đường cong elliptic 256 bit chúng em thực hiện là một đề tài có tính ứng dụng cao trong thực tiễn. Các mạng máy chủ và cơ sở dữ liệu với sự phát triển của điện toán đám mây và Software Defined Network (SDN) đang dần thiếu hụt về sức mạnh tính toán và cần đẩy các tính toán bảo mật đang đè nặng trên CPU (SSL/TLS hoặc IPSec) lên các thiết bị PCIe Offload hoặc các NIC Offload nhiều tính năng và các Switch board của các hãng viễn thông lớn hỗ trợ ASIC hoặc FPGA để xử lý. Sự lựa chọn về đường cong của đề tài cũng nhằm để phù hợp với loại đường cong mã hóa đang được sử dụng phổ biến trong bảo mật.



Hình 6-1 Intel FPGA Acceleration Hub PCIe card for servers

Với sự phát triển của công nghệ mã hóa thông tin, những điểm vượt trội về sức mạnh bảo mật cũng như khả năng sáng tạo của mã hóa đường cong elliptic sẽ còn tiến xa hơn nữa trong tương lai. Hai đường cong 256-bit sử dụng trong thiết kế với sức mạnh bảo mật 128 bit là loại đường cong được sử dụng phổ biến nhất hiện nay và vẫn bảo mật trong nhiều năm tới trừ khi có một đột phá về công nghệ xử lý [34]. Dưới đây là so sánh về sức mạnh bảo mật của đường cong elliptic so với mã hóa RSA.

Parameters	Section	Strength	Size	RSA/DSA	Koblitz or ran- dom
secp192k1	2.2.1	96	192	1536	k
secp192r1	2.2.2	96	192	1536	r
secp224k1	2.3.1	112	224	2048	k
secp224r1	2.3.2	112	224	2048	r
secp256k1	2.4.1	128	256	3072	k
→ secp256r1	2.4.2	128	256	3072	r
secp384r1	2.5.1	192	384	7680	r
secp521r1	2.6.1	256	521	15360	r

Hình 6-2 Sức mạnh bảo mật của các đường cong elliptic [4]

Đề tài nhóm chúng em đã thiết kế vẫn còn nhiều điểm cần cải thiện trong độ trễ của thiết kế để đạt được mức băng thông lớn. Đáng lưu ý là cần thêm cải thiện về xử lý song song trong giải thuật, một số gợi ý có thể là thêm block RAM để sử dụng nhiều bộ tính toán đọc ghi xử lý cùng lúc. Hỗ trợ thêm các đường cong 384-bit, 448-bit và 521-bit để tăng tính linh hoạt. Một hướng đi khác có thể làm nhỏ thiết kế lại, thêm tính năng kiểm tra chữ ký với một đường cong cụ thể để ứng dụng như một thiết kế phần cứng nhỏ để đảm nhận nhiệm vụ bảo mật trong thiết bị di động, kích thước nhỏ. Nếu thiết kế được thực hiện tối ưu cho người dùng cuối, có thể làm cho thiết kế nhỏ, hỗ trợ ít tính năng, tiết kiệm năng lượng và đạt được tốc độ tốt, giảm tải cho vi xử lý của thiết bị di động. Đề tài thiết kế bộ mã hóa đường cong elliptic 256 bit còn rất nhiều hướng phát triển và những điểm cần cải thiện để đạt được hiệu quả trong việc thay thế vi xử lý thực hiện các tính toán bảo mật.

7. TÀI LIỆU THAM KHẢO

- [1] NIST SP 800-56A Rev. 3, Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography (esp. Section 5.7.1.2)
- [2] AMERICAN NATIONAL STANDARD X9.62-1998 Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)©
- [3] FIPS PUB 186-4 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Digital Signature Standard (DSS)

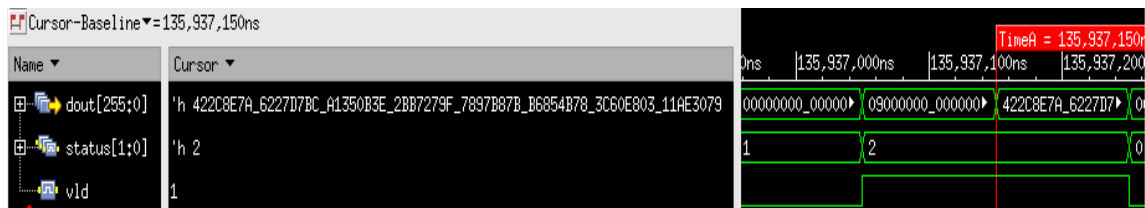
- [4] StandardsFor Efficient Cryptography SEC 2: Recommended Elliptic Curve Domain Parameters, Certicom Research Version 1.0
- [5] Transport Layer Security (TLS) Parameters – IANA - 22nd October 2018
- [6] Bernstein D.J. (2006) Curve25519: New Diffie-Hellman Speed Records. In: Yung M., Dodis Y., Kiayias A., Malkin T. (eds) Public Key Cryptography - PKC 2006. PKC 2006. Lecture Notes in Computer Science, vol 3958. Springer, Berlin, Heidelberg
- [7] FIPS PUB 140-3 - Security Requirements For Cryptographic Modules
- [8] A. Solinas, Jerome. (1999). “Generalized Mersenne Numbers”.
- [9] D.Hankerson, A.Menezes, S.Vanstone: “Guide to elliptic curve cryptography”, Springer Professional Computing (Springer, New York 2004)
- [10] ECDSA prime example number – NIST
- [11] Güneysu, Tim & Paar, Christof. (2008). “Ultra High Performance ECC over NIST Primes on Commercial FPGAs”. *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*(pp.62-78)
- [12] IEEE 1364-2005 - IEEE Standard for Verilog Hardware Description Language, 2005-11-08
- [13] RFC7748 - Elliptic Curves for Security - Internet Research Task Force (IRTF) - January 2016
- [14] Transport Layer Security (TLS) Parameters – IANA
- [15] P. Miranda, M. Siekkinen and H. Waris, "TLS and energy consumption on a mobile device: A measurement study," *2011 IEEE Symposium on Computers and Communications (ISCC)*, Kerkyra, 2011, pp. 983-989.
- [16] R. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, vol. 26, no. 1, pp. 96-99, 1983.
- [17] RFC4492 - Elliptic Curve Cryptography (ECC) Cipher Suitesfor Transport Layer Security (TLS)- Internet Research Task Force (IRTF) - January 2016

- [18] Benjamin Clement Sebastian et al, “ Advantage of using Elliptic curve cryptography in SSL/TLS”
- [19] Orlando, Gerardo. (2002). “A Scalable GF(p) Elliptic Curve Processor Architecture for Programmable Hardware”.
- [20] Johnson, D., Menezes, A. & Vanstone, S. IJIS (2001). “The Elliptic Curve Digital Signature Algorithm (ECDSA)”. *International Journal of Information Security* August 2001, Volume 1, Issue 1, pp 36–63
- [21] Izu T., Takagi T. (2002) “Fast Elliptic Curve Multiplications with SIMD Operations”. In: Deng R., Bao F., Zhou J., Qing S. (eds) *Information and Communications Security*. ICICS 2002. Lecture Notes in Computer Science, vol 2513. Springer, Berlin, Heidelberg
- [22] Aoki, Kazumaro & Hoshino, Fumitaka & Kobayashi, Tetsutaro & Oguro, Hiroaki. (2001). “Elliptic Curve Arithmetic Using SIMD”. *Information Security, 4th International Conference, ISC 2001, Malaga, Spain, October 1-3, 2001, Proceedings*(pp.235-247)
- [23] Itoh K., Takenaka M., Torii N., Temma S., Kurihara Y. (1999) Fast “Implementation of Public-Key Cryptography on a DSP TMS320C6201”. In: Koç Ç.K., Paar C. (eds) *Cryptographic Hardware and Embedded Systems*. CHES 1999. Lecture Notes in Computer Science, vol 1717. Springer, Berlin, Heidelberg
- [24] Paul C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", *CRYPTO '96 Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, pp. 104-113, 1996.
- [25] W. Fischer, C. Giraud, E. Knudsen, and J.-P. Seifert, “Parallel scalar multiplication on general elliptic curves over F_p hedged against Non-Differential Side Channel Attacks”, 2002, IACR
- [26] Elliptic Curve Point Multiply and Verify Core – IP cores
- [27] Bay[®] Acceleration Software for Security and CryptographyFPGA Acceleration for Cloud Computing, Data Centers and NFV Networking - Arrive Technologies

- [28] Joye M. (2008) “Fast Point Multiplication on Elliptic Curves without Precomputation”. In: von zur Gathen J., Imaña J.L., Koç Ç.K. (eds) *Arithmetic of Finite Fields. WAIFI 2008. Lecture Notes in Computer Science*, vol 5130. Springer, Berlin, Heidelberg
- [29] C. Costello and B. Smith, "Montgomery curves and their arithmetic", *Journal of Cryptographic Engineering*, vol. 8, no. 3, pp. 227-240, 2017.
- [30] Md. Rafiqul Islam, Md. Sajjadul Hasan and Ikhtear Sharif Muhammad Asaduzzaman, "A New Point Multiplication Method for Elliptic Curve Cryptography Using Modified Base Representation", *International Journal of The Computer, the Internet and Management*, vol. 16, no. 2, pp. 9-16, 2008.
- [31] Pedro Maat C. Massolino, Joost Renes and Lejla Batina, "Implementing Complete Formulas on Weierstrass Curves in Hardware", 2016
- [32] IETF RFC 7748 – Elliptic Curve For Securiy – S. Turner
- [33] HACL Verified libraries - <https://github.com/project-everest/hacl-star>
- [34] Bos, Joppe & Kaihara, Marcelo & Kleinjung, Thorsten & K. Lenstra, Arjen & L. Montgomery, Peter. (2009) “On the Security of 1024-bit RSA and 160-bit Elliptic Curve Cryptography”. *IACR Cryptology ePrint Archive*. 2009. 389.
- [35] INTEL Integer Arithmetic IP Cores User Guide Document Archives
- [36] G. Marsaglia, "Xorshift RNGs", *Journal of Statistical Software*, vol. 8, no. 14, 2003.
- [37] Roy Ward and Tim Molteno, "Table of Linear Feedback Shift Registers", 2007.
- [38] Sakai Y., Sakurai K. (2005), “Simple Power Analysis on Fast Modular Reduction with NIST Recommended Elliptic Curves”. In: Qing S., Mao W., López J., Wang G. (eds) *Information and Communications Security. ICICS 2005. Lecture Notes in Computer Science*, vol 3783. Springer, Berlin, Heidelberg
- [39] Tech invite, “IPsec guide: architecture and traffic processing”

8. PHỤ LỤC

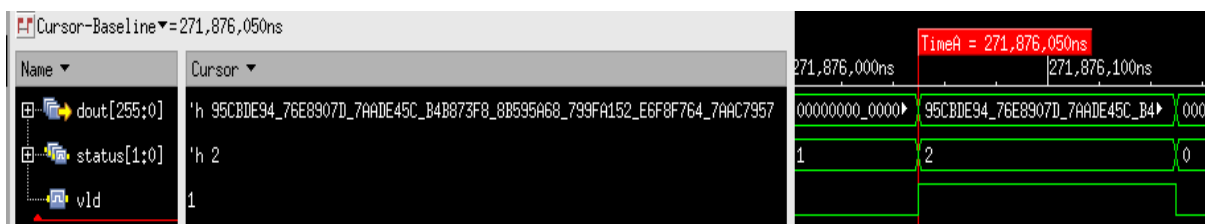
Dạng sóng và bộ nhớ của thiết kế NO_CLA_PL (kết quả tích vô hướng / ECDHE ở địa chỉ 15, ECDSA có thêm kết quả ở địa chỉ 14) với ngõ vào như trên mục kết quả đã ghi:



Hình 8-1 Kết quả waveform ECDHE GEN X25519 thiết kế NO_CLA_PL



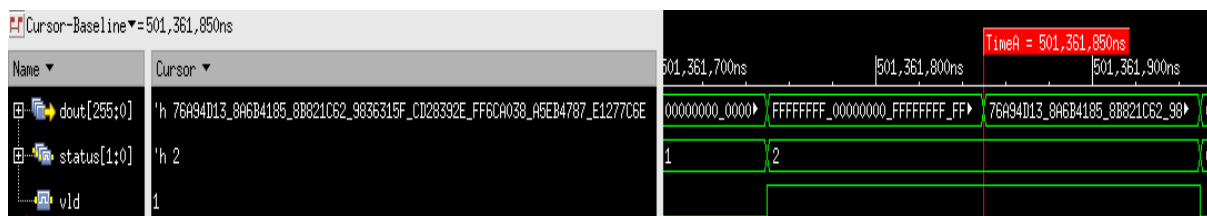
Hình 8-2 Kết quả bộ nhớ ECDHE GEN X25519 thiết kế NO_CLA_PL



Hình 8-3 Kết quả waveform ECDHE COMP X25519 thiết kế NO_CLA_PL



Hình 8-4 Kết quả bộ nhớ ECDHE GEN X25519 thiết kế NO_CLA_PL



Hình 8-5 Kết quả waveform ECDHE GEN NIST-P256 thiết kế NO_CLA_PL



Hình 8-6 Kết quả bộ nhớ ECDHE GEN NIST-P256 thiết kế NO_CLA_PL

[illegible]

```

Input
k = ffffffff00000000ffffffffffffffffbce6faada7179e84f3b9cac2fc632541
ECDHE GEN P256 test is done at          2448236950000

Output: 76a94d138a6b41858b821c629836315fcd28392eff6ca038a5eb4787e1277c6e

Expected :
76a94d138a6b41858b821c629836315fcd28392eff6ca038a5eb4787e1277c6e
ECDHE GEN P256 is working
ECDSA P256 begin at          2448239250000

Input
hash = a41a41a12a799548211c410c65d8133afde34d28bdd542e4b680cf2899c8a8c4
private key =
c477f9f65c22cce20657faa5b2d1d8122336f851a508a1ed04e479c34985bf96
k = 7a1a7e52797fc8caaa435d2a4dace39158504bf204fbe19f14dbb427faee50ae
ECDSA P256 test is done at          3665875950000

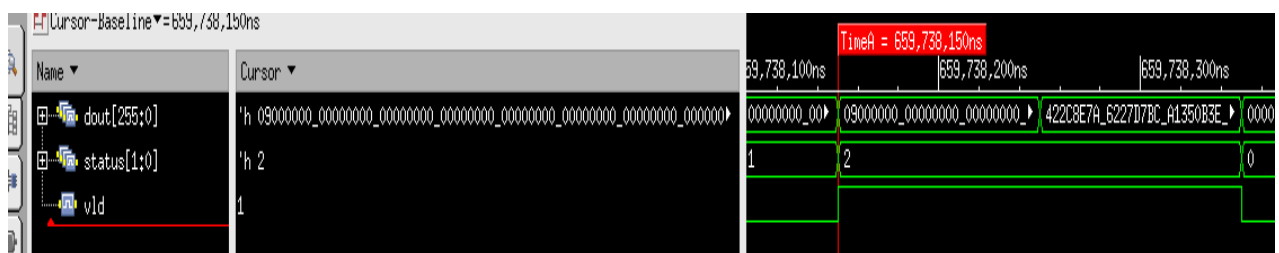
Output: 2b42f576d07f4165ff65d1f3b1500f81e44c316f1f0b3ef57325b69aca46104f

Expected :
2b42f576d07f4165ff65d1f3b1500f81e44c316f1f0b3ef57325b69aca46104f
ECDSA P256 R is working

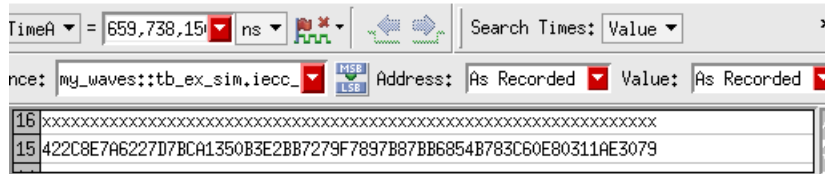
Output: dc42c2122d6392cd3e3a993a89502a8198c1886fe69d262c4b329bdb6b63faf1

Expected :
dc42c2122d6392cd3e3a993a89502a8198c1886fe69d262c4b329bdb6b63faf1
ECDSA P256 S is working
    
```

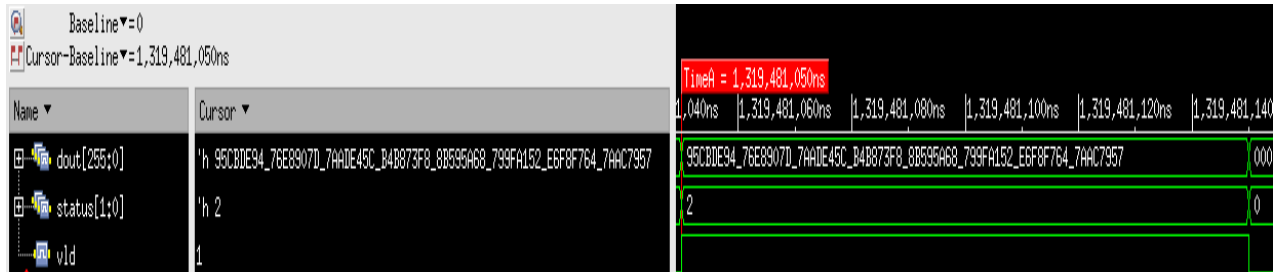
Các kết quả mô phỏng trên dạng sóng và bộ nhớ của thiết kế 2_CLA_PL (kết quả tích vô hướng / ECDHE nằm ở địa chỉ thứ 15 trong bộ nhớ, ECDSA có thêm kết quả ở địa chỉ 14) với ngõ vào không đổi:



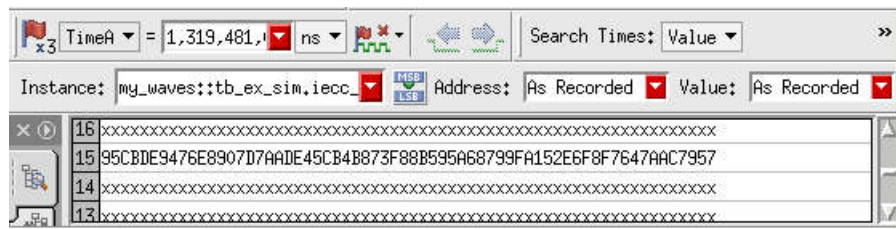
Hình 8-9 Kết quả waveform ECDHE GEN X25519 thiết kế 2_CLA_PL



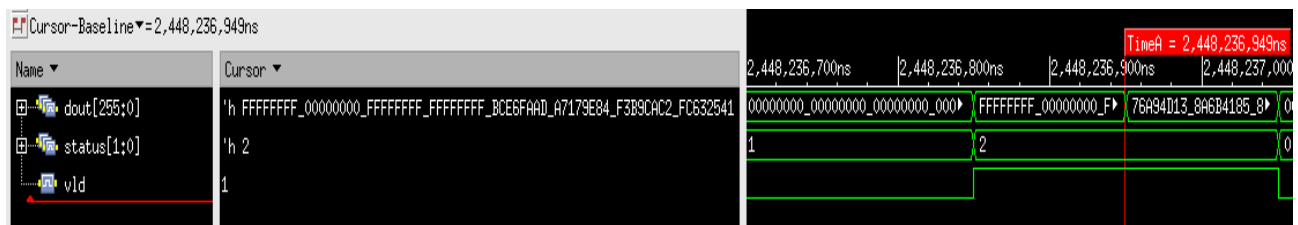
Hình 8-10 Kết quả bộ nhớ ECDHE GEN X25519 thiết kế 2_CLA_PL



Hình 8-11 Kết quả waveform ECDHE COMP X25519 thiết kế 2_CLA_PL



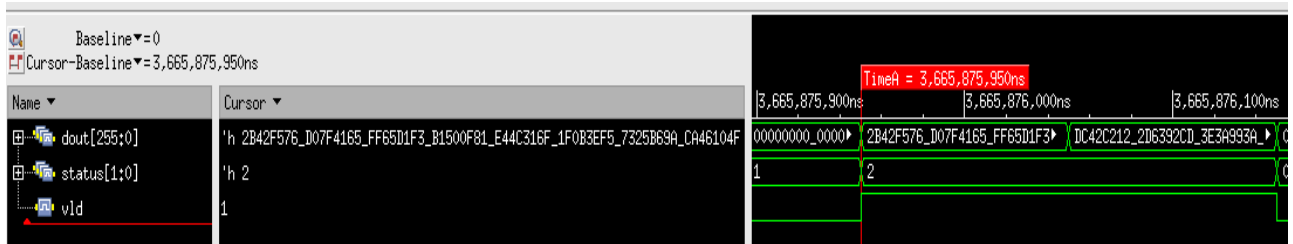
Hình 8-12 Kết quả bộ nhớ ECDHE COMP X25519 thiết kế 2_CLA_PL



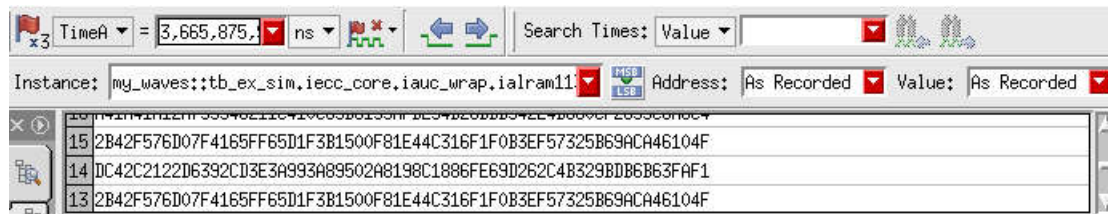
Hình 8-13 Kết quả waveform ECDHE GEN NIST-P256 thiết kế 2_CLA_PL



Hình 8-14 Kết quả bộ nhớ ECDHE GEN NIST-P256 thiết kế 2 CLA PL



Hình 8-15 Kết quả waveform ECDSA NIST-P256 thiết kế 2_CLA_PL



Hình 8-16 Kết quả bộ nhớ ECDSA NIST-P256 thiết kế 2_CLA_PL