1module four\_var(a,b,c,d,e) ;

input a,b,c,d ;

output e ;

assign e = ( a&b | c & d ) ;

endmodule

2module beh (a, b, y);

input a, b;

output y;

reg y;

always @ (a or b)

begin

if ((a = = 0) && (b = = 0))

y = 0;

else

y = 1;

end

endmodule

module dataflow (a, b, y);

input a, b;

output y;

assign y = a | b;

endmodule

module structural(a, b, y);

input a, b;

output y;

or g1(y, a, b);

endmodule

module halfadder(a,b,s,c);

input a,b ;

output s,c ;

assign s = a ^ b;

assign c = a & b;

endmodule

module full\_adder(a,b,cin,s,cout);

input a,b,cin;

output s,cout;

assign s = a ^ b ^ cin;

module full\_adder(a,b,cin,s,cout);

input a,b,cin;

output s,cout;

assign s = a ^ b ^ cin;assign cout = (a & b) | (b & cin) | (a & cin);

endmodule

module half\_subtractor(a,b,d,bout);

input a,b;

output d,bout;

assign d = a ^ b;

assign bout = ~ a & b;

endmodule

module full\_subtractor (a, b , bin , d , bout);

input a,b,bin;

output d,bout;

assign d = a ^ b ^ bin;

assign bout = (~a & b) | (~(a ^ b) & bin);

endmodule

module mux\_2\_1( I, sel, y);

input [1:0] I ;

input sel ;

output y ;

reg y;

always@ (sel , I)

begin

case (sel)

1'b0: y = I [0] ;

1'b1: y = I [1] ;

endcase

end

endmodule

module mux\_4\_1 (I, sel, y);

input [3:0] I;

input [1:0] sel;

output y;

reg y;

always@ (sel, I)

begin

case (sel)

module mux\_4\_1 (I, sel, y);

input [3:0] I;

input [1:0] sel;

output y;

reg y;

always@ (sel, I)

begin

case (sel)

2'b00: y = I [0];

2'b01: y = I [1];

2'b10: y = I [2];

default: y = I [3];

endcase

end

endmodule

module mux\_8\_1 (I, sel, y);

input [7:0] I;

input [2:0] sel;

output y;

reg y;

always@ (sel, I )

begin

case (sel)

3'b000: y = I [0];

3'b001: y = I [1];

3'b010: y = I [2];

3'b011: y = I [3];

3'b100: y = I [4];

3'b101: y = I [5];

3'b110: y = I [6];

default: y = I [7];

endcase

end

endmodule

module sr(input clk, rst\_n, input s,r,output reg q, output q\_bar);

// always@(posedge clk or negedge rst\_n) // for asynchronous reset

always @(posedge clk) begin // for synchronous reset

if (!rst\_n) q <= 0;

module sr(input clk, rst\_n, input s,r,output reg q, output q\_bar);

// always@(posedge clk or negedge rst\_n) // for asynchronous reset

always @(posedge clk) begin // for synchronous reset

if (!rst\_n) q <= 0; else begin

case ({s,r})

2'b00: q <= q; // No change

2'b01: q <= 1'b0; // reset

2'b10: q <= 1'b1; // set

2'b11: q <= 1'bx; // Invalid inputs

endcase

end

end

assign q\_bar = ~ q;

endmodule

