`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

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

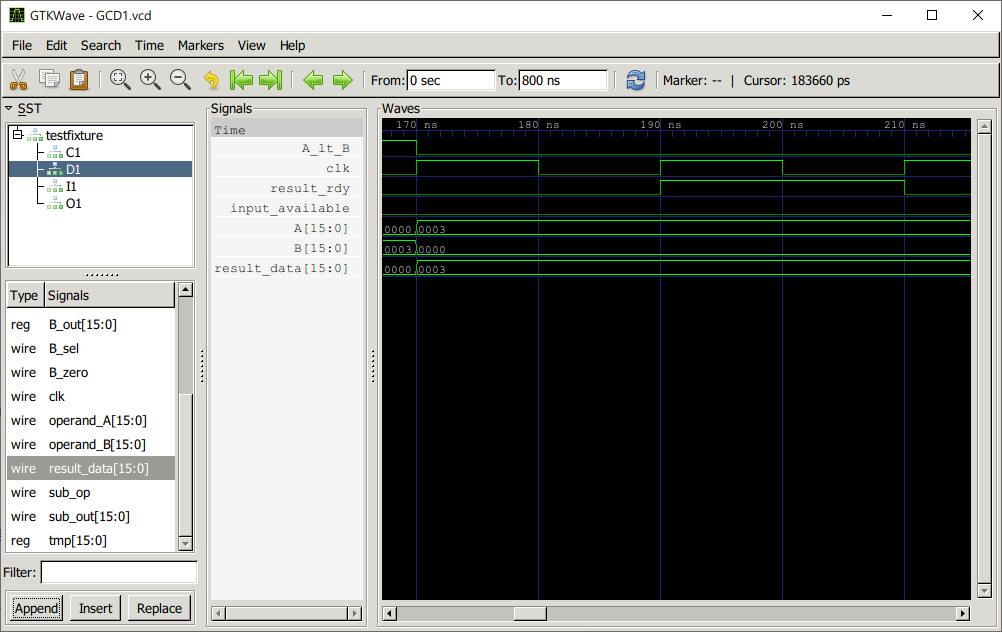
begin

$dumpfile("GCD1.vcd");

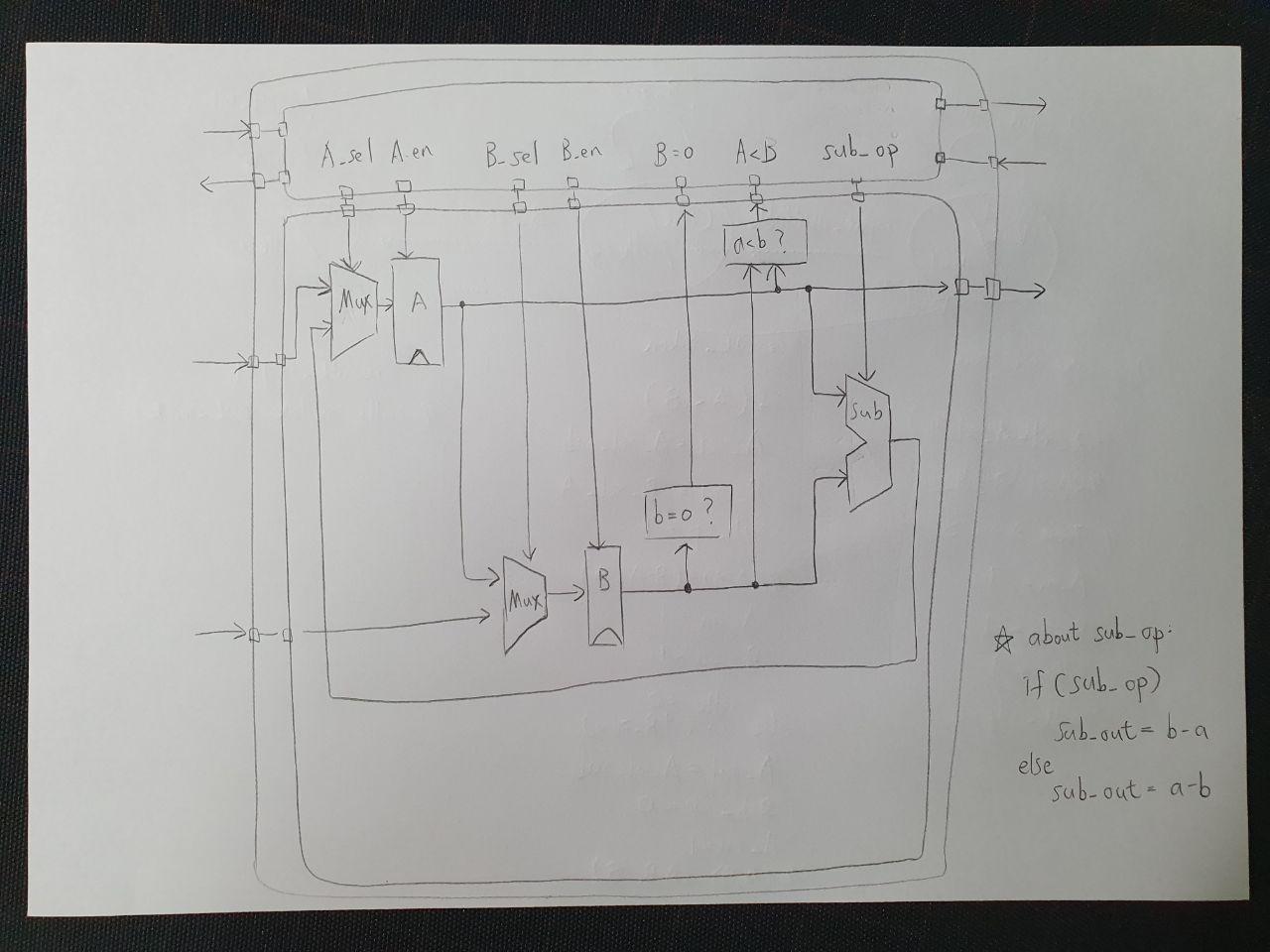
$dumpvars;

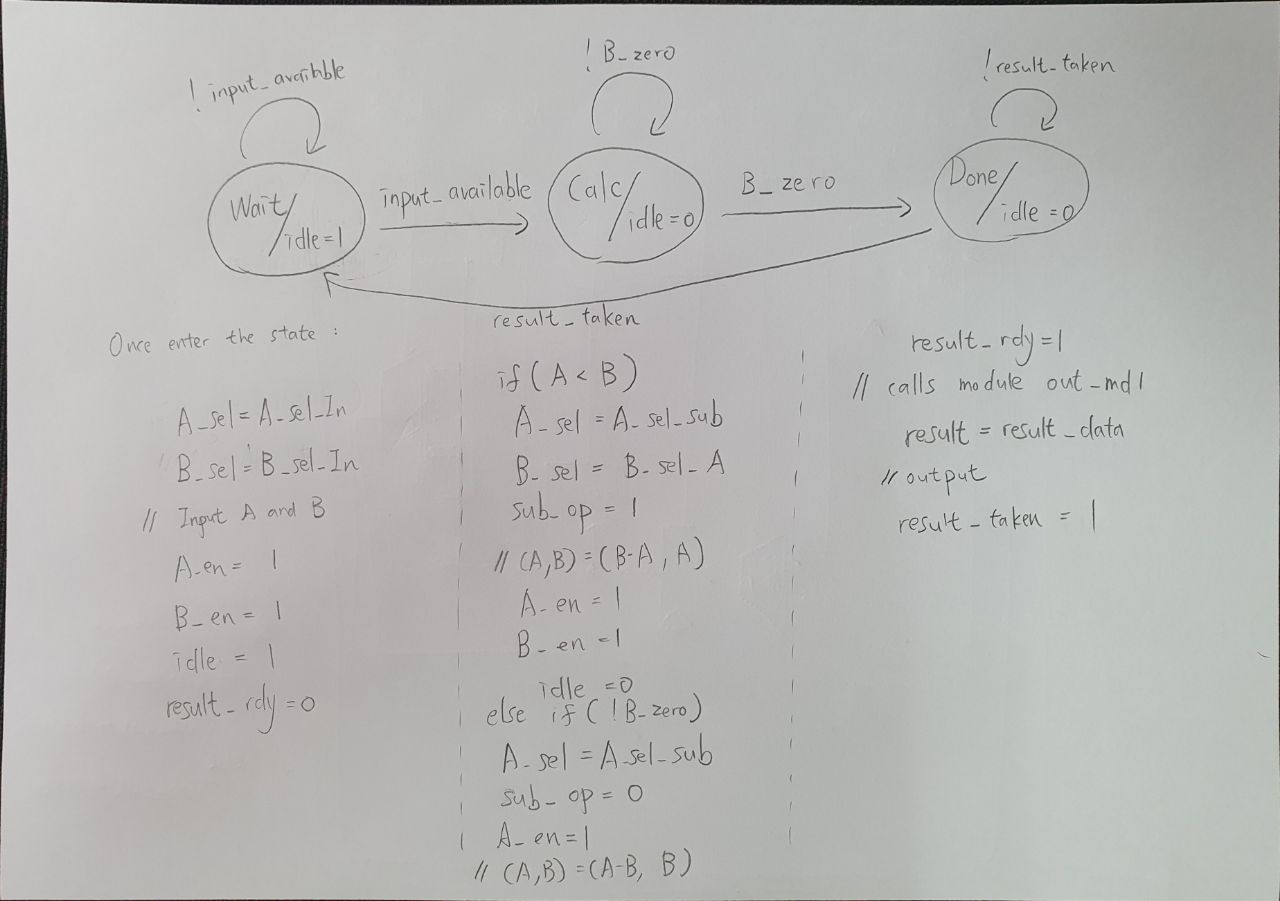
end

endmodule

Timing diagram：（shows result before 200ns）

Revised diagram of datapath and control unit :



FSM :