THIẾT KẾ SoC CHO MẠCH HỆ MẬT MÃ KHỐI AES

SoC DESIGN FOR AES BLOCK CIPHER SYSTEM

Đàm Đức Trung[1], Tạ Tiến Đạt[2], Lục Như Quỳnh[2]*

[1]Viện Vũ khí, Tổng cục Công nghiệp Quốc phòng, Hà Nội, Việt Nam
[2]Học viện Kỹ thuật Mật mã, Hà Nội, Việt Nam

*Tác giả liên hệ / Corresponding author: quynhln@actvn.edu.vn

(Nhận bài / Received: 01/3/2025; Sửa bài / Revised: 28/5/2025; Chấp nhận đăng / Accepted: 02/6/2025)

DOI: 10.31130/ud-jst.2025.23(7).115

Tóm tắt - Bài báo này trình bày thiết kế của một SoC cho mạch hệ mật mã khối AES. Thiết kế này được triển khai dựa trên thuật toán mã hóa và giải mã Rijndael theo tiêu chuẩn AES để có thể thực hiện cả mở rộng khóa và các thuật toán mã hóa/giải mã, thiết kế hỗ trợ cả ba độ dài khóa là 128, 192 và 256-bit. Nhóm tác giả đã sử dụng công cụ EDA Altera Quartus II với ngôn ngữ Verilog để tổng hợp thiết kế và sử dụng Modelsim để mô phỏng chức năng của thiết kế. Kết quả khi thiết kế được cài đặt trên FPGA Altera Cyclone III DE0 EP3C16F484C6, tổng số phần tử logic được sử dụng là 5106 chiếm 33% tài nguyên có sẵn trên FPGA được chọn, thiết kế ước tính tiêu thụ năng lượng là 73,53mW và cần 50 chu kỳ xung clock cho quá trình mã hóa/giải mã với tần số xung clock là 50MHZ. Đồng thời thiết kế được triển khai trên Cadence SoC Encounter với bộ công cụ FreePDK45, kết quả cho thấy thiết kế này tiêu thụ năng lượng là 1,18W.

Từ khóa - Thiết kế SoC; hệ mật mã khối AES; thuật toán Rijndael; SoC Encounter; FreePDK45

Abstract - This paper presents the design of a System-on-Chip (SoC) for an AES block cipher system. The design is implemented based on the Rijndael encryption and decryption algorithm following the AES standard, enabling both key expansion and encryption/decryption operations. It supports all three key lengths: 128, 192, and 256-bits. The authors utilized the Altera Quartus II EDA tool with Verilog for design synthesis and ModelSim for functional simulation. When implemented on the Altera Cyclone III DE0 EP3C16F484C6 FPGA, the design utilized a total of 5,106 logic elements, accounting for 33% of the available FPGA resources. The estimated power consumption was 73.53 mW, and the encryption/decryption process required 50 clock cycles with a clock frequency of 50 MHz. Additionally, the design was implemented on Cadence SoC Encounter using the FreePDK45 toolkit, where the results showed a power consumption of 1.18 W.

Key words - SoC design; AES block cipher system; Rijndael algorithm; SoC Encounter; FreePDK45

1. Đặt vấn đề

Năm 2001, Viện Tiêu chuẩn và Công nghệ Hoa Kỳ (NIST) lựa chọn hệ mật mã khối tiên tiến AES (Advanced Encryption Standard) đã thay thế chuẩn DES và trở thành thuật toán mã hóa được sử dụng rộng rãi nhất, phù hợp với các nền tảng phần cứng và phần mềm có hiệu suất và mức an toàn tốt [1], [2]. Hiện nay, chưa có nhiều nghiên cứu chỉ ra có phương pháp tấn công thực tế nào có thể phá vỡ AES (với các phiên bản AES 128-bit, 192-bit và 256-bit) [1]. Hầu hết các tấn công chủ yếu tập trung vào phân tích năng tiêu thụ, khóa yếu hoặc do quá trình triển khai không an toàn, như tấn công kênh kề [3]. Do đó, khi triển khai AES trên phần cứng yêu cầu cần có các biện pháp bảo vệ, làm nhiễu thời gian thực thi hoặc các kỹ thuật chống tấn công kênh kề để đảm bảo tính bảo mật cao nhất.

Theo công bố [4], thiết kế AES trên FPGA với kiến trúc pipeline và song song hóa có thể đạt tốc độ mã hóa lên tới hàng Gbps với độ trễ thấp. Trong công bố này cũng chỉ ra nhược điểm của AES trên FPGA tiêu thụ năng lượng cao hơn so với ASIC và chi phí sản xuất lớn. Mặt khác, công bố [5] đề xuất một kỹ thuật mới để tạo S-hộp (S-box) động trong mật mã AES và module mở rộng khóa AES được cải tiến với mục tiêu giảm hiệu suất tiêu thụ và diện tích phần cứng. Gần đây, với công nghệ FPGA, công bố [6] đưa ra thiết kế SoC (System on Chip) cho hệ mật AES đã cải thiện được về tối ưu hóa mạch RTL, tốc độ, diện tích và công suất tiêu thụ. Với công nghệ ARM, công bố [7] đã đề xuất kiến trúc RISC-V của hệ mật AES với thiết kế SoC 32-bit dựa trên công nghệ CMOS 180nm, trong đó đã thiết kế Sbox với kiến trúc pipeline. Điều này giúp giảm yêu cầu về bộ nhớ, tiết kiệm năng lượng hơn 100 lần so với phần mềm.

Trong nghiên cứu này, đầu tiên nhóm tác giả đã khảo sát một số công bố về hiệu quả, tài nguyên của hệ mật AES trên phần cứng. Sau đó, thực hiện cài đặt, thực thi hệ mật AES trên chíp FPGA. Từ đó sẽ đưa ra các phân tích, đánh giá hiệu suất hoạt động của AES về sử dụng tài nguyên, năng lượng. Đồng thời, đưa ra kết quả bước đầu trong thiết kế SoC của hệ mật AES bằng công cụ SoC Encounter sử dụng thư viện FreePDK45. Điều này được nhóm tác giả trình bày chi tiết trong các mục tiếp theo.

2. Cơ sở để thực hiện cứng hóa AES

2.1. Các nghiên cứu liên quan

Theo công bố [4], Kala cùng cộng sự đã đề xuất một triển khai FPGA của thuật toán AES 128-bit song song thông lượng cao với cơ chế mở rộng khóa công suất thấp cho các giai đoạn lặp. Một khóa đối xứng 128-bit đã được sử dụng để thực hiện 10 vòng chuyển đổi. Tất cả các quá trình mã hóa và giải mã đều được mô phỏng bằng phương pháp thiết kế lặp để giảm thiểu mức tiêu thụ phần cứng.

[1] Institute of Weapons, General Department of Defense Industry, Hanoi, Vietnam (Duc Trung Dam)
[2] Academy of Cryptography Techniques, Hanoi, Vietnam (Tien Dat Ta, Nhu Quynh Luc)

Kết quả cho thấy, quá trình mã hóa được thực hiện với tốc độ cao 68 Gb/giây và mức tiêu thụ năng lượng thấp 7 pJ/bit trên thiết bị FPGA Xilinx Artix-7.

Công bố [5] đã đề xuất một kỹ thuật mới để tạo hộp S động (S-box) trong mật mã AES. Module mở rộng khóa AES được cải tiến để giảm tiêu thụ điện năng và diện tích phần cứng. Phương pháp triển khai sử dụng cổng NAND có thể áp dụng trên FPGA để giảm mức tiêu thụ năng lượng so với các thiết kế truyền thống. Ngoài ra, việc hiện thực phần cứng của Substitution BOX bằng cổng NAND giúp tối ưu hóa hiệu suất và yêu cầu năng lượng. Thiết kế sử dụng công cụ Cadence Virtuoso và được ứng dụng trong các giao thức truyền tệp bảo mật như FTPS, HTTPS. Tiếp theo, công bố [6] đã đưa ra kết quả về thiết kế SoC bộ xử lý cho giải pháp sử dụng hệ mật mã khối AES có năng lượng tiêu thụ thấp chỉ còn 44,32mW.

Các phương pháp tiếp cận khác nhau cho các lược đồ tăng tốc AES-128/256 Sbox bằng cách đưa ra thiết kế một đơn vị Sbox để được kết nối như một đơn vị logic [7]. Nhóm tác giải đã chế tạo một SoC 32 bit dựa trên RISC-V trong công nghệ CMOS 180nm tiêu chuẩn bao gồm tính toán Sbox bằng cách sử dụng đường ống và phương pháp kết hợp thuần túy. Kết quả đo lường cho thấy, khả năng truy cập bộ nhớ thấp hơn và yêu cầu năng lượng thấp hơn so với các phương pháp tiếp cận phần mềm thuần túy lên đến 100 lần với lệnh Sbox tùy chỉnh.

Srinivas và cộng sự [8] đã đề xuất các kỹ thuật mã hóa và giải mã AES có kiến trúc Rijndael dựa trên các bảng tra cứu đã được tính toán trước. Trong công bố [8] đã chỉ ra kiến trúc do tác giả cùng cộng sự thực hiện có độ phức tạp thấp hơn nhưng vẫn đảm bảo độ trễ thấp, có thông lượng cao. Ngoài ra, thông qua kết quả có công suất là 289 mW tương đối cao, điều này đòi hỏi cần có chip xử lý phải có công suất cao. Với các kết quả đó, trong nghiên cứu đã có đề xuất lựa chọn giải pháp giúp giảm thiểu về hiệu xuất và năng lượng tiêu thụ của hệ mật AES trên chip.

Năm 2020, công bố [9] đưa ra thiết kế AES trên FPGA bằng công cụ HLS, đã được tối ưu hóa bằng cách sử dụng các cú pháp lệnh đặc biệt của công cụ HLS, có thông lượng tốt và công suất 124 mW. Do đó, việc cứng hóa AES mang lại hiệu quả và hiệu suất tốt hơn, giảm tiêu thụ tài nguyên so với thực hiện trên phần mềm thuần túy.

2.2. Hệ mật mã khối AES

2.2.1. Mã hóa AES

Cấu trúc của AES được thiết kế dựa trên kiến trúc Rijndael. Phương pháp mã hóa AES bao gồm nhiều bước biến đổi được thực hiện tuần tự, kết quả đầu ra của bước biến đổi trước là đầu vào của bước biến đổi tiếp theo. Kết quả trung gian giữa các bước biến đổi được gọi là trạng thái (state). Một trạng thái có thể được biểu diễn dưới dạng một ma trận gồm 4 hàng các byte với Nb bằng với độ dài của khối block chia cho 32. Mảng trạng thái ký hiệu là s trong đó mỗi byte của mảng có 2 chỉ số hàng r ($0 < r < 4$) và cột c ($0 < c < Nb$). Với thuật toán AES thì độ dài của input block, output block và State là 128-bit. Nên Nb = 4 (đây chính là số cột – số từ 32 bit) trong State. Mã khóa chính (Cipher Key) cũng được biểu diễn dưới dạng một ma trận gồm 4 dòng và Nk cột với Nk bằng với độ dài của khóa chia cho 32.

Theo chuẩn AES, độ dài của từ khóa K có thể là 128, 192, hoặc 256-bit. Độ dài của Key tương ứng là Nk = 4, 6, hoặc 8 (chính số cột của từ khóa), số vòng thực hiện mã hóa dữ liệu của thuật toán sẽ phụ thuộc vào độ lớn của từ khóa. Số vòng quay kí hiệu là *Nr,* với *Nr* = 10 khi *Nk = 4, Nr* = 12 khi *Nk* = 6, và *Nr* = 14 khi *Nk* = 8.

Quy trình mã hóa sử dụng bốn phép biến đổi chính:

- AddRoundKey: cộng XOR mã khóa của chu kỳ vào trạng thái hiện hành. Độ dài của mã khóa của chu kỳ bằng với kích thước của trạng thái là 128-bit.

- SubBytes: thay thế phi tuyến mỗi byte trong trạng thái hiện hành thông qua bảng thay thế (S-box).

- ShiftRows: dịch chuyển xoay vòng từng dòng của trạng thái hiện hành với di số khác nhau.

- MixColumns: trộn thông tin của từng cột trong trạng thái hiện hành. Mỗi cột được xử lý độc lập.

Hoạt động mã hóa của AES 128-bit, đầu tiên toàn bộ dữ liệu đầu vào được chép vào mảng trạng thái hiện hành. Sau khi thực hiện cộng mã khóa đầu tiên, mảng trạng thái sẽ được trải qua Nr = 10, 12 hay 14 chu kỳ biến đổi (tùy thuộc vào độ dài của mã khóa chính cũng như độ dài của khối được xử lý). Nr − 1 chu kỳ đầu tiên là các chu kỳ biến đổi bình thường và hoàn toàn tương tự nhau. Riêng vòng cuối cùng có một sự khác biệt là không thực hiện MixColumns. Cuối cùng, giá trị của mảng trạng thái sẽ được chép lại vào mảng chứa dữ liệu đầu ra. Điều này, cho thấy luồng hoạt động mã hóa là: Ở vòng đầu tiên (vòng 0), bộ mã hóa chỉ thực hiện AddRoundKey với khóa đầu tiên, sau đó thực hiện luôn vòng 1. Từ vòng 1 đến vòng Nr-1, bộ mã hóa thực hiện đầy đủ 4 phép biến đổi SubBytes, ShiftRows, MixColumns và AddRoundKey. Vòng cuối cùng, bộ mã hóa chỉ thực hiện SubBytes, ShiftRows và AddRoundKey mà không thực hiện MixColumns.

2.2.2. Giải mã AES.

Quá trình giải mã được thực hiện qua các giai đoạn sau:

- Thực hiện thao tác AddRoundKey đầu tiên trước khi thực hiện các chu kỳ giải mã.

- Nr – 1 chu kỳ giải mã bình thường: mỗi chu kỳ bao gồm bốn biến đổi liên tiếp nhau: InvShiftRows, InvSubBtes, AddRoundKey, InvMixColumns.

- Thực hiện chu kỳ giải mã cuối cùng. Trong chu kỳ này, thao tác InvMixColumns được bỏ qua.

2.2.3. Mở rộng khóa

KeyExpansion (128-bit): Lấy 128-bit khóa và tính toán khóa vòng cho bước lặp mã hóa và bước tạo ngõ ra. Kết quả của một lần thực thi KeyExpansion là một khóa vòng sử dụng cho chức năng AddRoundKey. Mã hóa AES-128 có 1 khóa mã và 10 khóa vòng nên tổng số từ là 44 và được đánh số từ 0 đến 43. Khóa mã có 4 từ là w[0], w[1], w[2] và w[3]. Khóa vòng 1 có 4 từ là w[4], w[5], w[6] và w[7]. Tương tự, khóa vòng 10 có 4 từ là w[40], w[41], w[42] và w[43]. Từ w[j] tính theo công thức sau, với $3 < j < 44$.

$$w[j] = AddW[j - 4] = w[j - 1] + w[j - 4]$$

Khi tính các từ ở vị trí j là bội số của 4, như w[4], w[8],… và w[40], thì w[j-1] phải được biến đổi qua 3 chức năng RotWord, SubWord và AddRcon, gọi là trans(w[j-

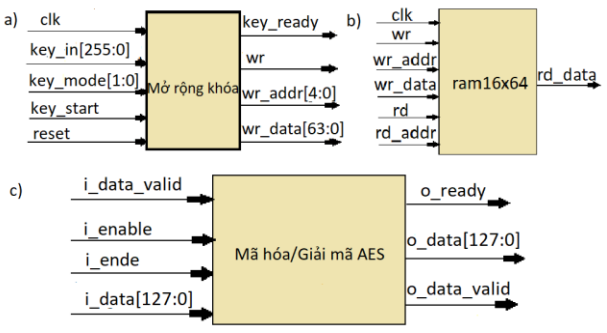
1]), trước khi XOR với w[j-4]. RotWord: Thực hiện quay trái từ w[j] một byte. SubWord thực hiện thay thế các phi tuyến từng byte của kết quả RotWord theo bảng S-box. AddRcon thực hiện XOR kết quả SubWord và giá trị Rcon[j/4] với j là bội số của 4.

Việc triển khai AES trên phần cứng chuyên dụng giúp giảm thời gian xử lý so với phần mềm, chống lại các tấn công kênh kề như DPA, LPA (bằng cách tích hợp kỹ thuật mặt nạ (masking), chèn độ trễ ngẫu nhiên [10]), đảm bảo với các ứng dụng yêu cầu thời gian thực. Do đó, khi cứng hóa AES, việc cập nhật thuật toán hoặc thay đổi kích thước khóa trở nên khó khăn hơn so với phần mềm. Trong nghiên cứu này, thiết kế AES, khối có thể thực hiện cả mở rộng khóa (cần thực hiện mỗi lần khi thay đổi khóa, kể cả sau khi reset hoặc bật nguồn) và các thuật toán mã hóa/giải mã. Khối được thiết kế với cả ba độ dài khóa là 128, 192 và 256-bit, có thể chọn thông qua tín hiệu đầu vào. Các chế độ mã hóa hoặc giải mã cũng được chọn tức thì qua tín hiệu đầu vào. Khối sử dụng một bộ nhớ RAM nhỏ nội bộ để lưu trữ khóa đã mở rộng. Các khối ngoại vi được thiết kế với chiều rộng tối đa cần thiết của dữ liệu để giảm thiểu thời gian cần thiết để nạp thông tin. Vì vậy, khóa và dữ liệu có chiều rộng lần lượt là 256-bit và 128-bit. Để có thể hỗ trợ tốc độ dữ liệu cao, khối AES cho phép vận hành pipeline với bốn vector đồng thời. Bằng cách sử dụng tín hiệu đầu ra sẵn sàng hoặc một thời gian cố định, người dùng có thể nạp tối đa bốn vector văn bản gốc hoặc đã mã hóa vào khối để mã hóa hoặc giải mã.

2.3. Thiết kế SoC cho mạch hệ mật mã khối AES

2.3.1. Thiết kế khối mở rộng khóa và khối RAM

Khối mở rộng khóa (Hình 1a) thực hiện chức năng mở rộng khóa, sử dụng 2 module ram 64-bit (Hình 1b) để lưu trữ giá trị khóa vòng sau mỗi lần tính toán. Khóa chính được đưa vào bộ mở rộng khóa qua tín hiệu key_in và xác định chế độ khóa qua tín hiệu key_mode. Quá trình mở rộng khóa được bắt đầu khi tín hiệu key_start được kích hoạt. Mỗi khóa con sẽ được ghi vào bộ nhớ ram qua tín hiệu wr và wr_addr. Tín hiệu key_ready được kích hoạt khi hoàn thành tất cả quá trình mở rộng khóa. Khối này được thiết kế gồm 4 chức năng là RotWord, SubWord, AddRcon và AddW. Sau khi khóa đầu vào key_in được nạp và chế độ khóa key_mode được chọn, quá trình mở rộng khóa sẽ bắt đầu. Các chế độ khóa có thể chọn là 0 cho 128 bit, 1 cho 192 bit, và 2 cho 256 bit. Giá trị rcon được sinh ra để phục vụ cho việc tính toán khóa con, rcon được cập nhật qua mỗi vòng lặp và có thể thay đổi giá trị dựa trên điều kiện đầu ra của S-Box.



Hình 1. *a) Sơ đồ khối mở rộng khóa; b) Sơ đồ tín hiệu khối RAM; c) Sơ đồ tín hiệu khối mã hóa/giải mã AES*

Quá trình mở rộng khóa, S-Box nhận đầu vào từ các từ khóa hiện tại và tạo ra đầu ra sau mỗi phép biến đổi. Các khóa con được tính toán dựa trên các công thức XOR giữa đầu ra của S-Box, giá trị rcon, và các từ khóa trước đó. Khi các khóa con được tính toán, giá trị sẽ được ghi vào RAM thông qua các tín hiệu điều khiển wr và wr_addr. Kết thúc mở rộng khóa, tín hiệu key_ready sẽ được kích hoạt.

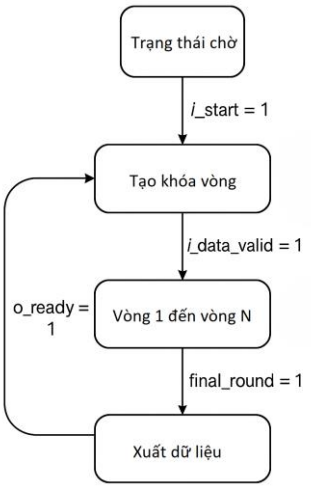
2.3.2. Thiết kế khối mã hóa/giải mã AES

Khối mã hóa/giải mã AES (Hình 1c) có thể chuyển đổi linh hoạt giữa mã hóa và giải mã dựa trên tín hiệu i_ende. Tín hiệu này quyết định chế độ hoạt động của khối:

- Khi i_ende = 1'b0: Khối hoạt động ở chế độ mã hóa.
- Khi i_ende = 1'b1: Khối chuyển sang chế độ giải mã.

Khối sẽ chỉ thực hiện quá trình mã hóa hoặc giải mã khi tín hiệu i_enable ở mức cao, dữ liệu đầu vào i_data nhận đủ 128-bit và quá trình mở rộng khóa đã hoàn thành, tương ứng với tín hiệu i_data_valid và key_ready được đặt ở mức cao. Có thể nhập 4 dữ liệu 128-bit vào khối trước dữ liệu đầu ra hợp lệ đầu tiên, tín hiệu o_ready sẽ được đặt lên mức cao khi có thể tiếp tục nạp dữ liệu đầu vào bằng cách triển khai pipeline.

Các vòng xử lý của AES, mỗi bước được chia thành các giai đoạn tương ứng, với ba stage chính được thể hiện qua thanh ghi sb_valid[2:0] và một stage round_valid phía trước để đưa dữ liệu vào pipeline, round_valid dùng để khởi động pipeline mỗi khi dữ liệu hợp lệ được đưa vào, sb_valid[2:0] dịch tuần tự giá trị hợp lệ qua ba stage: stage 1 (SubBytes), stage 2 (ShiftRows), stage 3 (MixColumns và AddRoundKey) cho phép dữ liệu được xử lý liên tục ở các stage khác nhau mà không chờ từng vòng hoàn thành xong. Trong các vòng xử lý, tín hiệu round_cnt được dùng để đếm số vòng hiện tại. sb_round_cnt1, sb_round_cnt2, sb_round_cnt3 được sử dụng để delay 3 chu kỳ tương ứng với 3 stage pipeline giúp đồng bộ chỉ số vòng ứng với dữ liệu đang đi qua pipeline, điều này giúp đọc đúng round_key tương ứng với vòng đang xử lý.



Hình 2. *Sơ đồ trạng thái mã hóa/giải mã AES*

Trong thiết kế khối mã hóa/giải mã AES, Hình 2 mô tả toàn bộ quá trình xử lý được điều khiển bởi một máy trạng thái hữu hạn gồm các trạng thái sau:

- Trạng thái chờ: bộ AES đợi tín hiệu khởi động i_start = 1 để bắt đầu quá trình mở rộng khóa.

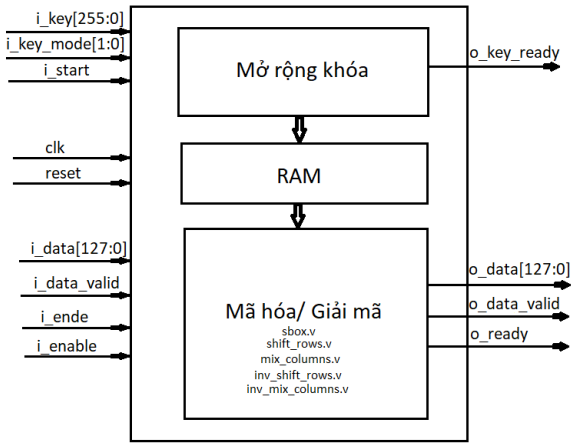
- Tạo khóa vòng: khởi tạo logic và thanh ghi cho quá trình tạo khóa vòng bao gồm reset bộ đếm và xác lập trạng thái khởi đầu. Khi quá trình mở rộng khóa hoàn tất và dữ liệu plaintext/ciphertext đã sẵn sàng (i_data_valid = 1), hệ thống chuyển sang trạng thái tiếp theo. Khi tín hiệu o_ready =1 có thể tiếp tục nạp dữ liệu đầu vào.

- Vòng 1 đến vòng N: thực hiện lần lượt các vòng mã hóa/giải mã theo số vòng tương ứng với loại khóa: 10 vòng cho 128-bit, 12 cho 192-bit, 14 cho 256-bit. Khi final_round = 1, hệ thống chuyển sang trạng thái tiếp theo.

- Xuất dữ liệu: dữ liệu đầu ra đã sẵn sàng, kết thúc một chu trình xử lý AES.

2.3.3. Nguyên lý hoạt động của AES trên phần cứng

Chế độ mã hóa: chu kỳ đầu tiên dữ liệu đầu vào i_data sẽ được XOR với khóa vòng đầu tiên. Trong các vòng mã hóa chính sẽ gồm các phép biến đổi SubBytes, ShiftRows, MixColumns và AddRoundKey. Ở vòng cuối cùng bước MixColumns sẽ được bỏ qua và thực hiện AddRoundKey với khóa cuối cùng, dữ liệu mã hóa cuối cùng được gửi qua o_data và được đánh dấu hợp lệ bởi o_data_valid. Sơ đồ khối AES của thiết kế được thể hiện trong Hình 3.



***Hình 3**. Sơ đồ khối AES của thiết kế*

Chế độ giải mã: Việc giải mã tương tự mã hóa, chỉ khác nhau ở một số công đoạn và thứ tự thực hiện các phép biến đổi, nên có thể sử dụng câu lệnh để phân biệt chế độ mã hóa và giải mã thay vì tách riêng ra hai trường hợp. Chu kỳ đầu tiên dữ liệu đầu vào i_data sẽ được XOR với khóa vòng cuối cùng để tạo ra dữ liệu đầu vào cho các vòng giải mã. Các vòng giải mã chính sẽ bao gồm các phép biến đổi Inverse Shift Rows, InvSubBytes, Add Round Key và Inverse Mix Columns. Ở vòng cuối cùng phép biến đổi Inverse Mix Columns sẽ không được thực hiện. Sau khi thực hiện phép XOR cuối cùng với khóa ban đầu sẽ tạo ra dữ liệu đầu ra của giải mã là o_data và được đánh dấu là hợp lệ bởi tín hiệu o_data_valid.

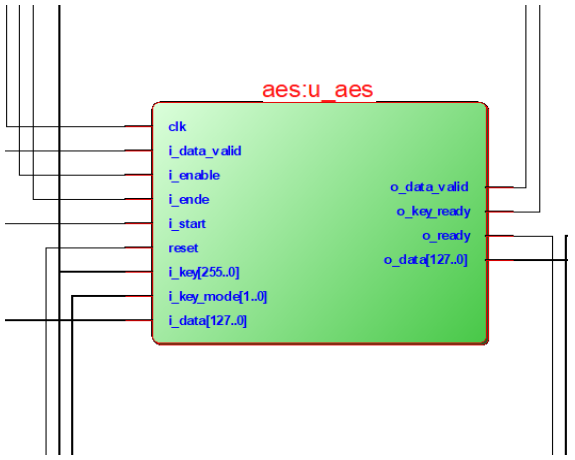
3. Kết quả và thảo luận

3.1. Phân tích, đánh giá thiết kế và hiệu năng thực thi của AES trên FPGA

Thiết kế được triển khai trên Altera Cyclone III DE0 EP3C16F484C6 với ngôn ngữ Verilog, được tổng hợp trên công cụ Quartus 13.0. Báo cáo tổng hợp của thiết kế được thể hiện trong Bảng 1 chỉ ra thiết kế của khối AES tiêu tốn tài nguyên là 4565 phần tử logic, kết quả này cho thấy thiết kế tiêu tốn lượng tài nguyên nhỏ. Hình 4 mô tả khối AES sau khi được thực hiện tổng hợp cho thấy được các tín hiệu đầu vào và đầu ra của khối.

***Bảng 1.** Báo cáo tổng hợp trên Quartus của khối AES*

Tổng số phần tử logic	Tổng số hàm tổ hợp	Tổng số thanh ghi	Tổng số chân
4565	4121	1676	80

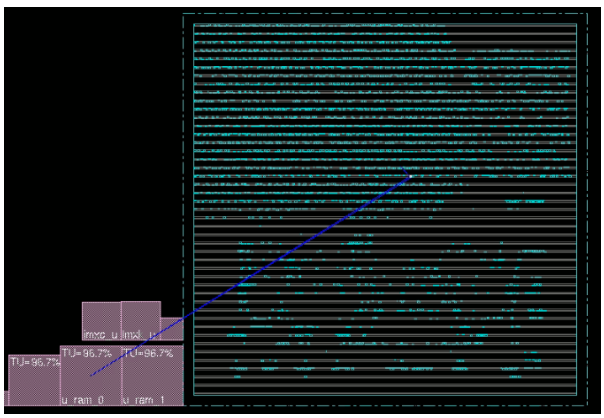


***Hình 4.** Mạch RTL của khối AES*

Thực hiện mô phỏng các chức năng mở rộng khóa, mã hóa/giải mã AES 128-bit trên công cụ ModelSim. Kết quả mô phỏng cho thấy, với tần số xung clock là 50MHZ, việc thực hiện mở rộng khóa cần thực hiện trong 45 chu kỳ, mã hóa và giải mã của khối AES (với khối dữ liệu khóa 128 bit) cần 42 chu kỳ xung clock. Do đó, thông lượng của thiết kế khối AES được đề xuất tính được là 609Mbps, tính toán được giả định rằng khối đang mã hóa hoặc giải mã bốn vector văn bản gốc hoặc đã mã hóa sử dụng chế độ pipeline.

3.2. Phân tích quá trình thực thi layout của SoC AES bằng Cadence SoC Encounter

Thiết kế được triển khai trên ASIC Cadence SoC Encounter với thư viện FreePDK45 Thư viện bao gồm tất cả các quy tắc thiết kế bố trí cần thiết và các bộ lệnh trích xuất để nắm bắt sự thay đổi có hệ thống phụ thuộc vào bố trí và thực hiện phân tích mạch thống kê. Ngoài ra, cũng bao gồm một thư viện ô và pad tiêu chuẩn với các tệp hỗ trợ cần thiết để cho phép đặt và định tuyến chip đầy đủ và xác minh cho các thiết kế hệ thống trên Chip.



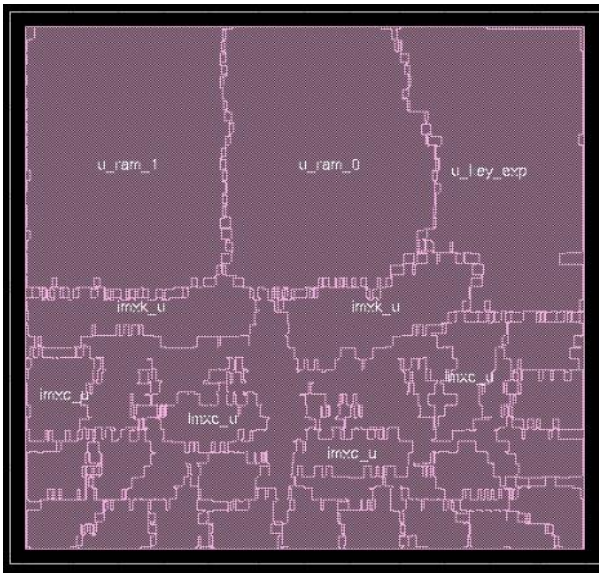
***Hình 5.** Thiết kế trong chế độ xem Floorplan*

Kết quả triển khai ở Hình 5 cho thấy thiết kế đang được hiển thị ở chế độ Floorplan View. Giúp quan sát cấu trúc bố trí tổng thể của các khối và tuyến dây trong thiết kế, bao gồm vị trí các module, các đường routing, và các cell. Thiết kế đang hoạt động ở mức hiệu suất cao với Timing Utilization = 96,7%. Vùng hình vuông chính giữa là nơi chứa các cell logic. Các đường kẻ ngang trong core là nơi các cổng logic tiêu chuẩn được đặt. Đường màu xanh dương nối từ vị trí cell đang được trỏ vào tới module tương ứng trong thiết kế cho biết cell được trỏ tới nằm trong module ram của thiết kế.

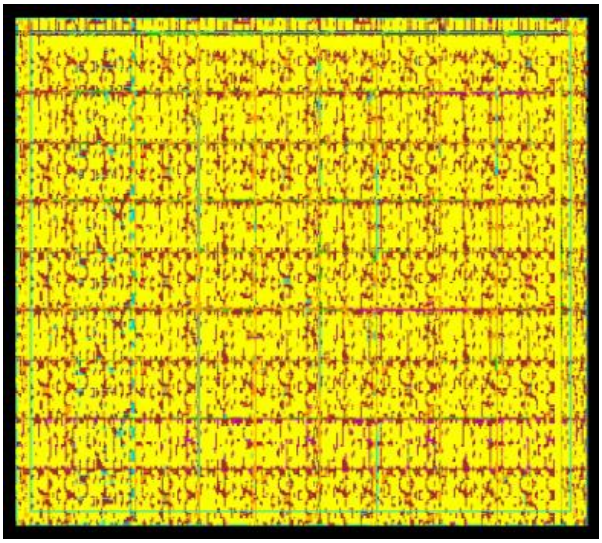
Bảng 2 là kết quả báo cáo thời gian của quá trình mã hóa/giải mã AES, thể hiện một đường truyền tín hiệu vi phạm yêu cầu về thời gian thiết lập trong thiết kế. Độ lệch có giá trị âm (-3.273 ns) cho thấy tín hiệu đến trễ hơn so với yêu cầu, dẫn đến vi phạm thiết lập.

Bảng 2. *Kết quả báo cáo thời gian*

Điểm bắt đầu	Điểm kết thúc	Thời gian yêu cầu	Thời gian đến	Độ lệch
i_enable	o_data_reg[88] /D	0.414	3.687	-3.273



Hình 6. *Thiết kế trong chế độ xem Amoeba*



Hình 7. *Kết quả layout AES trên SoC Encounter*

Hình 6 cho thấy kết quả về sắp xếp các module trong layout. Các module chiếm nhiều diện tích nhất là các module ram và module mở rộng khóa. Kết quả layout của khối AES được thể hiện trong Hình 7 sau khi thực hiện phủ đến lớp kim loại 10.

Bảng 3 cho biết số lượng và diện tích của các thành phần logic trong thiết kế. Diện tích tổng thể của thiết kế là 123510,374 đơn vị diện tích. Thành phần chiếm nhiều diện tích nhất là các cổng logic chiếm khoảng 48,5% diện tích. Các thanh ghi D Flip-Flop chiếm khoảng 26,5% tổng diện tích.Trong đó, khối mở rộng khóa sử dụng 7160 cell chiếm 19% tổng cell, khối RAM sử dụng 13751 cell chiếm 36,6% tổng cell, phần còn lại cho các khối thực thi thuật toán AES.

Bảng 3. *Kết quả tổng hợp Cell*

Kiểu	Số lượng	Diện tích sử dụng của AES	Diện tích sử dụng (%)
Phần tử tuần tự	3168	32708,333	26,5
Cổng NOT	9788	14149,864	11,5
Bộ đệm	7130	16763,865	13,6
Logic tổng hợp	17502	59888,312	48,5
Tổng	37588	123510,374	100,0

3.3. Đánh giá thiết kế AES trên các nền tảng phần cứng

Bảng 4 tổng hợp các kết quả đạt được khi triển khai thiết kế và so sánh với một công bố đã được khảo sát. Thực hiện mô phỏng quá trình mã hóa và giải mã đều hoàn thành trong 42 chu kỳ Clock (~840 ns) trong khi thử nghiệm thực tế trên board cần 50 chu kỳ Clock (~1000ns) cho quá trình mã hóa và giải mã. Nguyên nhân có sự chênh lệch về chu kỳ giữa mô phỏng và thực nghiệm của quá trình mã hóa và giải mã là do trễ vật lý cùng sự chuyển trạng thái không đồng thời của tín hiệu điều khiển o_data_valid. Mặt khác, do trong quá trình mô phỏng nhóm chỉ thực hiện với mô đun mã hóa và giải mã. Trong thực nghiệm có thiết kế thêm các khối giao tiếp ngoại vi.

Bảng 4. *Kết quả về tài nguyên chiếm dụng phần cứng AES*

Kết quả		Trên Quartus (Tần số 50Mhz)	Cyclone III DE0 EP3C16F 484C6	Công cụ SoC-Encounter	Công bố [8] (Tần số 200Mhz)
Chức năng	Mở rộng khóa	45 chu kỳ Clock	-	-	-
	Mã hóa (chu kỳ)	42 Clock (~840ns)	50 Clock (~1000ns)	-	20-30 clock (100ns-150ns)
	Giải mã (chu kỳ)	42 Clock (~840ns)	50 Clock (~1000ns)	-	30-40 clock (150-200ns)
Tài nguyên chiếm dụng của AES	Tổng số phần tử logic	4565	5106	17502	29917
	Tổng số thanh ghi	1676	2018	3168	9848
	Tổng số chân	80	11	523 (lõi IP AES256)	385 (lõi IP AES128)
	Công suất tiêu thụ	71,7mW	73,53mW	1,188W	289mW

Thiết kế chỉ sử dụng 11 cổng chiếm 3% số lượng cổng của thiết bị khi thử nghiệm thực tế. Tổng số phần tử logic là 5106 chiếm 33% tài nguyên của board và ước tính tiêu

thụ 73,53mW cho thấy lượng tài nguyên chiếm dụng là thấp khi so sánh với thiết kế của công bố [8].

Quá trình mô phỏng, khối AES được bao bọc bởi một module cấp cao để giảm thiểu số chân cần thiết còn 80 nhằm phù hợp với phần cứng được chọn. Số chân cần thiết khi thử nghiệm trên phần cứng ít hơn khi thực hiện mô phỏng do một module bao bọc thiết kế khối AES được thực hiện để giao tiếp giữa phần cứng với máy tính qua giao thức UART. Trong khi số chân cần thiết khi thực hiện layout lại tăng lên 523 chân do chỉ thực hiện layout cho thiết kế khối AES mà không có module bao bọc. Kết quả công suất ASIC trên chíp FPGA lớn, vì trong thiết kế nhóm đã sử dụng nhiều chân I/O, cụ thể công suất I/O chiếm 32% của tổng năng lượng tiêu thụ. Với mục tiêu trong nghiên cứu này chưa tính đến việc thiết kế I/O. Do đó, để triển khai trên thực tế, giải pháp có thể giảm số lượng I/O tối đa để có thể đưa công suất tiêu thụ về mức thấp hơn 1,188W.

4. Kết luận

Kết quả đạt được, hệ mật AES tiêu thụ tài nguyên hợp lý với mức năng lượng tiêu thụ là 73,53mW, đồng thời có thể hoạt động ổn định và hiệu quả việc mã hóa và giải mã dữ liệu trên chíp FPGA. Mặt khác, kết quả đạt được thiết kế SoC bằng công cụ SoC Encounter sử dụng 17502 phần tử logic, năng lượng tiêu tốn là 1,188W. Các kết quả này cho thấy thiết kế yêu cầu tài nguyên phần cứng thấp và phù hợp với các phần cứng có tài nguyên hạn chế. Tuy nhiên, thiết kế SoC cho hệ mật AES trong nghiên cứu còn ghi nhận một số vấn đề về thời gian trễ giữa các tín hiệu không đảm bảo yêu cầu. Trong các nghiên cứu tiếp theo, nhóm tác giả hướng tới thiết kế SoC của các nguyên thủy mật mã phù hợp với các thiết bị tài nguyên hạn chế với mục tiêu tối ưu hóa tín hiệu, phân bổ clock và công suất tiêu thụ thấp.

Lời cảm ơn: Các tác giả chân thành cảm ơn Học viện Kỹ thuật Mật mã đã hỗ trợ trong nghiên cứu này.

TÀI LIỆU THAM KHẢO

[1] N. Mouha, "Review of the advanced encryption standard", *National Institute of Standards and Technology Interagency or Internal Report 8319*, Jul. 2021. doi: 10.6028/NIST.IR.8319.

[2] D. Dangwal, M. Cowan, A. Alaghi, V. T. Lee, B. Reagen, and C. Trippel, "SoK: Opportunities for Software-Hardware-Security Codesign for Next Generation Secure Computing", in *Hardware and Architectural Support for Security and Privacy*, Oct. 2020, pp. 1–9. doi: 10.1145/3458903.3458911.

[3] S. F. Medeiros, "*The Schedulability of AES as a Countermeasure against Side Channel Attacks*", Springer-Verlag Berlin Heidelberg, 2012, pp. 16–31. doi: 10.1007/978-3-642-34416-9_2.

[4] J. Kala, J. Panda, and L. Tanwar, "FPGA Implementation Of a High Throughput Low Power Advanced Encryption Standard (AES-128) Cipher", in *2023 10th International Conference on Signal Processing and Integrated Networks (SPIN)*, Mar. 2023, pp. 367–372. doi: 10.1109/SPIN57001.2023.10116674.

[5] M. Deepa, B. Varssha, E. Abinaya, S. Ajitha, S. Dhaarani, and K. A. Sharmila, "An Efficient Implementation of AES Algorithm for Cryptography Using CADENCE", in *2023 7th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, Nov. 2023, pp. 1478–1484. doi: 10.1109/ICECA58529.2023.10395793.

[6] S. Islam, M. L. Ali, and R. Ruhana, "Design of a Low Power Double AES Crypto-Processor SoC", in *2021 IEEE International Conference on Telecommunications and Photonics (ICTP)*, Dec. 2021, pp. 1–5. doi: 10.1109/ICTP53732.2021.9744235.

[7] C. Duran and E. Roa, "A 10pJ/bit 256b AES-SoC Exploiting Memory Access Acceleration", *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 3, pp. 1612–1616, Mar. 2022, doi: 10.1109/TCSII.2021.3126984.

[8] N. S. S. Srinivas and M. Akramuddin, "FPGA based hardware implementation of AES Rijndael algorithm for Encryption and Decryption", in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, Mar. 2016, pp. 1769–1776. doi: 10.1109/ICEEOT.2016.7754990.

[9] P. Sikka, A. R. Asati, and C. Shekhar, "Speed optimal FPGA implementation of the encryption algorithms for telecom applications", *Microprocessors and Microsystems*, vol. 79, p. 103324, Nov. 2020, doi: 10.1016/j.micpro.2020.103324.

[10] Y. Han, X. Zou, Z. Liu, and Y. Chen, "Efficient DPA Attacks on AES Hardware Implementations", *International Journal of Communications, Network and System Sciences*, vol. 01, no. 01, pp. 68–73, 2008, doi: 10.4236/ijcns.2008.11010.