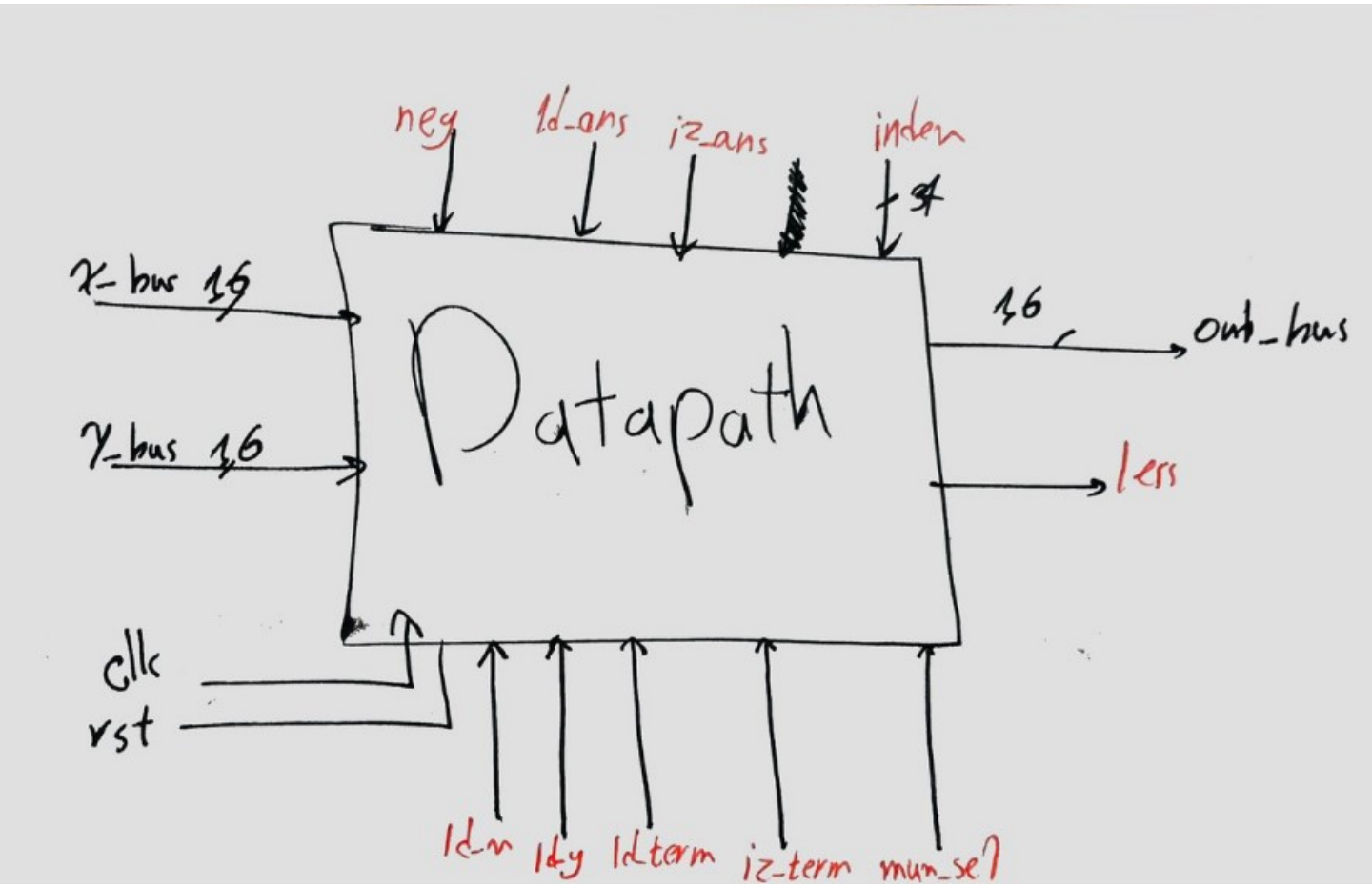
Digital Systems, CA6

Design phase

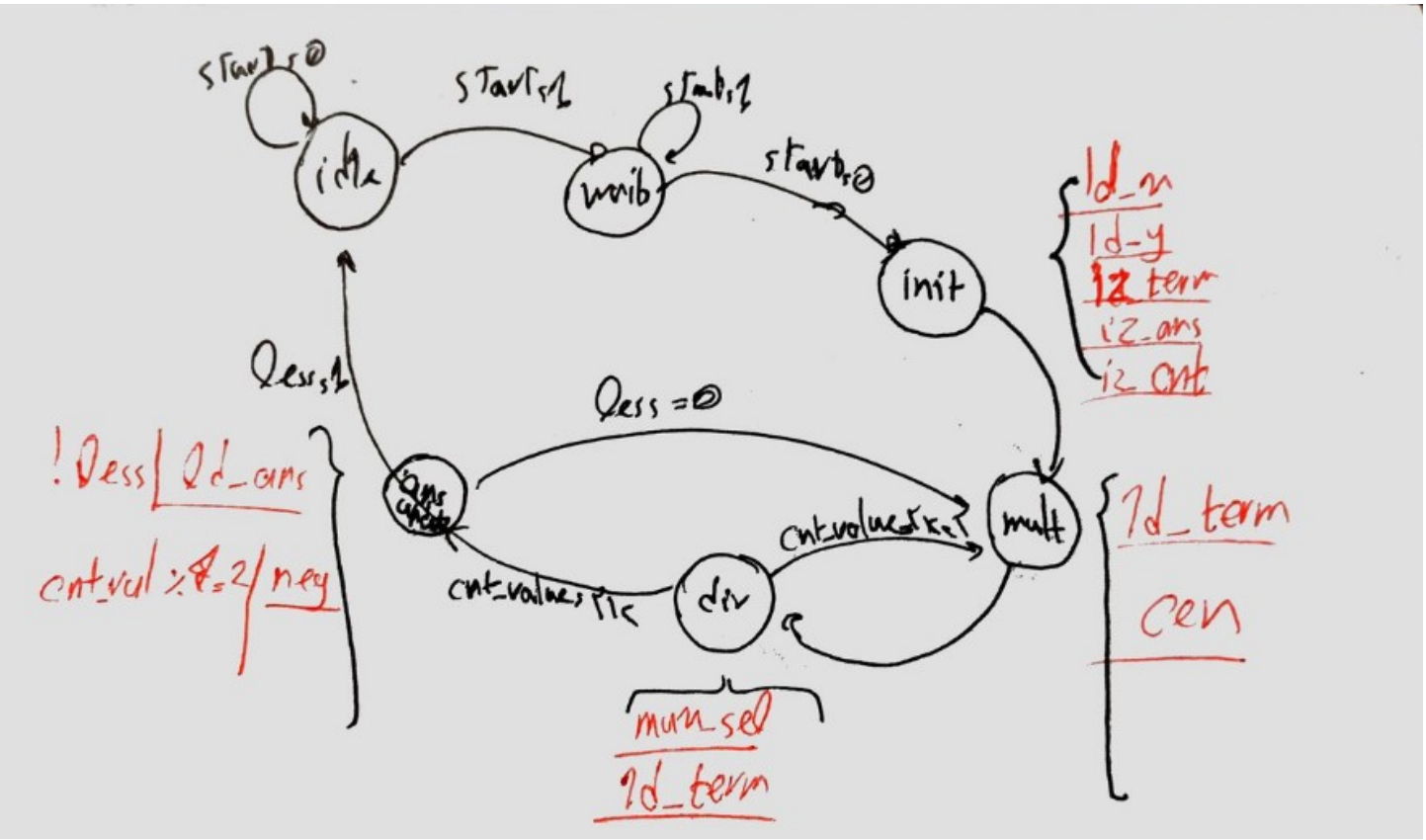
Porblem 1:

The schematic diagram for my datapath:



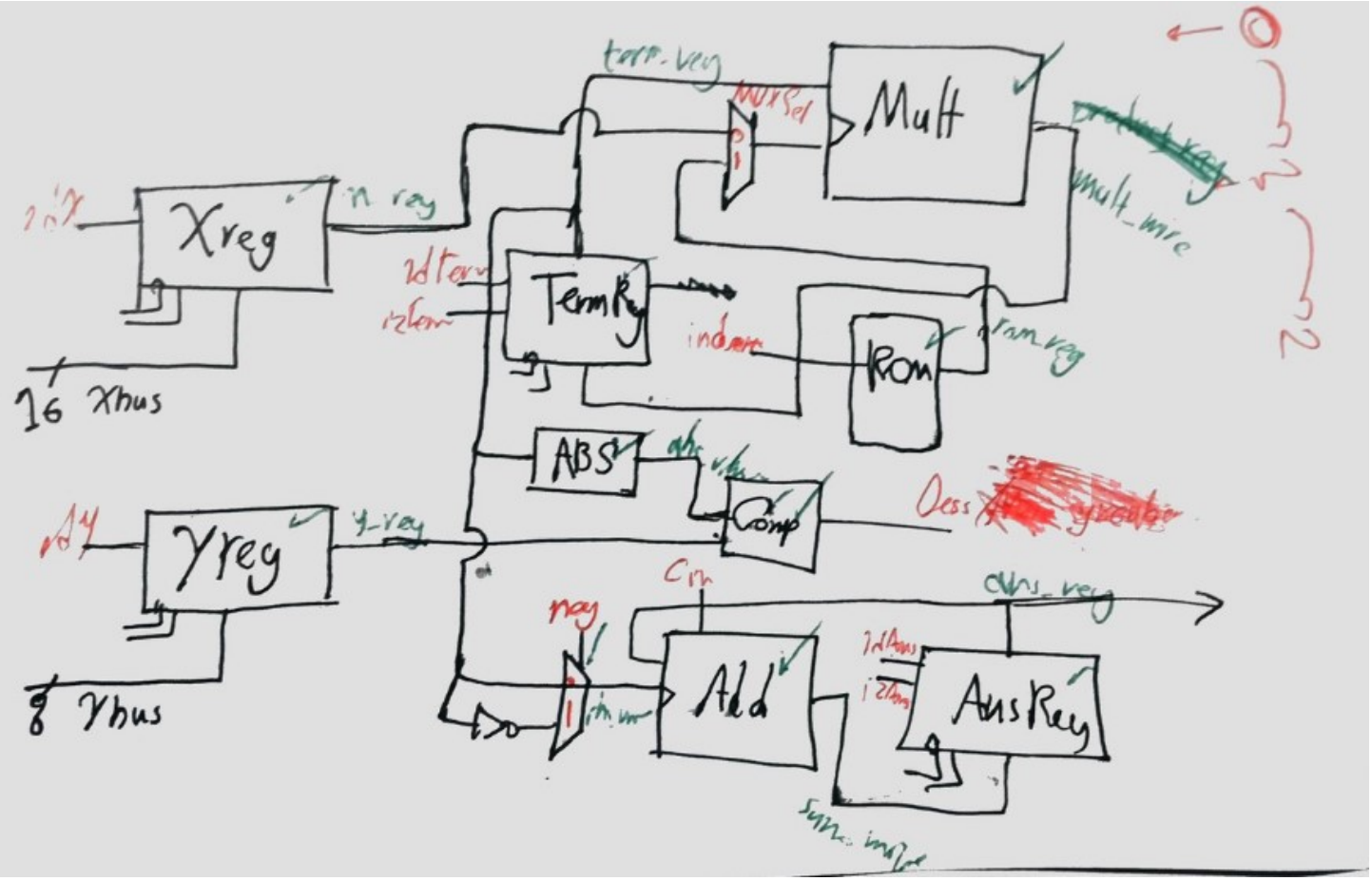
Problem 2:

The state diagram of controller:



Problem 3:

The implemented circuit diagram of datapath:



And the Verilog description for our datapath:

```
`timescale 1ns/1ns
module Datapath(input clk, rst, input [15:0] x_bus, input [7:0] y_bus,
    input ld_x, ld_y, ld_term, iz_term, mux_sel, neg, ld_ans, iz_ans, input [3:0] index,
    output less, output [15:0] out_bus
    );

    reg [15:0] x_reg = 0, y_reg = 0, term_reg = 0, ans_reg = 0;
    wire [15:0] mult_wire, rom_wire, sum_wire, add_rhs_wire, mult_rhs_wire;

    // X Register
    always @(posedge clk, posedge rst) begin
        if (rst)
            x_reg <= 16'b0;
        else begin
            if (ld_x)
                x_reg <= x_bus;
            else
                x_reg <= x_reg;
        end
    end

    // Y Register
    always @(posedge clk, posedge rst) begin
        if (rst)
```

```
            y_reg <= 16'b0;
        else begin
            if (ld_y)
                y_reg <= {8'b0, y_bus};
            else
                y_reg <= y_reg;
        end
    end

    // Term Register
    always @(posedge clk, posedge rst) begin
        if (rst)
            term_reg <= 16'b0;
        else begin
            if (ld_term)
                term_reg <= mult_wire;
            else begin
                if (iz_term)
                    term_reg <= 16'b0000000100000000;
                else
                    term_reg <= term_reg;
            end
        end
    end

    // Answer Register
    always @(posedge clk, posedge rst) begin
        if (rst)
            ans_reg <= 16'b0;
        else begin
            if (ld_ans)
                ans_reg <= sum_wire;
            else begin
                if (iz_ans)
                    ans_reg <= 16'b0000000100000000;
                else
                    ans_reg <= ans_reg;
            end
        end
    end

    // ROM
    wire [7:0] rom_8_bit_out;
    LUT rom_lut(index, rom_8_bit_out);
    assign rom_wire = {8'b0, rom_8_bit_out};

    // Multiplier
    reg [31:0] mult_product_reg_32;
    reg [15:0] mult_product_reg_16;
    assign mult_rhs_wire = mux_sel ? rom_wire : x_reg;
    always @(term_reg, mult_rhs_wire) begin
        mult_product_reg_32 = term_reg * mult_rhs_wire;
        mult_product_reg_16 = mult_product_reg_32[23:8];
    end
```

```
    assign mult_wire = mult_product_reg_16;

    // Adder
    assign add_rhs_wire = neg ? ~term_reg : term_reg;
    assign sum_wire = ans_reg + add_rhs_wire + neg;

    // Comperator
    assign less = term_reg < y_reg;

    assign out_bus = ans_reg;
endmodule
```

Porblem 4

The Verilog description of controller:

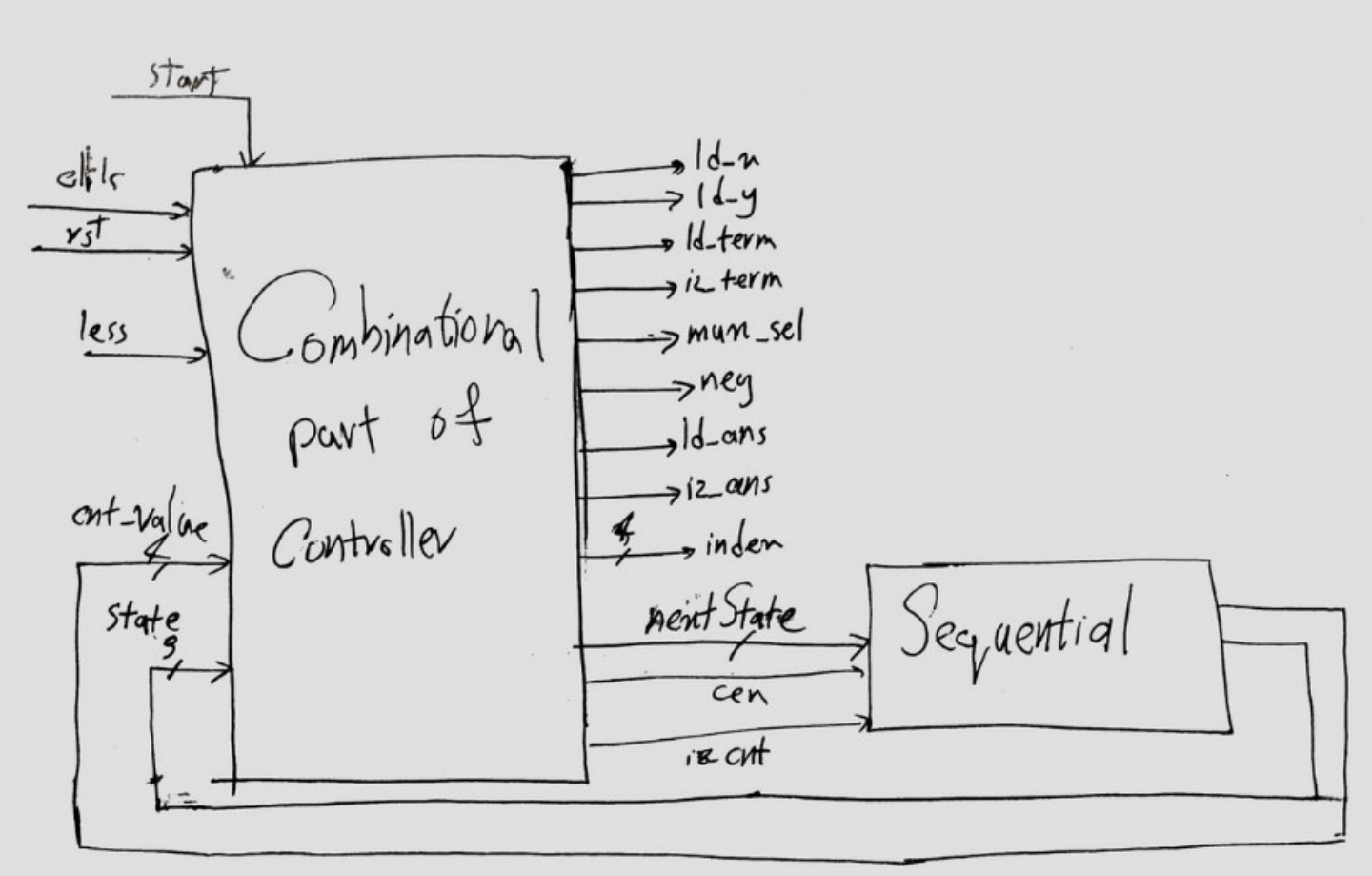
```
`timescale 1ns/1ns
module Controller(input clk, rst, start, less,
    output ld_x, ld_y, ld_term, iz_term, mux_sel, neg, ld_ans, iz_ans, output [3:0] index
    );

    parameter [2:0] idle = 0, waiting = 1, initalization = 2, multipy = 3, divide = 4, update_ans = 5;
    reg [2:0] state = idle, nextState = idle;

    reg ld_x_reg = 0, ld_y_reg = 0, ld_term_reg = 0, iz_term_reg = 0, mux_sel_reg = 0, neg_reg = 0, ld_an

    assign ld_x = ld_x_reg;
    assign ld_y = ld_y_reg;
    assign ld_term = ld_term_reg;
    assign iz_term = iz_term_reg;
    assign ld_ans = state == update_ans;
    assign iz_ans = iz_ans_reg;
    assign mux_sel = mux_sel_reg;
    assign neg = neg_reg;

    reg cen, iz_cnt;
    reg [3:0] cnt_val = 4'b0;

    assign index = cnt_val;

    // Counter
    always @(posedge clk, posedge rst) begin
        if (rst)
            cnt_val <= 4'b0;
        else begin
            if (iz_cnt)
                cnt_val <= 4'b0;
            else begin
                if (cen)
                    cnt_val <= cnt_val + 1;
                else
                    cnt_val <= cnt_val;
```

```
        end
    end
end

always @(state, start, less) begin
    cen = 0;
    iz_cnt = 0;
    ld_x_reg = 0;
    ld_y_reg = 0;
    ld_term_reg = 0;
    iz_term_reg = 0;
    mux_sel_reg = 0;
    neg_reg = 0;
    ld_ans_reg = 0;
    iz_ans_reg = 0;

    case (state)
        idle: begin
            if (start)
                nextState = waiting;
        end

        waiting: begin
            if (~start)
                nextState = initalization;
        end

        initalization: begin
            ld_x_reg = 1;
            ld_y_reg = 1;
            iz_cnt = 1;
            iz_term_reg = 1;
            iz_ans_reg = 1;

            nextState = multipy;
        end

        multipy: begin
            ld_term_reg = 1;
            cen = 1;
            nextState = divide;
        end

        divide: begin
            mux_sel_reg = 1;
            ld_term_reg = 1;
            if (cnt_val[0])
                nextState = multipy;
            else
                nextState = update_ans;
        end

        update_ans: begin
            if (less)
```

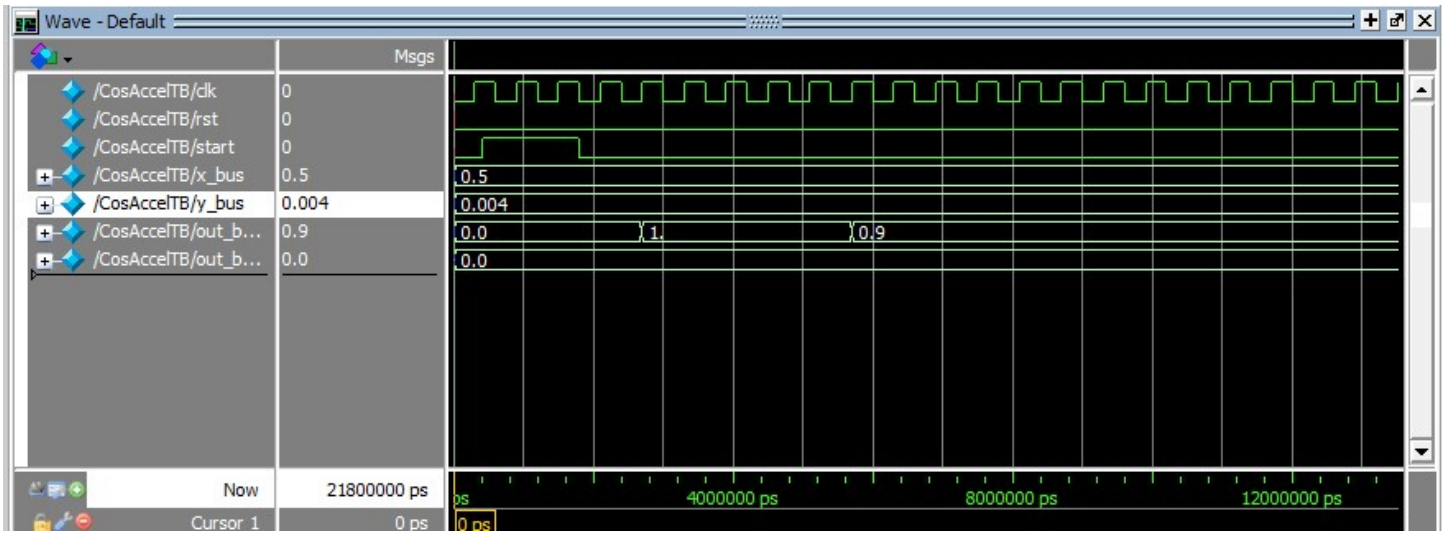
```
                    nextState = idle;
                else begin
                    ld_ans_reg = 1;
                    nextState = multipy;
                    if (cnt_val[1:0] == 2'b10)
                        neg_reg = 1;
                end
            end

            default: nextState = idle;
        endcase
    end

    always @(posedge clk, posedge rst) begin
        if (rst)
            state <= idle;
        else
            state <= nextState;
    end
endmodule
```

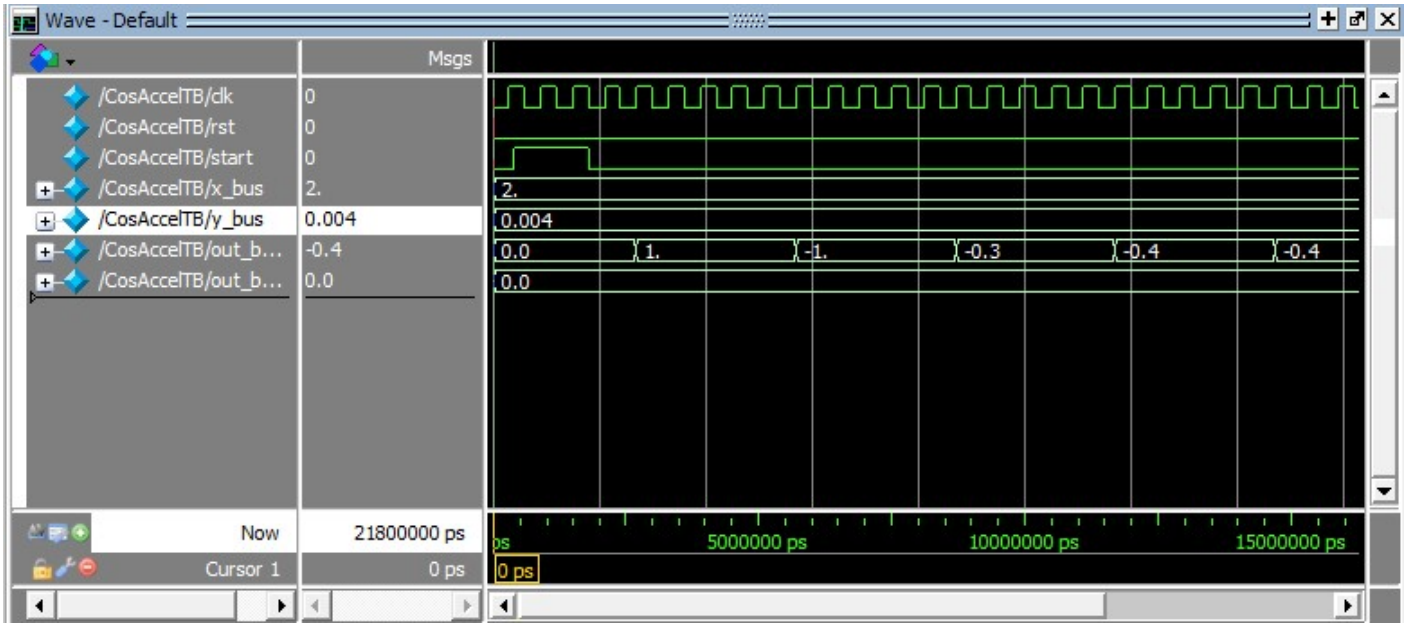
Problem 5

The huffman model of the circuit:



Implementation phase





Flow Status	Successful - Tue Jan 09 20:01:14 2024
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	CosAccel
Top-level Entity Name	CosAccel
Family	Cyclone IV GX
Device	EP4CGX15BF14A7
Timing Models	Final
Total logic elements	390 / 14,400 (3 %)
Total registers	66
Total pins	43 / 81 (53 %)
Total virtual pins	0
Total memory bits	0 / 552,960 (0 %)
Embedded Multiplier 9-bit elements	0
Total GXB Receiver Channel PCS	0 / 2 (0 %)
Total GXB Receiver Channel PMA	0 / 2 (0 %)
Total GXB Transmitter Channel PCS	0 / 2 (0 %)
Total GXB Transmitter Channel PMA	0 / 2 (0 %)
Total PLLs	0 / 3 (0 %)