Functional Electronic Circuits Lab3

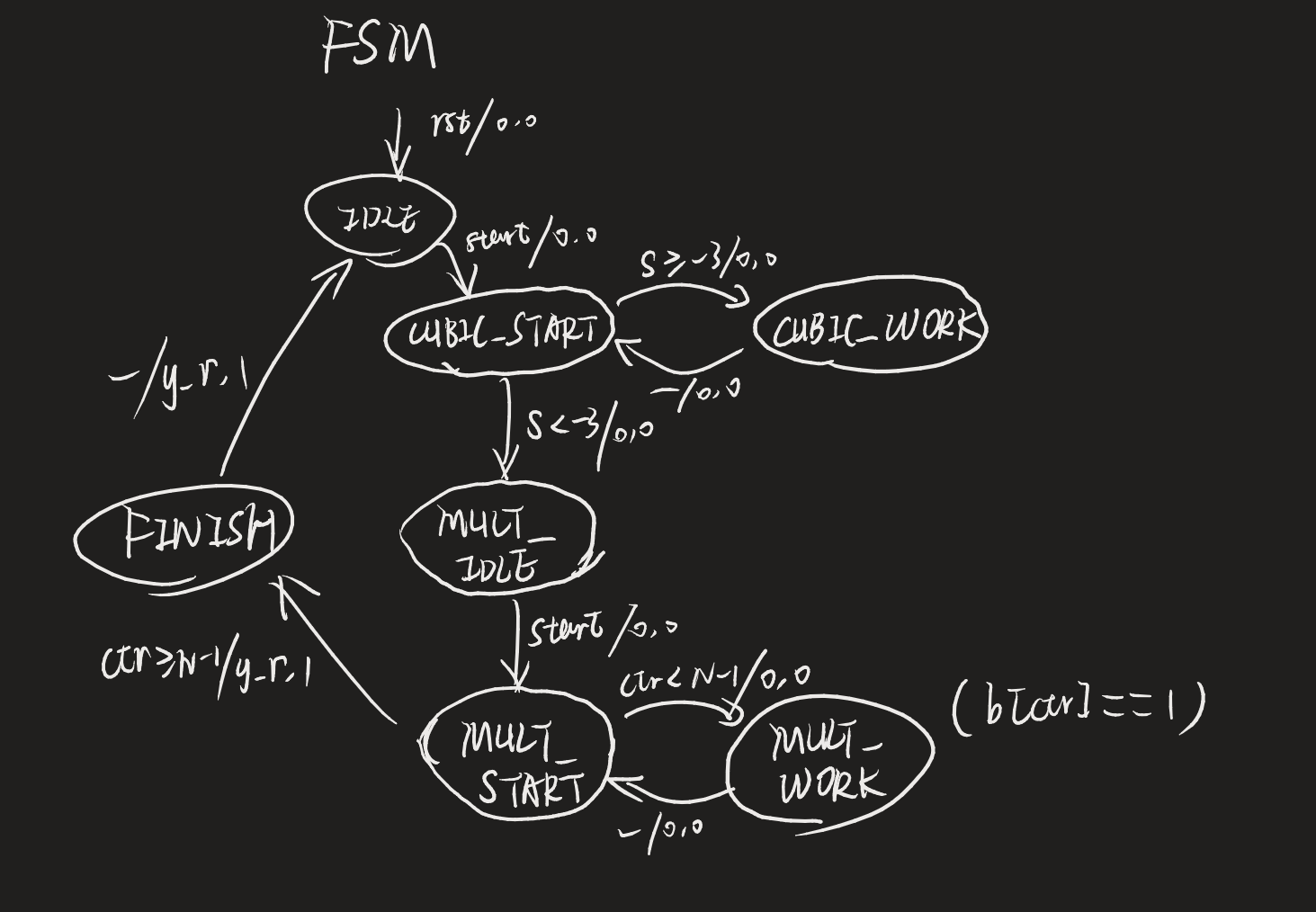
Student Name: CAO Xinyang

Student ID: 20321308

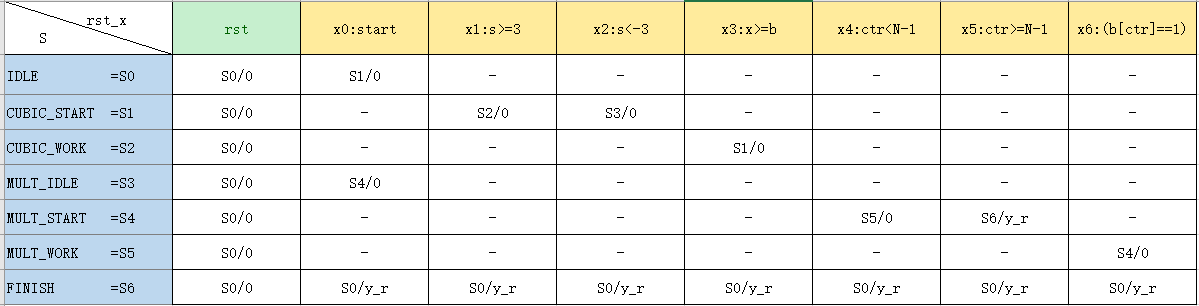
Variant: 8



**1. The picture with FSM**

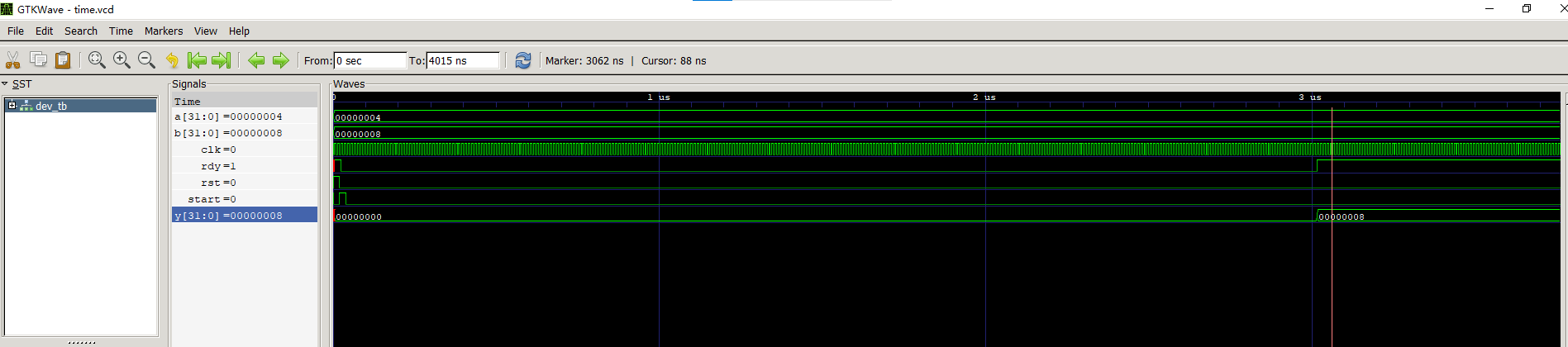


**2. The table of transition**

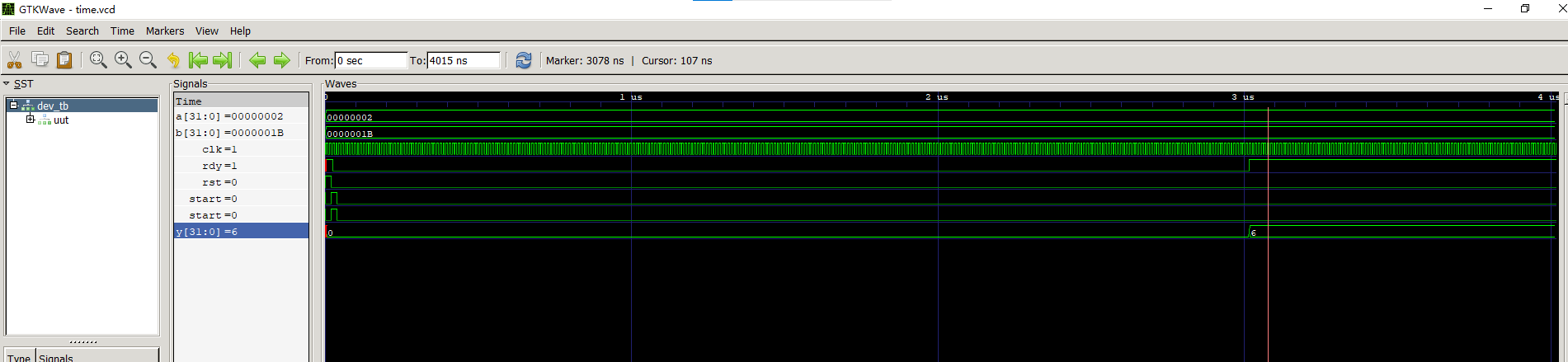
****

**3. The timing diagram with simulation results**

a = 4, b = 8 **🡺** y = 8

****

a = 2, b = 27 🡺 y = 6

****

**4. Code of the testbench and the device.**

***dev.v***

`include "adder.v"

`include "mult.v"

module dev (

    input clk\_i,

    input rst\_i,

    input [31:0] x\_bi,

    input [31:0] y\_bi,

    input start\_i,

    output reg [31:0] y\_bo,

    output reg rdy\_o

);

reg signed [31:0] a1\_op1, a1\_op2;

reg signed [31:0] a2\_op1, a2\_op2;

wire signed [31:0] a1\_res, a2\_res;

adder a1(

    .a\_bi(a1\_op1),

    .b\_bi(a1\_op2),

    .c\_bo(a1\_res)

);

adder a2(

    .a\_bi(a2\_op1),

    .b\_bi(a2\_op2),

    .c\_bo(a2\_res)

);

reg [31:0] m1\_op1, m1\_op2, y1;

reg m1\_start;

wire [15:0] m1\_res16;

wire [31:0] m1\_res = {16'h0, m1\_res16};

mult m1 (

    .clk\_i(clk\_i),

    .rst\_i(rst\_i),

    .a\_bi(m1\_op1[7:0]),

    .b\_bi(m1\_op2[7:0]),

    .start\_i(m1\_start),

    .busy\_o(m1\_busy),

    .y\_bo(m1\_res16)

);

// FSM control unit code

localparam N = 32;

localparam IDLE               = 0;

localparam CMP\_S              = 1;

localparam CALC\_MULT\_S1\_START = 2;

localparam CALC\_MULT\_S1       = 3;

localparam CALC\_MULT\_S2\_START = 4;

localparam CALC\_MULT\_S2       = 5;

localparam CALC\_B             = 6;

localparam CMP\_B              = 7;

localparam CALC\_S             = 8;

localparam CALC\_MULT\_S3 = 9;

localparam FINISH\_START       = 10;

localparam FINISH             = 11;

reg [3:0] state\_r;

reg [31:0] x\_r;

reg signed [31:0] s\_r;

reg [31:0] y\_r;

reg [31:0] b\_r;

always@(posedge clk\_i)

    if(rst\_i) begin

        x\_r <= 0;

        m1\_op1 <= 0;

        m1\_op2 <= 0;

        m1\_start <= 0;

        a1\_op1 <= 0;

        a1\_op2 <= 0;

        a2\_op1 <= 0;

        a2\_op2 <= 0;

        y\_bo   <= 0;

        y\_r    <= 0;

        rdy\_o  <= 1;

        s\_r    <= 0;

        y1     <= 0;

        state\_r <= IDLE;

    end else begin

        case(state\_r)

            IDLE:

                if(start\_i) begin

                    x\_r <= x\_bi;

                    y\_r <= 0;

                    rdy\_o <= 0;

                    b\_r <= 0;

                    s\_r <= 0;

                    // s = N-2

                    a1\_op1 <= N;

                    a1\_op2 <= -2;

                    state\_r <= CMP\_S;

                end

            CMP\_S:

                begin

                    s\_r <= a1\_res;

                    if(a1\_res > -3) begin

                        y\_r = y\_r << 1;

                        // 3\*y

                        m1\_op1 <= 3;

                        m1\_op2 <= y\_r;

                        m1\_start <= 1;

                        // y+1

                        a1\_op1 <= y\_r;

                        a1\_op2 <= 1;

                        state\_r <= CALC\_MULT\_S1\_START;

                    end else begin

                        y1 <= y\_r;

                        m1\_start <= 0;

                        state\_r <= CALC\_MULT\_S3;

                    end

                end

            CALC\_MULT\_S3:

                if(m1\_busy == 0) begin

                    m1\_op1 <= y1;

                    m1\_op2 <= y\_bi;

                    m1\_start <= 1;

                    state\_r <= FINISH\_START;

                end

            FINISH\_START:

                begin

                    m1\_start <= 0;

                    state\_r <= FINISH;

                end

            FINISH:

                if(m1\_busy == 0) begin

                    y\_bo <= m1\_res;

                    rdy\_o <= 1;

                    state\_r <= IDLE;

                end

            CALC\_MULT\_S1\_START: //waiting 1 clk for starting of the mult module

                begin

                    m1\_start <= 0;

                    state\_r <= CALC\_MULT\_S1;

                end

            CALC\_MULT\_S1:

                if(m1\_busy == 0) begin

                    // m1\_res = 3\*y

                    // a1\_res = y+1

                    // m1\_res \* a1\_res = 3\*y\*(y+1)

                    m1\_op1 <= m1\_res;

                    m1\_op2 <= a1\_res;

                    m1\_start <= 1;

                    state\_r <= CALC\_MULT\_S2\_START;

                end

            CALC\_MULT\_S2\_START: //waiting 1 clk for starting of the mult module

                begin

                    m1\_start <= 0;

                    state\_r <= CALC\_MULT\_S2;

                end

            CALC\_MULT\_S2:

                if(m1\_busy == 0) begin

                    // 3\*y(y+1)+1

                    a1\_op1 <= m1\_res;

                    a1\_op2 <= 1;

                    state\_r <= CALC\_B;

                end

            CALC\_B:

                begin

                    b\_r <= a1\_res << s\_r;

                    state\_r <= CMP\_B;

                end

            CMP\_B:

                if(x\_r >= b\_r) begin

                    // x-b

                    a1\_op1 <= x\_r;

                    a1\_op2 <= -b\_r;

                    // y+1

                    a2\_op1 <= y\_r;

                    a2\_op2 <= 1;

                    state\_r <= CALC\_S;

                end else begin

                    state\_r <= CALC\_S;

                end

            CALC\_S:

                begin

                    if(x\_r >= b\_r) begin

                        //x = x-b

                        x\_r <= a1\_res;

                        //y = y+1

                        y\_r <= a2\_res;

                    end

                    // s-3

                    a1\_op1 <= s\_r;

                    a1\_op2 <= -3;

                    state\_r <= CMP\_S;

                end

        endcase

    end

endmodule

***dev\_tb.v***

`include "dev.v"

`timescale 1ns/1ps

module dev\_tb;

reg [31:0] a,b;

reg clk, rst;

reg start;

wire rdy;

wire [31:0] y;

dev uut(

    .clk\_i(clk),

    .rst\_i(rst),

    .x\_bi(b),

    .y\_bi(a),

    .start\_i(start),

    .y\_bo(y),

    .rdy\_o(rdy)

);

always #5 clk = ~clk;

initial begin

    $dumpfile("time.vcd");

    $dumpvars(0, dev\_tb);

    clk   = 0;

    rst   = 1;

    a     = 4;

    b     = 8;

    start = 0;

    #20

    rst   = 0;

    start = 1;

    #20

    start = 0;

    @(posedge rdy);

    #1000

    $finish;

end

endmodule

***adder.v***

module adder (

    input [31:0] a\_bi,

    input [31:0] b\_bi,

    output [31:0] c\_bo

);

assign c\_bo = a\_bi + b\_bi;

endmodule

***mult.v***

module mult(

    input clk\_i,

    input rst\_i,

    input [7:0] a\_bi,

    input [7:0] b\_bi,

    input start\_i,

    output reg busy\_o,

    output reg [15:0] y\_bo

);

    localparam IDLE = 2'b00;

    localparam WORK = 2'b01;

    localparam END  = 2'b10;

    reg   [2:0] ctr;

    wire  [2:0] end\_step;

    wire  [7:0] part\_sum;

    wire [15:0] shifted\_part\_sum;

    reg   [7:0] a, b;

    reg  [15:0] part\_res;

    reg   [1:0] state;

    assign part\_sum = a & {8{b[ctr]}};

    assign shifted\_part\_sum = part\_sum << ctr;

    always @(posedge clk\_i)

        if (rst\_i) begin

           ctr      <= 0;

           part\_res <= 0;

           y\_bo     <= 0;

           busy\_o   <= 0;

           state <= IDLE;

        end else begin

            case (state)

                IDLE:

                    if (start\_i) begin

                        state  <= WORK;

                        a        <= a\_bi;

                        b        <= b\_bi;

                        ctr      <= 0;

                        part\_res <= 0;

                        busy\_o   <= 1;

                    end

                WORK:

                    begin

                        if (ctr == 3'h7) begin

                            state  <= END;

                        end

                        part\_res <= part\_res + shifted\_part\_sum;

                        ctr <= ctr + 1;

                    end

                END:

                    begin

                        y\_bo   <= part\_res;

                        busy\_o <= 0;

                        state  <= IDLE;

                    end

                default: state <= IDLE;

            endcase

        end

endmodule