**ĐẠI HỌC QUỐC GIA THÀNH PHỒ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA ĐIỆN TỬ - VIỄN THÔNG**

**BỘ MÔN MÁY TÍNH – HỆ THỐNG NHÚNG**

---🙞🕮🙜---



**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**Môn học:** Thực hành Thiết kế SoC

Giảng viên hướng dẫn: Trần Tuấn Kiệt

Lớp: 20Nhung

Nhóm sinh viên thực hiện: Nhóm 11

Trần Nguyễn Đạt Văn 20200417

Tôn Đức Phú Vĩnh 20200420

Phan Thành Văn 20200416

Lê Hoàng Việt Quốc 20200323

Thành phố Hồ Chí Minh, ngày 26 tháng 11 năm 2023

**MỤC LỤC**

[**MỤC LỤC** 2](#_Toc155980988)

[**DANH MỤC HÌNH ẢNH** 2](#_Toc155980989)

[**TỔNG QUAN** 4](#_Toc155980990)

[**I.** **GIỚI THIỆU** 5](#_Toc155980991)

[**1.** **Giới thiệu về môi trường phần mềm** 5](#_Toc155980992)

[*a.* *Quartus II* 5](#_Toc155980993)

[*b.* *ModelSim* 5](#_Toc155980994)

[**2.** **Giới thiệu về môi trường phần cứng** 6](#_Toc155980995)

[*a.* *Kit DE10-Standard* 6](#_Toc155980996)

[**3.** **Tương tác giữa phần cứng và phần mềm trong dự án sử dụng Board DE10 Standard và Quartus II ModelSim** 8](#_Toc155980997)

[**II.** **Thuật toán CRC** 10](#_Toc155980998)

[**1.** **Sơ lược về thuật toán CRC** 10](#_Toc155980999)

[*a.* *Cách biểu diễn chuỗi bit đa thức sinh (Polynomial Generator Representation)* 10](#_Toc155981000)

[*b.* *Thứ tự bit của các byte dữ liệu đầu vào* 11](#_Toc155981001)

[*c.* *Thứ tự bit của kết quả tính CRC* 12](#_Toc155981002)

[*d.* *XOR ngõ ra* 12](#_Toc155981003)

[*e.* *Thứ tự áp dụng các thông số tính toán trong quá trình tính CRC* 12](#_Toc155981004)

[**2.** **Áp dụng thiết kế thuật toán** 13](#_Toc155981005)

[*a.* *Ý tưởng* 13](#_Toc155981006)

[*b.* *Tiến hành code* 15](#_Toc155981007)

[*c.* *Kết quả mô phỏng và chạy trên board thật* 20](#_Toc155981008)

**DANH MỤC HÌNH ẢNH**

[Hình 1: Kit DE10-Standard 7](#_Toc155980749)

[Hình 2: Sơ đồ khối kit DE10-Standard 8](#_Toc155980750)

[Hình 3: Quá trình tính CRC 13](#_Toc155980751)

[Hình 4: Đa thức sinh 14](#_Toc155980752)

[Hình 5: Bit MSB điều khiển MUX 14](#_Toc155980753)

[Hình 6: Xét bit MSB 15](#_Toc155980754)

[Hình 7: Mạch tính CRC-1 bit 15](#_Toc155980755)

[Hình 8: Mạch tính CRC-3 bit 15](#_Toc155980756)

[Hình 9: CRC top signal 16](#_Toc155980757)

[Hình 10: Sơ đồ khối CRC top signal 17](file:///C:\Users\Admin\Desktop\baocaothsoc.docx#_Toc155980758)

[Hình 11: CRC sub signal 18](#_Toc155980759)

[Hình 12: Sơ đồ khối CRC sub signal 19](file:///C:\Users\Admin\Desktop\baocaothsoc.docx#_Toc155980760)

[Hình 13: Kết quả mô phỏng 21](#_Toc155980761)

[Hình 14: Kết quả 21](#_Toc155980762)

# **TỔNG QUAN**

SoC là viết tắt của “System on a Chip” là một khái niệm trong lĩnh vực kỹ thuật điện tử và vi mạch.

Một SoC là một chip tích hợp tất cả hoặc hầu hết các thành phần hệ thống điện tử cần thiết vào một chip duy nhất. Các thành phần này có thể bao gồm vi xử lý trung tâm (CPU), bộ nhớ, đơn vị xử lý đồ hoạ (GPU), bộ điều khiển giao tiếp, các khối kết nối, cảm biến và nhiều thành phần khác tuỳ thuộc vào mục đích sử dụng cụ thể.

Thiết kế SoC (System on a Chip) là quá trình tích hợp tất cả hoặc hầu hết các thành phần hệ thống điện tử vào một chip duy nhất. SoC thường bao gồm vi xử lý trung tâm (GPU), bộ nhớ, đơn vị xử lý đồ hoạ (GPU), bộ điều khiển giao tiếp, các khối kết nối và nhiều thành phần khác nhau như cảm biến, bộ điều khiển năng lượng, và nhiều khác nữa.

Quá trình SoC thường đi qua nhiều giai đoạn, bắt đầu từ đặt ra yêu cầu chức năng và hiệu suất, sau đó đến thiết kế logic, mô phỏng và cuối cùng là sản xuất và kiểm thử. Mục tiêu của việc tích hợp tất các thành phần vào một chip duy nhất là tối ưu hoá hiệu suất, giảm tiêu thụ năng lượng và giảm kích thước của hệ thống.

Thiết kế SoC CRC (System on a Chip with Cyclic Redundancy Check) là quá trình tạo ra một chip tích hợp nhiều chức năng hệ thống (SoC) mà trong đó có tính năng kiểm tra Cyclic Redundancy Check (CRC).

CRC là một kỹ thuật kiểm tra lỗi được sử dụng rộng rãi trong truyền thông dữ liệu để đảm bảo tính toàn vẹn của thông tin. Khi tích hợp vào SoC, chức năng CRC có thể được sử dụng để kiểm tra lỗi trong quá trình truyền thông dữ liệu giữa các thành phần của chip hoặc giữa chip và các thiết bị ngoại vi khác.

Quá trình thiết kế SoC CRC bao gồm các bước từ phân tích yêu cầu, thiết kế logic, đến mô phỏng và kiểm thử để đảm bảo tính đúng đắn và hiệu suất của hệ thống. Mục tiêu của việc tích hợp CRC vào SoC thường là tăng cường khả năng xác định và sửa lỗi trong dữ liệu truyền, đặc biệt là trong các ứng dụng đòi hỏi độ chắc chắn và độ tin cậy cao, như trong các hệ thống truyền thông và lưu trữ dữ liệu.

# **GIỚI THIỆU**

## **Giới thiệu về môi trường phần mềm**

### *Quartus II*

Quartus II là một phần mềm thiết kế và tổng hợp logic của hãng Altera (nay là một phần của Intel) được sử dụng trong quá trình phát triển hệ thống dựa trên các thiết bị FPGA (Field-Programmale Gate Array) và CPLD (Complex Programmable Logic Device). Phần mềm này cung cấp một môi trường tích hợp để thiết kế và triển khai các hệ thống số, bao gồm việc mô phỏng, tổng hợp và triển khai logic trên các chip lập trình được.

Các tính năng chính của Quartus II bao gồm:

* Thiết kế Logic: Quartus II hỗ trợ thiết kế logic từ ngôn ngữ HDL như Verilog và VHDL.
* Tổng hợp và Tối ưu hoá: Cung cấp các công cụ tổng hợp logic và tối ưu hoá hiệu suất của hệ thống.
* Định nghĩa và Mô phỏng Thiết kế: Cho phép mô phỏng thiết kế trước khi triển khai, giúp kiểm tra tính đúng đắn và hiệu suất.
* Triển khai trên FPGA và CPLD: Hỗ trợ quá trình thiết kế lên các thiết bị lập trình được.

### *ModelSim*

ModelSim là một công cụ mô phỏng và kiểm thử hệ thống dựa trên ngôn ngữ HDL (Hardware Description Language) như Verilog, VHDL. ModelSim được sử dụng để mô phỏng và kiểm thử các mô-đun và hệ thống số trước khi triển khai lên thiết bị FPGA hoặc CPLD.

Các tính năng chính của ModelSim bao gồm:

* Mô phỏng và Kiểm thử: Cho phép mô phỏng toàn bộ hoặc một phần của hệ thông để kiểm tra tính đúng đắn và hiệu suất.
* Debugging: Cung cấp công cụ mạnh mẽ cho quá trình debug, giúp phát triển và sửa lỗi trong mã nguồn HDL.
* Hỗ trợ nhiều Ngôn ngữ HDL: Hỗ trợ mô phỏng cho cả Verilog và VHDL.
* Tích hợp với Quartus II: Có thể tích hợp với ModelSim với Quartus II để tạo ra một luồng làm việc tích hợp từ mô phỏng đến triển khai.

Cả hai công cụ Quartus II và ModelSim đều là công cụ quan trọng trong quy trình thực hiện thiết kế SoC CRC dựa trên FPGA.

## **Giới thiệu về môi trường phần cứng**

### *Kit DE10-Standard*

Bộ công cụ phát triển tiêu chuẩn DE10 trình bày một nền tảng thiết kế phần cứng mạnh mẽ được xây dựng xung quanh FPGA Hệ thống trên chip (SoC) của Intel, kết hợp các lõi nhúng Cortex-A9 lõi kép mới nhất với logic khả trình hàng đầu trong ngành để mang lại tính linh hoạt cao nhất cho thiết kế. Giờ đây, người dùng có thể tận dụng sức mạnh của khả năng tái cấu hình cực lớn kết hợp với hệ thống bộ xử lý hiệu suất cao, tiêu thụ ít năng lượng. SoC của Altera tích hợp hệ thống bộ xử lý cứng (HPS) dựa trên ARM bao gồm bộ xử lý, thiết bị ngoại vi và giao diện bộ nhớ được liên kết liền mạch với cấu trúc FPGA bằng cách sử dụng đường trục kết nối băng thông cao. Bo mạch phát triển DE10-Standard bao gồm phần cứng như bộ nhớ DDR3 tốc độ cao, khả năng video và âm thanh, mạng Ethernet, v.v.

A close-up of a computer

Description automatically generated with low confidence

Hình 1: Kit DE10-Standard

Các phần cứng được tích hợp trên kit DE10-Standard:

* FPGA Cyclone V 5CSXFC6D6F31C6N (110.000 phần tử logic, 41509 ALMs, 5.761K bit RAM, 6 FPGA PLLs và 3 HPS PLLs)
* USB Blaster II (cấu hình cho FPGA)
* 64MB (32Mx16) SDRAM (FPGA)
* 1GB (2x256Mx16) DDR3 SDRAM (HPS)
* Khe cắm thẻ nhớ SD
* Bộ giải mã Audio 24-bit
* VGA DAC
* Bộ giải mã TV
* Bộ điều khiển Ethernet 10/100/1000
* Bộ điều khiển chuẩn USB với các đầu kết nối loại A và B
* PS/2 (nối với bàn phím hoặc chuột)
* Bộ chuyển đổi hồng ngoại

Ngoài ra, kit DE10-Standard còn hỗ trợ thêm các nút gạt, nút bấm, LED thường, LED 7 đoạn, các khe mở rộng,…

A screenshot of a computer

Description automatically generated with low confidence

Hình 2: Sơ đồ khối kit DE10-Standard

## **Tương tác giữa phần cứng và phần mềm trong dự án sử dụng Board DE10 Standard và Quartus II ModelSim**

* Mô phỏng và kiểm thử với ModelSim:
* Mô phỏng CRC logic: ModelSim sẽ giúp bạn mô phỏng logic CRC trước khi triển khai lên FPGA. Bằng cách này, bạn có thể đảm bảo rằng logic CRC hoạt động đúng đắn và đáp ứng yêu cầu của dự án.
* Kiểm thử tương tác CPU và FPGA: Sử dụng ModelSim, bạn có thể kiểm thử tương tác giữa CPU và FPGA, đảm bảo rằng dữ liệu được truyền đúng cách và CRC được tính toán chính xác.

1. Giao tiếp GPIO và đầu vào dữ liệu:

* Giao tiếp GPIO với Quartus II: Nếu có các cổng GPIO trên DE10 Board, bạn có thể sử dụng chúng để giao tiếp với môi trường ngoại vi. Quartus II cung cấp công cụ để đặt và kiểm soát các chân GPIO.

1. Xử lý dữ liệu và tính toán CRC:

* Lập trình CPU: Việc lập trình CPU (thường là ARM Cortex-A9 trên DE10) sẽ bao gồm việc kết hợp logic CRC vào mã nguồn và quản lý việc kích thích tính toán CRC khi cần thiết.
* Tính toán CRC trên FPGA: Logic CRC trên FPGA sẽ được thiết kế để tính toán CRC dựa trên dữ liệu đầu vào từ CPU.

1. Kiểm thử thời gian thực:

* Kiểm soát thời gian thực: Nếu dự án yêu cầu tính thời gian thực, sử dụng ModelSim để kiểm thử và đảm bảo rằng quá trình tính toán CRC và tương tác với CPU đáp ứng các yêu cầu thời gian.

1. Kiểm thử toàn diện:

* Tổng hợp và kiểm thử toàn diện: Sử dụng Quartus II để tổng hợp logic và triển khai lên FPGA, sau đó sử dụng ModelSim để kiểm thử toàn bộ hệ thống, đảm bảo tính đúng đắn và hiệu suất mong muốn.

1. **Thuật toán CRC**
2. **Sơ lược về thuật toán CRC**

**Cyclic Redundancy Check** thường viết tắt là **CRC**, là thuật ngữ tiếng Anh trong kỹ thuật số là phương pháp kiểm tra và phát hiện lỗi, được sử dụng trong các mạng số và thiết bị lưu trữ để phát hiện sự thay đổi tình cờ đối với dữ liệu được truyền đi hay lưu trữ.

1. *Cách biểu diễn chuỗi bit đa thức sinh (Polynomial Generator Representation)*

Một đa thức sinh có nhiều các biểu diễn chuỗi bit khác nhau. Mỗi cách biểu diễn sẽ cho giá trị chuỗi bit đa thức sinh khác nhau. Đa thức sinh tổng quát của bộ tính CRC-n như sau:

Chuỗi bit của đa thức sinh có các cách biểu diễn sau:

* Cách biểu diễn thông thường (Normal representation).
* Cách biểu diễn đảo ngược (Reversed representation).
* Cách biểu diễn Koopman (Koopman representation).
* Cách biểu diễn nghịch đảo (Reciprocal representation).

Ví dụ: Ta có đa thức sinh CRC-8 sau:

**Cách biểu diễn thông thường** sẽ loại bỏ bit phần tử có trọng số cao nhất x^n. Trong một đa thức sinh, an luôn bằng 1. Chuỗi bit đa thức sinh, tính từ trái qua phải, sẽ chứa các bit từ n-1 đến 0. Áp dụng vào ví dụ trên, x^8 sẽ được loại bỏ, chuỗi bit đa thức sinh sẽ từ x^7 đến x^0 là:

1101\_0101 (0xD5)

**Cách biểu diễn đảo ngược** sẽ loại bỏ bit phần tử có trọng số cao nhất x^n. Chuỗi bit đa thức sinh, tính từ trái qua phải, sẽ chứa các bit từ 0 đến n-1. Thứ tự này ngược với cách biểu diễn thông thường. Áp dụng vào ví dụ trên, x^8 sẽ được loại bỏ, chuỗi bit đa thức sinh sẽ từ x^0 đến x^7 là:

1010\_1011 (0xAB)

**Cách biểu diễn Koopman** sẽ loại bỏ bit phần tử có trọng số thấp nhất x^0. Trong một đa thức sinh, a0 luôn bằng 1 giống như an. Chuỗi bit đa thức sinh, tính từ trái qua phải, sẽ chứa các bit từ n đến 1. Áp dụng vào ví dụ trên, x^0 sẽ được loại bỏ, chuỗi bit đa thức sinh sẽ từ x^8 đến x^1 là:

1110\_1010 (0xEA)

**Cách biểu diễn nghịch đảo** sẽ đảo lại toàn bộ thứ tự bit của đa thức sinh, x^0 thành x^n, x^1 thành x^n-1, ... sau đó, bit đầu tiên bên trái sẽ bị loại bỏ. Chuỗi bit đa thức sinh, tính từ trái qua phải, sẽ chứa các bit từ 1 đến n. Áp dụng vào ví dụ trên, x^0 sẽ được loại bỏ, chuỗi bit đa thức sinh sẽ từ x^1 đến x^8 là:

0101\_0111 (0x57)

1. *Thứ tự bit của các byte dữ liệu đầu vào:*

* Đây là thứ tự các bit trong các byte dữ liệu được truyền vào để tính CRC.
* Normal input: là thứ tự bit được đưa vào bộ tính CRC và bộ tính CRC không thay đổi thứ tự này.
* Tùy giao thức, thứ tự bit trong từng byte dữ liệu (8 bit) có thể bị đảo ngược trước khi tính CRC. Việc đảo ngược thứ tự bit trong từng byte gọi là "Reflected Input".

Ví dụ: Xét chuỗi dữ liệu sau:

10011010\_10111100

Nếu tính CRC với cơ chế "Reflected input", thì chuỗi bit đầu vào sẽ được biến đổi thành:

**01011001\_00111101**

1. *Thứ tự bit của kết quả tính CRC:*

Tùy giao thức, thứ tự bit của toàn bộ chuỗi CRC có thể bị đảo ngược, "MSB đến LSB" bị đảo thành "LSB đến MSB". Việc đảo ngược thứ tự bit của chuỗi CRC gọi là "Reflected Output". Lưu ý, việc đảo ngược thực hiện trên toàn bộ chuỗi CRC chứ không phải theo từng byte như "Reflected input". Ví dụ, nếu ta có một chuỗi kết quả tính CRC thông thường như sau:

10011010\_10111100  
Nếu tính CRC với cơ chế "Reflected output", thì giá trị tính toán CRC sẽ là:

00111101\_01011001

* Như vậy, tính CRC với "normal output" và "reflected output" sẽ cho kết quả chuỗi CRC khác nhau.

1. *XOR ngõ ra:*

Tùy giao thức, kết quả thông thường có thể bị điều chỉnh bằng cách XOR với một giá trị cố định, gọi là "Final XOR hay XOR output". Ví dụ, nếu ta có một chuỗi kết quả tính CRC thông thường như sau: 10011010\_10111100

Nếu tính CRC dùng cơ chế "Final XOR" với giá trị được XOR là: 11111111\_11111111 thì giá trị tính toán CRC sẽ là: 01100101\_0100011

Kết quả thông thường ứng với trường hợp XOR với giá trị toàn bit 0. Như vậy, hằng số XOR ngõ ra khác nhau sẽ tạo ra chuỗi bit CRC khác nhau.

1. *Thứ tự áp dụng các thông số tính toán trong quá trình tính CRC:*

A diagram of a computer program

Description automatically generated

Hình 3: Quá trình tính CRC

1. **Áp dụng thiết kế thuật toán**
2. *Ý tưởng*

Ta sẽ dựa vào cách tính CRC thông thường như sau:

* Xét đa thức sinh g(x) = x + 1, đây là đa thức CRC-1. Tính CRC cho chuỗi 8 bit b10100010 và chuỗi b10011111.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Chuỗi dữ liệu | | | | | | | | | Đa thức sinh |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 1 |
| 1 | 1 |  |  |  |  |  |  |  | 1 1 0 0 0 0 1 1 |
| 0 | 1 | 1 |  |  |  |  |  |  |
|  | 1 | 1 |  |  |  |  |  |  |
|  | 0 | 0 | 0 |  |  |  |  |  |
|  |  | 0 | 0 |  |  |  |  |  |
|  |  | 0 | 0 | 0 |  |  |  |  |
|  |  |  | 0 | 0 |  |  |  |  |
|  |  |  | 0 | 0 | 0 |  |  |  |  |
|  |  |  |  | 0 | 0 | 1 |  |  |  |
|  |  |  |  |  | 0 | 0 |  |  |  |
|  |  |  |  |  | 0 | 1 | 0 |  |  |
|  |  |  |  |  |  | 1 | 1 |  |  |
|  |  |  |  |  |  | 0 | 1 | 0 |  |
|  |  |  |  |  |  |  | 1 | 1 |  |
|  |  |  |  |  |  |  | 1 | 1 |  |

Hình 4: Đa thức sinh

→ Từ ví dụ trên, CRC được tính theo nguyên tắc:

* Nếu bit MSB của lần tính hiện tại bằng 1 thì nó sẽ được XOR (modulo-2) với đa thức sinh
* Nếu bit MSB của lần tính hiện tại bằng 0 thì nó sẽ không đổi

Từ 2 nguyên tắc trên ta sẽ xây dựng mạch tính CRC-1 bit như sau:

* Mạch cần 2 FF để lưu giá trị sau mỗi lần XOR và mạch sẽ dịch 1 bit sau mỗi lần XOR để lấy 1 bit dữ liệu mới như hình sau:

A diagram of a block diagram

Description automatically generated

Hình 5: Bit MSB điều khiển MUX

* Ở hình trên, bit MSB sẽ điều khiển MUX chọn có XOR với đa thức sinh x+1 hay không? Tuy nhiên, sau mỗi chu kỳ tính, bit MSB luôn bị loại bỏ nên mạch MUX và XOR của bit MSB là không cần thiết. Mạch được rút gọn như hình sau:

A diagram of a block diagram

Description automatically generated

Hình 6: Xét bit MSB

* Xét mạch MUX, nếu bit MSB bằng 1 thì bit 0 XOR với 1, nếu bit MSB bằng 0 thì tương ứng với việc bit 0 XOR với 0 nên mạch MUX được loại bỏ để thay bằng bit 1 XOR bit 0.

A diagram of a block diagram

Description automatically generated

Hình 7: Mạch tính CRC-1 bit

* Từ mạch tính CRC-1 bit như trên ta sẽ mở rộng thành CRC-3 bit như sau:

A diagram of a block diagram

Description automatically generated

Hình 8: Mạch tính CRC-3 bit

* Từ mạch trên ta sẽ tiến hành code

1. *Tiến hành code*

Ta sẽ có 2 module chính:

* Module CRC: module chính để tính toán CRC dựa vào data truyền vào và các parameter mà ta đã set
* Module CRC\_Slave: module bao bọc module CRC. Module này chứa các tín hiệu để giao tiếp với các component trên hệ thống sử dụng Avalon bus: CPU NIOS II, on\_chip\_memory, jtag, …

*CRC top signal:*

A diagram of a computer code

Description automatically generated

Hình 9: CRC top signal

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal** | **Width** | **Direction** | **Description** |
| clk | 1 | Input | Synchronous clock signal |
| Rst\_n | 1 | Input | Reset signal |
| cs | 1 | Input | Chip select signal |
| wr | 1 | Input | Write active signal |
| rd | 1 | Input | Read active signal |
| addr | 2 | Input | Address |
| wr\_data | 32 | Input | Data for write operation |
| rd\_data | 32 | Output | Data for read operation |

Sơ đồ khối:

A black background with white squares

Description automatically generated

Hình 10: Sơ đồ khối CRC top signal

Hoạt động:

* Khối CRC sẽ lấy 32 chu kỳ để tính toán và trả về kết quả CRC
* Tín hiệu done báo hiệu khối CRC đã tính toán xong
* Thanh ghi control sẽ điều khiển khối CRC hoạt động như thế nào:
  + Control = 0: tiến hành nạp data vào khối CRC
  + Control = 1: bắt đầu tính toán

*CRC sub signal:*

A diagram of a computer system

Description automatically generated

Hình 11: CRC sub signal

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal** | **Width** | **Direction** | **Description** |
| clk | 1 | Input | Synchronous clock signal |
| Rst | 1 | Input | Reset signal |
| control | 1 | Input | 0: write data  1: calculate + read data |
| enable | 1 | Input | Cho phep module nay hoat dong |
| data\_in | 32 | Input | Data input |
| done | 1 | Input | Active when complete |
| result | 32 | Input | Final result after calculate |

Sơ đồ khối:

A diagram of a circuit

Description automatically generated

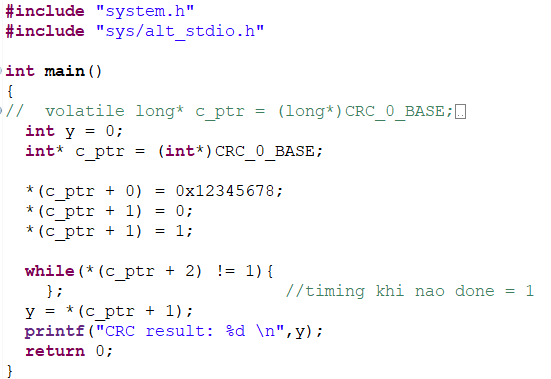
Hình 12: Sơ đồ khối CRC sub signal

Hoạt động:

* Khi control = 0 thì tiến hành nạp data vào thanh ghi lsb và thanh ghi crc\_result
* Khi control = 1 bắt đầu tính toán như đã mô tả ở phần ý tưởng. Sau 32 chu kỳ sẽ trả về kết quả. Số chu kỳ chờ đợi bằng số bit data truyền vào.

1. *Kết quả mô phỏng và chạy trên board thật*

Phần software để test



* Giải thích:
* Sử dụng con trỏ để ghi data xuống module CRC\_Slave. Data ở đây là 0x12345678.
* Sau đó ghi vào thanh ghi control để CRC\_Slave bắt đầu nạp data sau đó tính toán.
* Vòng lặp while để timing khi nào dữ liệu đã được tính toán xong thông qua việc đọc dữ liệu từ done trong module CRC\_Slave.
* Đọc kết quả từ read data sau đó lưu vào biến y và in ra màn hình.

Kết quả mô phỏng dạng sóng:

A computer screen shot of a black and green screen

Description automatically generated

Hình 13: Kết quả mô phỏng

* Giải thích:
* Dữ liệu được ghi xuống 0x12345678 = 32’b00010010001101000101011001111000 tương ứng với tín hiệu data\_in.
* Dữ liệu sẽ được tính toán trong 32 chu kỳ. Sau khi được tính toán xong là crc\_result = 101 = 5.
* Dữ liệu xuất ra để CRC\_Slave đọc và truyền lên lại software là result = 5.

A screenshot of a computer program

Description automatically generated

Hình 14: Kết quả

−−−🙞🕮🙜−−−