

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



**Tài liệu hướng dẫn thực hành:**  
**THIẾT KẾ VI MẠCH VỚI VERILOG HDL**

Tác giả: Lâm Đức Khải

LUU HÀNH NỘI BỘ  
Thành phố Hồ Chí Minh – Tháng 10/2017

# **MỤC LỤC**

Bài 1. HƯỚNG DẪN THỰC HÀNH THIẾT KẾ VI MẠCH DÙNG VERILOG TRÊN MODELSIM	1
Bài 2. THIẾT KẾ MẠCH ĐẾM ĐỒNG BỘ CÓ KHẢ NĂNG NẠP GIÁ TRỊ BAN ĐẦU	13
2.1 Mục tiêu	13
2.2 Nội dung thực hành	13
2.3 Sinh viên chuẩn bị trước ở nhà	13
2.4 Hướng dẫn thực hành	13
Bài 3. THIẾT KẾ MẠCH TUẦN TỰ BẰNG MÔ HÌNH MÁY TRẠNG THÁI HỮU HẠN	15
3.1 Mục tiêu	15
3.2 Nội dung thực hành	15
3.3 Sinh viên chuẩn bị trước ở nhà	15
3.4 Hướng dẫn thực hành	15
Bài 4. THIẾT KẾ ALU	17
4.1 Mục tiêu	17
4.2 Nội dung thực hành	17
4.3 Sinh viên chuẩn bị ở nhà	17
4.4 Hướng dẫn thực hành	18
Bài 5. THIẾT KẾ DATAPATH ĐƠN GIẢN	20
5.1 Mục tiêu	20
5.2 Nội dung thực hành	20

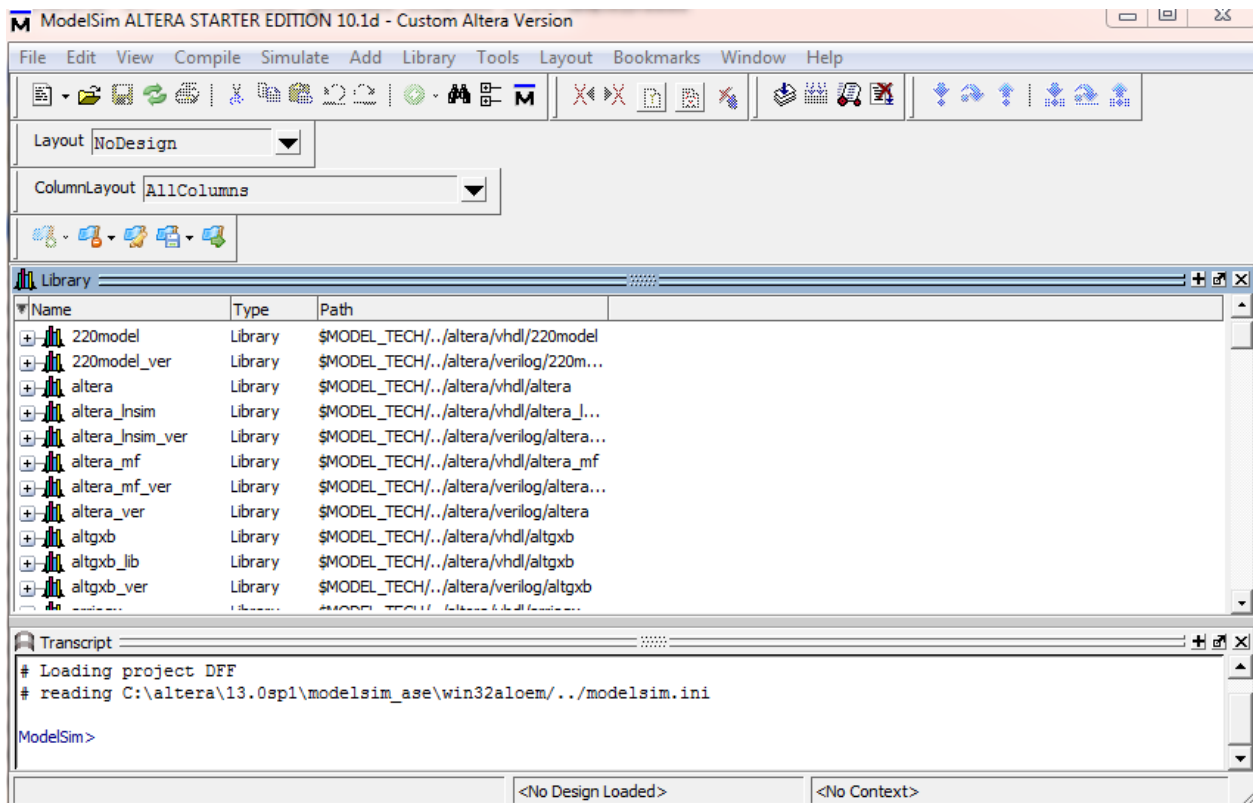
5.3	Sinh viên chuẩn bị ở nhà	21
5.4	Hướng dẫn thực hành	21
Bài 6. THIẾT KẾ SIMPLE CONTROL UNIT		22
6.1	Mục tiêu	22
6.2	Nội dung thực hành	22
6.3	Sinh viên chuẩn bị ở nhà	23
6.4	Hướng dẫn thực hành	24
Bài 7. THIẾT KẾ SIMPLE PROCESSOR		25
7.1	Mục tiêu	25
7.2	Nội dung thực hành	25
7.3	Sinh viên chuẩn bị ở nhà	26
7.4	Hướng dẫn thực hành	26

## **NỘI QUY THỰC HÀNH**

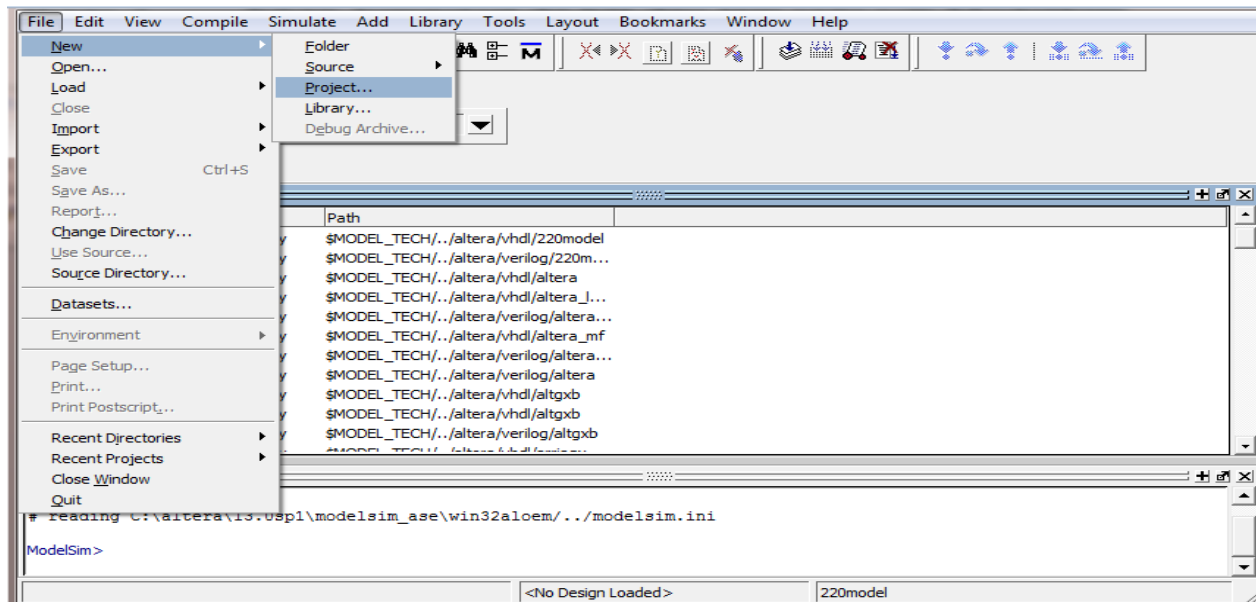
1. Sinh viên tham dự đầy đủ các buổi thực hành theo quy định của giảng viên hướng dẫn (GVHD) (6 buổi với lớp học cách tuần và 10 buổi với lớp học liên tục).
2. Sinh viên phải chuẩn bị các phần từ mục 1 tới mục 3 trong phần “Hướng dẫn thực hành” trước khi đến lớp. GVHD sẽ kiểm tra bài chuẩn bị của sinh viên trong 15 phút đầu của buổi học (nếu không có bài chuẩn bị -> tính vắng buổi học đó).
3. Sinh viên phải chạy thiết kế của mình trên phần mềm mô phỏng ModelSim để kiểm tra thiết kế, sau đó giảng viên sẽ kiểm tra để hoàn thành bài thực hành.

# Bài 1. HƯỚNG DẪN THỰC HÀNH THIẾT KẾ VI MẠCH DÙNG VERILOG TRÊN MODELSIM

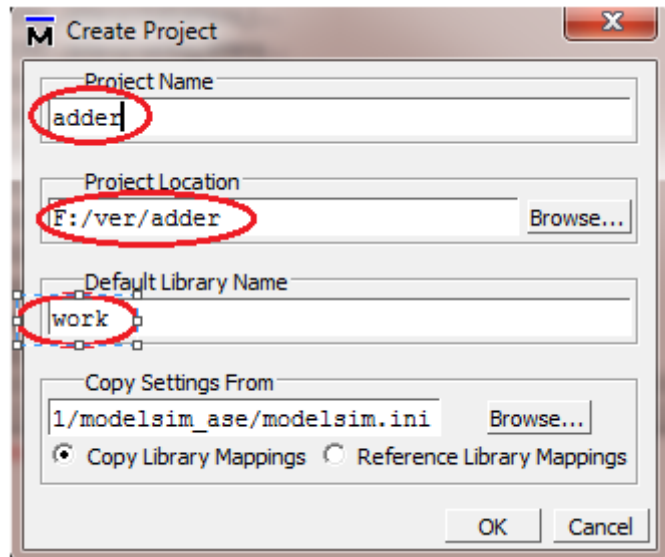
## Bước 1. Mở phần mềm ModelSim



## Bước 2. Tạo project. Bằng cách chọn file -> new -> project

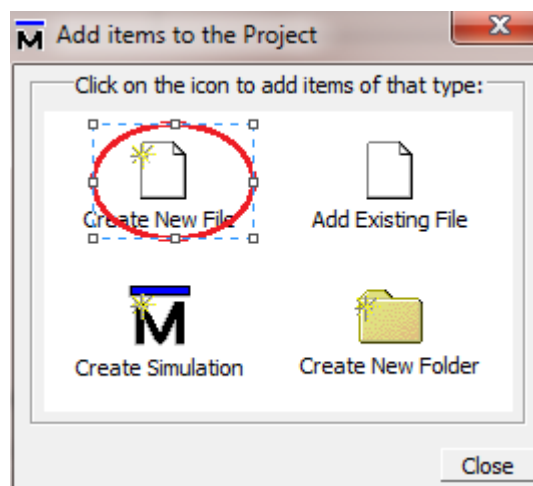


**Bước 3.** Điền thông tin về tên project, đường dẫn cũng như tên Library cho project mới. Ta có thể đặt tên cho project cũng như đường dẫn của project tùy ý



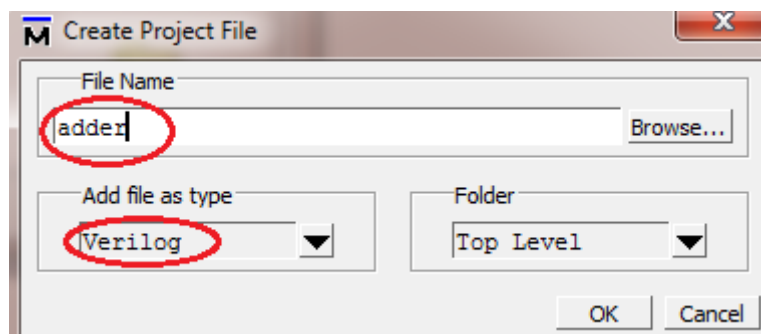
Hình 5.3 Cửa sổ điền thông tin cho project

**Bước 4.** Nhấn **OK**, một cửa sổ mới hiện ra, chọn **Create New File**



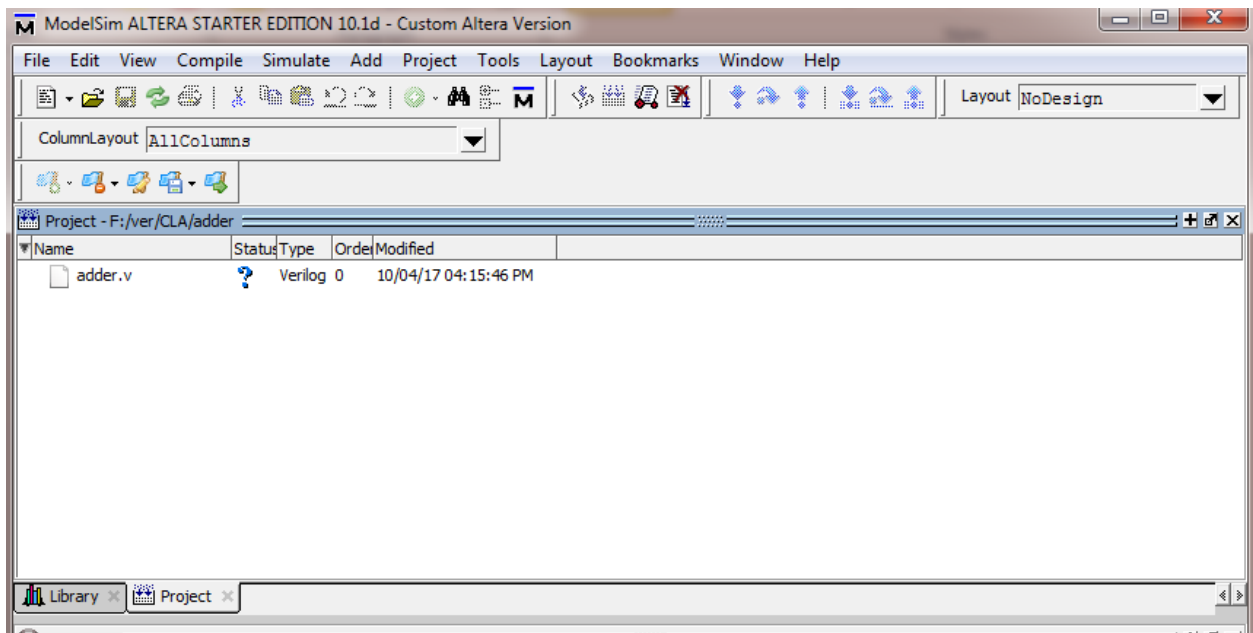
Hình 5.4 Tạo new file

**Bước 5.** Nhập tên file muốn tạo và chọn loại



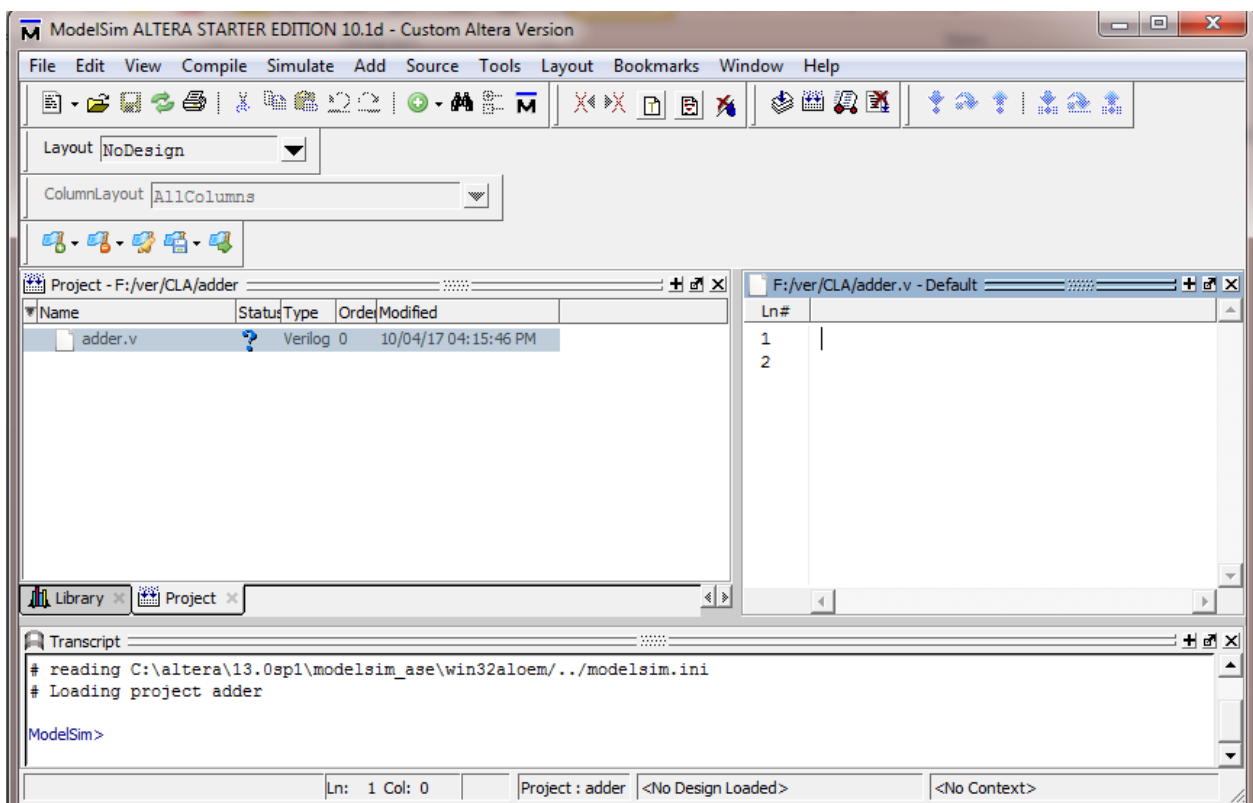
Hình 5.5 Điền thông tin cho new file

**Bước 6.** Nhấn **OK**, quay trở về cửa sổ trong bước 4, nhấn **Close**



Hình 5.6 Cửa sổ sau khi tạo project

**Bước 7.** Double click vào Adder.v bên màn hình Workspace, một cửa sổ dùng cho việc mô tả thiết kế dùng Verilog xuất hiện



Hình 5.7 Cửa sổ dùng cho mô tả thiết kế

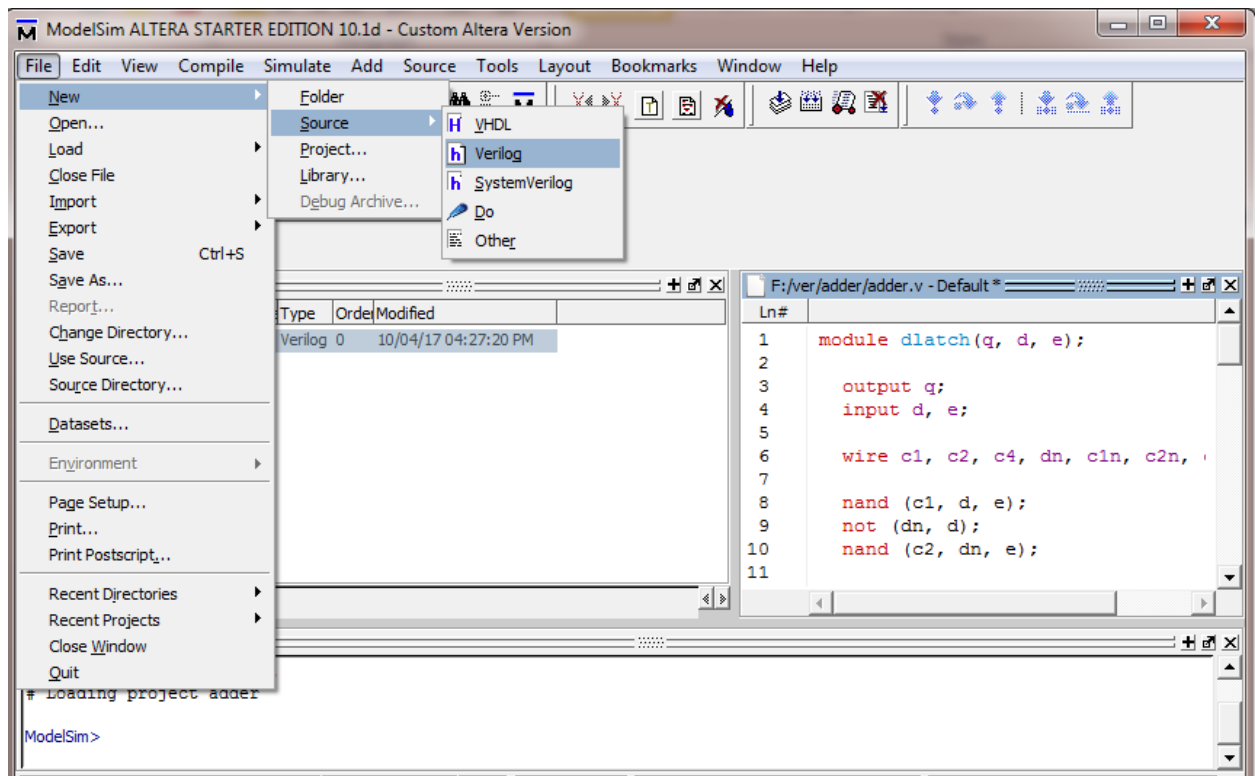
**Bước 8.** Dùng ngôn ngữ Verilog mô tả thiết kế, ở đây ta đang thực hiện ví dụ là mô tả thiết kế một bộ adder 3bit.

Ln#	
1	module CLA_3b(A, B, Cin, S);
2	input [2:0] A, B;
3	input Cin;
4	wire Cout;
5	output [3:0] S;
6	wire [2:0] G, P;
7	wire [1:0] C;
8	wire [5:0] W;
9	
10	and a0(G[0], A[0], B[0]);
11	xor x0(P[0], A[0], B[0]);
12	and a1(G[1], A[1], B[1]);
13	xor x1(P[1], A[1], B[1]);
14	and a2(G[2], A[2], B[2]);
15	xor x2(P[2], A[2], B[2]);
16	
17	and pc0(W[0], P[0], Cin);
18	or o0(C[0], W[0], G[0]);
19	
20	and pc1(W[1], W[0], P[1]);
21	and pc2(W[2], G[0], P[1]);
22	or o1(C[1], G[1], W[1], W[2]);
23	
24	and pc3(W[3], W[1], P[2]);
25	and pc4(W[4], W[2], P[2]);
26	and pc5(W[5], G[1], P[2]);
27	or o2(Cout, G[2], W[3], W[4], W[5]);
28	
29	xor s0(S[0], Cin, P[0]);
30	xor s1(S[1], C[0], P[1]);
31	xor s2(S[2], C[1], P[2]);
32	and s4(S[3], Cout, 1);
33	
34	endmodule

Hình 5.8 Mô tả thiết kế adder

**Bước 9.** Tạo Testbench để định nghĩa dạng sóng cho các tín hiệu input của thiết kế. Ta mở tiếp một cửa sổ khác cũng dùng Verilog để thiết kế bằng cách chọn tab File -> New -> Source -> Verilog





Hình 5.9 Tạo new file

**Bước 10.** Một cửa sổ mới dùng Verilog để mô tả thiết kế khác xuất hiện, ta mô tả testbench cho thiết kế như sau

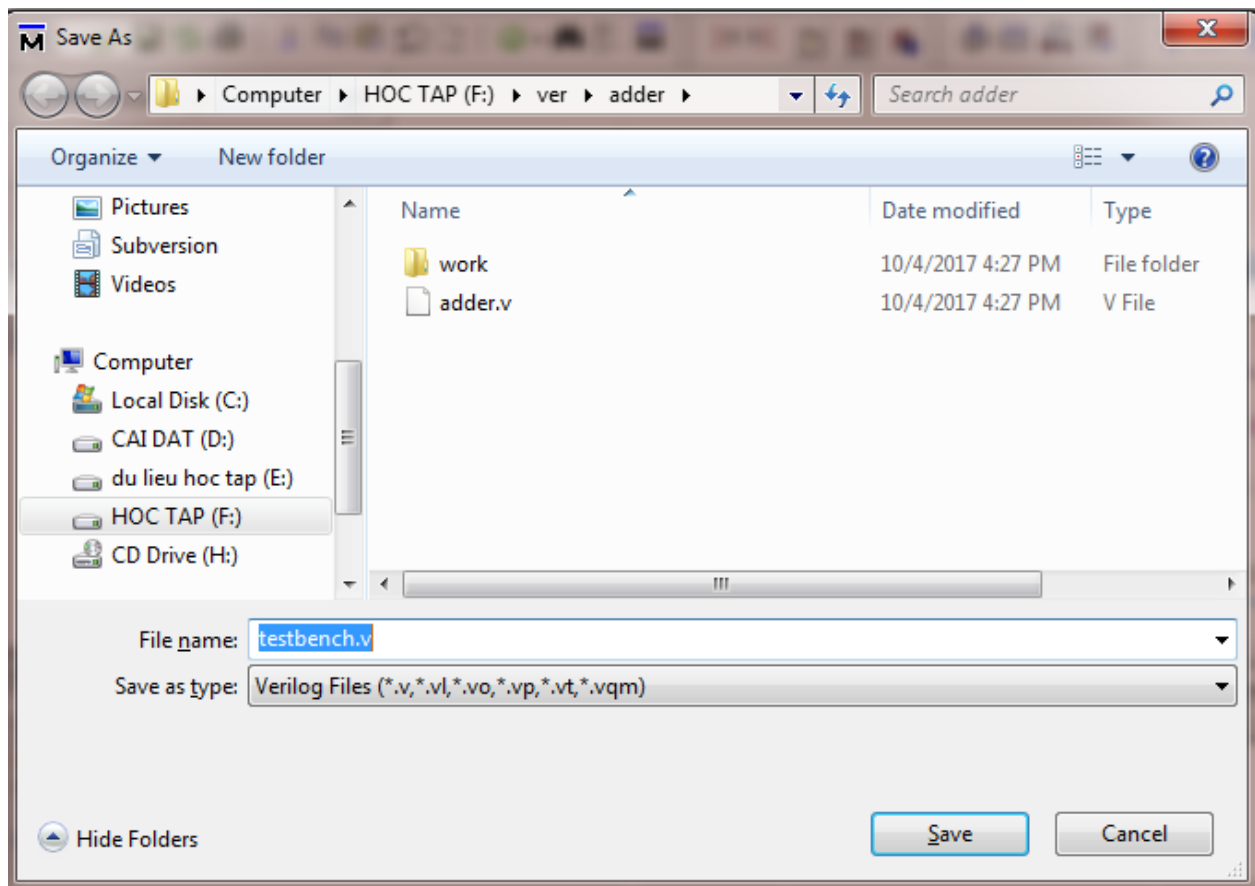
```

1  module CLA_M(A, B, Cin, S);
2      input [2:0] A, B;
3      input Cin;
4      output [3:0] S;
5
6      assign S = (A + B + Cin);
7  endmodule
8
9  `timescale 1ns/100ps
10 module CLA_testbench();
11     reg[2:0] A, B;
12     reg Cin;
13     wire[3:0] S, mS;
14
15     initial
16     begin
17         #640 $stop;
18     end
19
20     initial
21     begin
22         Gen();
23     end
24
25     CLA_3b Inst1(.A(A), .B(B), .Cin(Cin), .S(S));
26     CLA_M Inst2(.A(A), .B(B), .Cin(Cin), .S(mS));
27
28     task Gen;
29     begin
30         A <= 0;
31         B <= 0;
32         Cin <= 0;
33         forever begin
34             #5 A <= (A + 1);
35             if(A == 3'b111)
36                 begin

```

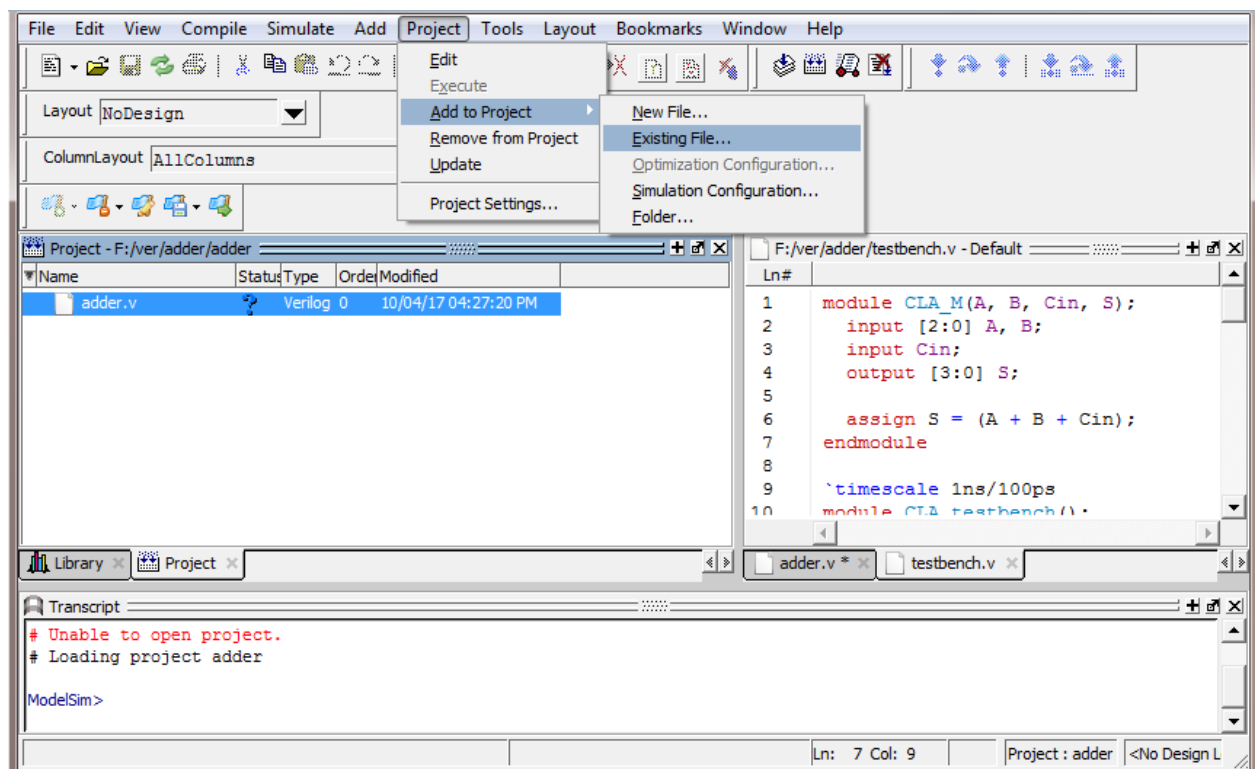
Hình 5.10 Mô tả Testbench cho thiết kế

**Bước 11.** Lưu testbench cho thiết kế với tên Testbench.v



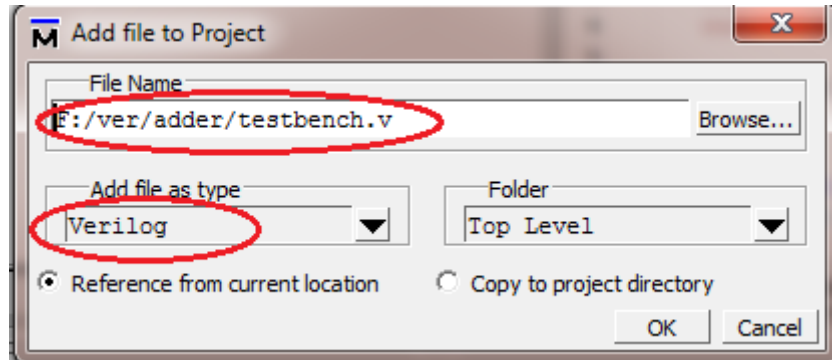
Hình 5.11 Lưu mô tả testbench

**Bước 12.** Add file Testbench.v vừa tạo vào project, chọn tab Project ->  
Add to Project -> Existing File



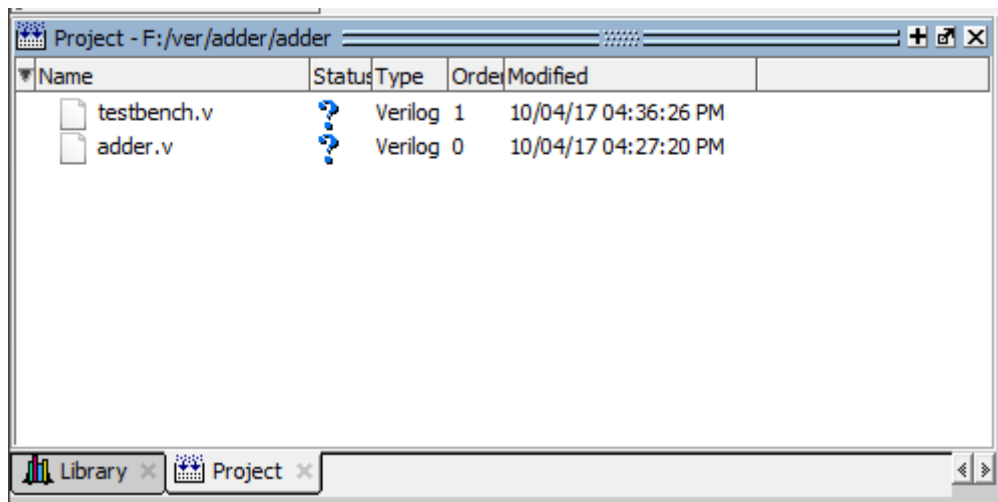
Hình 5.12 Add file Testbench vào project

### Bước 13. Chỉ đường dẫn đến file Testbench



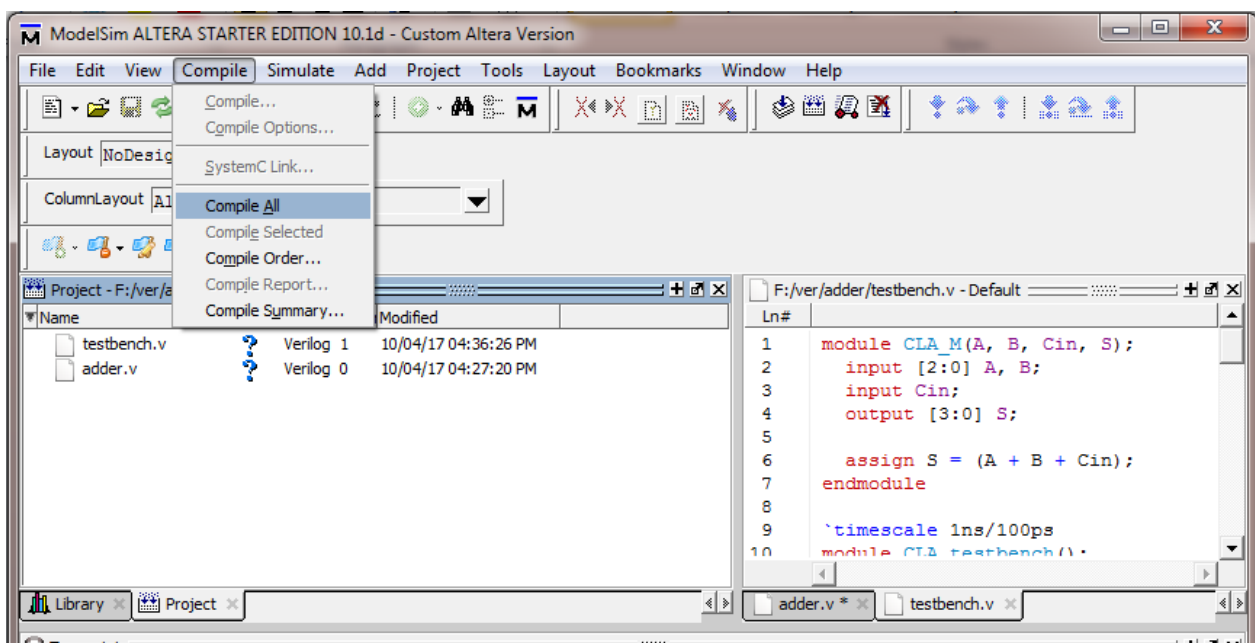
Hình 5.13 Chỉ đường dẫn đến file Testbench

### Bước 14. Cửa sổ Workspace xuất hiện như sau



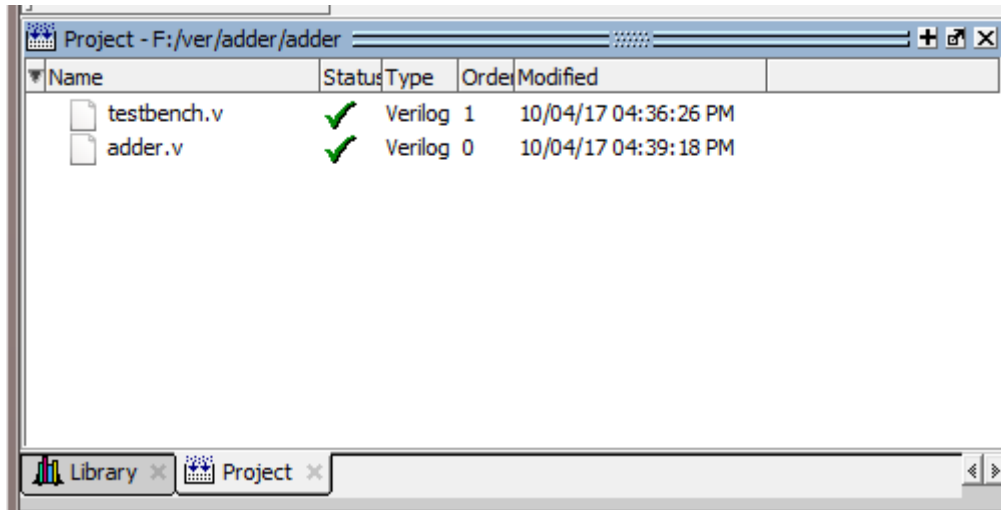
Hình 5.14 Cửa sổ Workspace sau thiết kế

### Bước 15. Compile mô tả thiết kế, chọn tab Compile -> Compile All



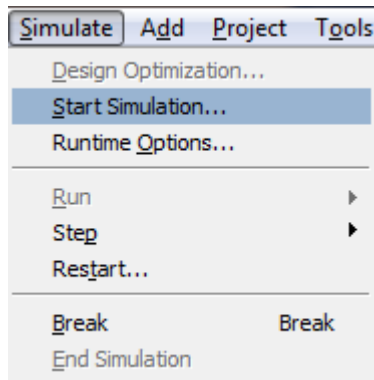
Hình 5.15 Compile thiết kế

**Bước 16.** Nếu mô tả thiết kế không có lỗi thì sẽ có thông báo sau



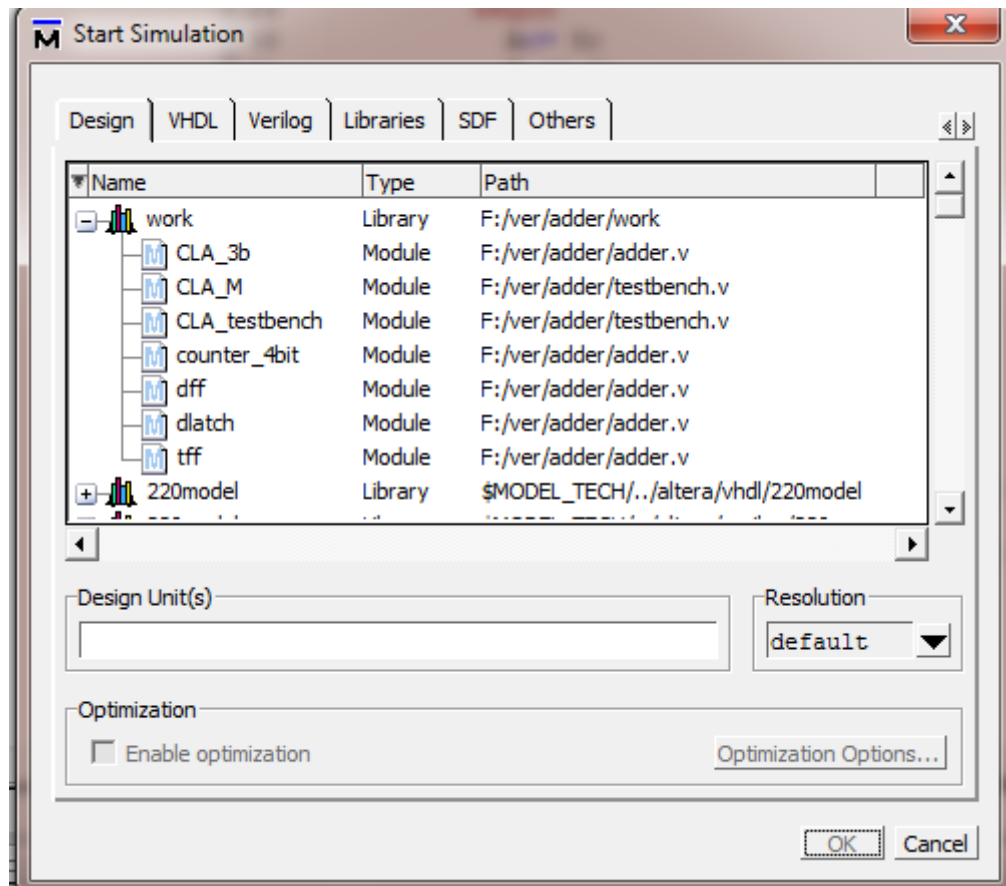
Hình 5.16 Compile thành công

**Bước 17.** Sau khi ta mô tả thiết kế và mô tả testbench cho thiết kế thành công ta sẽ thực hiện các bước chạy mô phỏng, chọn tab Simulate -> Start Simulation



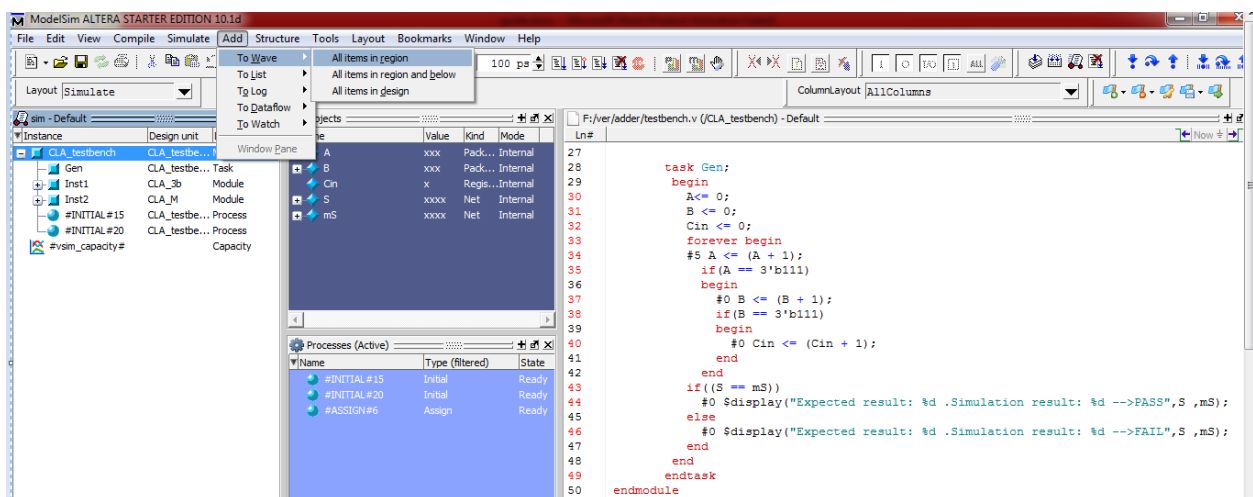
Hình 5.17 Thiết lập mô phỏng

**Bước 18.** Một cửa sổ như hình dưới xuất hiện, ta chọn tab Design -> work -> Testbench. Nhấn OK



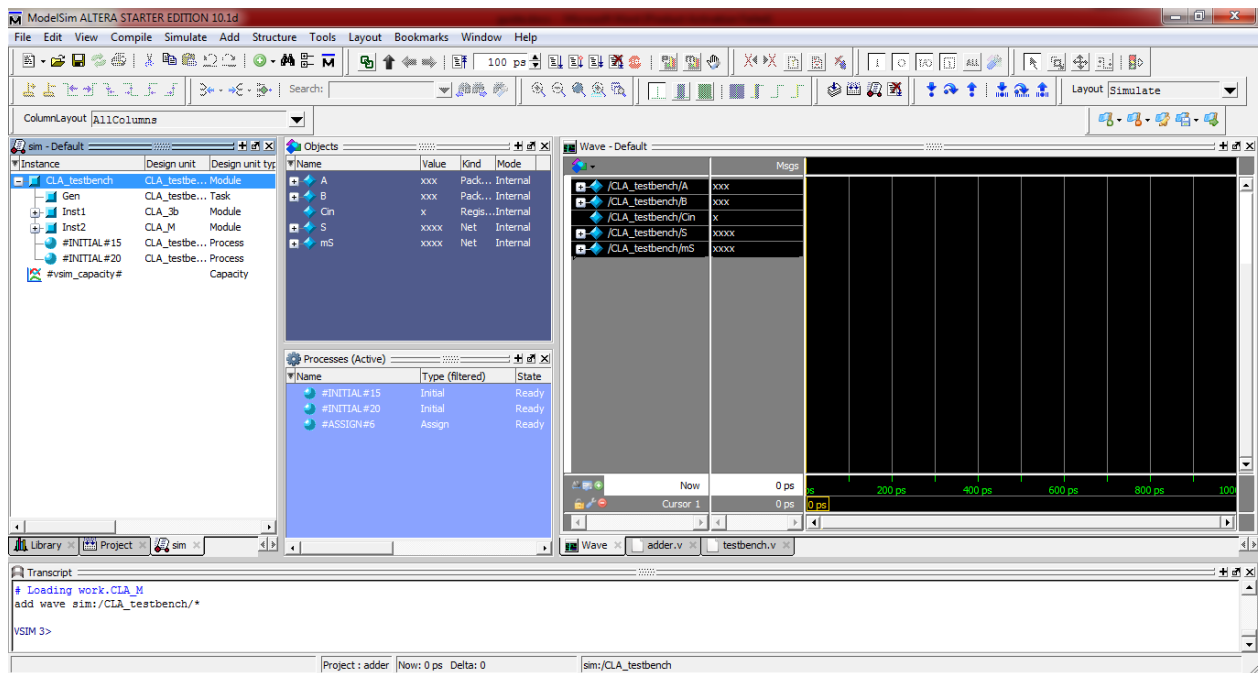
Hình 5.18 Chọn thiết kế cần mô phỏng

**Bước 19.** Một cửa sổ như sau xuất hiện, Click phải chuột lên Testbench ->Add -> To Wave -> All items in region



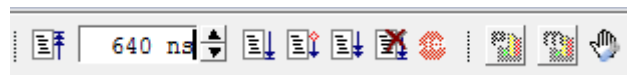
Hình 5.19 Chọn tín hiệu dạng sóng cần quan sát

**Bước 20.** Cửa sổ dạng sóng được mở ra



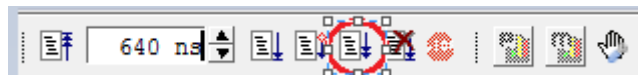
Hình 5.20 Cửa sổ dạng sóng

**Bước 21.** Chọn khoảng thời gian chạy mô phỏng



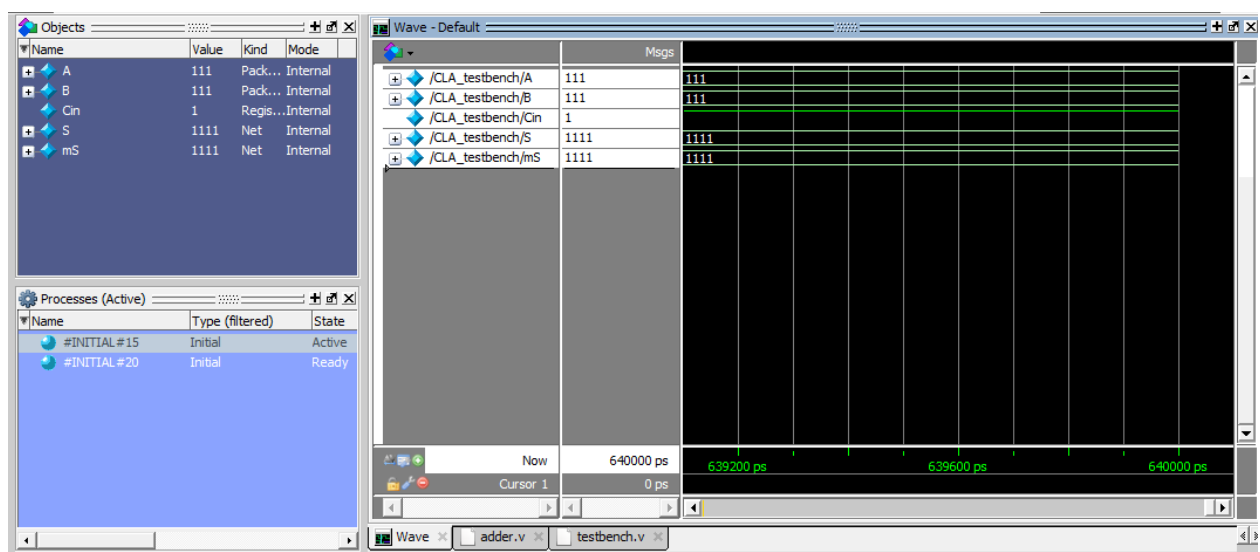
Hình 5.21 Thiết lập thời gian chạy mô phỏng

**Bước 22.** Nhấn nút chạy mô phỏng

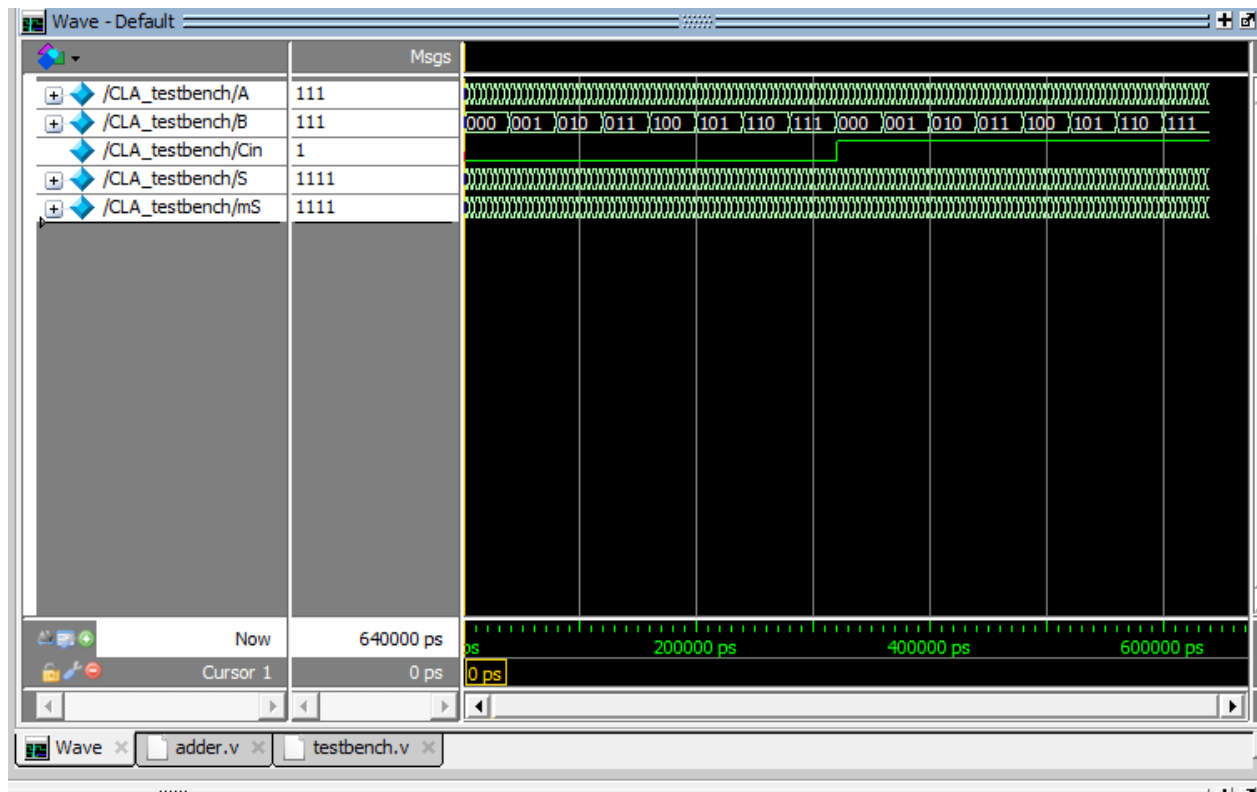


Hình 5.22 Chạy mô phỏng

**Bước 23.** Sau khi chạy xong



**Bước 24.**nhấn nút “Zoom Full” để thấy toàn bộ dạng sóng



Hình 5.24 Dạng sóng sau mô phỏng

**Bước 25.** Quan sát ta thấy, tín hiệu ngõ ra không có delay timing so với các tín hiệu ngõ vào. Bởi vì đây là mô phỏng pre-synthesis nên trong quá trình chạy mô phỏng, thông tin về định thời cho các node bên trong thiết kế không được cung cấp, do đó delay timing giữa các node này là bằng 0.

**Bước 26.** Quan sát dạng sóng và debug function nếu có sai.

**Kiểm tra kết quả sử dụng Testbench :** Nếu các bạn thấy việc quan sát dạng sóng và debug khá tốn thời gian. Bạn cũng có thể kiểm tra xem mạch của mình đúng hay sai bằng cách sau. Bạn có thể viết một module khác thực hiện chức năng đó, sau đó kiểm tra kết quả của 2 module và in nó ra.

**Bước 1:** Thiết kế module

```

1  module CLA_M(A, B, Cin, S);
2      input [2:0] A, B;
3      input Cin;
4      output [3:0] S;
5
6      assign S = (A + B + Cin);
7  endmodule
8

```

Bước 2 : Chạy testbench bao gồm cả module viết thêm và module chính

```

9   `timescale 1ns/100ps
10  module CLA_testbench();
11      reg[2:0] A, B;
12      reg Cin;
13      wire[3:0] S ,mS;
14
15      initial
16      begin
17          #640 $stop;
18      end
19
20      initial
21      begin
22          Gen();
23      end
24
25      CLA_3b Inst1(.A(A), .B(B), .Cin(Cin), .S(S));
26      CLA_M Inst2(.A(A), .B(B), .Cin(Cin), .S(mS));
27
28      task Gen;
29      begin
30          A <= 0;
31          B <= 0;
32          Cin <= 0;
33          forever begin
34              #5 A <= (A + 1);
35              if(A == 3'b111)
36              begin
37                  #0 B <= (B + 1);
38                  if(B == 3'b111)
39                  begin
40                      #0 Cin <= (Cin + 1);
41                  end
42              end

```

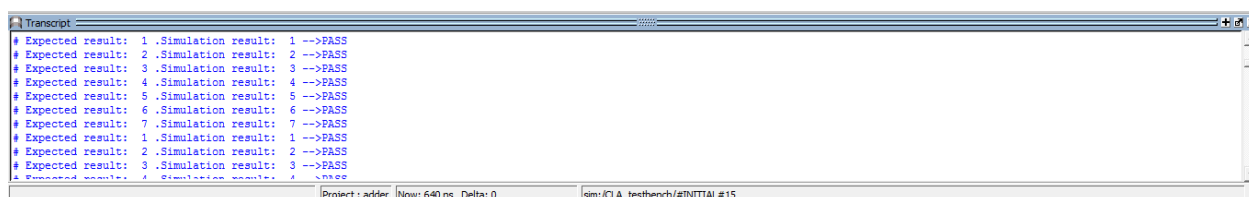
### Bước 3: Hiện thị kết quả

```

43      if ( (S == mS) )
44          #0 $display("Expected result: %d .Simulation result: %d -->PASS",S ,mS);
45      else
46          #0 $display("Expected result: %d .Simulation result: %d -->FAIL",S ,mS);
47      end
48      end
49      endtask
50  endmodule
--

```

### Bước 4: Quan sát kết quả nhận được



```

Transcript
# Expected result: 1 .Simulation result: 1 -->PASS
# Expected result: 2 .Simulation result: 2 -->PASS
# Expected result: 3 .Simulation result: 3 -->PASS
# Expected result: 4 .Simulation result: 4 -->PASS
# Expected result: 5 .Simulation result: 5 -->PASS
# Expected result: 6 .Simulation result: 6 -->PASS
# Expected result: 7 .Simulation result: 7 -->PASS
# Expected result: 1 .Simulation result: 1 -->PASS
# Expected result: 2 .Simulation result: 2 -->PASS
# Expected result: 3 .Simulation result: 3 -->PASS
# Expected result: 4 .Simulation result: 4 -->PASS
# Expected result: 5 .Simulation result: 5 -->PASS
# Expected result: 6 .Simulation result: 6 -->PASS
# Expected result: 7 .Simulation result: 7 -->PASS
Project : adder Now: 640 ns Delta: 0 sim:/CLA_testbench/#INITIAL#15

```



---

## Bài 2. THIẾT KẾ MẠCH ĐẾM ĐỒNG BỘ CÓ KHẢ NĂNG NẠP GIÁ TRỊ BAN ĐẦU

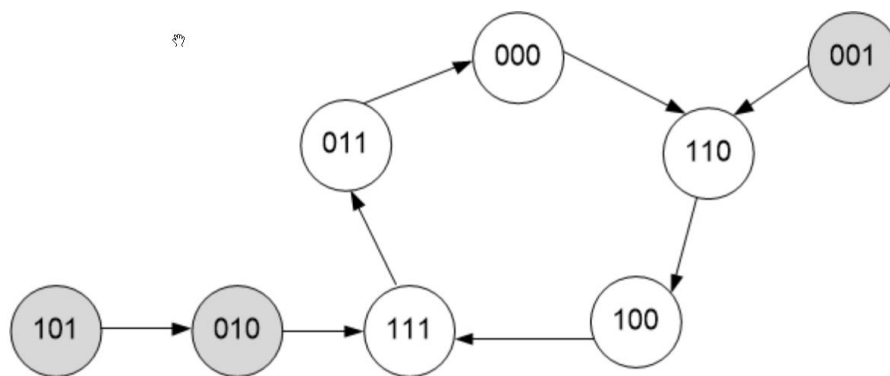
### 2.1 Mục tiêu

Mục tiêu của bài này là giúp sinh viên:

- ✚ Sử dụng ngôn ngữ Verilog HDL, thiết kế được một mạch đếm đồng bộ.

### 2.2 Nội dung thực hành

Trong Lab này, sinh viên sử dụng mô hình hành vi (behavioral model) trong ngôn ngữ Verilog HDL để thiết kế một mạch đếm đồng bộ có chu trình đếm như sơ đồ trong Hình 1-1.



*Hình 2-1 Sơ đồ chuyển trạng thái của bộ đếm*

### 2.3 Sinh viên chuẩn bị trước ở nhà

- ✚ Sử dụng ngôn ngữ Verilog HDL thiết kế bộ đếm như trên Hình 1-1 với giá trị ban đầu của mạch đếm được nạp vào thông qua các chân Preset và Clear trên phần mềm mô phỏng ModelSim.
- ✚ Viết testbench tạo giá trị cho các tín hiệu input và chạy mô phỏng kiểm tra chức năng của thiết kế.

### 2.4 Hướng dẫn thực hành

#### Bài thực hành 1:

1. Tạo một project mới trên phần mềm ModelSim, đặt tên: E/CE221-Lab/Lab1\_1-MSSV.
2. Sử dụng ngôn ngữ Verilog HDL thiết kế bộ đếm như trên Hình 1-1 với giá trị ban đầu của mạch đếm được nạp vào thông qua các chân Preset và Clear trên phần mềm mô phỏng ModelSim.
3. Viết testbench tạo giá trị cho các tín hiệu input và chạy mô phỏng kiểm tra chức năng của thiết kế.

#### Bài thực hành 2:

- 
1. Tạo một project mới trên phần mềm ModelSim, đặt tên: E/CE221-Lab/Lab1\_2-MSSV.
  2. Sử dụng ngôn ngữ Verilog HDL, thiết kế một tập gồm 32 thanh ghi, mỗi thanh ghi 4 byte. Tập thanh ghi (Register File) có các tín hiệu sau: ReadAddress1[4:0], ReadAddress2[4:0], WriteAddress[4:0], WriteData[31:0], ReadData1[31:0], ReadData2[31:0], ReadWriteEn.
  3. Viết testbench tạo giá trị cho các tín hiệu input và chạy mô phỏng kiểm tra chức năng của thiết kế.

---

# Bài 3. THIẾT KẾ MẠCH TUẦN TỰ BẰNG MÔ HÌNH MÁY TRẠNG THÁI HỮU HẠN

## 3.1 Mục tiêu

- ✚ Sử dụng ngôn ngữ Verilog HDL, thiết kế một mạch tuần tự bằng mô hình máy trạng thái hữu hạn (FSM).

## 3.2 Nội dung thực hành

Sử dụng ngôn ngữ Verilog HDL thiết kế một mạch tuần tự theo mô hình máy trạng thái kiểu Moore. Hệ tuần tự này có chức năng phát hiện 3 bit ngõ vào (X) liên tiếp có giá trị là 010 thì ngõ ra  $Z = 1$ .

## 3.3 Sinh viên chuẩn bị trước ở nhà

- ✚ Sử dụng ngôn ngữ Verilog HDL thiết kế một mạch tuần tự theo mô hình máy trạng thái kiểu Moore có sơ đồ khối như trong Hình 2-1. Hệ tuần tự này có chức năng phát hiện 3 bit ngõ vào (X) liên tiếp có giá trị là 010 thì ngõ ra  $Z = 1$ .
- ✚ Viết testbench tạo giá trị cho các tín hiệu input và chạy mô phỏng kiểm tra chức năng của thiết kế.

## 3.4 Hướng dẫn thực hành

---

### **Bài thực hành 1:**

1. Tạo một project mới trên phần mềm Quartus II, đặt tên: E/CE221-Lab/Lab2\_1-MSSV.
2. Sử dụng ngôn ngữ Verilog HDL thiết kế một mạch tuần tự theo mô hình máy trạng thái kiểu Moore. Hệ tuần tự này có chức năng phát hiện 3 bit ngõ vào (X) liên tiếp có giá trị là 010 thì ngõ ra  $Z = 1$ .
3. Viết testbench tạo giá trị cho các tín hiệu input và chạy mô phỏng kiểm tra chức năng của thiết kế.

### **Bài thực hành 2:**

1. Tạo một project mới trên phần mềm Quartus II, đặt tên: E/CE221-Lab/Lab2\_2-MSSV.
2. Sử dụng ngôn ngữ Verilog HDL, hiện thức thiết kế bộ nhớ dữ liệu (Data Memory) SRAM có các tín hiệu sau: Address[31:0], WriteData[31:0], ReadData[31:0], WriteEn, ReadEn và viết testbench kiểm tra chức năng trên phần mềm mô phỏng ModelSim.
3. Viết testbench tạo giá trị cho các tín hiệu input và chạy mô phỏng kiểm tra chức năng của thiết kế.

---

## Bài 4. THIẾT KẾ ALU

### 4.1 Mục tiêu

- ✚ Sử dụng ngôn ngữ Verilog HDL, thiết kế được một ALU 32-bit đơn giản.

### 4.2 Nội dung thực hành

Sử dụng ngôn ngữ Verilog HDL, thiết kế bộ ALU 32-bit có sơ đồ khối và các chức năng như Hình 3-1 bên dưới. Trong đó, lệnh Add và Subtract được thực hiện trên 2 số có dấu – bù 2 32-bit. Cờ báo add\_sub\_overflow sẽ được bật lên 1 khi mạch phát hiện có overflow xảy ra.

<i>M</i>	<i>S<sub>1</sub></i>	<i>S<sub>0</sub></i>	<i>ALU Operations</i>
0	0	0	Complement A
0	0	1	AND
0	1	0	EX-OR
0	1	1	OR
1	0	0	Decrement A
1	0	1	Add
1	1	0	Subtract
1	1	1	Increment A

*Hình 3-1 Bảng trạng thái ALU*

### 4.3 Sinh viên chuẩn bị ở nhà

- ✚ Sử dụng ngôn ngữ Verilog HDL, thiết kế bộ ALU 32-bit có các chức năng như Hình 3-1.

- 
- ✚ Viết testbench tạo giá trị cho các tín hiệu input và chạy mô phỏng kiểm tra chức năng của thiết kế.

## **4.4 Hướng dẫn thực hành**

### **Bài thực hành 1:**

1. Tạo một project mới trên phần mềm Quartus II, đặt tên: E/CE221-Lab/Lab3\_1-MSSV.
2. Sử dụng ngôn ngữ Verilog HDL, thiết kế bộ ALU 32-bit có các chức năng như Hình 3-1.
3. Viết testbench tạo giá trị cho các tín hiệu input và chạy mô phỏng kiểm tra chức năng của thiết kế.

### **Bài thực hành 2:**

1. Tạo một project mới trên phần mềm Quartus II, đặt tên: E/CE221-Lab/Lab3\_2-MSSV.
2. Sử dụng ngôn ngữ Verilog HDL, thiết kế bộ MUX 2-1 với ngõ vào/ngõ ra là 32 bits.
3. Viết testbench tạo giá trị cho các tín hiệu input và chạy mô phỏng kiểm tra chức năng của thiết kế.

---

## Bài 5. THIẾT KẾ DATAPATH ĐƠN GIẢN

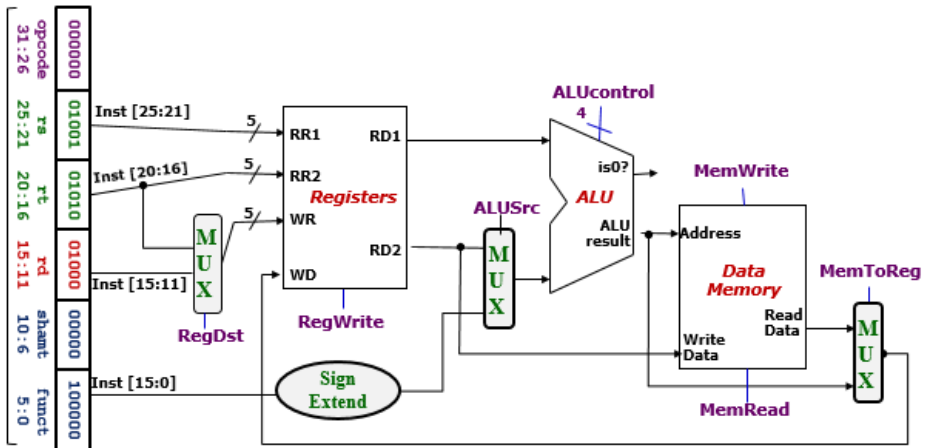
### 5.1 Mục tiêu

- ✚ Sử dụng ngôn ngữ Verilog HDL, thiết kế được một DATAPATH đơn giản theo kiến trúc MIPS.

### 5.2 Nội dung thực hành

Sinh viên dựa vào lý thuyết về DATA PATH của kiến trúc MIPS đã học trong môn Kiến trúc máy tính như trên Hình 4-1 và sử dụng lại các module đã thiết kế trong các lab trước để thiết kế một DATAPATH thực hiện lệnh các lệnh sau dùng ngôn ngữ Verilog HDL:

- ❖ add \$1, \$2, \$3
- ❖ lw \$1, 0(\$2)
- ❖ sw \$1, 0(\$2)



Hình 4-1 Data path theo kiến trúc MIPS



---

### 5.3 Sinh viên chuẩn bị ở nhà

1. Kiểm tra lại các module MUX, Register File, ALU, Data Memory đã thiết kế trong các bài Lab trước.
2. Tìm hiểu lại kiến trúc và cách thức hoạt động của DATAPATH trong kiến trúc MIPS.
3. Phân tích hoạt động DATAPATH trong Hình 4-1.

### 5.4 Hướng dẫn thực hành

1. Tạo một project mới trên phần mềm ModelSim, đặt tên: E/CE221-Lab/Lab4-MSSV.
2. Đưa thiết kế DATAPATH đã chuẩn bị ở nhà vào project.
3. Viết testbench kiểm tra thiết kế trên phần mềm mô phỏng ModelSim ứng với lệnh add \$1, \$2, \$3.
4. Viết testbench kiểm tra thiết kế trên phần mềm mô phỏng ModelSim ứng với lệnh lw \$1, 0(\$2).
5. Viết testbench kiểm tra thiết kế trên phần mềm mô phỏng ModelSim ứng với lệnh sw \$1, 0(\$2).

---

## Bài 6. THIẾT KẾ SIMPLE CONTROL UNIT

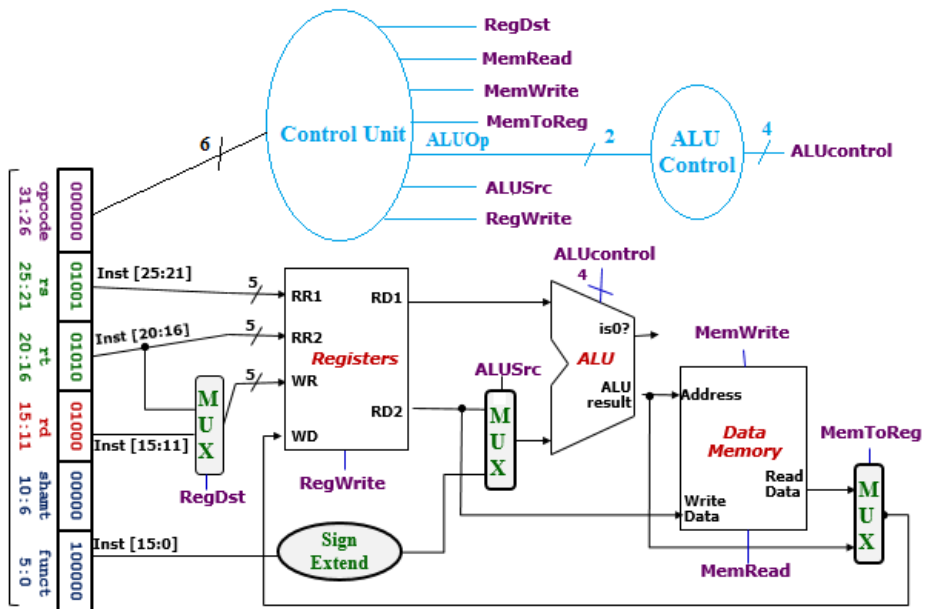
### 6.1 Mục tiêu

- ✚ Sử dụng ngôn ngữ Verilog HDL, thiết kế được một Control Unit đơn giản.

### 6.2 Nội dung thực hành

Dựa vào DATAPATH đã được thiết kế trong lab 4 và các lý thuyết liên quan, sinh viên sẽ tiến hành thiết kế 2 khối Control Unit và ALU Control nhằm điều khiển DATAPATH này như trên Hình 5-1 để thực hiện các lệnh sau sử dụng ngôn ngữ Verilog HDL:

- ❖ add \$1, \$2, \$3
- ❖ lw \$1, 0(\$2)
- ❖ sw \$1, 0(\$2)



**Hình 5-1 Data path và Control Unit theo kiến trúc MIPS**

## 6.3 Sinh viên chuẩn bị ở nhà

1. Lập bảng sự thật:

Op cod e	Lệ nh	ALU Op [1:0]	ALUc ontrol [3:0]	Re gD st	Me mR ead	me m Wr ite	Me mT oR eg	A L U S r c	Re g W rit e
000 001	add	10	0101						

---

000 010	sw	00	0101						
000 100	lw	00	0101						

## 6.4 Hướng dẫn thực hành

1. Tạo một project mới trên phần mềm ModelSim, đặt tên: E/CE221-Lab/Lab5-MSSV.
2. Đưa thiết kế khối Control Unit và khối ALU Control đã chuẩn bị ở nhà vào project.
3. Viết testbench kiểm tra thiết kế trên phần mềm mô phỏng ModelSim ứng với bảng chức năng sau:

Opcode	Lệnh	ALUOp [1:0]	ALUcontrol [3:0]
000001	add	10	0101
000010	sw	00	0101
000100	lw	00	0101

---

## Bài 7. THIẾT KẾ SIMPLE PROCESSOR

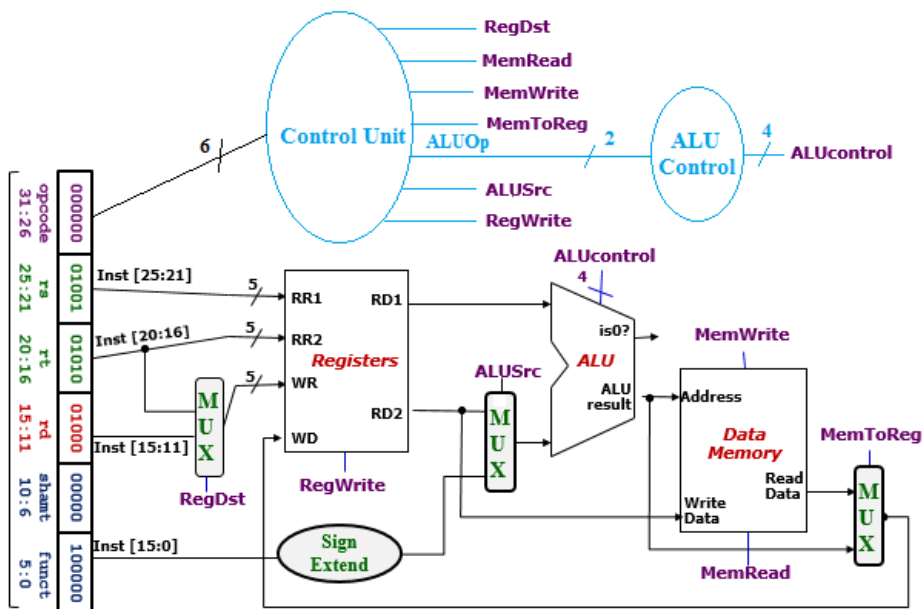
### 7.1 Mục tiêu

✚ Sử dụng ngôn ngữ Verilog HDL, thiết kế được một Processor đơn giản

### 7.2 Nội dung thực hành

Sử dụng khối thiết kế DATAPATH và khối thiết kế Control Unit, ALU Control để thiết kế một Processor đơn giản thực hiện các lệnh sau:

- ❖ add \$1, \$2, \$3
- ❖ lw \$1, 0(\$2)
- ❖ sw \$1, 0(\$2)



**Hình 6-1 Data path và Control Unit theo kiến trúc MIPS**

### 7.3 Sinh viên chuẩn bị ở nhà

Mạch kết nối hoàn chỉnh của simple Processor từ simple DATAPATH (lab 4) và simple Control Unit (lab 5).

### 7.4 Hướng dẫn thực hành

1. Tạo một project mới trên phần mềm ModelSim, đặt tên: E/CE221-Lab/Lab6-MSSV.
2. Đưa thiết kế khối Control Unit và khối ALU Control đã chuẩn bị ở nhà vào project.

- 
3. Viết testbench kiểm tra thiết kế trên phần mềm mô phỏng ModelSim ứng với bảng chức năng sau.

