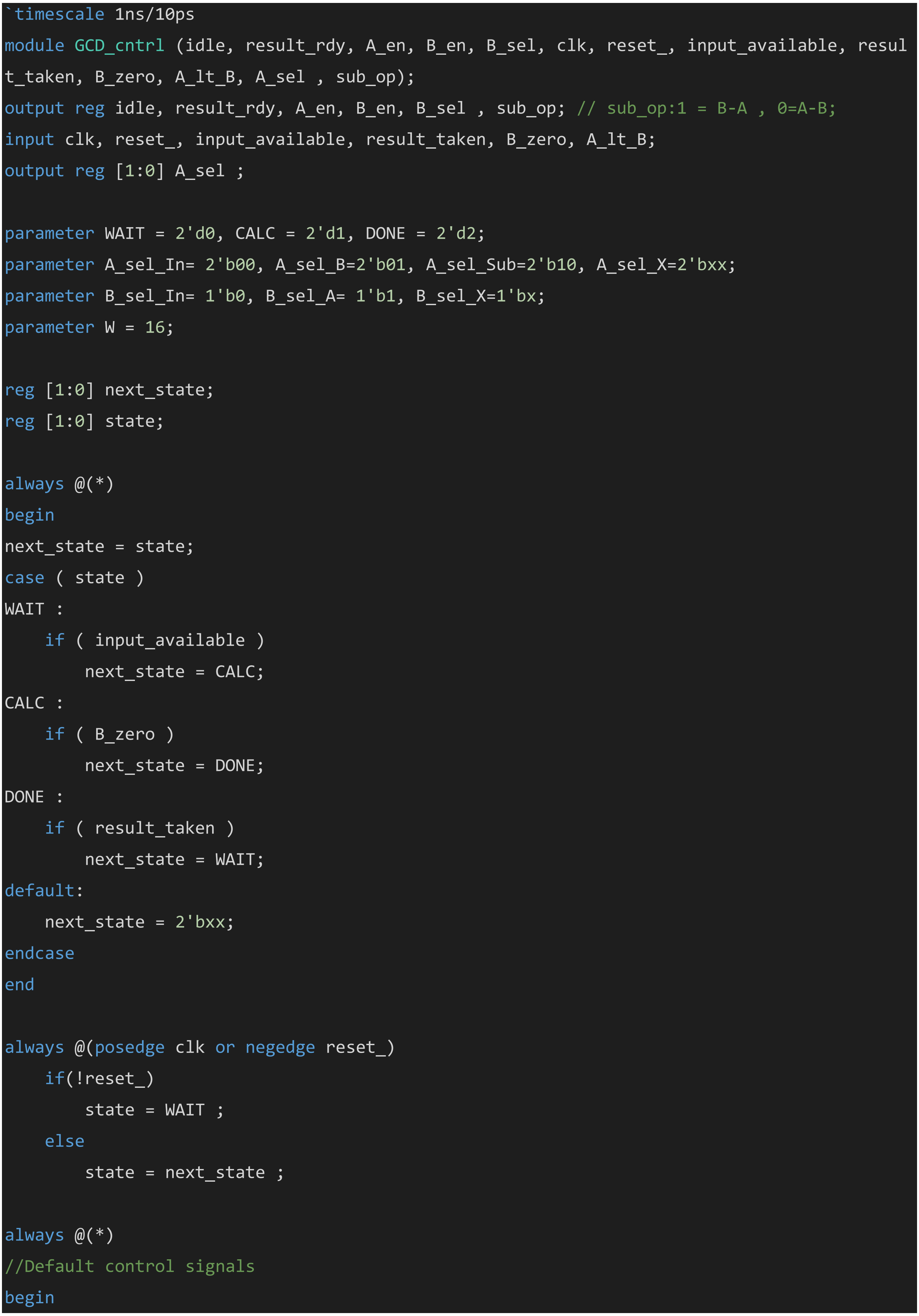
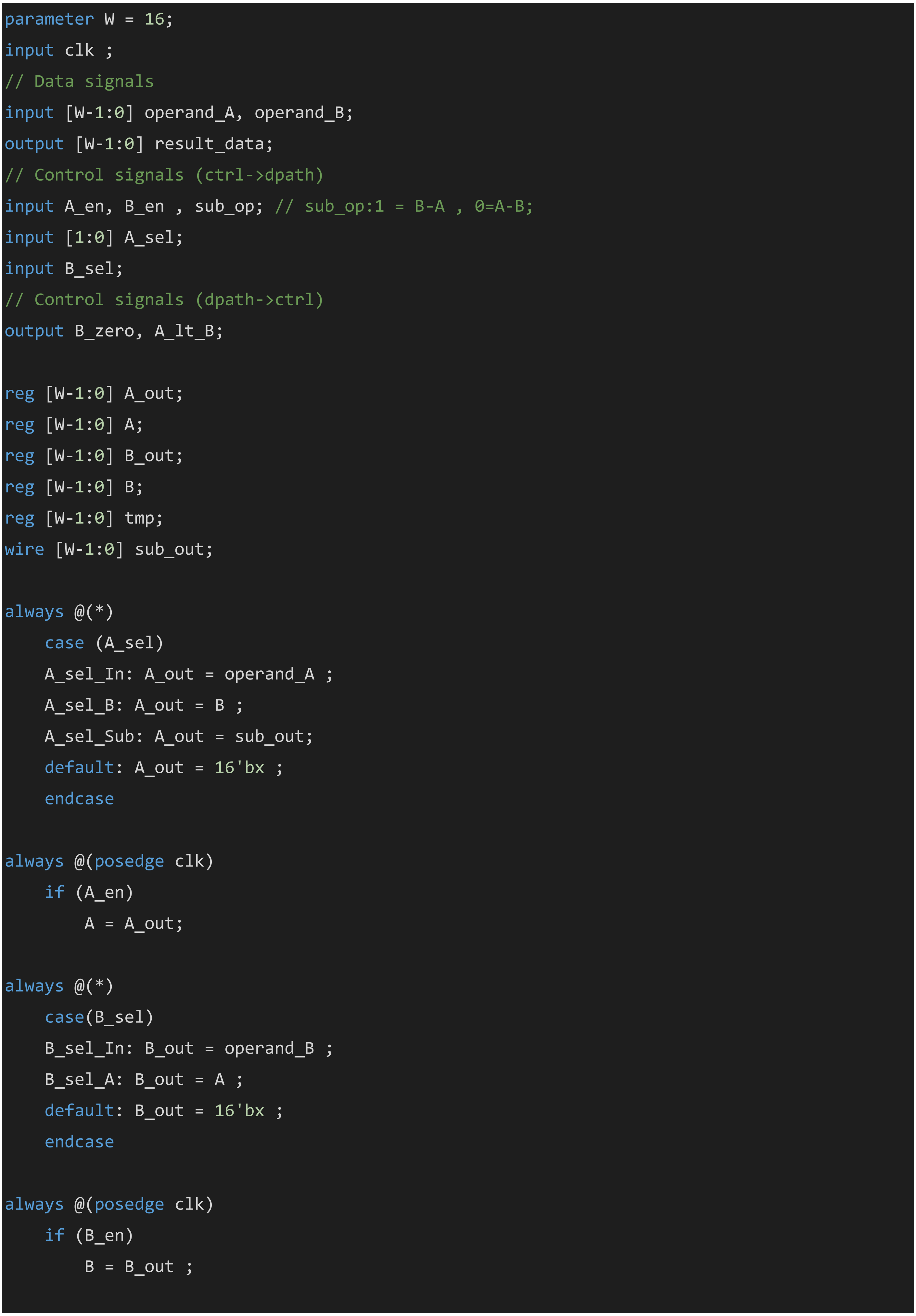
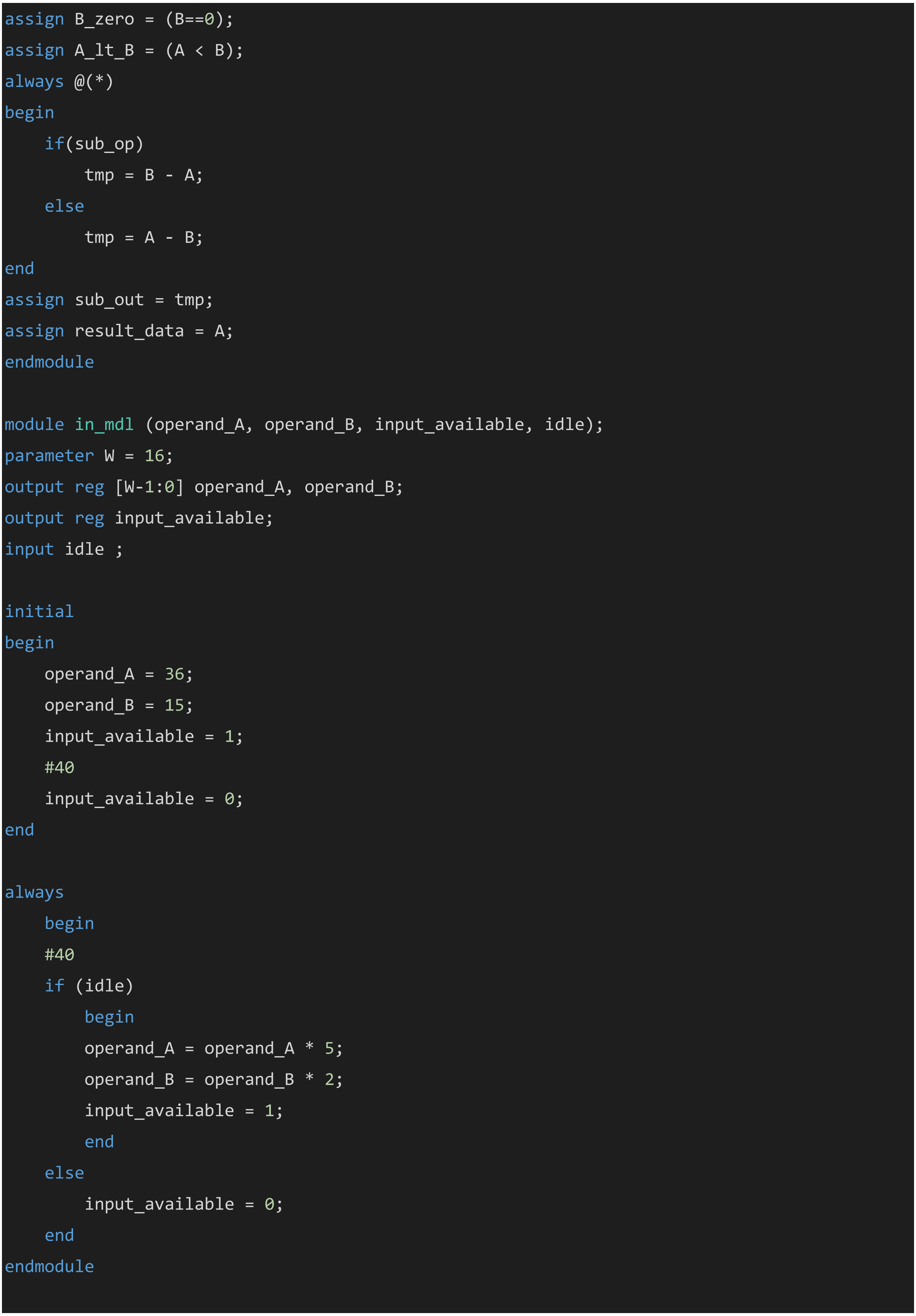


```
`timescale 1ns/10ps
module GCD_cntrl (idle, result_rdy, A_en, B_en, B_sel, clk, reset_, input_available, result_taken, B_zero, A_lt_B, A_sel , sub_op);
output reg idle, result_rdy, A_en, B_en, B_sel , sub_op; // sub_op:1 = B-A , 0=A-B;
input clk, reset_, input_available, result_taken, B_zero, A_lt_B;
output reg [1:0] A_sel ;

parameter WAIT = 2'd0, CALC = 2'd1, DONE = 2'd2;
parameter A_sel_In= 2'b00, A_sel_B=2'b01, A_sel_Sub=2'b10, A_sel_X=2'bxx;
parameter B_sel_In= 1'b0, B_sel_A= 1'b1, B_sel_X=1'bx;
parameter W = 16;

reg [1:0] next_state;
reg [1:0] state;

always @(*)
begin
next_state = state;
case ( state )
WAIT :
    if ( input_available )
        next_state = CALC;
CALC :
    if ( B_zero )
        next_state = DONE;
DONE :
    if ( result_taken )
        next_state = WAIT;
default:
    next_state = 2'bxx;
endcase
end

always @(posedge clk or negedge reset_)
    if(!reset_)
        state = WAIT ;
    else
        state = next_state ;

always @(*)
//Default control signals
begin
```

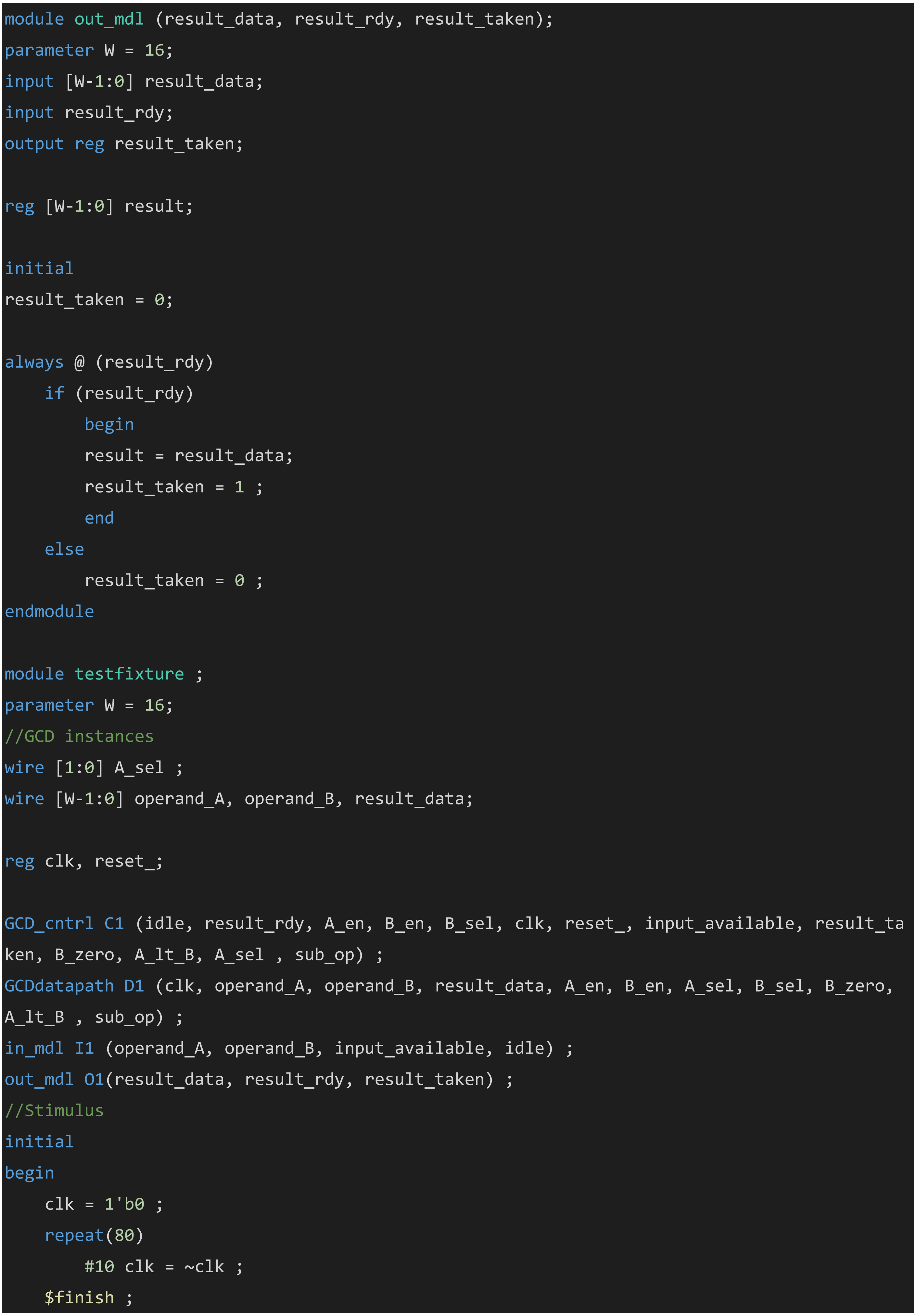
```
    A_sel = A_sel_X;
    A_en = 1'b0;
    B_sel = B_sel_X;
    B_en = 1'b0;
    idle = 1'b0;
    result_rdy = 1'b0;
    case ( state )
    WAIT: begin
        A_sel = A_sel_In;
        A_en = 1'b1;
        B_sel = B_sel_In;
        B_en = 1'b1;
        idle = 1'b1;
        end
    CALC: if ( A_lt_B ) begin
            A_sel = A_sel_Sub;
            sub_op = 1'b1;
            A_en = 1'b1;
            B_sel = B_sel_A;
            B_en = 1'b1;
            end
        else if ( !B_zero ) begin
            A_sel = A_sel_Sub;
            sub_op = 1'b0;
            A_en = 1'b1;
            end
    DONE: begin
        result_rdy = 1'b1;
        end
    endcase
end
endmodule

module GCDdatapath (clk, operand_A, operand_B, result_data, A_en, B_en, A_sel, B_sel, B_zero, A_lt_B , sub_op);
parameter A_sel_In= 2'b00;
parameter A_sel_B=2'b01;
parameter A_sel_Sub=2'b10;
parameter A_sel_X=2'bxx;
parameter B_sel_In= 1'b0;
parameter B_sel_A= 1'b1;
parameter B_sel_X=1'bx;
```



```
parameter W = 16;
input clk ;
// Data signals
input [W-1:0] operand_A, operand_B;
output [W-1:0] result_data;
// Control signals (ctrl->dpath)
input A_en, B_en , sub_op; // sub_op:1 = B-A , 0=A-B;
input [1:0] A_sel;
input B_sel;
// Control signals (dpath->ctrl)
output B_zero, A_lt_B;

reg [W-1:0] A_out;
reg [W-1:0] A;
reg [W-1:0] B_out;
reg [W-1:0] B;
reg [W-1:0] tmp;
wire [W-1:0] sub_out;

always @(*)
    case (A_sel)
    A_sel_In: A_out = operand_A ;
    A_sel_B: A_out = B ;
    A_sel_Sub: A_out = sub_out;
    default: A_out = 16'bx ;
    endcase

always @(posedge clk)
    if (A_en)
        A = A_out;

always @(*)
    case(B_sel)
    B_sel_In: B_out = operand_B ;
    B_sel_A: B_out = A ;
    default: B_out = 16'bx ;
    endcase

always @(posedge clk)
    if (B_en)
        B = B_out ;
```



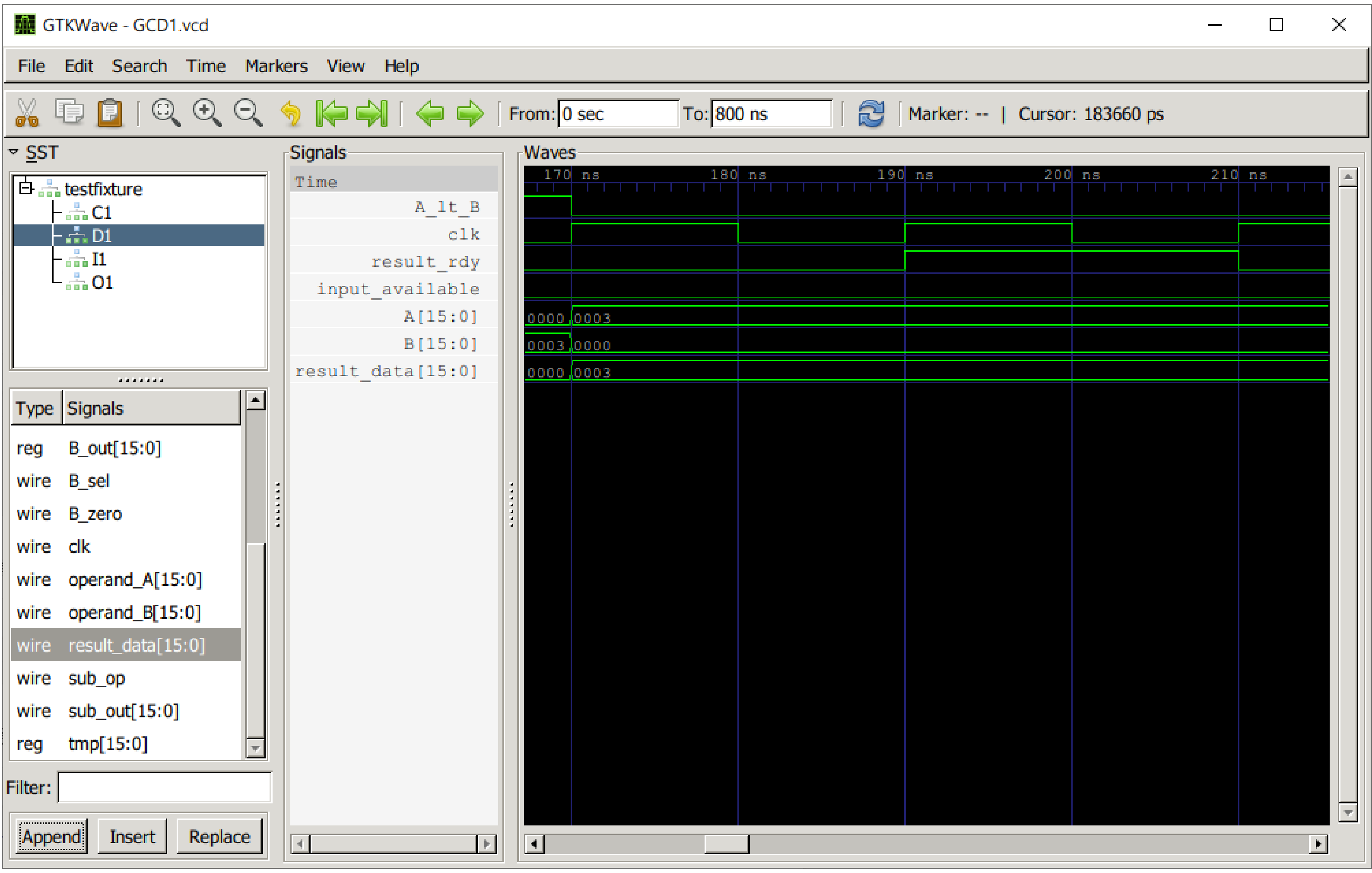
```
assign B_zero = (B==0);
assign A_lt_B = (A < B);
always @(*)
begin
    if(sub_op)
        tmp = B - A;
    else
        tmp = A - B;
end
assign sub_out = tmp;
assign result_data = A;
endmodule

module in_mdl (operand_A, operand_B, input_available, idle);
parameter W = 16;
output reg [W-1:0] operand_A, operand_B;
output reg input_available;
input idle ;

initial
begin
    operand_A = 36;
    operand_B = 15;
    input_available = 1;
    #40
    input_available = 0;
end

always
    begin
    #40
    if (idle)
        begin
        operand_A = operand_A * 5;
        operand_B = operand_B * 2;
        input_available = 1;
        end
    else
        input_available = 0;
    end
endmodule
```



```verilog
module out_mdl (result_data, result_rdy, result_taken);
parameter W = 16;
input [W-1:0] result_data;
input result_rdy;
output reg result_taken;

reg [W-1:0] result;

initial
result_taken = 0;

always @ (result_rdy)
    if (result_rdy)
        begin
        result = result_data;
        result_taken = 1 ;
        end
    else
        result_taken = 0 ;
endmodule

module testfixture ;
parameter W = 16;
//GCD instances
wire [1:0] A_sel ;
wire [W-1:0] operand_A, operand_B, result_data;

reg clk, reset_;

GCD_cntrl C1 (idle, result_rdy, A_en, B_en, B_sel, clk, reset_, input_available, result_taken, B_zero, A_lt_B, A_sel , sub_op) ;
GCDdatapath D1 (clk, operand_A, operand_B, result_data, A_en, B_en, A_sel, B_sel, B_zero, A_lt_B , sub_op) ;
in_mdl I1 (operand_A, operand_B, input_available, idle) ;
out_mdl O1(result_data, result_rdy, result_taken) ;
//Stimulus
initial
begin
    clk = 1'b0 ;
    repeat(80)
        #10 clk = ~clk ;
    $finish ;
```

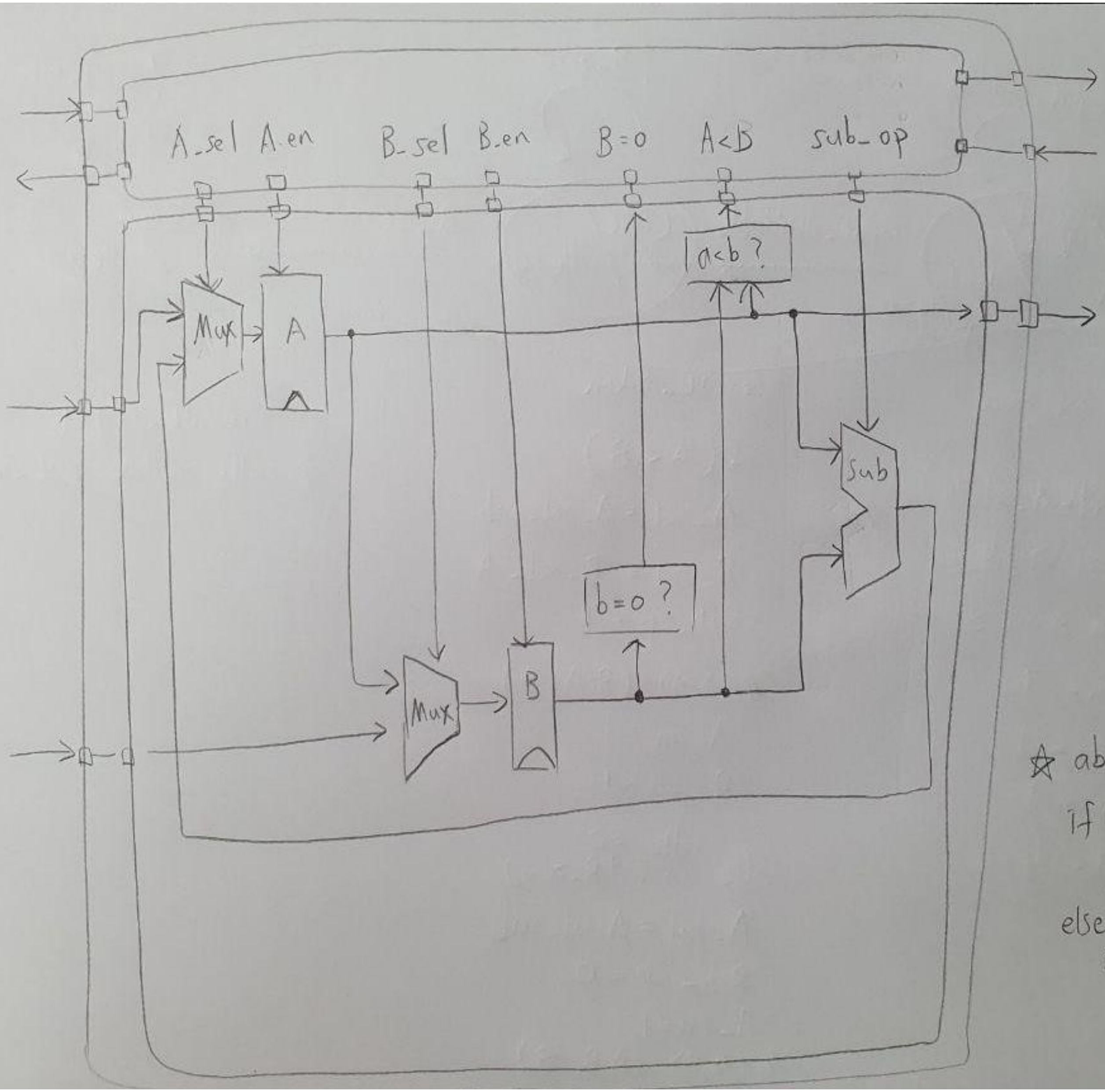
```
end

initial
begin
  reset_ = 0;
  #15  reset_ = 1'b1;
end


initial
begin
$dumpfile("GCD1.vcd");
$dumpvars;
end

endmodule
```

Timing diagram：（shows result before 200ns）


From: 0 sec To: 800 ns Marker: -- | Cursor: 183660 ps

Type	Signals
reg	B_out[15:0]
wire	B_sel
wire	B_zero
wire	clk
wire	operand_A[15:0]
wire	operand_B[15:0]
wire	result_data[15:0]
wire	sub_op
wire	sub_out[15:0]
reg	tmp[15:0]

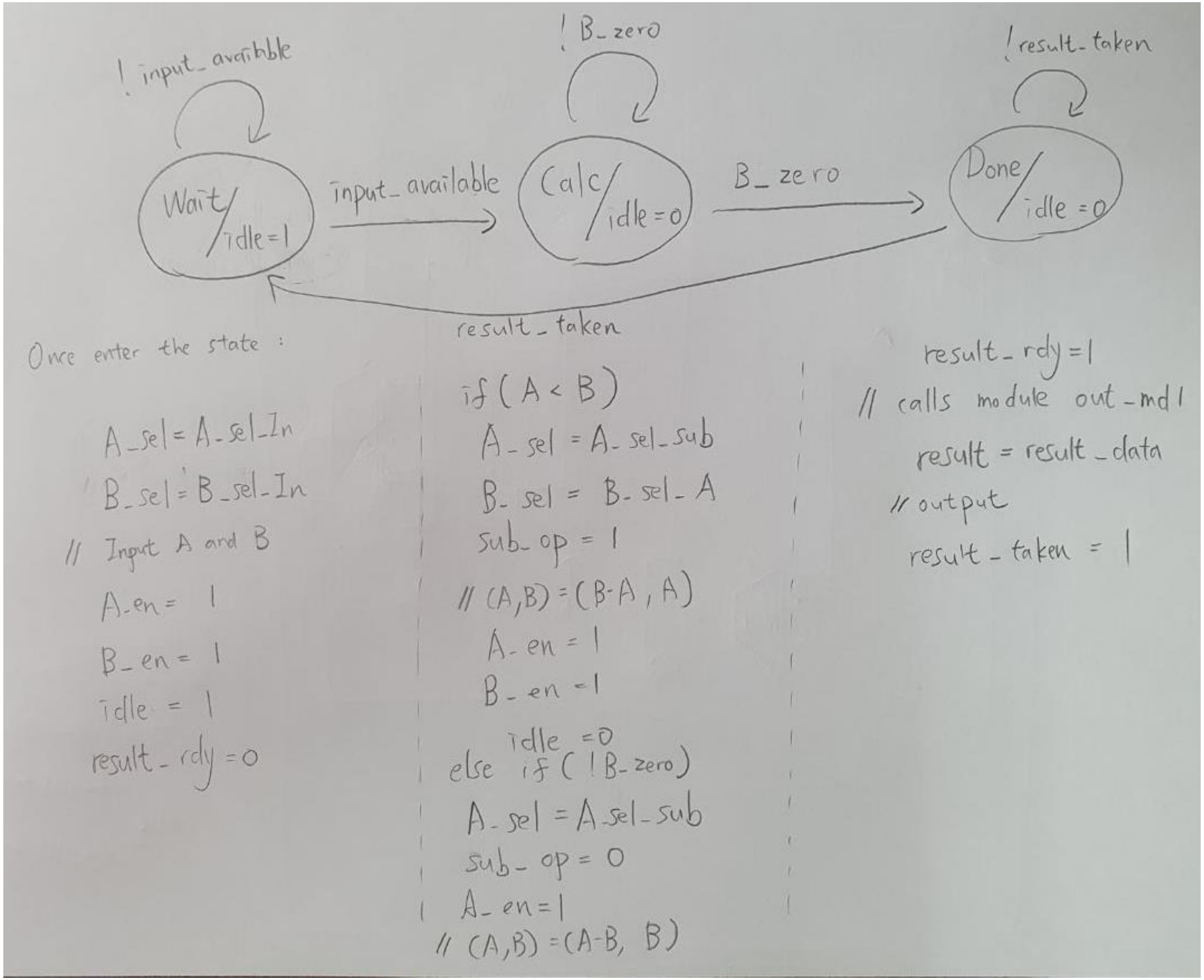
Revised diagram of datapath and control unit :



☆ about sub_op:

```
if (sub_op)
    sub_out = b-a
else
    sub_out = a-b
```

FSM :



Once enter the state :

```
A_sel = A_sel_In
B_sel = B_sel_In
// Input A and B
A_en = 1
B_en = 1
idle = 1
result_rdy = 0
```

```
if (A < B)
  A_sel = A_sel_sub
  B_sel = B_sel_A
  sub_op = 1
// (A,B) = (B-A, A)
  A_en = 1
  B_en = 1
  idle = 0
else if (!B_zero)
  A_sel = A_sel_sub
  sub_op = 0
  A_en = 1
// (A,B) = (A-B, B)
```

```
result_rdy = 1
// calls module out_mdl
result = result_data
// output
result_taken = 1
```