

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA ĐIỆN TỬ-VIỄN THÔNG

BÁO CÁO ĐỀ TÀI CẤP TRƯỜNG

Đề tài:

**TÌM HIỂU LÝ THUYẾT VỀ CPU DUAL
CORE VÀ MÔ PHỎNG MỘT SỐ
CHỨC NĂNG**

Mã đề tài: T2008 - 71

Chủ nhiệm đề tài:

ThS. Nguyễn Anh Vinh

Tham gia thực hiện:

TS. Huỳnh Hữu Thuận

CN. Tô Đình Thi



Tp. Hồ Chí Minh - 2010

GIỚI THIỆU ĐỀ TÀI

Trong thời đại ngày nay, khoa học kỹ thuật phát triển không ngừng để đáp ứng, phục vụ nhu cầu của con người. Cuộc sống con người ngày càng phong phú hơn nhờ những thiết bị công nghệ hỗ trợ : máy tính, điện thoại, tivi ... Bên cạnh đó để góp phần vào việc bảo vệ cuộc sống của con người, các nhà khoa học cần những cỗ máy hiện đại phân tích dữ liệu để đưa ra những cảnh báo về hiểm họa thiên nhiên như : sóng thần, động đất, núi lửa...

Máy tính là một phần không thể thiếu trong bất kì ngành công nghiệp nào. Chiếc máy tính với bộ xử lý CPU có thể giúp công việc tính toán nhanh hơn. Tuy nhiên để hỗ trợ tốt hơn các bộ vi xử lý ngày càng được cải tiến để đạt được tốc độ cao trong xử lý đồng thời giá thành cũng phải giảm. Chính vì vậy, những năm gần đây, chúng ta chứng kiến sự ra đời của nhiều bộ CPU có nhiều nhân do nhiều hãng sản xuất như: Intel, Amd...

Tìm hiểu về lý thuyết Dual core trong CPU là điều cần thiết, tạo tiền đề cơ bản cho việc phát triển những hệ thống CPU có nhiều core hơn. Do đó đề tài đã tiến hành xây dựng một hệ thống dual core kết hợp thuật toán RSA trong mã hóa dữ liệu sử dụng FPGA.

MỤC LỤC

GIỚI THIỆU ĐỀ TÀI

MỤC LỤC

CHƯƠNG 1 - TỔNG QUAN VỀ HỆ THỐNG DUAL CORE	1
1.1 Mở đầu.....	1
1.2 Kiến trúc vi xử lý Nios II	2
1.3 Avalon bus.....	3
1.4 Hệ thống nhiều vi xử lý	77
1.4.1 Khái niệm hệ thống nhiều vi xử lý	77
1.4.2 Hệ thống những vi xử lý tự trị	78
1.4.3 Hệ thống nhiều vi xử lý chia sẻ tài nguyên	79
1.4.3.1 Chia sẻ dữ liệu	79
1.4.3.2 Phản ứng Mutex core	80
1.4.3.2 Định vị chương trình của vi xử lý.....	80
CHƯƠNG 2 - THIẾT KẾ HỆ THỐNG DUALCORE KẾT HỢP THUẬT TOÁN RSA TẠO RA HỆ THỐNG MÃ HÓA TRÊN PHẦN CỨNG	22
2.1 Giới thiệu kít DE2	22
2.2 Khái quát chung về mã hóa	1
2.2.1 Định nghĩa mã hóa.....	1
2.2.2 Một số phương pháp mã hóa.....	1
2.3 Giới thiệu phương pháp mã hóa RSA	4
2.4 Thiết kế hệ thống phần cứng	77
CHƯƠNG 3 - KẾT QUẢ THỰC HIỆN TRÊN PHẦN CỨNG VÀ SO SÁNH VỚI PHẦN MỀM	85

3.1 Kết quả.....	85
3.2 Kết luận	88
TÀI LIỆU THAM KHẢO	93

CHƯƠNG 1 - TỔNG QUAN VỀ HỆ THỐNG DUAL CORE VÀ THUẬT TOÁN MÃ HÓA DỮ LIỆU.

1.1 Mở đầu

Với sự phát triển mạnh mẽ của khoa học kỹ thuật, máy tính ngày càng được gia tăng tốc độ xử lý của nó đồng thời chi phí cũng được giảm đáng kể. Sự quan sát này được thực hiện đầu tiên bởi Gordon Moore vào năm 1965, và thường được gọi là định luật Moore. Định luật phát biểu rằng : “Những tiến bộ của công nghệ sản xuất điện tử có khả năng tăng gấp đôi số transistor trên một khu vực đơn vị khoảng mỗi 12 đến 18 tháng”. Sự tiến bộ này khiến cho các hiện tượng tăng trưởng tốc độ máy tính và khả năng tiếp cận xảy ra trong hơn bốn thập kỷ qua. Chúng ta có thể tăng số lượng transistor bằng cách tạo ra những transistor nhỏ hơn và nhờ đó khoảng cách tín hiệu phải di chuyển trong những vi xử lý giảm xuống giúp tần số clock xử lý đạt được tốc độ cao. Điều này làm tăng hiệu năng hệ thống và giảm giá thành. Thế nhưng, định luật Moore gần đây bắt đầu có dấu hiệu không đúng vì những tác dụng phụ ngoài mong đợi.

Chúng ta luôn luôn mong muốn kết hợp được hiệu năng và tần số xử lý cao. Trong quá khứ, việc giảm kích thước của transistor có nghĩa là giảm khoảng cách giữa những transistor và tăng số lần chuyển mạch của transistor. Cùng với nhau, hai tác dụng đó đã góp phần đáng kể vào việc làm nhanh hơn tần số xử lý. Một cách khác để tần số xử lý có thể tăng là số transistor có khả năng thực hiện chức năng xử lý ví dụ như bộ cộng số nguyên có thể thực hiện trong nhiều phương pháp. Một phương pháp sử dụng rất ít transistor, nhưng thời gian thực hiện là rất lâu. Một phương pháp khác có thể rút ngắn thời gian nhưng cần nhiều transistor hơn. Vì vậy có nhiều transistor hơn sẽ cho phép thực hiện những tác vụ phức tạp hơn mà vẫn nhanh hơn.

Thế nhưng vẫn có những vấn đề khác cần quan tâm. Khi tần số xử lý lên cao, thì nhiệt sinh ra cũng tăng theo nó. Khả năng làm mát bộ vi xử lý đã trở thành một nhân tố chính hạn chế tốc độ của bộ vi xử lý. Điều này được bù đắp phần nào bởi giảm kích thước transistor vì những transistor nhỏ hơn có thể hoạt động ở mức điện

thế thấp và cho phép sản sinh ít nhiệt hơn. Thật không may, những transistor bây giờ là quá nhỏ, những hoạt động của chúng có thể bị ảnh hưởng bởi khả năng lưỡng tử của các điện tích electron. Theo cơ học lượng tử, những phần tử nhỏ như electron có thể tự phát đường hầm một cách ngẫu nhiên trên một khoảng cách ngắn. Mà các transistor cơ bản và emitter đủ gần đến nỗi chỉ cần một số electron cũng có thể tạo đường hầm tạo ra một dòng rò rỉ giữa chúng làm hiệu hụt transistor. Nếu điện thế quá thấp, điểm khác biệt giữa logic 1 và 0 quá ít có thể tạo đường hầm lưỡng tử và vi xử lý sẽ không hoạt động. Như vậy chúng ta có thể tăng số lượng transistor trên một đơn vị nhưng tần số xử lý phải giảm xuống để có khả năng giữ mát vi xử lý.

Vấn đề làm mát làm cho việc thiết kế vi xử lý trong thế tiến thoái lưỡng nan. Thị trường mong đợi các thế hệ vi xử lý sau sẽ nhanh hơn các vi xử lý trước. Nhưng cơ học lưỡng tử và hằng số nhiệt gây ra nhiều khó khăn không thể giải quyết được. Mặt khác các thế hệ sau này cũng sẽ có nhiều transistor hơn để làm việc và chúng cũng yêu cầu ít năng lượng. Công nghệ sản xuất bây giờ đạt đến mức có thể đặt hai lõi vi xử lý trên cùng một chip – một bộ xử lý lõi kép. Mặc dù có thể mỗi vi xử lý trong lõi kép chậm hơn bộ vi xử lý đơn bình thường nhưng khi chúng kết hợp lại thành lõi kép thì sẽ nhanh hơn. Vì vậy các thế hệ sau sẽ tăng số core và giảm tần số xử lý.

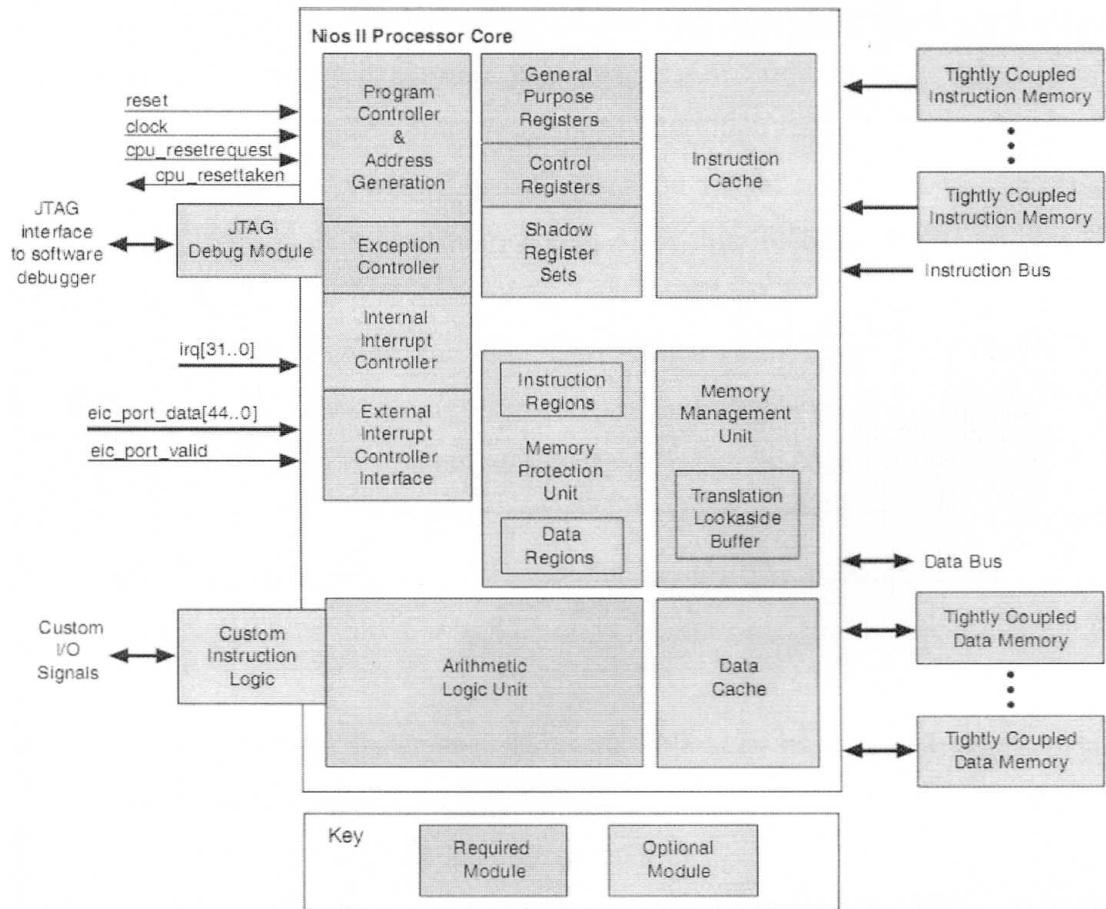
Do đó mục tiêu của đề tài là tìm hiểu mô hình của hệ thống sử dụng dual core kết hợp một ứng dụng mã hóa dữ liệu thực hiện trên FPGA. Để rút ngắn quá trình thiết kế, đề tài đã kế thừa một số thành tựu của Altera như để kết hợp các CPU NIOS tạo nên một hệ thống dual core.

1.2 Kiến trúc vi xử lý Nios

Kiến trúc Nios II bao gồm những đơn vị chức năng sau:

- Register file
- Arithmetic logic unit (ALU)
- Interface to custom instruction logic
- Exception controller
- Internal or external interrupt controller
- Instruction bus

- Data bus
- Memory management unit (MMU)
- Memory protection unit (MPU)
- Instruction and data cache memories
- Tightly-coupled memory interfaces for instructions and data
- JTAG debug module



Reister file: chứa đựng 32 thanh ghi interger mục đích chung 32 bit và 32 thanh ghi điều khiển 32 bit. Kiến trúc hỗ trợ chế độ giám sát và người dùng sử dụng mã hệ thống để bảo vệ những thanh ghi điều khiển khỏi những ứng dụng sai.

Arithmetic logic unit (ALU): hoạt động trên dữ liệu được lưu trữ trong những thanh ghi mục đích chung. Chúng lấy một hoặc hai input từ những thanh ghi đó và lưu lại kết quả. ALU hỗ trợ các phép toán số học, phép quan hệ, logic ...

- Bảo vệ bộ nhớ
- Gồm 32 khu vực lệnh và 32 khu vực dữ liệu
- Tổng khu vực bộ nhớ được định nghĩa bởi kích thước hoặc địa chỉ giới hạn trên
- Quyền truy suất đọc hoặc ghi đến khu vực dữ liệu
- Quyền cho phép thực hiện cho khu vực lệnh

Tightly-coupled memory : cung cấp việc truy suất bộ nhớ ít trễ cho những ứng dụng yêu cầu hiệu suất cao, đồng thời còn có những chức năng sau:

- Hiệu suất tương tự bộ nhớ cache
- Phần mềm có thể cảnh báo những đoạn code hoặc data không tốt.

Jtag debug module : cung cấp những chức năng trên chip để điều khiển vi xử lý Nios một cách từ xa từ máy tính. Những công cụ debug phần mềm dựa trên PC giao tiếp với Jtag debug module để thực hiện một số tác vụ sau :

- Download chương trình đến bộ nhớ
- Bắt đầu và dừng thực hiện lệnh
- Thiết lập breakpoint và watchpoint
- Phân tích những thanh ghi và bộ nhớ

1.3 Avalon bus

Avalon bus là một kiến trúc bus đơn giản, được thiết kế để kết nối các vi xử lý nhúng và các thiết bị ngoại vi trong một hệ-thống-trên-chip-khả-trình (SOPC). Avalon bus là một giao tiếp mà đặc tả các kết nối cổng giữa các thiết bị master và slave, đồng thời đặc tả thời gian liên lạc giữa các thiết bị.

Mục đích thiết kế chính của Avalon bus là:

- Đơn giản: cung cấp một giao thức dễ hiểu, dễ sử dụng.
- Tối ưu hóa việc sử dụng tài nguyên cho các bus logic: bảo toàn các yếu tố logic (logic element) bên trong thiết bị logic khả trình (PLD).
- Đồng bộ hóa các quá trình hoạt động: vừa tích hợp tốt với các logic của người dùng cùng tồn tại trên một PLD, lại vừa tránh được các vấn đề phân tích thời gian phức tạp.

Interface to custom instruction logic: ALU kết nối trực tiếp đến custom instruction logic, cho phép chúng ta thực hiện những phép toán được truy xuất và sử dụng như những lệnh tư nhiên.

Exception controller: dùng để xử lý tất cả các tín hiệu ngoại lệ. Ví dụ những tín hiệu ngắt phần cứng bên trong làm cho bộ vi xử lý dịch chuyển lệnh đến một địa chỉ ngoại lệ. Một bộ xử lý ngoại lệ tại địa chỉ này sẽ xác định nguyên nhân gây ra sự ngoại lệ và dò ra một chương trình con thích hợp để tiếp tục thực hiện.

External interrupt controller interface (EIC): được sử dụng kết hợp với những thanh ghi để cung cấp những ngắt phần cứng hiệu suất cao. Nios kết nối với EIC thông qua giao diện EIC. Khi EIC đang tồn tại thì những internal interrupt controller (bộ điều khiển ngắt trong) không được thực hiện, và SOPC builder kết nối những ngắt đến EIC. EIC chọn những ngắt đang hoạt động và miêu tả một ngắt cho vi xử lý Nios II đồng thời gửi địa chỉ xử lý ngắt và thông tin chọn lựa thiết lập thanh ghi. Thuật toán chọn lựa ngắt dựa vào quyền ưu tiên ngắt. Như vậy EIC miêu tả một cấp độ ngắt, Nios II sử dụng những cấp độ ngắt này để xác định khi nào phục vụ ngắt.

Internal interrupt controller: được hỗ trợ lên đến 32 cái. Core vi xử lý có 32 cấp độ ngắt (IRQ) từ irq0 đến irq32, cung cấp một ngắt duy nhất cho mỗi nguồn ngắt.

Instruction and data bus: kiến trúc Nios II hỗ trợ bus data và bus instruction riêng lẽ. Cả hai bus này thực hiện như port Avalon-master. Port master data kết nối đến cả thành phần bộ nhớ và thiết bị ngoại vi, trong khi port master instruction chỉ kết nối thành phần bộ nhớ.

Memory management unit: cung cấp những chức năng và đặc điểm sau:

- Kết nối địa chỉ ảo đến địa chỉ vật lý.
- Bảo vệ bộ nhớ.
- 32 bit địa chỉ ảo và vật lý, kết nối không gian 4GByte địa chỉ ảo vào 4GByte địa chỉ vật lý.
- 512MByte không gian địa chỉ vật lý để truy suất trực tiếp

Memory projection unit : gồm những chức năng sau

Avalon bus hỗ trợ bus đa chủ. Các kiến trúc đa chủ này tạo ra một sự linh hoạt rất lớn trong việc xây dựng các hệ thống SOPC, và tạo điều kiện cho các thiết bị ngoại vi có băng thông rộng. Ví dụ, một thiết bị ngoại vi chủ có thể thực hiện một quá trình truyền truy-xuất-bộ-nhớ-trực-tiếp (DMA) để truyền dữ liệu từ thiết bị ngoại vi vào bộ nhớ mà không cần một vi xử lý trong đường truyền dữ liệu.

Các master và slave giao tiếp với nhau dựa trên một kĩ thuật gọi là thỏa hiệp ở phía slave (slave-side arbitration). Kĩ thuật này xác định master nào được nắm quyền giao tiếp với slave trong trường hợp nhiều master cùng truy cập đến một slave. Kĩ thuật này có hai lợi điểm:

- Chi tiết về sự thỏa hiệp này được gói gọn bên trong Avalon bus. Do đó, các giao tiếp master và slave rất chắc, bất chấp số lượng các master và slave có trên bus. Mỗi bus master sẽ giao tiếp với Avalon bus như thể nó là master duy nhất trên đường Avalon bus.
- Các master có thể thực hiện quá trình truyền bất kì lúc nào, miễn là chúng không truy cập cùng một slave trong cùng một chu kì bus.

Avalon bus được thiết kế để thích nghi với môi trường SOPC, do đó nó là một kiến trúc bus tích hợp (on-chip) rất năng động, bao gồm các tài nguyên logic và truyền bên trong một thiết bị logic khả trình. Một vài nét chính của kiến trúc Avalon là:

- Giao tiếp với các thiết bị ngoại vi đồng bộ với xung clock Avalon. Do đó, sự sắp xếp các tín hiệu báo và tín hiệu bắt tay không đồng bộ và phức tạp là điều không cần thiết. Hoạt động của Avalon bus (và cũng là của toàn hệ thống) có thể được phân tích bằng các kĩ thuật phân tích chuẩn và đồng bộ thời gian.
- Tất cả các tín hiệu đều hoạt động ở LOW hoặc HIGH, điều này cho phép sự xoay vòng của hệ thống. Các bộ đa hợp bên trong Avalon bus xác định tín hiệu nào điều khiển thiết bị ngoại vi nào. Các thiết bị ngoại vi sẽ không cần phải có ngõ ra 3-trạng-thái ngay cả khi thiết bị không được chọn hoạt động.

- Các tín hiệu địa chỉ, dữ liệu và điều khiển sử dụng các cổng riêng biệt, điều này làm đơn giản hóa thiết kế của các thiết bị ngoại vi. Một thiết bị ngoại vi không cần thiết phải giải mã các chu kỳ bus dữ liệu và địa chỉ, và cũng không cần phải tắt các ngõ ra khi nó không được chọn hoạt động.
- Không gian địa chỉ lên tới 4Gbyte: các thiết bị ngoại vi và bộ nhớ có thể được gắn bất kì nơi nào trong không gian địa chỉ 32-bit.
- Tích hợp bộ giải mã địa chỉ: Avalon bus sẽ tự động tạo các tín hiệu Chip Select cho tất cả các thiết bị ngoại vi, rất thuận tiện trong việc thiết kế các thiết bị ngoại vi để giao tiếp với Avalon.
- Điều chỉnh kích thước bus một cách tự động: Avalon bus tự động quản lý chi tiết dữ liệu truyền giữa các thiết bị ngoại vi có kích thước dữ liệu không trùng khớp, cho phép các thiết bị ngoại vi với kích thước dữ liệu khác nhau có thể liên lạc dễ dàng.

❖ Avalon bus module

Avalon bus module là xương sống của một module hệ thống. Đó là con đường chính để giao tiếp giữa các thành phần thiết bị ngoại vi trong một thiết kế SOPC. Avalon bus module là một tập hợp các tín hiệu điều khiển, dữ liệu, địa chỉ và các logic thỏa hiệp mà kết nối các thành phần tạo nên hệ thống. Avalon bus module thực hiện một kiến trúc bus có thể cấu hình được, có nghĩa là có thể thay đổi để phù hợp với nhu cầu kết nối các thiết bị ngoại vi của người thiết kế.

Avalon bus module được tạo một cách tự động bởi SOPC Builder, vì thế người thiết kế hệ thống có thể tiết kiệm thời gian trong việc nối các thiết bị với nhau.

Avalon bus module sẽ cung cấp các hỗ trợ sau cho những thiết bị ngoại vi kết nối tới đường bus này:

- Đa hợp đường dữ liệu: Các bộ đa hợp bên trong module Avalon bus sẽ truyền dữ liệu từ thiết bị slave được chọn đến thiết bị master thích hợp.
- Giải mã địa chỉ: bộ giải mã địa chỉ của module này sẽ tạo ra các tín hiệu chip select cho mỗi thiết bị ngoại vi, như vậy các thiết bị ngoại vi không

cần giải mã đường địa chỉ để tạo ra tín hiệu chip select → đơn giản hóa các thiết kế của thiết bị ngoại vi.

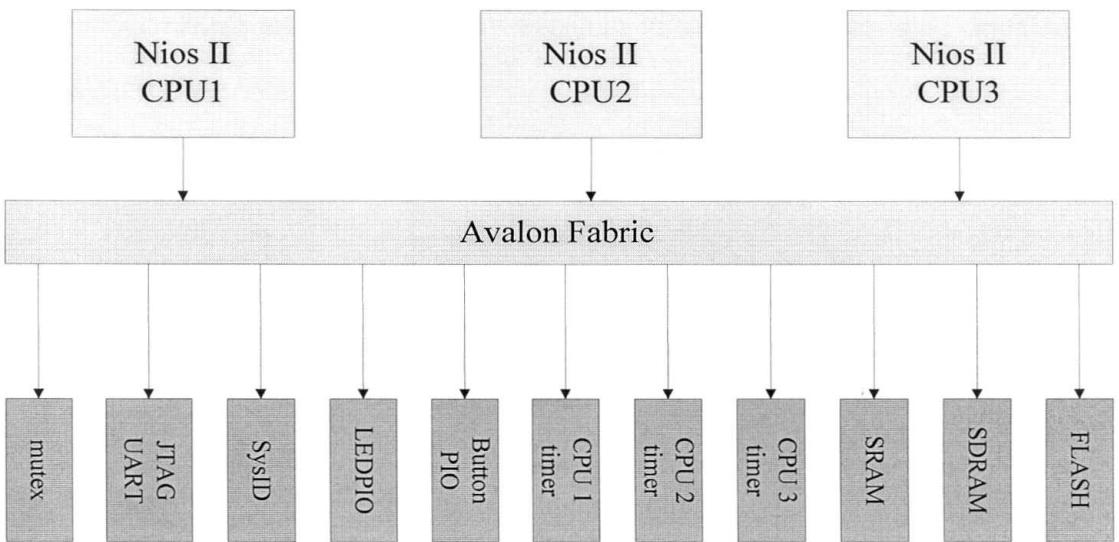
- Tạo trạng thái chờ: bộ tạo trạng thái chờ sẽ mở rộng thêm các quá trình truyền bằng một hay nhiều chu kỳ bus, điều này cần thiết cho một số thiết bị hoặc quá trình đòi hỏi sự đồng bộ đặc biệt. Trạng thái này có thể được tạo ra để dùng một thiết bị master trong trường hợp thiết bị slave không thể đáp ứng trong một chu kỳ bus.
- Phân công ưu tiên ngắn: khi một hay nhiều thiết bị slave tạo ra các tín hiệu ngắn, Avalon bus module sẽ chuyển các ngắn (theo thứ tự ưu tiên) đến cá thiết bị master kèm theo số yêu cầu ngắn (IRQ number) thích hợp.
- Khả năng trì hoãn các dữ liệu: các logic cần để thực hiện việc truyền với độ trễ mong muốn giữa các cặp master-slave đã được tích hợp sẵn trong Avalon bus module.

1.4 Hệ thống nhiều vi xử lý Nios

1.4.1 Khái niệm hệ thống nhiều vi xử lý

Một hệ thống mà có hai hoặc ba vi xử lý trở lên cùng thực hiện một hoặc nhiều nhiệm vụ được gọi là hệ thống nhiều vi xử lý.

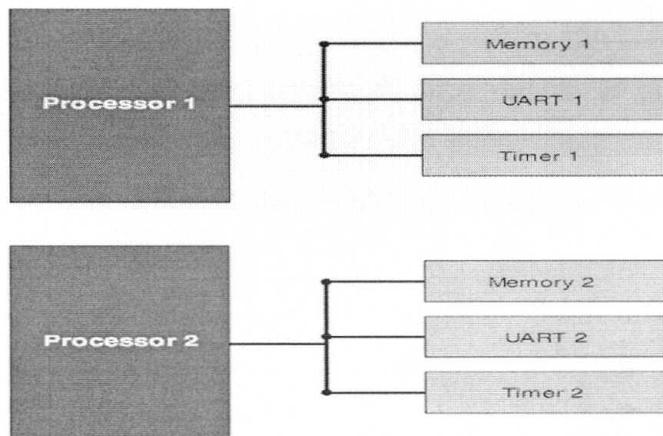
Trên thực tế, có rất nhiều ứng dụng phức tạp đòi hỏi nhiều vi xử lý cùng lúc hoặc để tăng hiệu suất hệ thống như thời gian, tính hiệu quả. Sự kết hợp của nhiều vi xử lý làm cho hệ thống thêm phức tạp, giá thành cao. Chính vì vậy, việc đưa nhiều vi xử lý vào hệ thống nhúng sẽ giảm được giá thành sản phẩm trong khi tính hiệu quả vẫn không suy giảm.



Hình 2.40 : Hệ thống gồm nhiều vi xử lý và các thiết bị ngoại vi

1.4.2 Hệ thống những vi xử lý tự trị

Hệ thống này có nhiều vi xử lý, mỗi vi xử lý làm một nhiệm vụ riêng lẻ và không làm ảnh hưởng tới những vi xử lý khác. Những hệ thống kiểu này ít phức tạp, được thiết kế dễ dàng hơn.

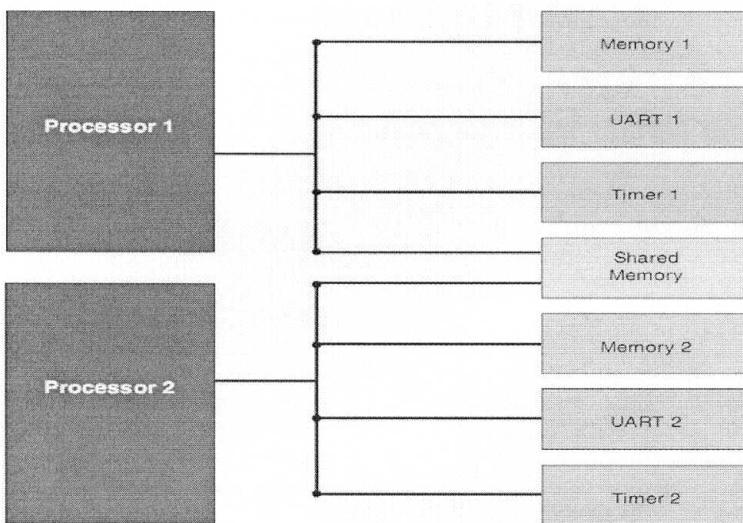


Hình 2.48 : Hệ thống những vi xử lý tự trị

1.4.3 Hệ thống nhiều vi xử lý chia sẻ tài nguyên

Đối với những thiết bị nhúng, tài nguyên là hạn chế nhất là thiếu bộ nhớ. Vì vậy khả năng chia sẻ tài nguyên cho nhiều vi xử lý là cần thiết. Nhưng việc chia sẻ những tài nguyên này cho nhiều bộ vi xử lý làm cho thiết kế phức tạp hơn. Việc nhiều vi xử lý cùng truy xuất vào bộ nhớ cùng một thời điểm là không thể.

Việc những chương trình chạy trên mỗi vi xử lý cũng cần phải được đặt ở những vị trí thích hợp trong bộ nhớ.



Hình 2.49 : Hệ thống nhiều vi xử lý có chung bộ nhớ

Lập trình cho những vi xử lý này cũng cần phải cẩn thận để tránh tình trạng xảy ra xung đột.

1.4.3.1 Chia sẻ dữ liệu

Nếu một thành phần bộ nhớ chứa đựng chương trình lập trình cho hơn một vi xử lý. Mỗi vi xử lý có bộ nhớ chia sẻ phải được yêu cầu sử dụng những không gian riêng để chứa code thực hiện. Mỗi vi xử lý không được chia sẻ không gian chứa code lập trình nghĩa là chúng phải có những khu vực sở hữu .text, .rodata, .rwdta, .heap và .stack duy nhất.

Nếu một vi xử lý viết dữ liệu đến vùng nhớ chia sẻ cùng lúc với một vi xử lý khác cũng đang viết hoặc đọc vùng nhớ đó, hiện tượng dữ liệu bị sai sẽ xuất hiện làm cho hệ thống bị lỗi. Chính vì vậy cần phải có một cơ chế để cảnh báo cho một vi xử lý biết bộ nhớ chia sẻ có đang được sử dụng hay không.

1.4.3.2 Phần cứng Mutex core

Mutex core không phải một chức năng có sẵn trong vi xử lý NIOS mà là một thành phần phần cứng nhỏ được thiết kế hỗ trợ trong SOPC.

Những môi trường nhiều vi xử lý có thể sử dụng mutex core với giao tiếp Avalon bus để sắp xếp việc truy suất bộ nhớ chia sẻ. Mutex core cung cấp một giao thức chắc chắn bảo đảm sự sở hữu duy nhất một cách luân phiên các bộ nhớ.

Mutex core cung cấp hoạt động tự kiểm tra và thiết lập hoàn toàn bằng phần cứng, cho phép phần mềm trong mỗi vi xử lý xác định vi xử lý nào đang sở hữu mutex cũng như bộ nhớ.

Cấu trúc phần cứng Mutex core như sau :

Table 26-1. Mutex Core Register Map

Offset	Register Name	R/W	Bit Description		
			31 ... 16	15 ... 1	0
0	mutex	RW	OWNER	VALUE	
1	reset	RW	-	-	RESET

Bảng 2.5 : Các thanh ghi của mutex core

Mutex core có những khả năng cơ bản sau :

- Nếu lớp VALUE có giá trị 0x0000, mutex không bị khóa. Ngược lại nếu nó khác không, mutex bị khóa.
- Thanh ghi của mutex luôn luôn có khả năng được đọc. Vì vậy mỗi vi xử lý đều có thể đọc những thanh ghi này để xác định trạng thái chính xác.
- Thanh ghi mutex chỉ được ghi khi những điều kiện sau đây thỏa :
 - + Giá trị VALUE phải là zezo.
 - + Lớp OWNER phải đang trong trạng thái ghi dữ liệu.
- Một vi xử lý có gắng yêu cầu mutex bởi việc ghi chỉ số ID của nó đến lớp OWNER, và ghi giá trị khác không đến lớp VALUE. Vi xử lý sau đó có thể kiểm tra việc yêu cầu có thành công bằng cách xác định lớp OWNER.

Cách lập trình :

Mutex là một phần cứng đã được xây dựng một số hàm tích hợp trong thư viện giúp người lập trình dễ dàng sử dụng. Sau đây là một số hàm cơ bản

Bảng 2.6 : Bảng chức năng một số hàm điều khiển mutex

Tên hàm	Chức năng

Alter_avalon_mutex_open()	Yêu cầu bắt tay đến thiết bị mutex, khả năng để cho phép các hàm khác truy suất thiết bị mutex
Altera_avalon_mutex_trylock()	Cố gắng thực hiện việc khóa mutex. Trả lại ngay lập tức giá trị 0 nếu khóa thành công
Altera_avalon_mutex_lock()	Khóa mutex và không trả lại giá trị 0 nếu vẫn chưa khóa được mutex
Altera_avalon_mutex_unlock()	Bỏ khóa mutex
Altera_avalon_mutex_is_mine()	Xác định nếu CPU đang sở hữu mutex
Altera_avalon_mutex_first_lock()	Kiểm tra khi nào mutex được giải phóng vì reset

Ví dụ về cách lập trình sử dụng mutex core:

```
#include <altera_avalon_mutex.h>
/* bắt tay với thiết bị mutex*/
alt_mutex_dev* mutex = altera_avalon_mutex_open( "/dev/mutex" );
altera_avalon_mutex_lock( mutex, 1 );/* yêu cầu mutex, thiết lập giá trị 1*/
/*
 * truy xuất bộ nhớ chia sẻ ở đây
 */
altera_avalon_mutex_unlock( mutex );/* giải phóng mutex*/
```

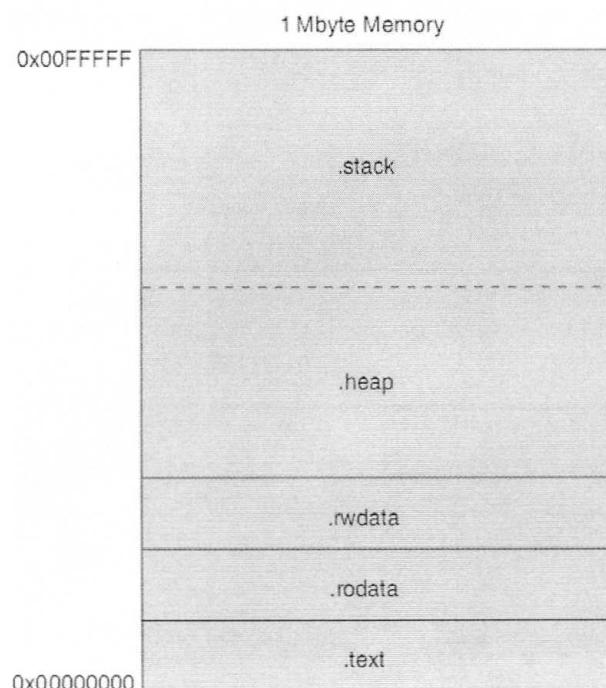
1.4.3.3 Định vị chương trình của vi xử lý

Khi tạo ra một hệ thống nhiều vi xử lý, chúng ta phải chạy phần mềm đặt trong cùng một bộ nhớ vật lý bên ngoài (memory_off_chip). Phần mềm cho mỗi vi xử lý phải được đặt riêng biệt trong khu vực bộ nhớ đó. Thủ tượng tưởng, một hệ thống gồm hai vi xử lý, mà phần mềm của chúng đều đặt trong SDRAM. Phần mềm của vi xử lý đầu tiên yêu cầu 128 Kbytes bộ nhớ, và phần mềm của vi xử lý thứ hai yêu cầu 64 Kbytes bộ nhớ. Như vậy vi xử lý thứ nhất có thể sử dụng khu vực bộ

nhớ từ 0x0 đến 0xFFFF trong SDRAM, và vi xử lý thứ hai có thể sử dụng khu vực từ 0x20000 đến 0x2FFF.

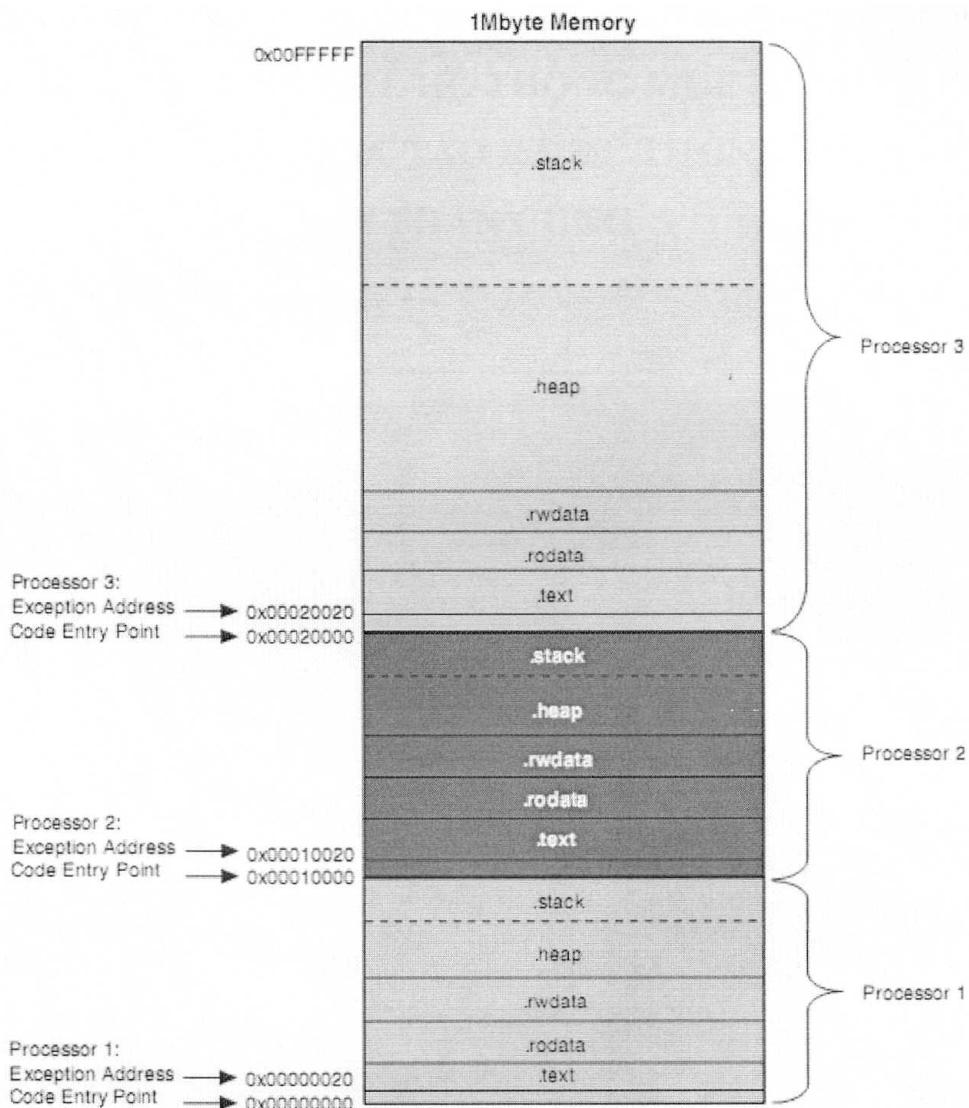
Mỗi vi xử lý bất kì đều có năm khu vực chứa code :

- .text : khu vực code thực hiện.
- .rodata : vùng dữ liệu chỉ đọc trong việc thực hiện code.
- .rwdtata : nơi mà những tham số và con trỏ có khả năng đọc và ghi.
- .heap : nơi vùng nhớ động được định vị.
- .stack :chứa đựng hàm con và một số dữ liệu tạm thời.



Hình 2.50: Tổ chức chương trình trong vi xử lý.

Đối với một hệ thống nhiều vi xử lý, ranh giới địa chỉ chứa code là nơi kết thúc của vi xử lý này và bắt đầu cho vi xử lý mới. Ví dụ SDRAM có địa chỉ từ 0x0 đến 0xFFFF và mỗi vi xử lý A, B, và C yêu cầu 64 Kbytes SDRAM để chạy phần mềm của chúng, thì sự sắp xếp vùng chương trình được trình diễn trong hình.



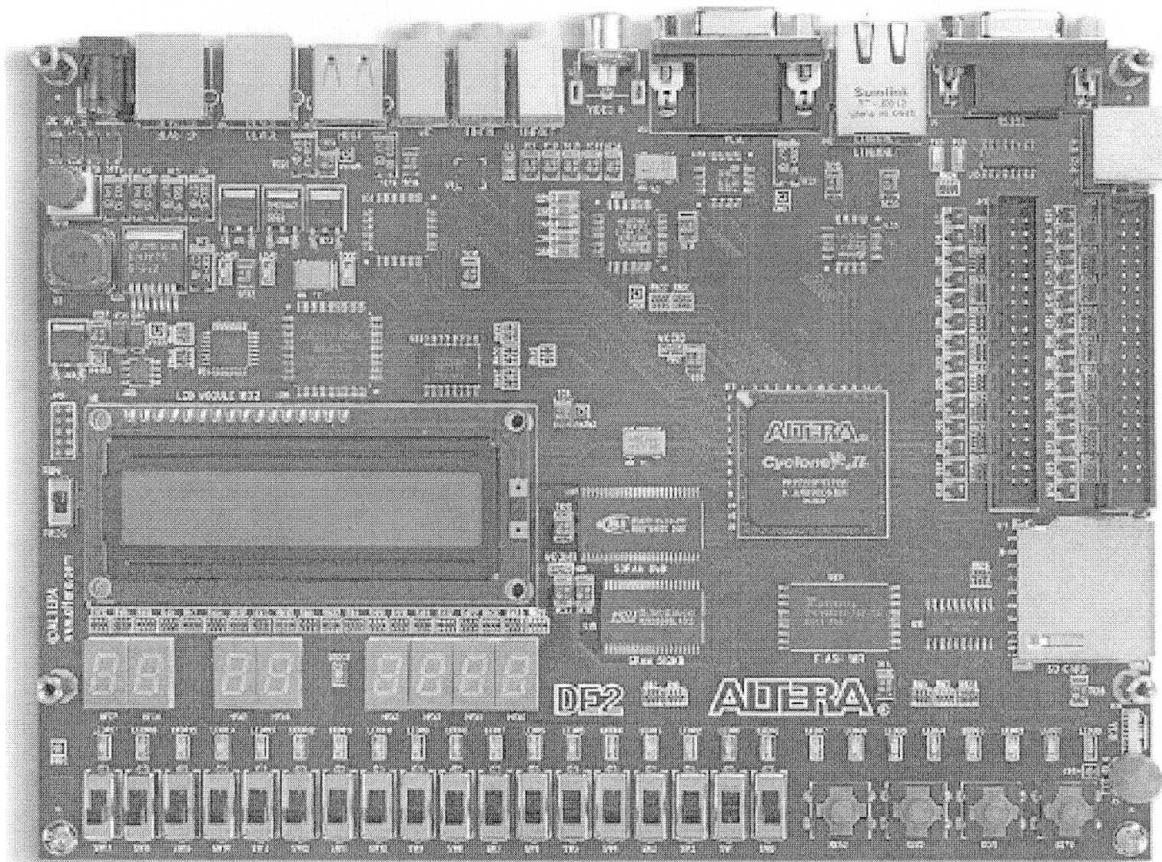
Hình 2.51: Sắp xếp chương trình mỗi vi xử lý trong SDRAM

Vùng địa chỉ ngoại lệ (address exception) luôn luôn được thiết lập là 0x20. Offset 0x0 là nơi mà vi xử lý phải chạy code reset của nó. Offset 0x20 được thiết lập như bộ nhớ đệm lệnh để bắt đầu đưa đến vùng code thực hiện.

CHƯƠNG 2 - THIẾT KẾ HỆ THỐNG MULTICPU KẾT HỢP THUẬT TOÁN RSA TẠO RA HỆ THỐNG MÃ HÓA TRÊN PHẦN CỨNG

2.1 Giới thiệu kít DE2

- ❖ Kit DE2:



Hình 2.0: Kit DE2

Kit DE2 cung cấp một môi trường phát triển các ứng dụng số hoàn thiện cho các kỹ sư thiết kế. Kit hỗ trợ toàn bộ quá trình thiết kế từ ý tưởng thiết kế cho đến thực hiện phần cứng. Kit bao gồm một board mạch sử dụng FPGA Stratix II, công cụ phát triển DSP Builder, phần mềm Quartus II, MATLAB/Simulink, các lõi IP.

- ❖ Các thành phần chức năng:

- Thiết bị xuất nhập
 - + 10/100 Ethernet, RS-232, cổng hồng ngoại
 - + Video Out (VGA 10-bit DAC)
 - + Video In (NTSC/PAL/Multi-format)
 - + USB 2.0 (type A and type B)
 - + PS/2 và port bàn phím
 - + Line-in, Line-out, microphone-in (24-bit audio CODEC).
 - + 76 chân mở rộng.
- Hệ thống nhớ:
 - + Một bộ nhớ SRAM dung lượng 512KB
 - + Một bộ nhớ flash dung lượng 4 Mbyte được cấu hình như một bus 8-bit
 - + Một bộ nhớ SDRAM gồm hai chip, tổng dung lượng 8 Mbyte và được cấu hình như một bus 64-bit
 - + Một khe cắm SDCARD
- Những thành phần khác:
 - + 18 switch
 - + Cấu hình bằng cách tải dữ liệu sử dụng cáp tải USB Blaster
 - + 18 led đỏ, 9 led xanh, 8 led 7 đoạn.
 - + Bốn nút nhấn
 - + 16 x 2 LCD
 - + Tám đèn led cho người dùng tự định nghĩa
 - + Một bộ dao động 27-MHz và 50 - MHz
 - + Nguồn cấp 16-V DC

2.2 Khái quát chung về mã hóa

2.2.1 Định nghĩa về mã hóa

Mã hóa (Encryption) là một kỹ thuật bảo mật làm biến đổi dữ liệu từ hình thức đơn giản, rõ ràng sang hình thức dữ liệu biến thành mật mã. Chỉ những ai có được những thông tin giải mã cần thiết mới có thể giải mã và đọc nội dung dữ liệu.

Mã hóa có thể là một chiều, có nghĩa là mã hóa được thiết kế để ẩn các dữ liệu thật đi và không bao giờ có thể giải mã được. Hoặc có thể là hai chiều chuỗi kí tự được mã hóa có thể chuyển lại thành dữ liệu thực.

Encryption được dùng như một kỹ thuật nhằm nâng vấn đề bảo mật lên một cấp độ quan trọng. Encryption hỗ trợ sự cần mật bởi vì nó bảo vệ dữ liệu chống các truy cập bất hợp pháp. Nó hỗ trợ tính tích hợp bởi vì rất khó có thể làm xáo trộn hay thay đổi dữ liệu mã hóa mà không bị phát hiện

2.2.2 Một số thuật toán mã hóa

Thuật toán mã hóa là một quy tắc, một hệ thống hoặc một cơ chế được sử dụng để mã hóa dữ liệu. Các thuật toán có thể là sự thay thế mang tính máy móc khá đơn giản, nhưng trong mã hóa thông tin điện tử, nhìn chung sử dụng những hàm toán học cực kì phức tạp. Thuật toán càng mạnh, càng phức tạp thì càng khó giải mã.

Khóa (key) : khóa mã hóa là một mẫu thông tin đặc biệt được kết hợp với thuật toán để thi hành mã hóa và giải mã. Mỗi khóa khác nhau có thể “chế tạo” ra các văn bản mã hóa khác nhau, và nếu không chọn đúng khóa thì không thể mở được dữ liệu đã mã hóa, cho dù biết được mã hóa văn bản dùng thuật toán gì. Sử dụng khóa càng phức tạp, mã hóa càng mạnh.

Mã hóa hàm băm (Hashing Encryption) : là cách thức mã hóa một chiều tiến hành biến đổi văn bản nhận dạng trở thành hình thái mã hóa mà không bao giờ có thể giải mã. Kết quả của quá trình hashing gọi là một hash (xử lý băm), giá trị hash (hash value) hay thông điệp đã được tiêu hóa (message digest) và tất nhiên không thể tái tạo lại dạng ban đầu. Hashing được sử dụng trong một số một hình xác thực mật khẩu. Một số thuật toán hashing như : Message Digest 5 (MD5), Secure Hash Algorithm (SHA)...

Mã hóa đối xứng (Symmetric Encryption) : hay mã hóa chia sẻ khóa là mô hình mã hóa hai chiều có nghĩa là tiến trình mã hóa và giải mã dùng chung một khóa. Khóa này phải được bí mật chuyển giao giữa hai đối tượng tham gia giao tiếp. Khóa này có thể được cấu hình trong phần cứng hoặc phần mềm. Một số thuật toán

mã hóa đối xứng : Data Encryption Standard (DES), Triple DES (3DES), Advanced Encryption Standard (AES), Rivest Cipher (RC)...

Mã hóa bất đối xứng hay mã hóa công khai (public-key encryption) là mô hình mã hóa hai chiều sử dụng một cặp khóa là khóa riêng (private key) và khóa công (public key). Thông thường, một thông điệp được mã hóa với private key, và chắc chắn key này là của người gửi thông điệp. Nó sẽ được giải mã với public key bất cứ người nhận nào cũng có thể truy cập nếu họ có key này. Một số thuật toán mã hóa bất đối xứng: Rivest Shamir Adelman (RSA), Diffie-Hellman, Elgamal, Paillier Cryptosystem...

2.3 Giới thiệu phương pháp mã hóa dùng RSA

Thuật toán RSA được đặt tên theo 3 người phát minh là Ronald Rivest, Adi Shamir và Len Adelman có hai khóa: khóa công khai (hay khóa công cộng) và khóa bí mật (hay khóa cá nhân). Mỗi khóa là những số cố định sử dụng trong quá trình mã hóa và giải mã. Khóa công khai được công bố rộng rãi cho mọi người và được dùng để mã hóa. Những thông tin được mã hóa bằng khóa công khai chỉ có thể được giải mã bằng khóa bí mật tương ứng. Nói cách khác, mọi người đều có thể mã hóa nhưng chỉ có người biết khóa cá nhân (bí mật) mới có thể giải mã được.

Ta có thể mô phỏng trực quan một hệ mật mã khoá công khai như sau : Bob muốn gửi cho Alice một thông tin mật mà Bob muốn duy nhất Alice có thể đọc được. Để làm được điều này, Alice gửi cho Bob một chiếc hộp có khóa đã mở sẵn và giữ lại chìa khóa. Bob nhận chiếc hộp, cho vào đó một tờ giấy viết thư bình thường và khóa lại (như loại khoá thông thường chỉ cần sập chốt lại, sau khi sập chốt khóa ngay cả Bob cũng không thể mở lại được-không đọc lại hay sửa thông tin trong thư được nữa). Sau đó Bob gửi chiếc hộp lại cho Alice. Alice mở hộp với chìa khóa của mình và đọc thông tin trong thư. Trong ví dụ này, chiếc hộp với khóa mở đóng vai trò khóa công khai, chiếc chìa khóa chính là khóa bí mật.

2.3.1 Tạo Khóa

Có 2 đối tượng cần trao đổi dữ liệu mật với nhau thông qua một kênh thông tin không an toàn thì trước tiên noi nhận dữ liệu phải tiến hành tạo ra 2 khóa bảo mật d và khóa công khai e theo các bước sau:

1. Chọn 2 số nguyên tố lớn p và q với p khác q , lựa chọn ngẫu nhiên và độc lập.
2. Tính: $n = pq$.
3. Tính: giá trị hàm số O-le: $\phi(n) = (p - 1)(q - 1)$.
4. Chọn một số tự nhiên e sao cho $1 < e < \phi(n)$ và là số nguyên tố cùng nhau với $\phi(n)$.
5. Tính: d sao cho $de \equiv 1 \pmod{\phi(n)}$.

Một số lưu ý:

- Các số nguyên tố thường được chọn bằng phương pháp thử xác suất.
- Các bước 4 và 5 có thể được thực hiện bằng giải thuật Euclid mở rộng.
- Bước 5 có thể viết cách khác: Tìm số tự nhiên x sao cho $d = \frac{x(p - 1)(q - 1) + 1}{e}$ cũng là số tự nhiên. Khi đó sử dụng giá trị $d \pmod{(p - 1)(q - 1)}$.
- Từ bước 3, PKCS#1 v2.1 sử dụng $\lambda = LCM(p - 1, q - 1)$ thay cho $\phi = (p - 1)(q - 1)$.

Khóa công khai bao gồm:

- n : môđun,
- e : số mũ công khai (cũng gọi là *số mũ mã hóa*).

Khóa bí mật bao gồm:

- n : môđun, xuất hiện cả trong khóa công khai và khóa bí mật,
- d : số mũ bí mật (cũng gọi là *số mũ giải mã*).

2.3.2 Mã Hóa:

Giả sử người gửi muốn gửi đoạn thông tin M cho người nhận. Đầu tiên người gửi chuyển M thành một số $m < n$ theo một hàm có thể đảo ngược (từ m có thể xác định lại M) được thỏa thuận trước

Lúc này người gửi có m và biết n cũng như e do bên nhận gửi. Người gửi sẽ tính c là bản mã hóa của m theo công thức: $c = m^e \pmod{n}$

Hàm trên có thể tính dễ dàng sử dụng phương pháp tính hàm mũ (theo môđun) bằng (thuật toán bình phương và nhân) Cuối cùng c được truyền đi.

2.3.3 Giải Mã:

C được chuyển tới máy nhận và có bí mật d . Máy nhận có thể tìm được m từ c theo công thức sau: $\mathbf{m} = c^d \pmod{\mathbf{n}}$

Biết m , máy nhận tìm lại M theo phương pháp đã thỏa thuận trước. Quá trình giải mã hoạt động vì ta có $c^d \equiv (m^e)^d \equiv m^{ed} \pmod{n}$

Do $ed \equiv 1 \pmod{p-1}$ và $ed \equiv 1 \pmod{q-1}$, (theo Định lý Fermat nhỏ) nên:

$$m^{ed} \equiv m \pmod{p}$$

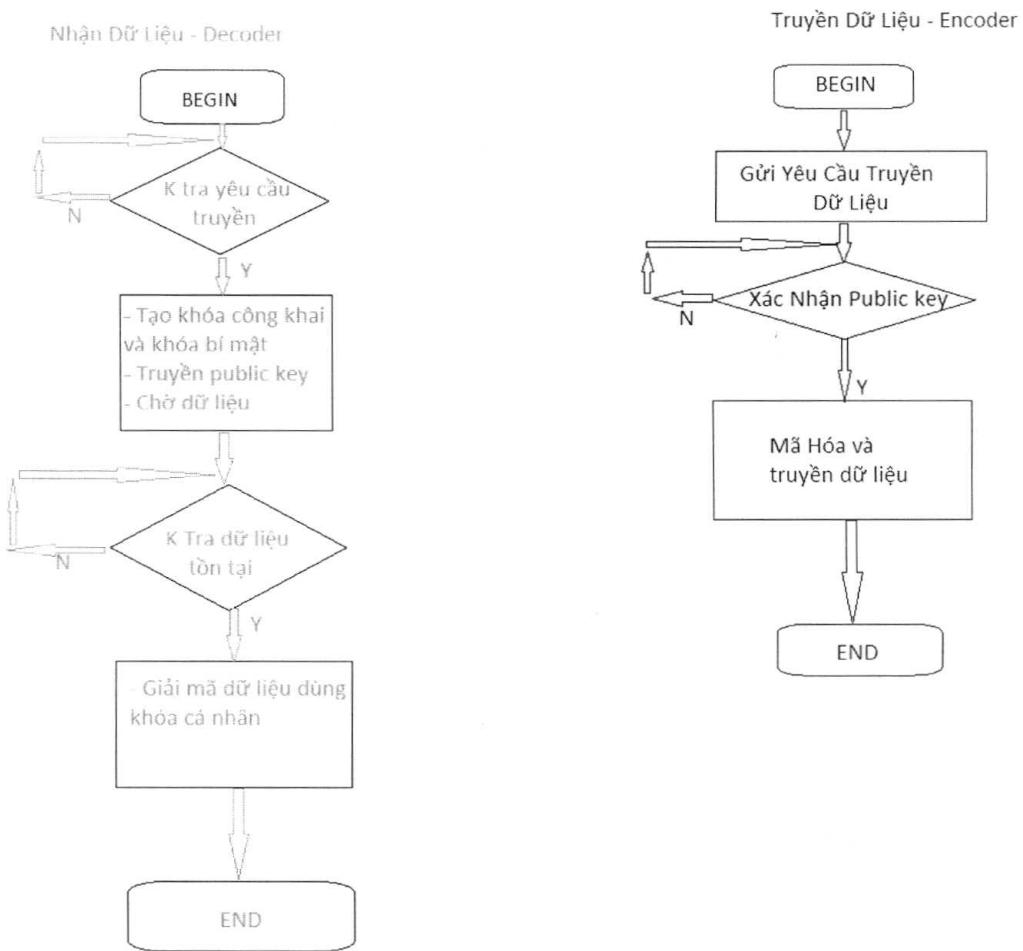
và

$$m^{ed} \equiv m \pmod{q}$$

Do p và q là hai số nguyên tố cùng nhau, áp dụng định lý số dư Trung Quốc, ta có:

$$m^{ed} \equiv m \pmod{pq}$$

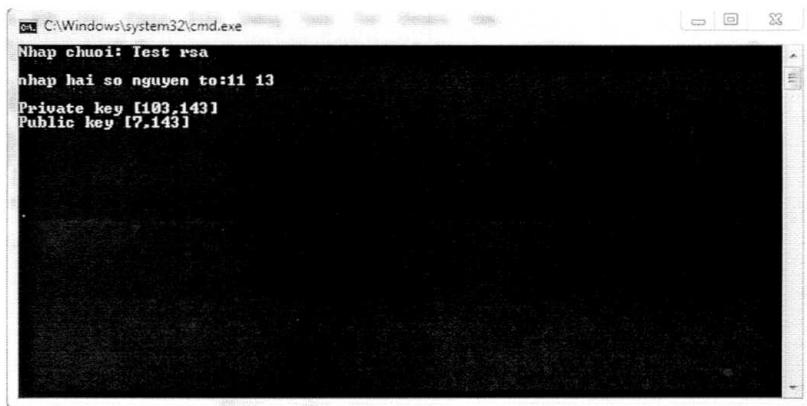
hay: $c^d \equiv m \pmod{n}$



2.3.4 Lưu đồ giải thuật

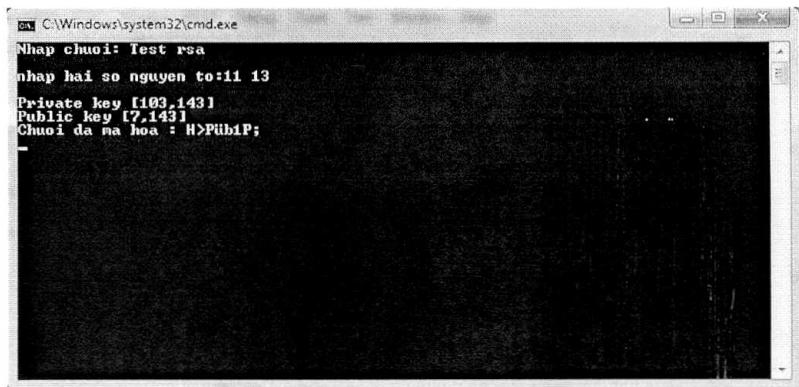
Kết quả trên phần mềm:

Key Generation :



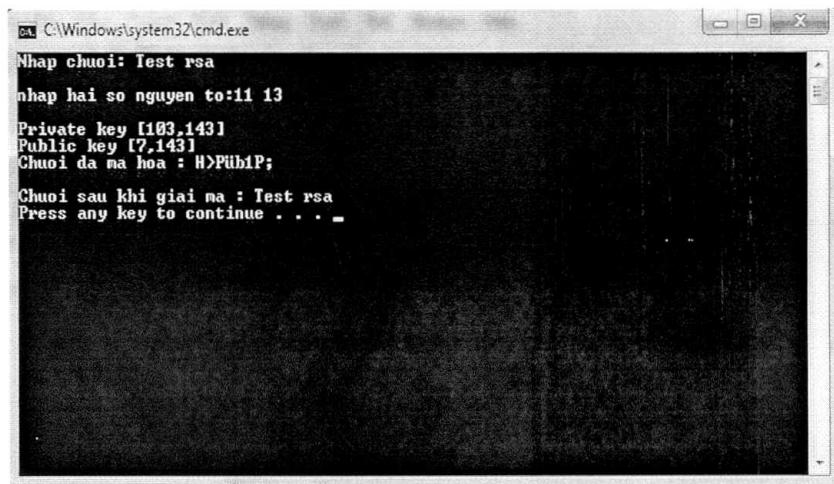
```
Nhap chuoi: Test rsa
nhap hai so nguyen to:11 13
Private key [103,143]
Public key [7,143]
```

Encoder:



```
Nhap chuoi: Test rsa
nhap hai so nguyen to:11 13
Private key [103,143]
Public key [7,143]
Chuoi da ma hoa : H>PühlP;
```

Decoder:

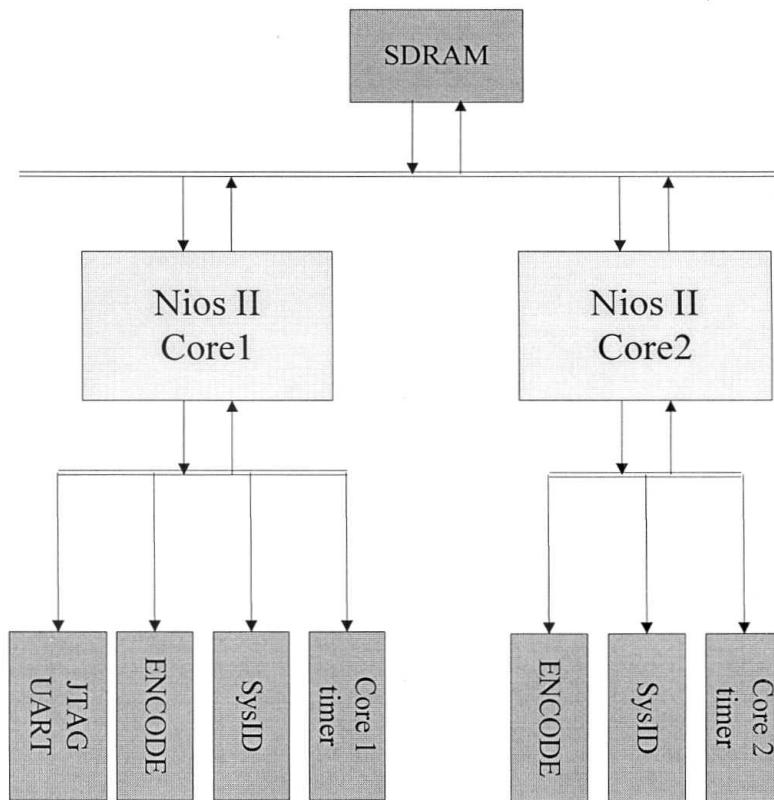


```
Nhap chuoi: Test rsa
nhap hai so nguyen to:11 13
Private key [103,143]
Public key [7,143]
Chuoi da ma hoa : H>PühlP;

Chuoi sau khi giao ma : Test rsa
Press any key to continue . . .
```

2.4 Thiết kế hệ thống phần cứng

Từ những thuận lợi mà hệ thống nhiều vi xử lý đem lại, đề tài thực hiện một hệ thống gồm hai vi xử lý, mỗi vi xử lý thực hiện một nhiệm vụ riêng biệt đó là : mã hóa, giải mã.



Hình 2.52 : Sơ đồ hệ thống nhiều vi xử lí thực hiện mã hóa và giải mã hình ảnh
Quá trình thực hiện việc mã hóa và giải mã của hai CPU :

1: CPU1 gửi yêu cầu truyền dữ liệu đến CPU2

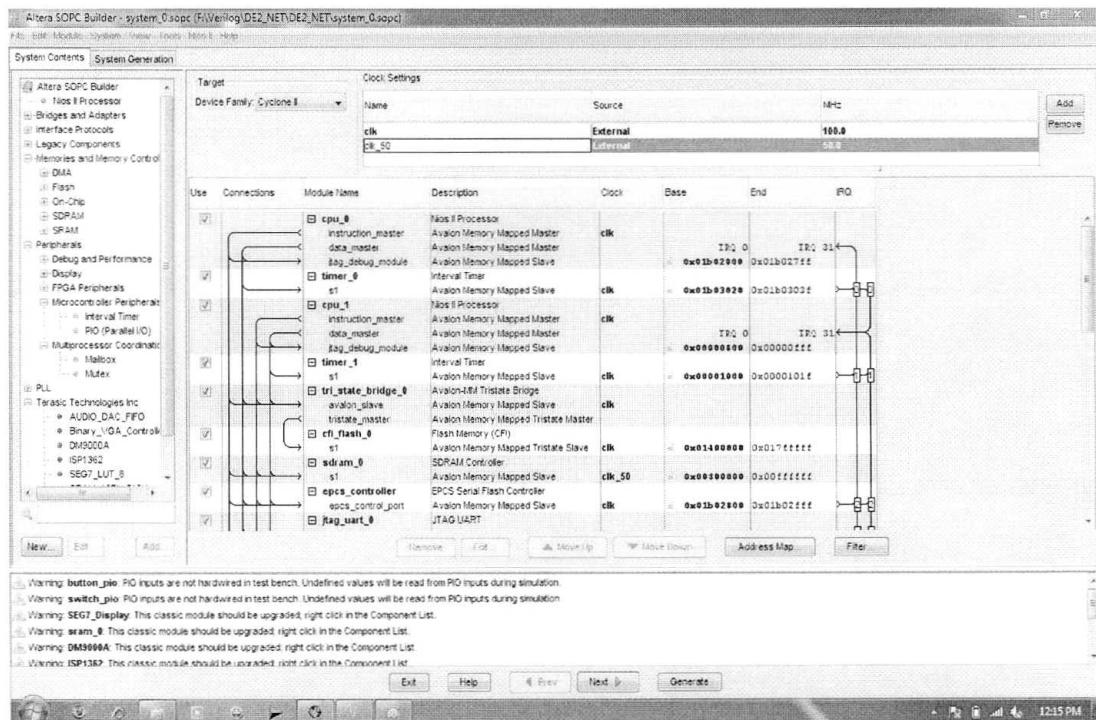
CPU2 tạo ra 2 khóa công khai và khóa bảo mật

2: CPU2 chấp nhận nhận dữ liệu và gửi khóa công khai cho CPU1 mã hóa

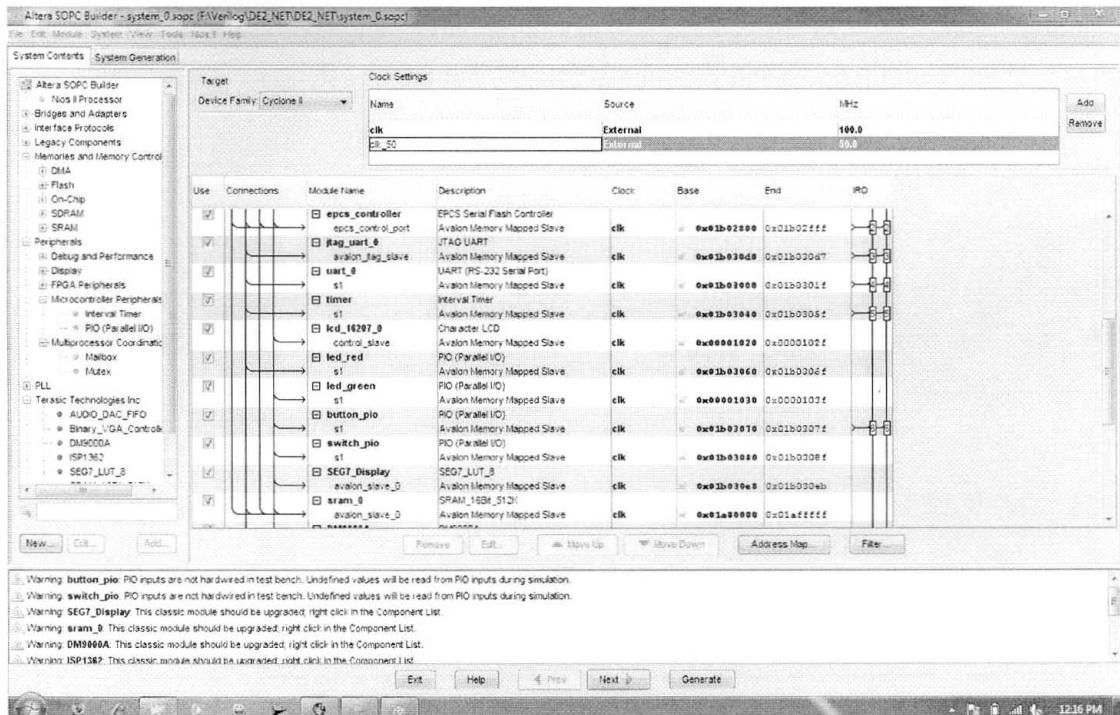
3: CPU1 nhận dữ liệu muốn gửi từ cổng jtag, mã hóa dữ liệu sử dụng khóa công khai của CPU2 tạo ra và truyền dữ liệu đã mã hóa đến CPU2.

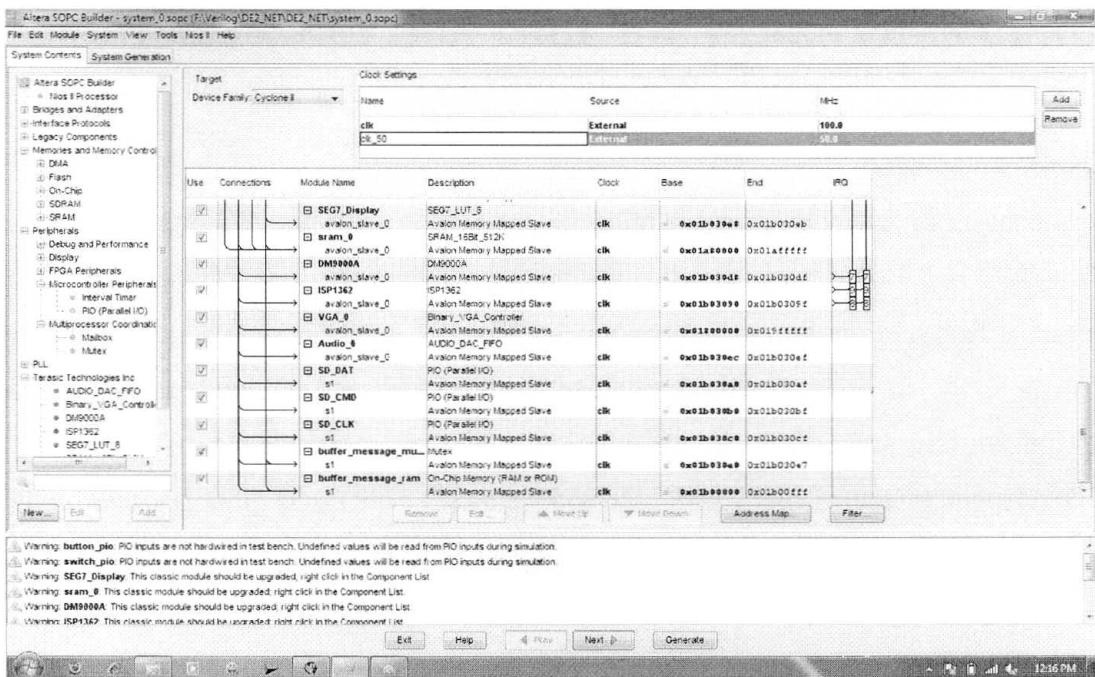
Tại CPU2 tiến hành giải mã dữ liệu và xuất dữ liệu mã hóa được ra LCD

Sơ đồ hệ thống trong SOPC :



Chủ nhiệm đề tài: Nguyễn Anh Vinh





CHƯƠNG 3 - KẾT QUẢ THỰC HIỆN TRÊN PHẦN CỨNG VÀ SO SÁNH VỚI PHẦN MỀM

3.1 Kết quả

So sánh kết quả giữa phần cứng và phần mềm ta thấy quá trình encode và decode đưa ra kết quả giống nhau và đúng với lí thuyết thuật toán RSA. Ở đây, với các ứng dụng nhỏ sự cải thiện về tốc độ sẽ không rõ ràng, nhưng nếu phát triển một hệ thống ứng dụng lớn hơn thì tốc độ sẽ khác biệt rõ ràng hơn.

Nếu RSA phát triển trên nền phần mềm rất dễ bị tấn công giải mã bởi các kiểu tấn công kênh phụ (Side Channel) bởi các phần mềm trojan và mã độc. Còn nếu phát triển trên nền tảng phần cứng có thể tránh khỏi các phần mềm phá hại. Ngoài ra, cấp độ bảo mật có thể tăng lên, hiện tại với các hệ thống PC độ dài khóa có thể xử lý tốt nhất là 512 bit, tuy nhiên nếu thiết kế trên phần cứng nhúng thì độ dài khóa có thể tăng lên 1024 bit và hiện nay đã là 2048 bit.

3.2 Kết luận

❖ Ưu điểm và hạn chế

- **Ưu điểm**
 - + Đề tài này đã giới thiệu tổng quát về các kỹ thuật mã hóa, các nguyên lý mã hóa, một số khái niệm quan trọng và các thuật toán trong mã hóa dữ liệu.
 - + Đề tài đã thiết kế hoàn chỉnh phần cứng thực hiện việc mã hóa và giải mã trên cở sở mô phỏng một cpu nhiều core, đồng bộ nhiều thành phần trong hệ thống.
 - + Tốc độ xử lý nhanh, hoạt động ở xung clock lên đến 50Mhz.
 - + Đồng thời điều quan trọng nhất là khôi thiết kế có khả năng tích hợp cao, phù hợp với nhiều cấu hình phần cứng khác nhau.
- **Hạn chế**
 - + Hạn chế của đề tài là chưa xây dựng một hệ thống mã hóa và giải mã dữ liệu truyền qua internet.
 - + Chưa thiết kế hệ thống display để làm rõ hơn kết quả đạt được.

❖ Hướng phát triển

RSA là một thuật toán ứng dụng rộng trong viễn thông, cho nên để tận dụng được thế mạnh của thuật toán thì đề tài phải phát triển và mở rộng ra môi trường internet. Trong thiết kế phần cứng của đề tài đã tích hợp sẵn card Ethernet DM9000A, cho nên tạo nhiều thuận lợi cho việc mở rộng đề tài truyền qua Ethernet thông qua các giao thức UDP, TCP/IP. Có thể sử dụng HĐH nhúng uLinux sẽ thuận lợi hơn trong việc phát triển các chuẩn UDP, TCP/IP.

TÀI LIỆU THAM KHẢO

Tiếng Việt

- [1]. Quách Tuấn Ngọc, *Ngôn ngữ lập trình C*, Nhà xuất bản Giáo Dục, 1998.
- [2]. Lương Mạnh Bá, Nguyễn Thanh Thúy, *Nhập môn xử lý ảnh số*, NXB Khoa Học và Kỹ Thuật, 1999.

Tiếng Anh

- [3]. Altera Corporation, *Avalon bus specification reference manual*, 2003
- [4]. Altera Corporation, *DMA Controller Core*, 2007
- [5]. Altera Corporation, *Tour of the SOPC Builder User Interface*, 2007
- [6]. Ben Atitallah, P. Kadionik, F. Ghazzi, P. Nouel, N. Masmoudi, Ph. Marchegay, *Real-Time Video System Design Based on the NIOS II Processor and μClinix*, 2007

Trang web

- [7]. <http://www.mathworks.com/products/rftoolbox>
- [8]. <http://www.ieexplore.ieee.org>
- [9]. <http://www.obrador.com/essentialjpeg/HeaderInfo.htm>
- [10]. <http://www.codeproject.com>

b. Các công trình khoa học tiêu biểu theo hướng đề tài đã hoàn thành và công bố:

.....

.....

.....

c. Khả năng hợp tác Quốc tế:

.....

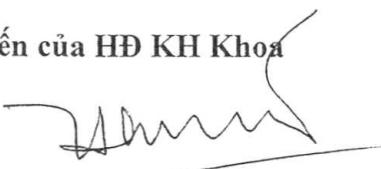
.....

.....

5. Dự kiến kinh phí hỗ trợ:

- Tổng kinh phí: 12.000.000 đồng
(Viết bằng chữ: Mười lăm triệu đồng)
- Diễn giải tóm tắt nội dung chi kinh phí nêu trên:
 - Mua, in và photo tài liệu tham khảo: 1.000.000 đồng
 - Thuê khoán chuyên môn : 4.800.000 đồng (400.000đồng / tháng x 12 tháng)
 - Chi phí xây dựng phần mềm : 2.200.000 đồng
 - Chi phí mua phần cứng : 2.000.000 đồng
 - Chi phí xét duyệt, nghiệm thu : 1.000.000 đồng

Ý kiến của HĐ KH Khoa

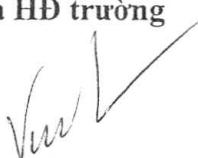

Nguyễn Văn Thiệu

Chủ nhiệm đề tài



Nguyễn Anh Vinh

Ý kiến của HĐ trường



HỢP ĐỒNG NGHIÊN CỨU KHOA HỌC
ĐỀ TÀI CẤP TRƯỜNG NĂM 2008

Hôm nay, ngày 22 tháng 04 năm 2008. Chúng tôi gồm:

Bên A: Đại diện là Ông: **TRẦN LINH THƯỚC**

Phó Hiệu trưởng trường Đại học Khoa học Tự nhiên - ĐHQG Tp. HCM

Bên B: Họ tên chủ nhiệm đề tài: **NGUYỄN ANH VINH**

Thuộc bộ môn: Máy tính – Viễn thông, Khoa Điện tử - Viễn thông ,
trường Đại học Khoa học Tự nhiên - ĐHQG Tp. HCM.

Thỏa thuận ký hợp đồng thực hiện đề tài Nghiên cứu khoa học cấp Trường như sau:

Điều 1: Bên B triển khai thực hiện đề tài (ghi tên đề tài đã được duyệt):

Tìm hiểu lý thuyết về CPU DUAL CORE và mô phỏng một số chức năng

Thời gian thực hiện đề tài: **12 tháng** (tính từ 04/2008 đến 04/2009)

Điều 2: Trách nhiệm bên B

- Thực hiện đúng đắn cương tổng quát và tiến độ đề tài theo như bản thuyết minh đã đăng ký.
- Báo cáo kết quả nghiên cứu và quyết toán kinh phí đề tài theo đúng qui định hiện hành:
 - + **Quyết toán kinh phí:** nộp chứng từ và báo cáo quyết toán kinh phí từ **03/11 – 28/11/2008**
 - + Báo cáo tiến độ thực hiện đề tài: tháng 11/2008
 - + Tiến hành nghiệm thu đề tài: tháng 04/2009
- Hết thời hạn thực hiện đề tài mà bên B vẫn không thể báo cáo nghiệm thu thì phải hoàn trả 100% kinh phí nghiên cứu.

Nếu vì lý do khách quan không thể nghiệm thu đề tài, chủ nhiệm phải đề nghị gia hạn thời gian làm đề tài (nhưng không quá 6 tháng).

Điều 3: Trách nhiệm bên A

- Hướng dẫn bên B hoàn tất các thủ tục cần thiết để tiến hành thực hiện và nghiệm thu đề tài.
- Cấp cho bên B toàn bộ kinh phí thực hiện đề tài ngay sau khi hợp đồng đã được ký kết.

Tổng số tiền: **11.000.000đ** (Mười một triệu đồng)

Điều 4: Hợp đồng được lập thành 03 bản. Chủ nhiệm đề tài giữ 1 bản, Phòng Tài Vụ giữ 1 bản và phòng KHCN&QHQT giữ 1 bản có giá trị như nhau.

Bên A

Trần Linh Thước

Bên B

Nguyễn Anh Vinh