**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

**BỘ MÔN ĐIỆN TỬ**

---------------o0o---------------

****

**BÀI TẬP LỚN MÔN THIẾT KẾ VI MẠCH NÂNG CAO**

**LAB 5**

**VERILOG LANGUAGE**

**Họ và Tên: Tiến Hoàng Trí Nghĩa**

**MSHV: 1870048**

**TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2018**

**MỤC LỤC**

[1. Exercise 1 4](#_Toc533202682)

[1.1 Requirement 4](#_Toc533202683)

[1.2 Source code 5](#_Toc533202684)

[1.3 Simulation 9](#_Toc533202685)

[2. Exercise 2 10](#_Toc533202686)

[2.1 Requirement - Develop the following hardware with Verilog HDL 10](#_Toc533202687)

[2.2 Source code 10](#_Toc533202688)

[2.3 Simulation 16](#_Toc533202689)

[3. Exercise 3 16](#_Toc533202690)

[3.1 Requirement - Develop the following hardware with Verilog HDL 16](#_Toc533202691)

[3.2 Source code 16](#_Toc533202692)

[3.3 Simulation 19](#_Toc533202693)

[4. Exercise 4 19](#_Toc533202694)

[4.1 Requirement - Develop the following hardware (4 bits addition) with Verilog HDL 19](#_Toc533202695)

[4.2 Source code 19](#_Toc533202696)

[4.3 Simulation 22](#_Toc533202697)

[5. Exercise 5 22](#_Toc533202698)

[5.1 Requirement - Develop the following hardware (4 bits multiplier) with Verilog HDL 22](#_Toc533202699)

[5.2 Source code 22](#_Toc533202700)

[5.3 Simulation 25](#_Toc533202701)

[6. Exercise 6 25](#_Toc533202702)

[6.1 Requirement - Develop the following hardware (Simple state machine) with Verilog HDL 25](#_Toc533202703)

[6.2 Source code 25](#_Toc533202704)

[6.3 Simulation 28](#_Toc533202705)

[7. Exercise 7 29](#_Toc533202706)

[7.1 Requirement - Develop the following hardware (4 bits addition) with Verilog HDL 29](#_Toc533202707)

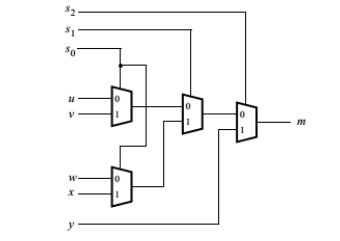
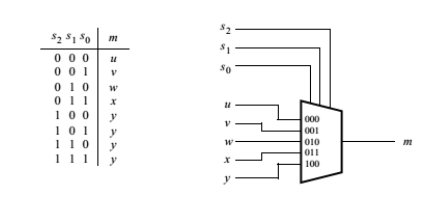
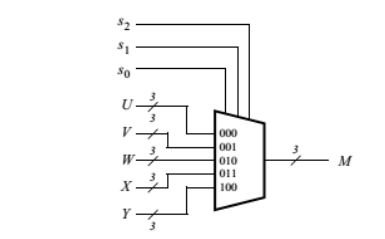
[7.2 Source code 29](#_Toc533202708)

[7.3 Simulation 32](#_Toc533202709)

# Exercise 1

## Requirement

* Develop the following hardware (for every figure) with Verilog HDL.

1. 
2. 
3. 

## Source code

**File ex01.v**

module ex01\_a(s, u, v, w, x, y, m);

parameter MUX\_WIDTH = 3;

input [MUX\_WIDTH-1:0] s;

input u;

input v;

input w;

input x;

input y;

output m;

wire uv;

wire wx;

wire uvwx\_s1;

assign uv = s[0] ? v : u ;

assign wx = s[0] ? x : w ;

assign uvwx\_s1 = s[1] ? wx : uv ;

assign m = s[2] ? y : uvwx\_s1 ;

endmodule

module ex01\_b(s, u, v, w, x, y, m);

parameter MUX\_WIDTH = 3;

parameter DATA\_WIDTH = 1;

input [MUX\_WIDTH-1:0] s;

input [DATA\_WIDTH-1:0] u;

input [DATA\_WIDTH-1:0] v;

input [DATA\_WIDTH-1:0] w;

input [DATA\_WIDTH-1:0] x;

input [DATA\_WIDTH-1:0] y;

output [DATA\_WIDTH-1:0] m;

reg [DATA\_WIDTH-1:0] m;

always @(\*)

begin

case(s)

3'b000: begin

m = u;

end

3'b001: begin

m = v;

end

3'b010: begin

m = w;

end

3'b011: begin

m = x;

end

default: begin

m = y;

end

endcase

end

endmodule

module ex01\_c(s, u, v, w, x, y, m);

parameter MUX\_WIDTH = 3;

parameter DATA\_WIDTH = 3;

input [MUX\_WIDTH-1:0] s;

input [DATA\_WIDTH-1:0] u;

input [DATA\_WIDTH-1:0] v;

input [DATA\_WIDTH-1:0] w;

input [DATA\_WIDTH-1:0] x;

input [DATA\_WIDTH-1:0] y;

output [DATA\_WIDTH-1:0] m;

reg [DATA\_WIDTH-1:0] m;

always @(\*)

begin

case(s)

3'b000: begin

m = u;

end

3'b001: begin

m = v;

end

3'b010: begin

m = w;

end

3'b011: begin

m = x;

end

default: begin

m = y;

end

endcase

end

endmodule

## Simulation a) Techbench File t\_ex01.v

`timescale 1ns/1ns

module t\_ex01;

wire [2:0] s;

wire u;

wire v;

wire w;

wire x;

wire y;

wire m\_a;

wire m\_b;

wire [2:0] m\_c;

wire [2:0] u\_c;

wire [2:0] v\_c;

wire [2:0] w\_c;

wire [2:0] x\_c;

wire [2:0] y\_c;

reg [2:0] t\_s;

reg [6:0] t\_data;

assign s = t\_s;

assign u = t\_data[0];

assign v = t\_data[1];

assign w = t\_data[2];

assign x = t\_data[3];

assign y = t\_data[4];

assign u\_c = t\_data[2:0];

assign v\_c = t\_data[3:1];

assign w\_c = t\_data[4:2];

assign x\_c = t\_data[5:3];

assign y\_c = t\_data[6:4];

ex01\_a ex01\_a\_01 (.s(s), .u(u), .v(v), .w(w), .x(x), .y(y), .m(m\_a));

ex01\_b #(.DATA\_WIDTH(1))

ex01\_b\_01 (.s(s), .u(u), .v(v), .w(w), .x(x), .y(y), .m(m\_b));

ex01\_c #(.DATA\_WIDTH(3))

ex01\_c\_01 (.s(s), .u(u\_c), .v(v\_c), .w(w\_c), .x(x\_c), .y(y\_c), .m(m\_c));

initial begin

# 10 t\_s = 3'b000;

t\_data = 7'b0000000;

# 10 t\_data = 7'b0000001;

# 10 t\_data = 7'b1110001;

# 10

# 10 t\_s = 3'b001;

t\_data = 7'b0000000;

# 10 t\_data = 7'b0000010;

# 10 t\_data = 7'b1100011;

# 10 t\_s = 3'b010;

t\_data = 7'b0000000;

# 10 t\_data = 7'b0000011;

# 10 t\_data = 7'b1000111;

# 10 t\_s = 3'b011;

t\_data = 7'b0000000;

# 10 t\_data = 7'b0000100;

# 10 t\_data = 7'b0001111;

# 10 t\_s = 3'b100;

t\_data = 7'b0000000;

# 10 t\_data = 7'b0000101;

# 10 t\_data = 7'b0011110;

# 10 t\_s = 3'b101;

t\_data = 7'b0000000;

# 10 t\_data = 7'b0000110;

# 10 t\_data = 7'b0111100;

# 10 t\_s = 3'b110;

t\_data = 7'b0000000;

# 10 t\_data = 7'b0000111;

# 10 t\_data = 7'b1111000;

# 10 t\_s = 3'b111;

t\_data = 7'b0000000;

# 10 t\_data = 7'b0000001;

# 10 t\_data = 7'b1110001;

# 20 $finish;

end

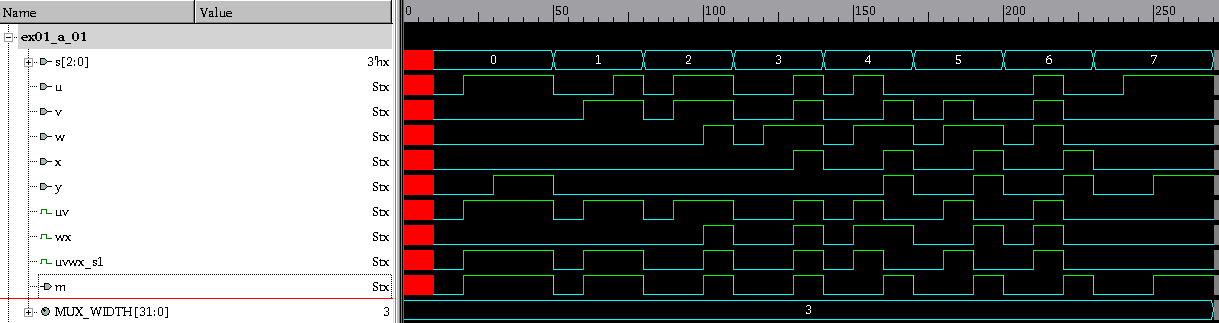
initial begin

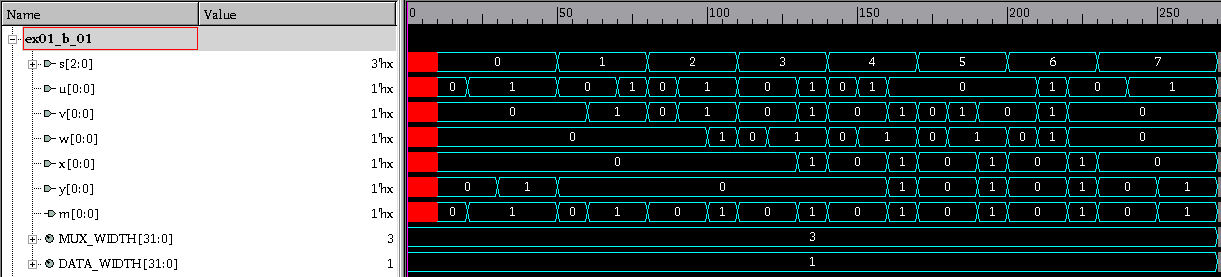
$vcdplusfile ("Ex01\_Waveform.vpd");

$vcdpluson();

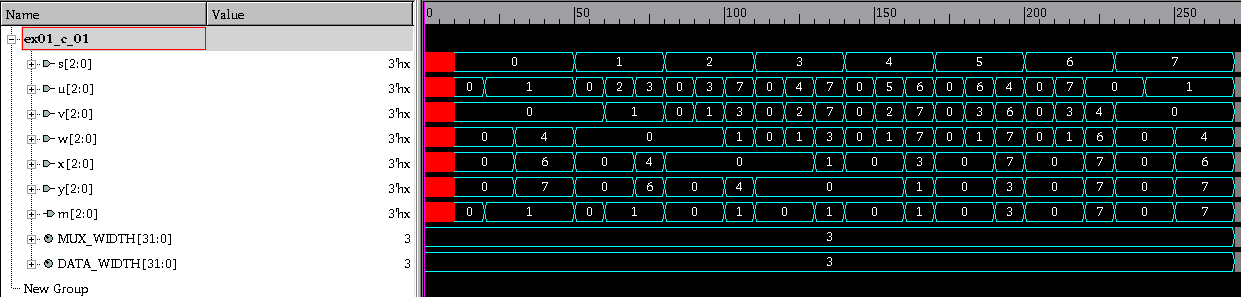
end

endmodule  
  
**b) Waveform**

**ex01\_a waveform**

**ex01\_b waveform**

**ex01\_c waveform**

****

## Đánh giá kết quả

# Exercise 2

## Requirement - Develop the following hardware with Verilog HDL

## Source code

**File ex02.v**

module ex02 (s, u, v, w, x, y, clk, led7\_out);

parameter MUX\_WIDTH = 3;

parameter DATA\_WIDTH = 3;

input [MUX\_WIDTH-1:0] s;

input [DATA\_WIDTH-1:0] u;

input [DATA\_WIDTH-1:0] v;

input [DATA\_WIDTH-1:0] w;

input [DATA\_WIDTH-1:0] x;

input [DATA\_WIDTH-1:0] y;

input clk;

output [6:0] led7\_out;

wire [DATA\_WIDTH-1:0] m;

wire [DATA\_WIDTH-1:0] led7\_in;

mux\_5 #(.DATA\_WIDTH(3))

mux\_5\_01 (.s(s), .u(u), .v(v), .w(w), .x(x), .y(y), .m(m));

signal signal (.clk(clk), .d(m), .qa(led7\_in));

led7\_segment\_decoder led7\_segment\_decoder (.in(led7\_in), .out(led7\_out));

endmodule

module signal (clk, d, qa);

input clk;

input [2:0] d;

output [2:0] qa;

reg [2:0] qa;

initial

begin

qa = 3'b000;

end

always @(posedge clk)

begin

qa <= d;

end

endmodule

module mux\_5 (s, u, v, w, x, y, m);

parameter MUX\_WIDTH = 3;

parameter DATA\_WIDTH = 3;

input [MUX\_WIDTH-1 :0] s;

input [DATA\_WIDTH-1:0] u;

input [DATA\_WIDTH-1:0] v;

input [DATA\_WIDTH-1:0] w;

input [DATA\_WIDTH-1:0] x;

input [DATA\_WIDTH-1:0] y;

output [DATA\_WIDTH-1:0] m;

reg [DATA\_WIDTH-1:0] m;

always @(\*)

begin

case(s)

3'b000: begin

m <= u;

end

3'b001: begin

m <= v;

end

3'b010: begin

m <= w;

end

3'b011: begin

m <= x;

end

default: begin

m <= y;

end

endcase

end

endmodule

module led7\_segment\_decoder (in , out);

input [2:0]in;

output [6:0]out;

reg [6:0]out;

always @(\*)

begin

case(in)

3'b000: begin

out = 7'b0111111;

end

3'b001: begin

out = 7'b0000110;

end

3'b010: begin

out = 7'b1011011;

end

3'b011: begin

out = 7'b1001111;

end

3'b100: begin

out = 7'b1100110;

end

3'b101: begin

out = 7'b1101101;

end

3'b110: begin

out = 7'b1111101;

end

3'b111: begin

out = 7'b0000111;

end

default: begin

out = 7'b0000000;

end

endcase

end

endmodule

## 2.3 Simulation a) Techbench File t\_ex02.v

`timescale 1ns/1ns

module t\_ex02;

wire [2:0] s;

wire [2:0] u;

wire [2:0] v;

wire [2:0] w;

wire [2:0] x;

wire [2:0] y;

wire [6:0] led;

reg [2:0] t\_s;

reg [2:0] t\_data;

reg t\_clk;

assign s = t\_s;

assign u = t\_data;

assign v = t\_data;

assign w = t\_data;

assign x = t\_data;

assign y = t\_data;

always begin

#0 t\_clk = 0;

#10 t\_clk = 1;

#10;

end

ex02 ex02 (.s(s), .u(u), .v(v), .w(w), .x(x), .y(y), .clk(t\_clk), .led7\_out(led));

initial begin

# 20 t\_s = 3'b000;

t\_data = 3'b000;

# 20 t\_data = 3'b001;

# 20 t\_data = 3'b010;

# 20 t\_s = 3'b001;

t\_data = 3'b011;

# 20 t\_data = 3'b100;

# 20 t\_data = 3'b101;

# 20 t\_s = 3'b010;

t\_data = 3'b110;

# 20 t\_data = 3'b111;

# 20 t\_data = 3'b000;

# 20 t\_s = 3'b011;

t\_data = 3'b001;

# 20 t\_data = 3'b010;

# 20 t\_data = 3'b011;

# 20 t\_s = 3'b100;

t\_data = 3'b100;

# 20 t\_data = 3'b101;

# 20 t\_data = 3'b110;

# 20 t\_s = 3'b101;

t\_data = 3'b111;

# 20 t\_data = 3'b000;

# 20 t\_data = 3'b001;

# 20 t\_s = 3'b110;

t\_data = 3'b010;

# 20 t\_data = 3'b011;

# 20 t\_data = 3'b100;

# 20 t\_s = 3'b111;

t\_data = 3'b101;

# 20 t\_data = 3'b110;

# 20 t\_data = 3'b111;

# 40 $finish;

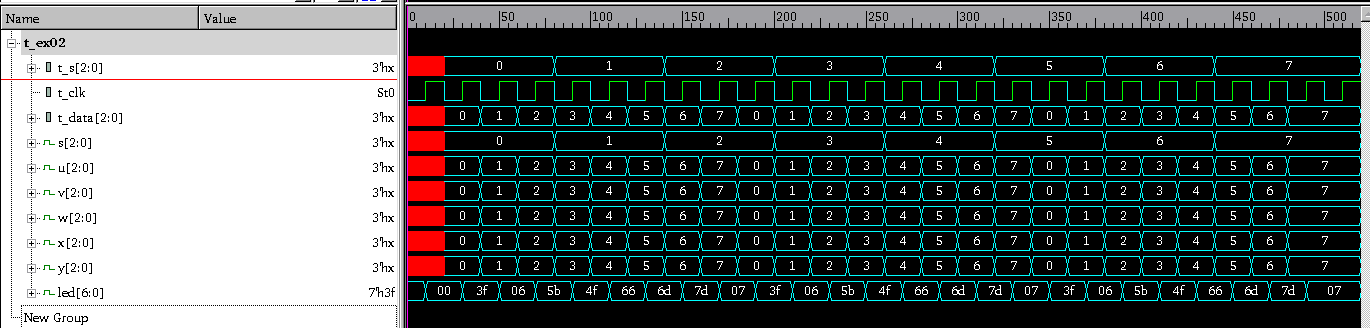
end

initial begin

$vcdplusfile ("Ex2\_WaveForm.vpd");

$vcdpluson();

end

endmodule  
  
**b) Waveform  
ex02 waveform**

## Đánh giá kết quả

# Exercise 3

## Requirement - Develop the following hardware with Verilog HDL

## Source code

**File ex03.v**

module ex03 (ena, clk, rst\_n, trigger);

input clk;

input rst\_n;

input ena;

output trigger;

wire [3:0] w\_ena;

wire [3:0] w\_q;

assign w\_ena[0] = ena;

d\_ff d\_ff\_01(.clk(clk), .rst\_n(rst\_n), .d(w\_ena[0]), .q(w\_q[0]) );

assign w\_ena[1] = w\_ena[0] && w\_q[0];

d\_ff d\_ff\_02(.clk(clk), .rst\_n(rst\_n), .d(w\_ena[1]), .q(w\_q[1]) );

assign w\_ena[2] = w\_ena[1] && w\_q[1];

d\_ff d\_ff\_03(.clk(clk), .rst\_n(rst\_n), .d(w\_ena[2]), .q(w\_q[2]) );

assign w\_ena[3] = w\_ena[2] && w\_q[2];

d\_ff d\_ff\_04(.clk(clk), .rst\_n(rst\_n), .d(w\_ena[3]), .q(w\_q[3]) );

assign trigger = w\_q[3];

endmodule

module d\_ff (clk, rst\_n, d, q);

input clk;

input rst\_n;

input d;

output q;

reg q;

always @(posedge clk or negedge rst\_n)

begin

if (!rst\_n) begin

q <= 0;

end

else begin

q <= d;

end

end

endmodule

## Simulation a) Techbench File t\_ex03.v

`timescale 1ns/1ns

module t\_ex03;

reg t\_clk;

reg t\_rst\_n;

reg t\_ena;

wire t\_trigger;

always begin

#0 t\_clk = 0;

#10 t\_clk = 1;

#10;

end

ex03 ex03\_01(.clk(t\_clk), .rst\_n(t\_rst\_n), .ena(t\_ena), .trigger(t\_trigger));

initial begin

#0 t\_rst\_n = 0;

t\_ena = 0;

#20 t\_rst\_n = 1;

t\_ena = 0;

#20 t\_ena = 1;

#200 t\_rst\_n = 0;

#20 t\_rst\_n = 1;

#20 t\_ena = 0;

#20 t\_ena = 1;

#180 t\_ena = 0;

#20 t\_ena = 1;

#160 t\_ena = 0;

#20 t\_ena = 1;

#140 t\_ena = 0;

#20 t\_ena = 1;

#120 t\_ena = 0;

#20 t\_ena = 1;

#100 t\_ena = 0;

#20 t\_ena = 1;

#80 t\_ena = 0;

#20 t\_ena = 1;

#60 t\_ena = 0;

#20 t\_ena = 1;

#40 t\_ena = 0;

#20 t\_ena = 1;

#20 t\_ena = 0;

#20 t\_rst\_n = 0;

t\_ena = 0;

#20 t\_rst\_n = 1;

t\_ena = 0;

#20 $finish;

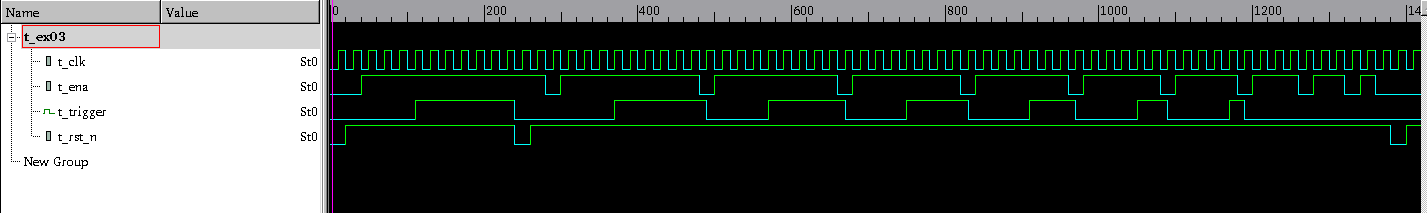
end

initial begin

$vcdplusfile ("Ex3\_WaveForm.vpd");

$vcdpluson();

end

endmodule  
  
**b) Waveform  
ex03 waveform**

## Đánh giá kết quả

# Exercise 4

## Requirement - Develop the following hardware (4 bits addition) with Verilog HDL

## Source code

**File ex04.v**

module ex04 (a, b, cin, s, cout);

input [3:0] a;

input [3:0] b;

input cin;

output [3:0] s;

output cout;

wire [4:0] c;

assign c[0] = cin;

f\_adder f\_adder\_01(.a(a[0]), .b(b[0]), .c\_in(c[0]), .s(s[0]), .c\_out(c[1]));

f\_adder f\_adder\_02(.a(a[1]), .b(b[1]), .c\_in(c[1]), .s(s[1]), .c\_out(c[2]));

f\_adder f\_adder\_03(.a(a[2]), .b(b[2]), .c\_in(c[2]), .s(s[2]), .c\_out(c[3]));

f\_adder f\_adder\_04(.a(a[3]), .b(b[3]), .c\_in(c[3]), .s(s[3]), .c\_out(c[4]));

assign cout = c[4];

endmodule

module f\_adder(a, b, c\_in, s, c\_out);

input a;

input b;

input c\_in;

output s;

output c\_out;

assign s = a ^ b ^ c\_in;

assign c\_out = (a & b)|(b & c\_in)|(a & c\_in);

endmodule

## Simulation a) Techbench File t\_ex04.v

`timescale 1ns/1ns

module t\_ex04;

reg [3:0] a;

reg [3:0] b;

reg cin;

wire [3:0] s;

wire cout;

ex04 ex04\_01(.a(a), .b(b), .cin(cin), .s(s), .cout(cout));

initial begin

#0 cin = 0;

a = 0;

b = 0;

#20 a = 1;

b = 0;

#20 a = 0;

b = 1;

#20 a = 1;

b = 1;

#20 a = 1;

b = 2;

#20 a = 3;

b = 4;

#20 a = 5;

b = 6;

#20 a = 7;

b = 8;

#20 a = 9;

b = 10;

#20 a = 11;

b = 12;

#20 a = 13;

b = 14;

#20 a = 15;

b = 15;

#20 cin = 1;

a = 0;

b = 0;

#20 a = 1;

b = 0;

#20 a = 0;

b = 1;

#20 a = 1;

b = 1;

#20 a = 1;

b = 2;

#20 a = 3;

b = 4;

#20 a = 5;

b = 6;

#20 a = 7;

b = 8;

#20 a = 9;

b = 10;

#20 a = 11;

b = 12;

#20 a = 13;

b = 14;

#20 a = 15;

b = 15;

#20 $finish;

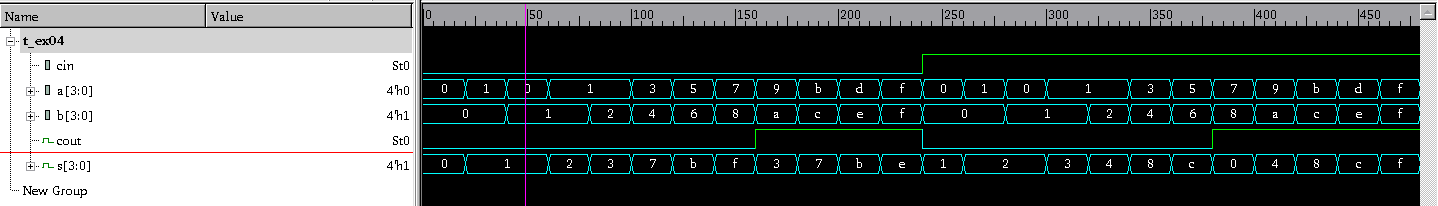
end

initial begin

$vcdplusfile ("Ex4\_Waveform.vpd");

$vcdpluson();

end

****endmodule  
  
**b) Waveform  
ex04 waveform**

## Đánh giá kết quả

# Exercise 5

## Requirement - Develop the following hardware (4 bits multiplier) with Verilog HDL

## Source code

**File ex05.v**

module ex05 (m, q, p);

input [3:0] m;

input [3:0] q;

output [7:0] p;

wire [4:0]c\_in[0:4];

wire [4:0]m\_in[0:4];

assign m\_in[0][4:0] = 5'b00000;

assign c\_in[0][0] = 1'b0;

m\_adder m\_adder\_01(.a(m[0]), .b(q[0]), .m\_in(m\_in[0][1]), .c\_in(c\_in[0][0]), .m\_out(m\_in[1][0]), .c\_out(c\_in[0][1]));

m\_adder m\_adder\_02(.a(m[0]), .b(q[1]), .m\_in(m\_in[0][2]), .c\_in(c\_in[0][1]), .m\_out(m\_in[1][1]), .c\_out(c\_in[0][2]));

m\_adder m\_adder\_03(.a(m[0]), .b(q[2]), .m\_in(m\_in[0][3]), .c\_in(c\_in[0][2]), .m\_out(m\_in[1][2]), .c\_out(c\_in[0][3]));

m\_adder m\_adder\_04(.a(m[0]), .b(q[3]), .m\_in(m\_in[0][4]), .c\_in(c\_in[0][3]), .m\_out(m\_in[1][3]), .c\_out(c\_in[0][4]));

assign m\_in[1][4] = c\_in[0][4];

assign p[0] = m\_in[1][0];

assign c\_in[1][0] = 1'b0;

m\_adder m\_adder\_05(.a(m[1]), .b(q[0]), .m\_in(m\_in[1][1]), .c\_in(c\_in[1][0]), .m\_out(m\_in[2][0]), .c\_out(c\_in[1][1]));

m\_adder m\_adder\_06(.a(m[1]), .b(q[1]), .m\_in(m\_in[1][2]), .c\_in(c\_in[1][1]), .m\_out(m\_in[2][1]), .c\_out(c\_in[1][2]));

m\_adder m\_adder\_07(.a(m[1]), .b(q[2]), .m\_in(m\_in[1][3]), .c\_in(c\_in[1][2]), .m\_out(m\_in[2][2]), .c\_out(c\_in[1][3]));

m\_adder m\_adder\_08(.a(m[1]), .b(q[3]), .m\_in(m\_in[1][4]), .c\_in(c\_in[1][3]), .m\_out(m\_in[2][3]), .c\_out(c\_in[1][4]));

assign m\_in[2][4] = c\_in[1][4];

assign p[1] = m\_in[2][0];

assign c\_in[2][0] = 1'b0;

m\_adder m\_adder\_09(.a(m[2]), .b(q[0]), .m\_in(m\_in[2][1]), .c\_in(c\_in[2][0]), .m\_out(m\_in[3][0]), .c\_out(c\_in[2][1]));

m\_adder m\_adder\_10(.a(m[2]), .b(q[1]), .m\_in(m\_in[2][2]), .c\_in(c\_in[2][1]), .m\_out(m\_in[3][1]), .c\_out(c\_in[2][2]));

m\_adder m\_adder\_11(.a(m[2]), .b(q[2]), .m\_in(m\_in[2][3]), .c\_in(c\_in[2][2]), .m\_out(m\_in[3][2]), .c\_out(c\_in[2][3]));

m\_adder m\_adder\_12(.a(m[2]), .b(q[3]), .m\_in(m\_in[2][4]), .c\_in(c\_in[2][3]), .m\_out(m\_in[3][3]), .c\_out(c\_in[2][4]));

assign m\_in[3][4] = c\_in[2][4];

assign p[2] = m\_in[3][0];

assign c\_in[3][0] = 1'b0;

m\_adder m\_adder\_13(.a(m[3]), .b(q[0]), .m\_in(m\_in[3][1]), .c\_in(c\_in[3][0]), .m\_out(m\_in[4][0]), .c\_out(c\_in[3][1]));

m\_adder m\_adder\_14(.a(m[3]), .b(q[1]), .m\_in(m\_in[3][2]), .c\_in(c\_in[3][1]), .m\_out(m\_in[4][1]), .c\_out(c\_in[3][2]));

m\_adder m\_adder\_15(.a(m[3]), .b(q[2]), .m\_in(m\_in[3][3]), .c\_in(c\_in[3][2]), .m\_out(m\_in[4][2]), .c\_out(c\_in[3][3]));

m\_adder m\_adder\_16(.a(m[3]), .b(q[3]), .m\_in(m\_in[3][4]), .c\_in(c\_in[3][3]), .m\_out(m\_in[4][3]), .c\_out(c\_in[3][4]));

assign m\_in[4][4] = c\_in[3][4];

assign p[3] = m\_in[4][0];

assign p[4] = m\_in[4][1];

assign p[5] = m\_in[4][2];

assign p[6] = m\_in[4][3];

assign p[7] = m\_in[4][4];

endmodule

module m\_adder(a, b, m\_in, c\_in, m\_out, c\_out);

input a;

input b;

input m\_in;

input c\_in;

output m\_out;

output c\_out;

wire c;

assign c = a & b;

assign m\_out = c ^ m\_in ^ c\_in;

assign c\_out = (c & m\_in)|(m\_in & c\_in)|(c\_in & c);

endmodule

## Simulation a) Techbench File t\_ex05.v

`timescale 1ns/1ns

module t\_ex05;

reg [3:0] a;

reg [3:0] b;

wire [7:0] p;

ex05 ex05\_01(.m(a), .q(b), .p(p));

initial begin

#0 a = 0;

b = 0;

#50 a = 1;

b = 0;

#50 a = 2;

b = 6;

#50 a = 1;

b = 8;

#50 a = 1;

b = 1;

#50 a = 7;

b = 2;

#50 a = 4;

b = 5;

#50 a = 1;

b = 8;

#50 a = 3;

b = 6;

#50 a = 8;

b = 7;

#50 a = 8;

b = 8;

#50 a = 10;

b = 7;

#50 a = 11;

b = 9;

#50 a = 14;

b = 10;

#50 a = 15;

b = 0;

#50 a = 4;

b = 11;

#50 a = 15;

b = 15;

#50 $finish;

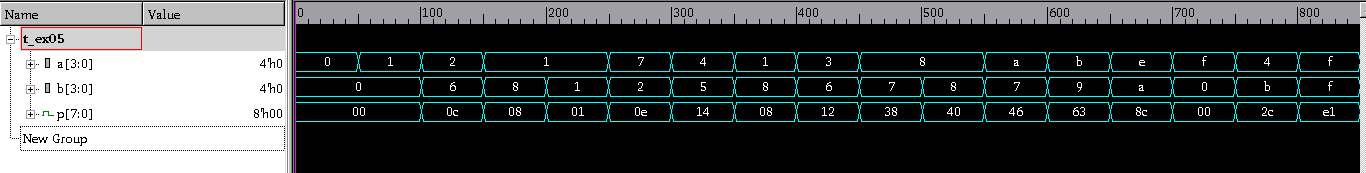
end

initial begin

$vcdplusfile ("Ex5\_Waveform.vpd");

$vcdpluson();

end

endmodule  
  
**b) Waveform  
ex05 waveform**

## Đánh giá kết quả

# Exercise 6

## Requirement - Develop the following hardware (Simple state machine) with Verilog HDL

## Source code

**File ex06.v**

module ex06 (clk, rst\_n, w, out);

input clk;

input rst\_n;

input w;

output out;

parameter ST\_A = 2'b00;

parameter ST\_B = 2'b01;

parameter ST\_C = 2'b10;

parameter ST\_F = 2'b11;

reg out;

reg [1:0] next\_state;

reg [1:0] state;

always @(posedge clk or negedge rst\_n) begin

if (!rst\_n) begin

state <= ST\_A;

end

else begin

if(state != next\_state) begin

state <= next\_state;

end

end

end

always @(\*) begin

case (state)

ST\_A: begin

next\_state = w ? ST\_F : ST\_B;

end

ST\_B: begin

next\_state = w ? ST\_F : ST\_C;

end

ST\_C: begin

next\_state = w ? ST\_F : ST\_C;

end

ST\_F: begin

next\_state = w ? ST\_F : ST\_B;

end

default: begin

next\_state = ST\_A;

end

endcase

end

reg [7:0] STATE\_CHAR;

always @\*begin

case(state)

ST\_A: begin

STATE\_CHAR = "A";

end

ST\_B: begin

STATE\_CHAR = "B";

end

ST\_C: begin

STATE\_CHAR = "C";

end

ST\_F: begin

STATE\_CHAR = "F";

end

default: begin

STATE\_CHAR = "X";

end

endcase

end

always @(\*) begin

case (state)

ST\_C, ST\_F: begin

out = 1'b1;

end

default: begin

out = 1'b0;

end

endcase

end

endmodule

## Simulation a) Techbench File t\_ex06.v

`timescale 1ns/1ns

module t\_ex06;

reg t\_clk;

reg t\_rst\_n;

reg t\_w;

wire t\_out;

always begin

#0 t\_clk = 0;

#10 t\_clk = 1;

#10;

end

ex06 ex06(.clk(t\_clk), .rst\_n(t\_rst\_n), .w(t\_w), .out(t\_out));

initial begin

#0 t\_rst\_n = 0;

t\_w = 0; // ST\_A

#20 t\_rst\_n = 1;

t\_w = 0; // ST\_B

#20 t\_w = 1; // ST\_F

#20 t\_w = 0; // ST\_B

#20 t\_w = 0; // ST\_C

#20 t\_w = 0; // ST\_C

#20 t\_w = 1; // ST\_F

#20 t\_rst\_n = 0;

t\_w = 0; // ST\_A -> ST\_A

#40 t\_rst\_n = 1;

t\_w = 0; // ST\_B -> ST\_C

#40 t\_w = 1; // ST\_F

#20 t\_w = 1; // ST\_F -> ST\_F -> ST\_F

#60 t\_w = 0; // ST\_B

#20 t\_w = 0; // ST\_C -> ST\_C -> ST\_C -> ST\_C

#80 t\_w = 1; // ST\_F

#100 $finish;

end

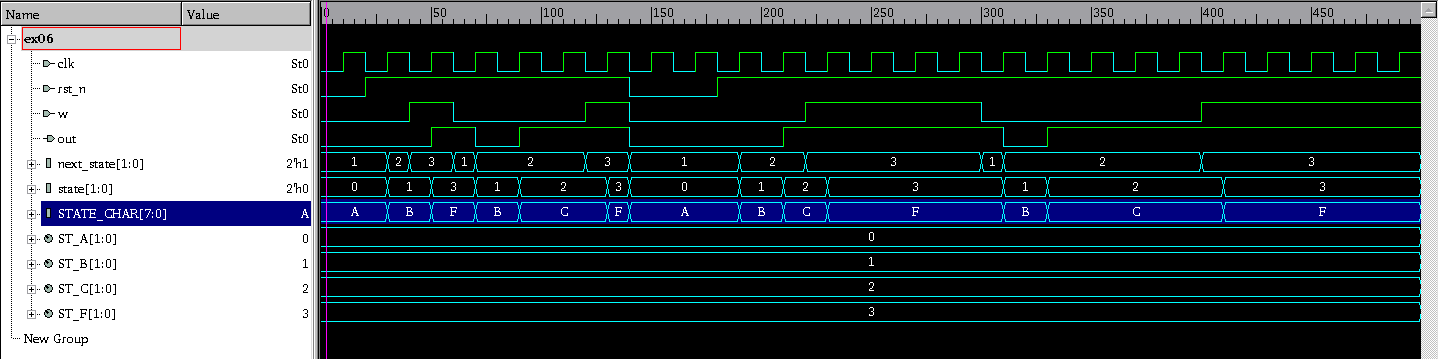
initial begin

$vcdplusfile ("Ex6\_Waveform.vpd");

$vcdpluson();

end

endmodule  
  
**b) Waveform  
ex06 waveform**

****

## Đánh giá kết quả

# Exercise 7

## Requirement - Develop the following hardware (Complicated State Machine) with Verilog HDL

## Source code

**File ex07.v**

module ex07 (clk, rst\_n, w, out);

input clk;

input rst\_n;

input w;

output out;

parameter ST\_A = 4'b1111;

parameter ST\_B = 4'b0000;

parameter ST\_C = 4'b0001;

parameter ST\_D = 4'b0010;

parameter ST\_E = 4'b0011;

parameter ST\_F = 4'b1000;

parameter ST\_G = 4'b1001;

parameter ST\_H = 4'b1010;

parameter ST\_I = 4'b1011;

reg out;

reg [4:0] next\_state;

reg [4:0] state;

always @(posedge clk or negedge rst\_n) begin

if (!rst\_n) begin

state <= ST\_A;

end

else begin

if(state!=next\_state) begin

state <= next\_state;

end

end

end

always @(\*) begin

case(state)

ST\_A: begin

next\_state = w ? ST\_F : ST\_B;

end

ST\_B,ST\_C,ST\_D: begin

next\_state = w ? ST\_F : (state + 1'b1);

end

ST\_F,ST\_G,ST\_H: begin

next\_state = w ? (state + 1'b1) : ST\_B;

end

ST\_E: begin

next\_state = w ? ST\_F : ST\_E;

end

ST\_I: begin

next\_state = w ? ST\_I : ST\_B;

end

default: begin

next\_state = ST\_A;

end

endcase

end

reg [7:0] STATE\_CHAR;

always@\*begin

case(state)

ST\_A: begin

STATE\_CHAR = "A";

end

ST\_B: begin

STATE\_CHAR = "B";

end

ST\_C: begin

STATE\_CHAR = "C";

end

ST\_D: begin

STATE\_CHAR = "D";

end

ST\_E: begin

STATE\_CHAR = "E";

end

ST\_F: begin

STATE\_CHAR = "F";

end

ST\_G: begin

STATE\_CHAR = "G";

end

ST\_H: begin

STATE\_CHAR = "SH";

end

ST\_I: begin

STATE\_CHAR = "I";

end

default: begin

STATE\_CHAR = "X";

end

endcase

end

always @(\*) begin

case(state)

ST\_I,ST\_E: begin

out = 1'b1;

end

default: begin

out = 1'b0;

end

endcase

end

endmodule

## Simulation a) Techbench File t\_ex07.v

`timescale 1ns/1ns

module t\_ex07;

reg t\_clk;

reg t\_rst\_n;

reg t\_w;

wire t\_out;

always begin

#0 t\_clk = 0;

#10 t\_clk = 1;

#10;

end

ex07 ex07(.clk(t\_clk), .rst\_n(t\_rst\_n), .w(t\_w), .out(t\_out));

initial begin

#0 t\_rst\_n = 0;

t\_w = 0; // ST\_A

#20 t\_rst\_n = 1;

t\_w = 0; // ST\_B

#20 t\_w = 0; // ST\_C

#20 t\_w = 1; // ST\_F

#20 t\_w = 0; // ST\_B -> ST\_C

#40 t\_w = 1; // ST\_F

#20 t\_w = 0; // ST\_B -> ST\_C -> ST\_D

#60 t\_w = 1; // ST\_F

#20 t\_w = 0; // ST\_B -> ST\_C -> ST\_D -> ST\_E

#80 t\_w = 0; // ST\_E

#20 t\_rst\_n = 0;

t\_w = 1; // ST\_A -> ST\_A

#40 t\_rst\_n = 1;

t\_w = 1; // ST\_F

#20 t\_w = 0; // ST\_B

#20 t\_w = 1; // ST\_F -> ST\_G

#40 t\_w = 0; // ST\_B

#20 t\_w = 1; // ST\_F -> ST\_G -> ST\_H

#60 t\_w = 0; // ST\_B

#20 t\_w = 1; // ST\_F -> ST\_G -> ST\_H -> ST\_I

#80 t\_w = 1; // ST\_I -> ST\_I -> ST\_I

#60 $finish;

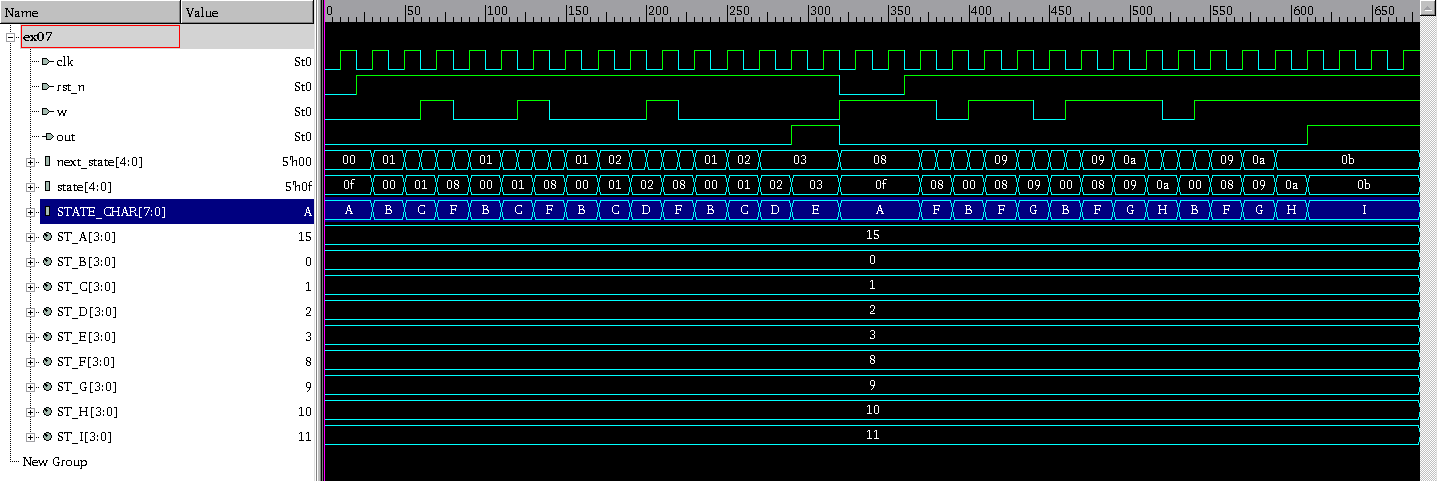
end

initial begin

$vcdplusfile ("Ex7\_Waveform.vpd");

$vcdpluson();

end

endmodule  
  
**b) Waveform  
ex07 waveform**

## Đánh giá kết quả