# Mục tiêu

Trong bài lab này sinh viên sẽ tham khảo thiết kế một Processor đơn giản trên phần mềm mô phỏng Quartus.

# Nội dung thực hành

## **Lí thuyết:**

* Processor (còn được gọi là central processing unit hoặc CPU) là một thành phần quan trọng của một máy tính hoặc thiết bị điện tử khác. Nó là bộ phận chịu trách nhiệm thực hiện các hoạt động tính toán và điều khiển trong hệ thống.
* Processor hoạt động bằng cách nhận dữ liệu từ bộ nhớ và các thiết bị khác trong hệ thống, sau đó thực hiện các phép tính và xử lý dữ liệu theo các chỉ thị (instructions) được cung cấp. Nó có khả năng thực hiện nhiều loại phép tính, bao gồm các phép toán số học, logic, truy xuất dữ liệu và điều khiển luồng.
* Processor thường được xây dựng trên một mạch tích hợp (integrated circuit) và chứa các thành phần điện tử như bộ nhớ đệm (cache), bộ điều khiển (control unit) và bộ xử lý (arithmetic logic unit). Nó còn có khả năng giao tiếp với các thành phần khác trong hệ thống thông qua các giao diện như bus.
* Một vi xử lý đơn giản sẽ bao gồm 2 khối chính đó là Controller và Datapath:
* Controller có nhiệm vụ điều khiển đường đi của dữ liệu (điều khiển việc đọc ghi của Register File, …) và điều kiển việc thực hiện tính toán (điều khiển Opcode của khối ALU, …) trong khối Datapath.
* Datapath chứa các khối cần thiết để thực hiện việc tính toán (Register File, ALU, bộ dịch,…) được điều khiển bởi Controller.

## **Nội dung thực hành:**

Thiết kế một vi xử lý đơn giản dùng để tính toán biểu thức sau:

D3I3 + D2I2 – D1I1 +D0I0

Trong đó:

* Dx là 4 ký số cuối của MSSV (ở đây 4 số cuối của MSSV là **0746 nhưng sẽ lấy giá trị 1746**)
* Ix là 4 ký số được nhập lần lượt tại ngõ vào (Ix có 4 bit)

# Thực hiện

Thực hiện thiết kế vi xử lý đơn giản để tính toán biểu thức: I3 + 7I2 – 4I1 + 6I0 . Các bước thực hiện để tính I3 + 7I2 – 4I1 + 6I0 :

|  |  |
| --- | --- |
| State | Công việc |
| S0 | I0 ➝ A000 |
| S1 | I1 ➝ A001 |
| S2 | I2 ➝ A010 |
| S3 | I3 ➝ A011 |
| S4 | Temp0 **←** I2 << 3  I2 AND I2 ;>> 3 ➝ A101:T0 |
| S5 | Data2 ← Temp0 – Data2  T0 – I2;>>0 ➝ A010:I2 |
| S6 | Data1 ← Data1 << 2  I1 AND I1;>>2 ➝ A001:I1 |
| S7 | Temp1 **←** Data0 << 2  I0 AND I0 ;>>2 ➝ A110:T1 |
| S8 | Temp0 **←** Data0 << 1  I0 AND I0;>>1 ➝ A101:T0 |
| S9 | Data0 **←** Temp0 + Temp1  T0 + T1;>>0 ➝ A000:I0 |
| S10 | Temp0 **←** Data3 + Data2  I3 + I2 ;>>0 ➝ A101:T0 |
| S11 | Temp0 **←** Temp0 + Data0  T0 + I0;>>0 ➝ A101:T0 |
| S12 | Sum **←** Temp0 – Data1  T0 – I1;>>0 ➝ A100:Sum |
| S13 | O **←** Sum  Sum AND Sum;>>0 ➝ A100:Sum |

Từ bảng trên, ta có thể thấy:

* Nhập dữ liệu (Ix) cần 4 bit, nhưng phải mở rộng dấu lên 16 bit để thực hiện tính toán sau này.
* Có 2 nguồn gán giá trị cho biến (một là từ đầu vào và hai là từ các phép tính toán) nên sẽ cần sử dụng Mux 2 to 1 16 bit.
* Chúng ta sẽ sử dụng 6 biến có dấu (Data0, Data1, Data2, Data3, Temp0, Temp1, Sum). Vì thế cần sử dụng Register File có tối thiểu 8 thanh ghi 16 bit có chức năng đọc và ghi dữ liệu.
* ALU 16 bit để thực hiện các phép toán.
* Một bộ dịch trái tối đa 3 bit.
* Xuất kết quả cần sử dụng thanh ghi 16 bit có khả năng cho phép/không cho phép xuất giá trị tại ngõ ra.

# 1.THIẾT KẾ DATAPATH

# Thiết Kế MUX2\_1\_1BIT:

# 

# Đóng gói MUX2\_1\_1BIT:

# 

# Từ MUX2\_1 ta thiết kế MUX2\_1\_16BIT :

# 

# 

# 

# Đóng gói MUX2\_1\_16BIT :

# 

# Ta thiết kế RFC 1 BIT:

# 

# Đóng gói RFC\_1 \_BIT

# 

# Từ Thanh ghi RFC\_1\_BIT ta tạo thanh ghi RFC\_16\_BIT:

# 

# 

# 

# Đóng gói thanh ghi RFC\_16\_BIT :

# 

# Ta thiết kế DECODER3\_To\_8

# 

# Ta Đóng gói DECODER3\_To\_8:

# 

# Từ Thanh ghi RFC\_16\_BIT và DECODER3\_TO\_8 ta tạo REGISTER FILE 8 thanh ghi 16 bit REG\_8\_16BIT :

# 

# 

# Đóng gói mạch REG\_8\_16BIT

# 

# Ta thiết kế ALU để tính toán giữa 2 thanh ghi 16 bit dựa vào bảng sau:

|  |  |  |
| --- | --- | --- |
| opcode |  |  |
| 000 | A + B | Full adder (16bit) |
| 001 | A + 1 |
| 010 | A – B | Full subtractor (16bit) |
| 011 | A – 1 |
| 100 | A and B | And (16bit) |
| 101 | A or B | Or(16bit) |
| 110 | A xor B | Xor(16bit) |
| 111 | A nand B | Nand(16bit) |

# ALU 16 BIT Ở TRÊN ĐƯỢC THIẾT KẾ TẠI LAB3 TA CÓ :

# 

# Đóng gói ALU 16 BIT :

# 

# Ta thiết kế MUX4\_1 để thiết kế Bộ dịch trái ( tối đa 3 bit )

# 

# Đóng gói MUX4\_1\_1BIT :

# 

# Thiết kế bộ dịch trái dựa vào MUX4\_1\_1BIT ta sẽ được mạch SHIFTLEFT\_1\_2\_3\_BIT :

# 

# 

# Đóng gói SHIFTLEFT\_1\_2\_3 :

# 

# Hạn chế kết quả đầu ra thanh ghi bằng tín hiệu OE ta thiết kế mạch

# 

# Đóng gói mạch

# 

# Từ Trên Ta thành 1 DATAPATH hoàn chỉnh :

# 

# Đóng gói DATAPATH

# 

# CHẠY WAVEFORM DATAPATH

# 

# Giải thích:

# Đầu tiên ta gán WA là A001 trong chu kì đầu để gán vào thanh ghi A001 (RAA và RBB không quan trọng trong chu kì đầu ). Giá trị gán vào A001 thông qua I là 8 (unsigned decimal) và IE là 1 cho phép nạp lệnh vào thanh ghi.Sau khi nạp xong IE = 0

# Trong chu kì tiếp theo thì WA , RAA và RBB đều gán thanh ghi A001 chỉnh OPCODE 000 để thực hiện phép cộng cho 2 thanh ghi là RAA và RBB và kết quả lưu vào WA. Ở đây ta cho tín hiệu RE =1 để đọc giá trị 2 thanh ghi A001 tại RAA và RBB là 8 và 8 nghĩa là 8 + 8 và tín hiệu OE =1 thì kết quả của A001 được xuất ra qua output RESULT là 16

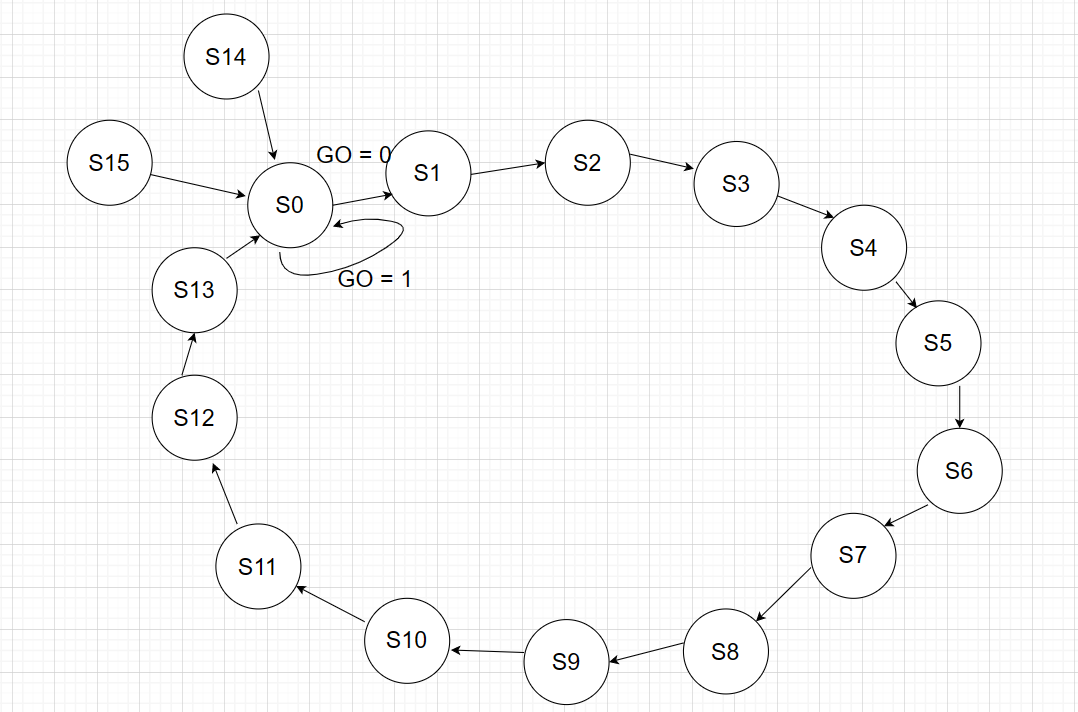
# Gán RAA và RBB là thanh ghi A001,để tín hiệu RE =1 để đọc thanh ghi A001 tại RAA và RBB . Ta gán thanh ghi A001 tại WA để tính toán kết quả, cho OPCODE là 100 để có A001 AND A001 và đặt S là 10 để dịch trái thanh ghi A001 2 bit. Giá trị A001 thay đổi ở chu kì trước là 16 và giờ thì 16 \* 2^2 = 64 thông qua tín hiệu OE =1 và xuất ra bởi output RESULT

# 2. THIẾT KẾ CONTROLLER

# a. Thiết kế máy trạng thái để điều kiển việc hoạt động các khối trong Datapath

* Vì cần 14 bước để hoàn thành việc tính toán nên máy trạng thái gồm 14 trạng thái, dùng 4 D flip flop.

Sơ đồ chuyển trạng thái:



* Thiết kế máy trạng thái khá đơn giản do chỉ cần xung clock để chuyển trạng thái. Riêng trạng thái cuối tùy thuộc vào giá trị biến Go mà quyết định giữ nguyên trạng thái hay quay về trạng thái S0

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| G | Q3 | Q2 | Q1 | Q0 | D3 | D2 | D1 | D0 |
| X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| X | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| X | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| X | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| X | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| X | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| X | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| X | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| X | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| X | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| X | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| X | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| X | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| X | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| X | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |

* Từ Biểu đồ trên ta rút ra biểu thức của D3 , D2 , D1 , D0 dựa vào G,Q3,Q2,Q1,Q0

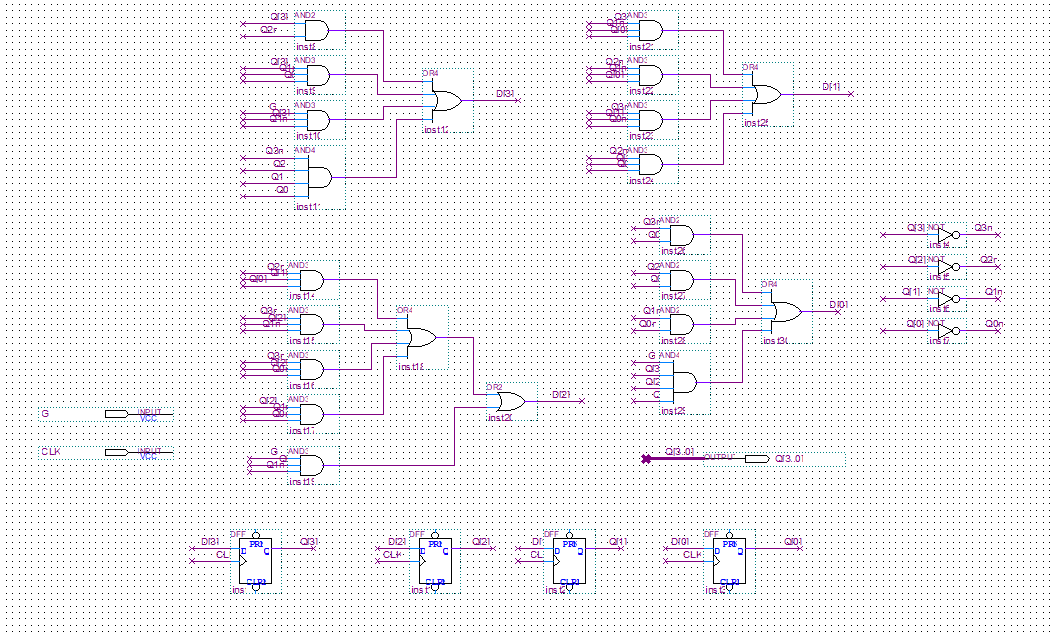
=> D3 = Q3Q2’ + Q3Q1’Q0’ + GQ3Q1’ + Q3’Q2Q1Q0

D2 = Q2’Q1Q0 + Q3’Q2Q1’ + Q3’Q2Q0’ + Q2Q1’Q0’ + GQ2Q1’

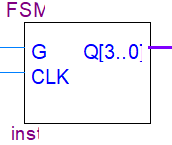
D1 = Q3’Q1’Q0 + Q2’Q1’Q0 + Q3’Q1Q0’ + Q2’Q1Q0’

D0 = Q3’Q0’ + Q2’Q0’ + Q1’Q0’ + GQ3Q2Q1’

Từ 4 biểu thức trên ta xây dựng FSM

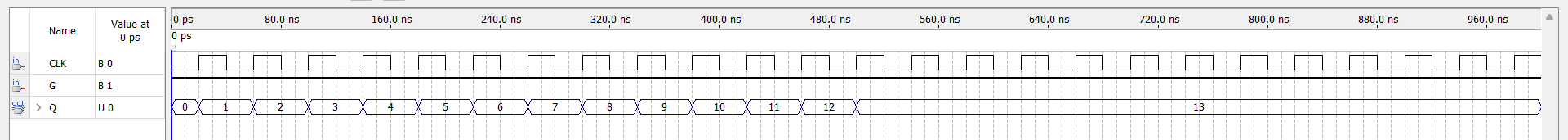


* Đóng gói FSM

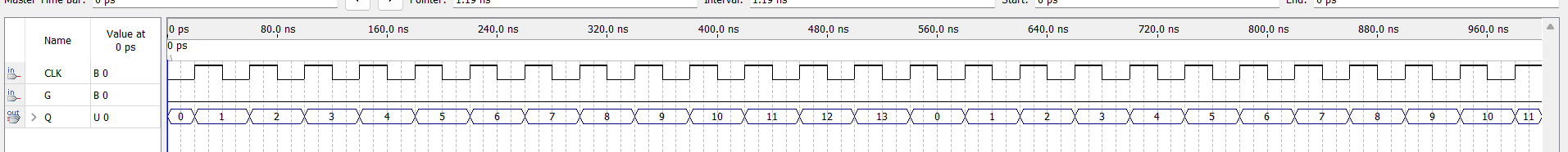


**Chạy WAVEFORM FSM**

Khi G = 1



Khi G = 0



* Giải thích : G = 0 thì vòng lặp từ State 13 về State0 và cứ tiếp tục. Còn G = 1 thì sẽ dừng tại State13

# b. Thiết kế khối Output để điều khiển và xuất kết quả:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| State | Q3 | Q2 | Q1 | Q0 | IE | WE | RE | WA | RAA | RBB | OPCODE | S | OE |
| S0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 000 | xxx | xxx | xxx | xx | x |
| S1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 001 | xxx | xxx | xxx | xx | x |
| S2 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 010 | xxx | xxx | xxx | xx | x |
| S3 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 011 | xxx | xxx | xxx | xx | x |
| S4 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 101 | 010 | 010 | 100 | 11 | 0 |
| S5 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 010 | 101 | 010 | 010 | 00 | 0 |
| S6 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 001 | 001 | 001 | 100 | 10 | 0 |
| S7 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 110 | 000 | 000 | 100 | 10 | 0 |
| S8 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 101 | 000 | 000 | 100 | 01 | 0 |
| S9 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 000 | 101 | 110 | 000 | 00 | 0 |
| S10 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 101 | 011 | 010 | 000 | 00 | 1 |
| S11 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 101 | 101 | 000 | 000 | 00 | 1 |
| S12 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 100 | 101 | 001 | 010 | 00 | 1 |
| S13 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 100 | 100 | 100 | 100 | 00 | 1 |
| S14 | 1 | 1 | 1 | 0 | x | x | x | xxx | xxx | xxx | xxx | xx | x |
| S15 | 1 | 1 | 1 | 1 | x | x | x | xxx | xxx | xxx | xxx | xx | x |

# Cấp input là Q3,Q2,Q1,Q0

# 

# Ta rút biểu thức của IE , WE, RE, WA, RAA. RBB, OPCODE, S, OE dựa vào Q3, Q2, Q1, Q0

# IE = Q3’Q2’ WE = 1 = VCC

# 

# RE = Q3 + Q2

# 

# WA2 = Q3Q0’ + Q3Q1 + Q3Q2 + Q2Q1’Q0’ + Q2Q1Q0

# WA1 = Q3’Q2’Q1 + Q3’Q2Q0

# WA0 = Q3Q1 + Q3’Q2’Q0 + Q3’Q2Q0’ + Q3Q2’Q0’

# 

# RAA2 = Q2’Q0 + Q1’Q0 +Q3Q2

# RAA1 = Q3’Q1’Q0’ + Q2’Q1Q0’

# RAA0 = Q2’Q0 + Q1Q0’ + Q3’Q1’Q0 + Q3Q2Q0’

# 

# RBB2 = Q3Q1’Q0

# RBB1= Q3’Q1’ + Q2’Q1’Q0 + Q2’Q1Q0’

# RBB0 = Q3’Q1Q0’ + Q3Q2Q0’

# 

# OPCODE2 = Q3’Q0’ + Q3’Q1 + Q2’Q1’Q0’ + Q3Q2Q0

# OPCODE1 = Q3’Q1’Q0 + Q3Q2Q0’

# OPCODE0 = 0 = GND

# 

# S1 = Q3’Q0’ + Q3’Q1

# S0 = Q3’Q1’Q0’ + Q2’Q1’Q0’

# 

# OE = Q2’Q1 + Q3Q2

# 

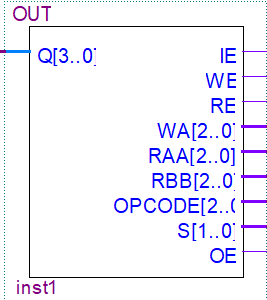
# Các đường output

# 

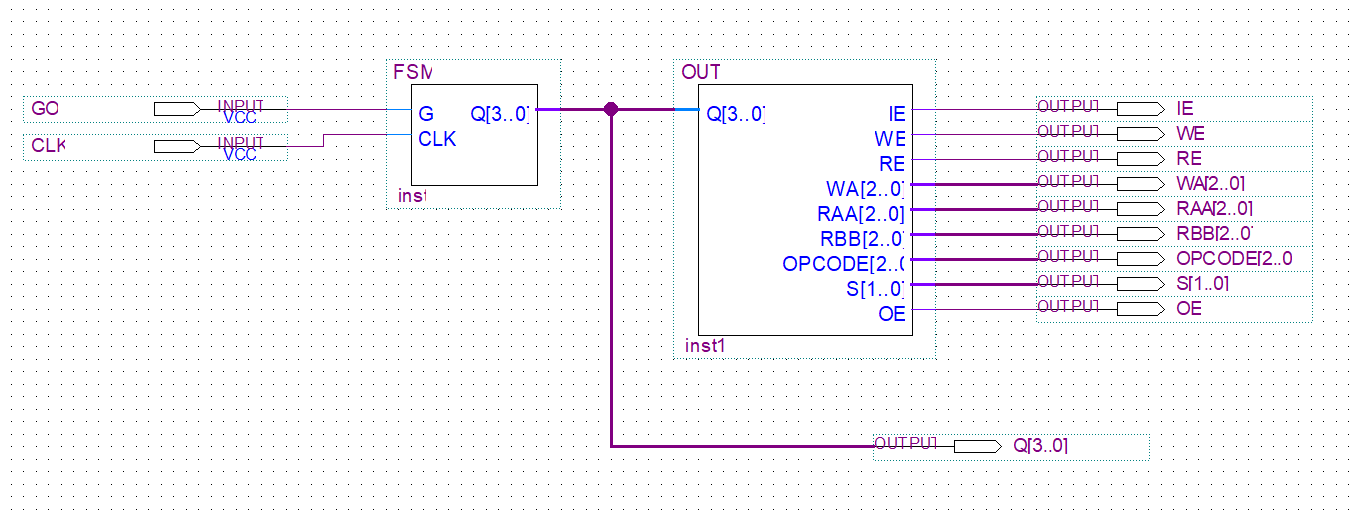
# Toàn bộ khối OUTPUT

# 

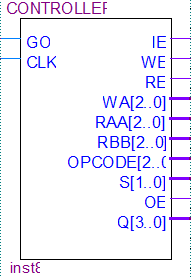
* Đóng gói khối Output



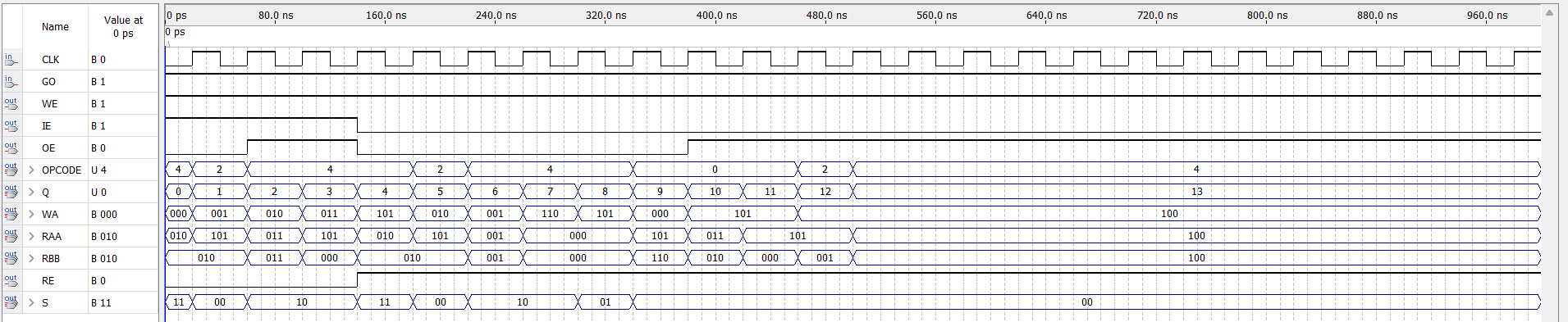
**KHỐI CONTROLLER HOÀN CHỈNH**

****

* Đóng gói CONTROLLER

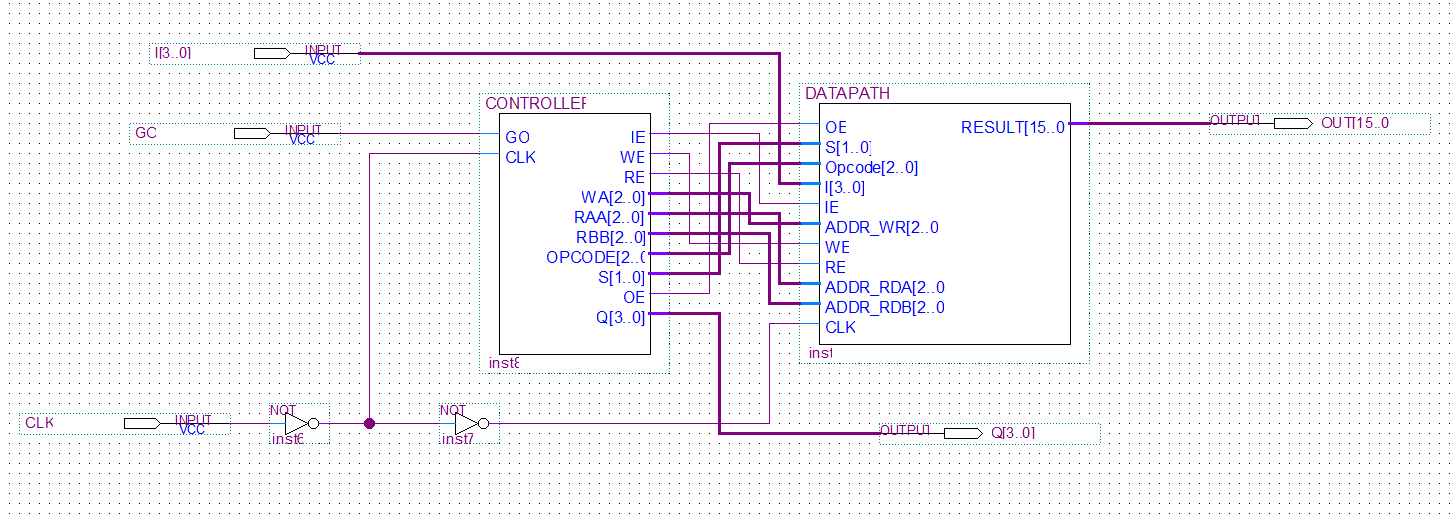


**CHẠY WAVEFORM CONTROLLER**

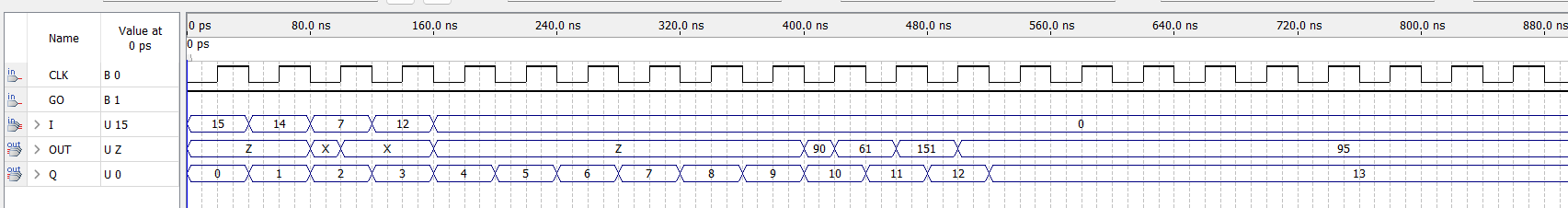
****

* Giải thích : WAVEFORM đã chạy đúng vì các thông tin GO, WE, IE, OE, OPCODE, Q, WA, RAA, RBB , RE,S đã có kết quả đúng với kết quả trên bảng đã xác định

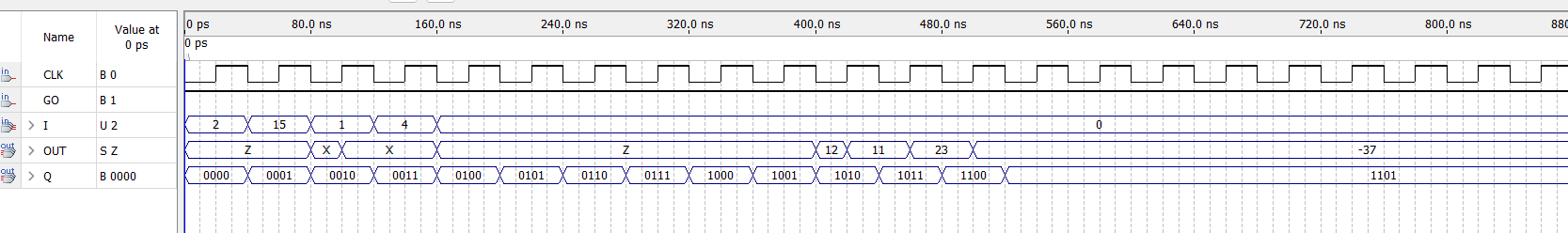
**TỪ KHỐI CONTROLLER VÀ DATAPATH TA CÓ KHỐI PROCESSOR HOÀN CHỈNH**

****

**CHẠY WAVEFORM PROCESSOR**

****

* Giải thích : ta nạp 4 giá trị đầu vào 4 thanh ghi A000, A001, A010, A011 lần lượt là 15, 14, 7, 12 tức là ta sẽ có biểu thức 12 + 7\*7 – 4\*14 + 6\*15 = 95 được xuất ra tại State 12 và State 13 trở đi ( Do State 13 SUM -> O ). Các State khác (ngoại trừ State 10,11,12,13 ) bị hạn chế đầu ra do tín hiệu OE = 0

****

* Giải thích : ta nạp 4 giá trị đầu vào 4 thanh ghi A000, A001, A010, A011 lần lượt là 2, 15, 1, 4 ta có 4 + 7\*1 – 4\*15 + 6\*2 = -37.

**=> Vậy ta đã thực hiện xong việc thiết kế vi xử lý đơn giản để tính toán biểu thức: I3 + 7I2 – 4I1 + 6I0 .**