DIGITAL SYSTEM DESIGN APPLICATION – EHB 436E

Final Project

Muhammed Velihan Bağcı

040170093

Yiğit Bektaş Gürsoy

040180063

Class Lecturer: Sıddıka Berna Örs Yalçın

Class Assistant:
Serdar Duran
Yasin Fırat Kula
Mehmet Onur Demirtürk

Muhammed Velihan BAĞCI
040170093
Yiğit Bektaş GÜRSOY
040180063

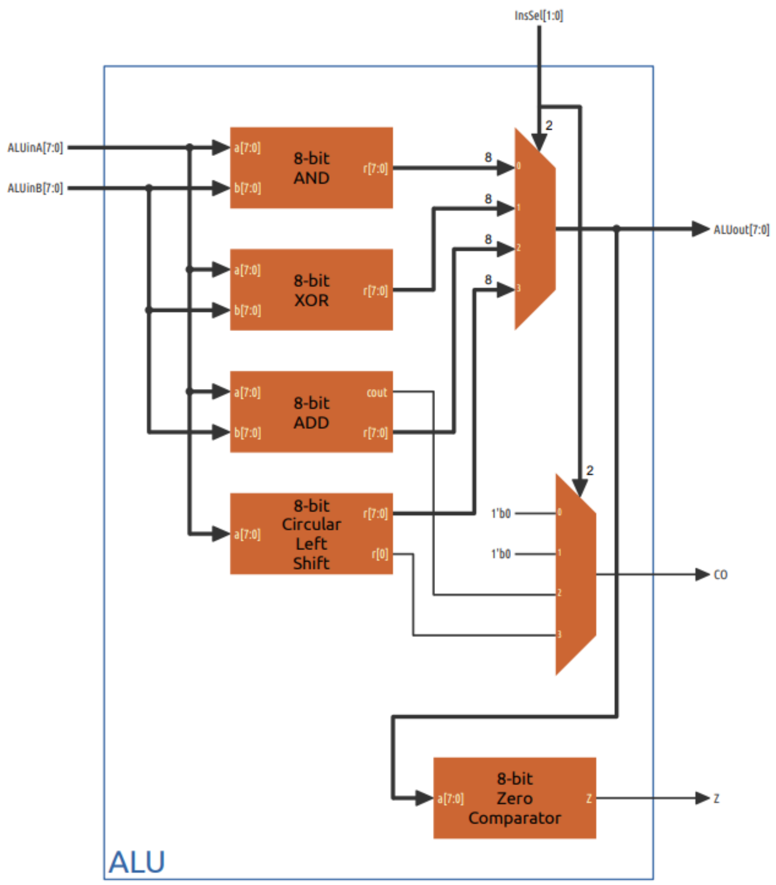
1. Description of The Problem

The assignment given in ninova is shown below. A is to take the power of these integers to be 0, 1 , 2 or 3. Force values can be 0.1,2 or 3. The minimum number that can output is 0. It has the value 0^0 which is undefined. The maximum possible number is 27. To define 27, we need to define a 5-bit result. These will be discussed in more detail in the algorithm section.

2) Calculate $C = A^B$, where A and B are 2-bit positive integers and C is a 6-bit positive integer.

2. ALU (Arithmetic Logic Unit)

The Arithmetic Logic Unit (ALU) is a digital circuit within a computer's processor that performs arithmetic and logical operations. It is responsible for executing instructions stored in memory. The ALU has two inputs for operands, performs various operations such as addition, bit shifting, and operation and xor operation, and it has a set of status or flag registers that indicate the results of the previous operation. These flags are used by the processor to make decisions and control the flow of the program. An image of the ALU is described below.

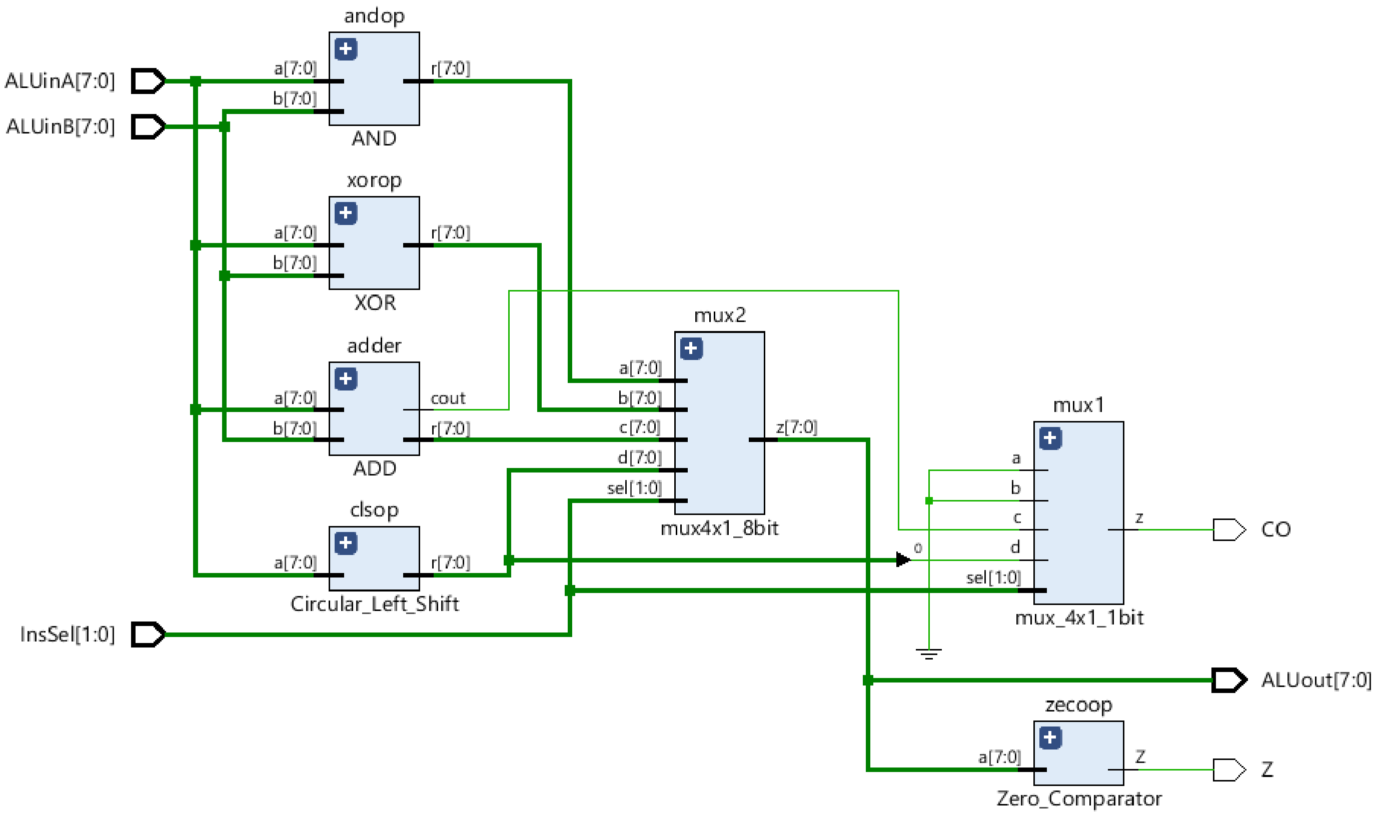


Verilog Code

```
`timescale 1ns / 1ps

module ALU(
    input [7:0] ALUinA, ALUinB,
    input [1:0] InsSel,
    output [7:0] ALUout,
    output CO,
    output Z
    );
    wire [7:0] AND_r, XOR_r, ADD_r, CLS_r, ALU_r;
    wire ADD_cout;

    ADD adder (
        .a(ALUinA),
        .b(ALUinB),
        .r(ADD_r),
        .cout(ADD_cout)
        );

    AND andop (
        .a(ALUinA),
        .b(ALUinB),
        .r(AND_r)
        );

    XOR xorop (
        .a(ALUinA),
        .b(ALUinB),
        .r(XOR_r)
        );

    Circular_Left_Shift clsop (
        .a(ALUinA),
        .r(CLS_r)
        );

    Zero_Comparator zecoop(
        .a(ALU_r),
        .Z(Z)
        );

    mux_4x1_1bit mux1(
        .a(1'b0),
        .b(1'b0),
        .c(ADD_cout),
        .d(CLS_r[0]),
        .sel(InsSel),
        .z(CO)
        );

    mux4x1_8bit mux2 (
        .a(AND_r),
        .b(XOR_r),
        .c(ADD_r),
        .d(CLS_r),
        .sel(InsSel),
        .z(ALU_r)
        );
        assign ALUout = ALU_r;
endmodule
```

RTL Schematic

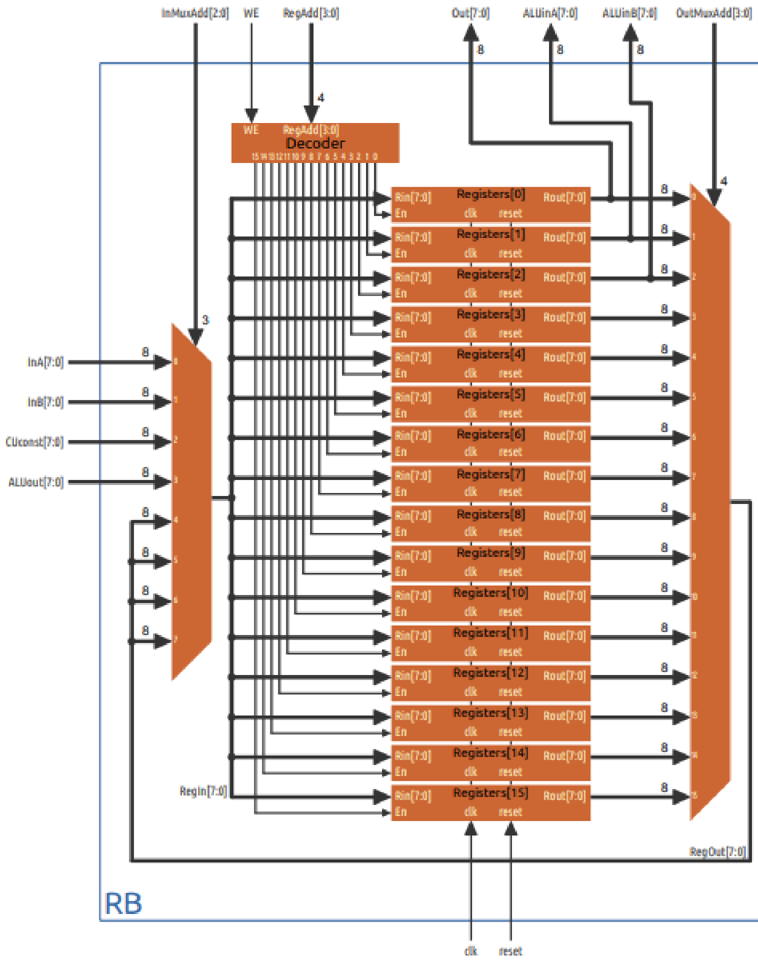
As seen on the RTL diagram below, we need to use the circular shift operator to get the powers of 2 and 3. Addition block is also defined with XOR and AND gates. Then the zero comperator circuit is defined for the undefined condition. where it is undefined , 0^0, will result in undefined case. As requested, the blocks here are structurally designed. In this way, it has been seen that the circuit diagram given with RTL Schematic is similar to each other.



Muhammed Velihan BAĞCI
040170093
Yiğit Bektaş GÜRSOY
040180063

3. Register Block

A register block is a crucial component of a computer's processor, it's a group of registers that are used to temporarily store data and instructions during the execution of a program. These registers are small and fast storage locations built into the processor. The number and size of registers in the block can vary depending on the design of the processor, some have a small number of large registers, while others have a large number of smaller ones. The register block also includes a control unit which manages the transfer of data to and from the registers and the ALU, and controls the flow of the program. Below is a representative design of the RB.



Muhammed Velihan BAĞCI
040170093
Yiğit Bektaş GÜRSOY
040180063

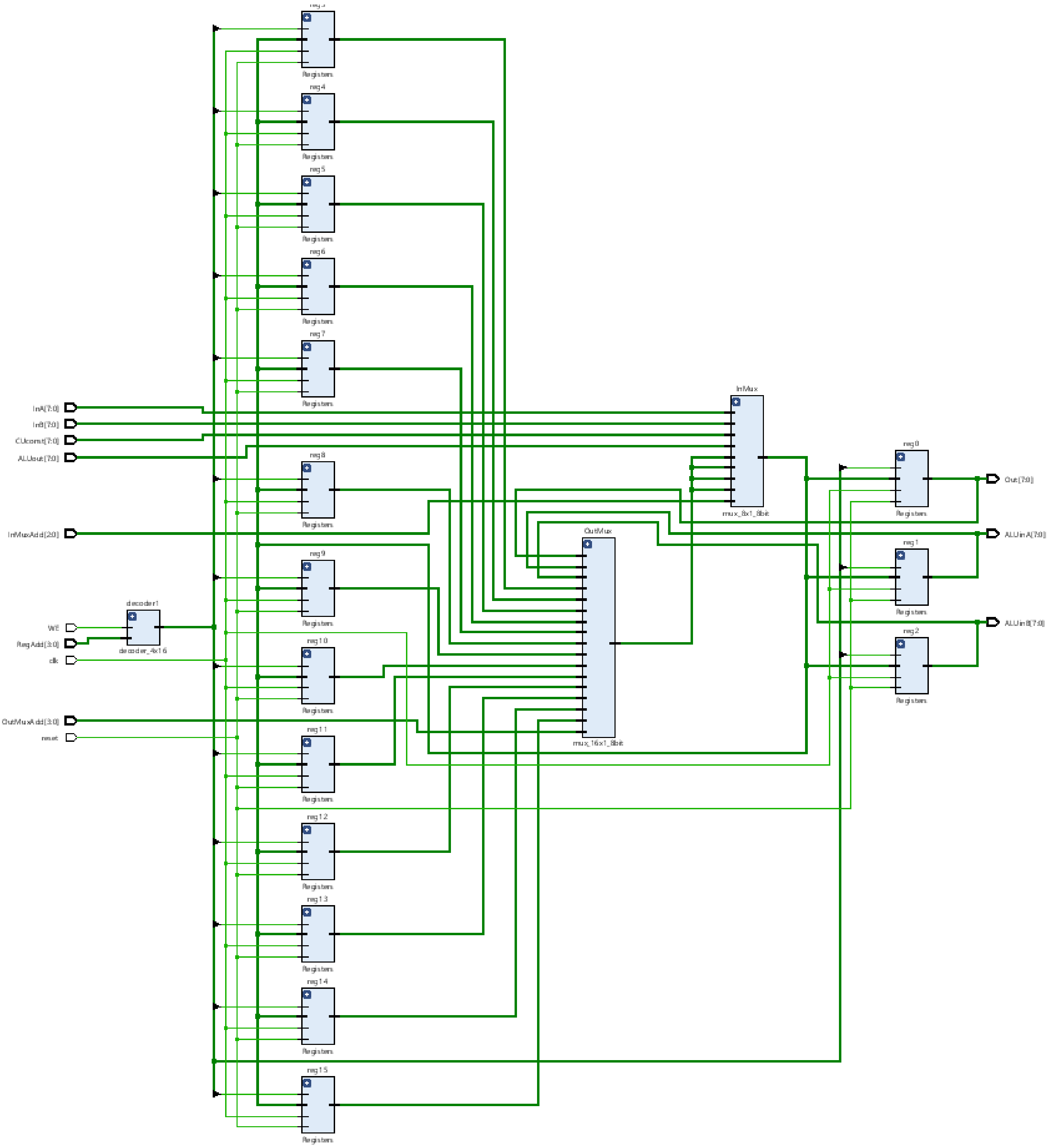
Verilog Code

```
`timescale 1ns / 1ps

module RB(
    input [7:0] InA, InB, CUconst, ALUout,
    input [3:0] RegAdd, OutMuxAdd,
    input [2:0] InMuxAdd,
    input WE, clk, reset,
    output [7:0] Out, ALUinA, ALUinB
    );

    wire [7:0] RegBlocksOut [15:0];
    wire [15:0] DecoderOut;
    wire [7:0] RegBlocksIn, RegOut;

    decoder_4x16 decoder1 (.sel(RegAdd), .WE(WE), .out(DecoderOut));
    mux_8x1_8bit InMux (.a0(InA), .a1(InB), .a2(CUconst), .a3(ALUout), .a4(RegOut), .a5(RegOut), .a6(RegOut), .a7(RegOut),
.sel(InMuxAdd), .out(RegBlocksIn));
    Registers reg0 (.En(DecoderOut[0]), .clk(clk), .reset(reset), .Rin(RegBlocksIn), .Rout(RegBlocksOut[0][7:0]));
    Registers reg1 (.En(DecoderOut[1]), .clk(clk), .reset(reset), .Rin(RegBlocksIn), .Rout(RegBlocksOut[1][7:0]));
    Registers reg2 (.En(DecoderOut[2]), .clk(clk), .reset(reset), .Rin(RegBlocksIn), .Rout(RegBlocksOut[2][7:0]));
    Registers reg3 (.En(DecoderOut[3]), .clk(clk), .reset(reset), .Rin(RegBlocksIn), .Rout(RegBlocksOut[3][7:0]));
    Registers reg4 (.En(DecoderOut[4]), .clk(clk), .reset(reset), .Rin(RegBlocksIn), .Rout(RegBlocksOut[4][7:0]));
    Registers reg5 (.En(DecoderOut[5]), .clk(clk), .reset(reset), .Rin(RegBlocksIn), .Rout(RegBlocksOut[5][7:0]));
    Registers reg6 (.En(DecoderOut[6]), .clk(clk), .reset(reset), .Rin(RegBlocksIn), .Rout(RegBlocksOut[6][7:0]));
    Registers reg7 (.En(DecoderOut[7]), .clk(clk), .reset(reset), .Rin(RegBlocksIn), .Rout(RegBlocksOut[7][7:0]));
    Registers reg8 (.En(DecoderOut[8]), .clk(clk), .reset(reset), .Rin(RegBlocksIn), .Rout(RegBlocksOut[8][7:0]));
    Registers reg9 (.En(DecoderOut[9]), .clk(clk), .reset(reset), .Rin(RegBlocksIn), .Rout(RegBlocksOut[9][7:0]));
    Registers reg10 (.En(DecoderOut[10]), .clk(clk), .reset(reset), .Rin(RegBlocksIn), .Rout(RegBlocksOut[10][7:0]));
    Registers reg11 (.En(DecoderOut[11]), .clk(clk), .reset(reset), .Rin(RegBlocksIn), .Rout(RegBlocksOut[11][7:0]));
    Registers reg12 (.En(DecoderOut[12]), .clk(clk), .reset(reset), .Rin(RegBlocksIn), .Rout(RegBlocksOut[12][7:0]));
    Registers reg13 (.En(DecoderOut[13]), .clk(clk), .reset(reset), .Rin(RegBlocksIn), .Rout(RegBlocksOut[13][7:0]));
    Registers reg14 (.En(DecoderOut[14]), .clk(clk), .reset(reset), .Rin(RegBlocksIn), .Rout(RegBlocksOut[14][7:0]));
    Registers reg15 (.En(DecoderOut[15]), .clk(clk), .reset(reset), .Rin(RegBlocksIn), .Rout(RegBlocksOut[15][7:0]));

    assign Out = RegBlocksOut[0][7:0];
    assign ALUinA = RegBlocksOut[1][7:0];
    assign ALUinB = RegBlocksOut[2][7:0];

    mux_16x1_8bit OutMux(
        .a0(RegBlocksOut[0][7:0]),
        .a1(RegBlocksOut[1][7:0]),
        .a2(RegBlocksOut[2][7:0]),
        .a3(RegBlocksOut[3][7:0]),
        .a4(RegBlocksOut[4][7:0]),
        .a5(RegBlocksOut[5][7:0]),
        .a6(RegBlocksOut[6][7:0]),
        .a7(RegBlocksOut[7][7:0]),
        .a8(RegBlocksOut[8][7:0]),
        .a9(RegBlocksOut[9][7:0]),
        .a10(RegBlocksOut[10][7:0]),
        .a11(RegBlocksOut[11][7:0]),
        .a12(RegBlocksOut[12][7:0]),
        .a13(RegBlocksOut[13][7:0]),
        .a14(RegBlocksOut[14][7:0]),
        .a15(RegBlocksOut[15][7:0]),
        .sel(OutMuxAdd),
        .out(RegOut)
        );
endmodule
```

Muhammed Velihan BAĞCI
040170093
Yiğit Bektaş GÜRSOY
040180063

RTL Schematic

As requested, the blocks here are structurally designed. In this way, it has been seen that the circuit diagram given with RTL Schematic is similar to each other.

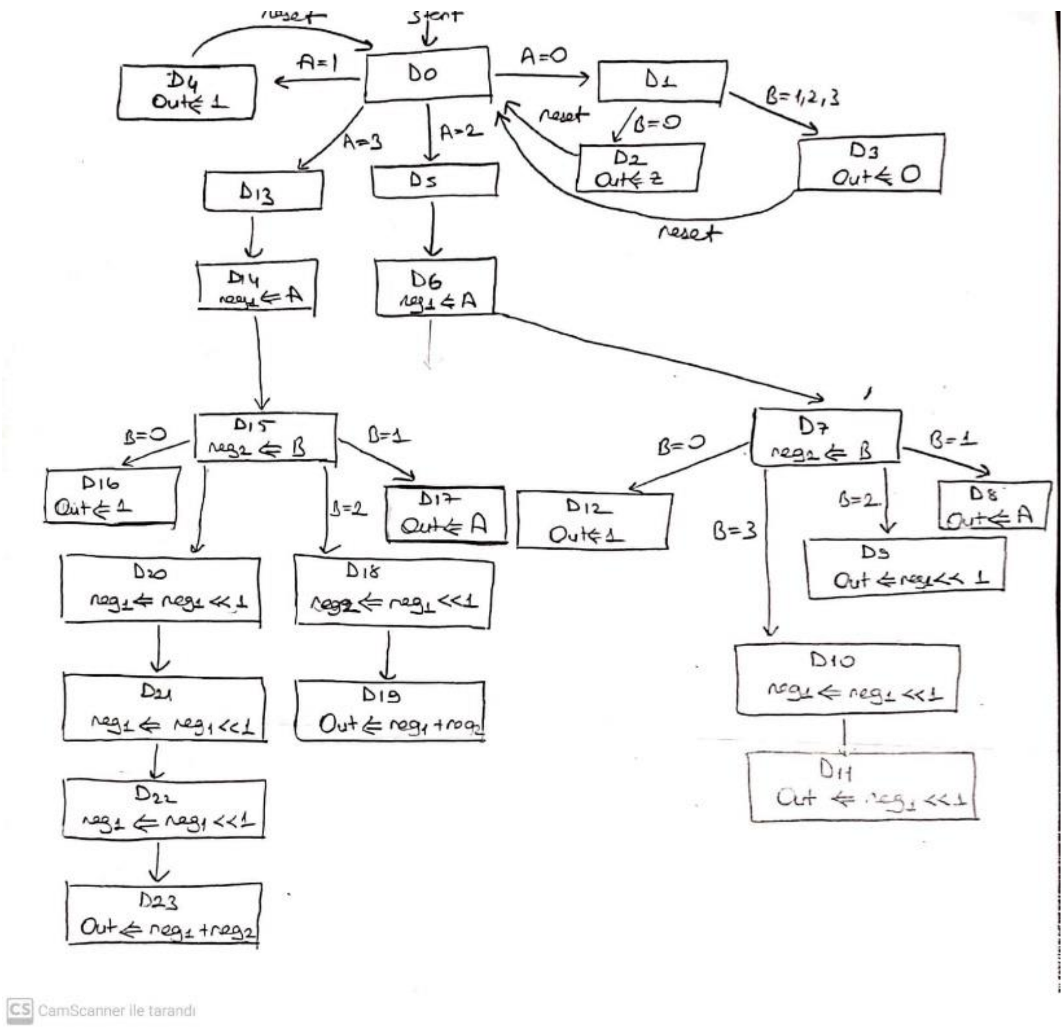


Muhammed Velihan BAĞCI
040170093
Yiğit Bektaş GÜRSOY
040180063

4. Control Unit

The control unit (CU) in a computer's processor manages the flow of data and instructions. It retrieves instructions from memory, decodes them to determine the required operation, and then sends control signals to other components such as the ALU and register block to execute the instruction. The CU also updates the program counter to manage the program flow and monitors the processor's status through flags generated by the ALU to make decisions. In summary, the control unit is responsible for coordinating the execution of instructions by decoding them, sending control signals to other components, managing the program flow, and monitoring the processor's status

ASM Diagram

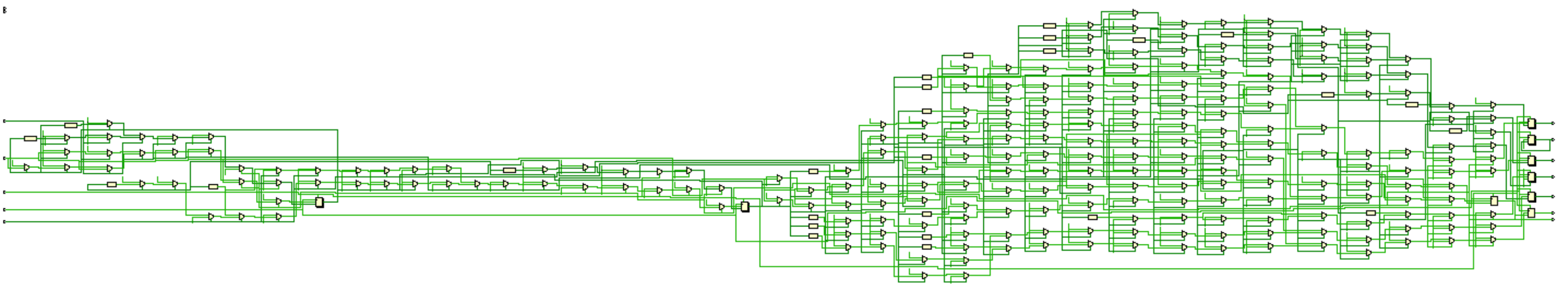


Verilog Code

```
`timescale 1ns / 1ps

module CU(
    input Start, clk, reset, CO, Z,
    input [7:0] InA, InB,
    output reg Busy, WE,
    output reg [7:0] CUconst,
    output reg [3:0] OutMuxAdd, RegAdd,
    output reg [2:0] InMuxAdd,
    output reg [1:0] InsSel
    );

    reg [3:0]durum;
    reg [3:0]durum2;

    always@(posedge clk or posedge reset) begin
        if (reset) begin
            InsSel <= 0;
            WE <= 0;
            Busy <= 0;
            CUconst <= 0;
            OutMuxAdd <= 0;
            RegAdd <= 0;
            InMuxAdd <=0;
            durum <= 0;
            durum2 <= 0;
        end
        else begin
            if (Start) begin
               Busy <=1'b1;
               durum <= 0;
               durum2 <= 0;
            end
            if (Busy == 0) begin
                durum <= 0;
                durum2 <= 0;
            end
            else if (Busy) begin
                if (InA == 8'd0)begin // A==0  => sonuç = 0
                    if (InB == 8'd0) begin
                        InMuxAdd <=3'd2;
                        CUconst <= 8'bzzzz_zzzz;
                        RegAdd <= 3'd0;
                        WE <= 1'b1;
                        Busy <=0;
                    end
                    else begin
                        InMuxAdd <=3'd0;
                        RegAdd <= 3'd0;
                        WE <= 1'b1;
                        Busy <=0;
                    end
                end

                else if (InA == 8'd1)begin // A==1  => sonuç = 1
                    InMuxAdd <=3'd0;
                    RegAdd <= 3'd0;
                    WE <= 1'b1;
                    Busy <=0;
                end
                else if (InA == 8'd2)begin // A==10  => sonuç = A << B
                    if (durum == 4'd0) begin // reg1 <= A
                        InMuxAdd <=3'd0;
                        RegAdd <= 3'd1;
                        WE <= 1'b1;
                        durum <= 4'd1;
                    end
                    if (durum == 4'd1) begin // reg2 <= B
                        InMuxAdd <=3'd1;
                        RegAdd <= 3'd2;
                        WE <= 1'b1;
                        durum <= 4'd2;
                    end
                    else if (durum == 4'd2) begin // sonuc = A << B
                        if(InB == 8'd0) begin // B==0 => sonuc = 1
                            InMuxAdd <=3'd2;
                            RegAdd <= 3'd0;
                            WE <= 1'b1;
                            CUconst <= 8'd1;
                            Busy <= 0;
                        end
                        else if(InB == 8'd1) begin // B==1 => sonuc = 10
                            InMuxAdd <=3'd0;
                            RegAdd <= 3'd0;
                            WE <= 1'b1;
                            Busy <= 0;
                        end
                        else if(InB == 8'd2) begin // B==10 => sonuc = 100
                            InsSel <= 2'd3;
                            InMuxAdd <= 3'd3;
                            RegAdd <= 3'd0;
                            WE <= 1'b1;
                            Busy <= 0;
                        end
```

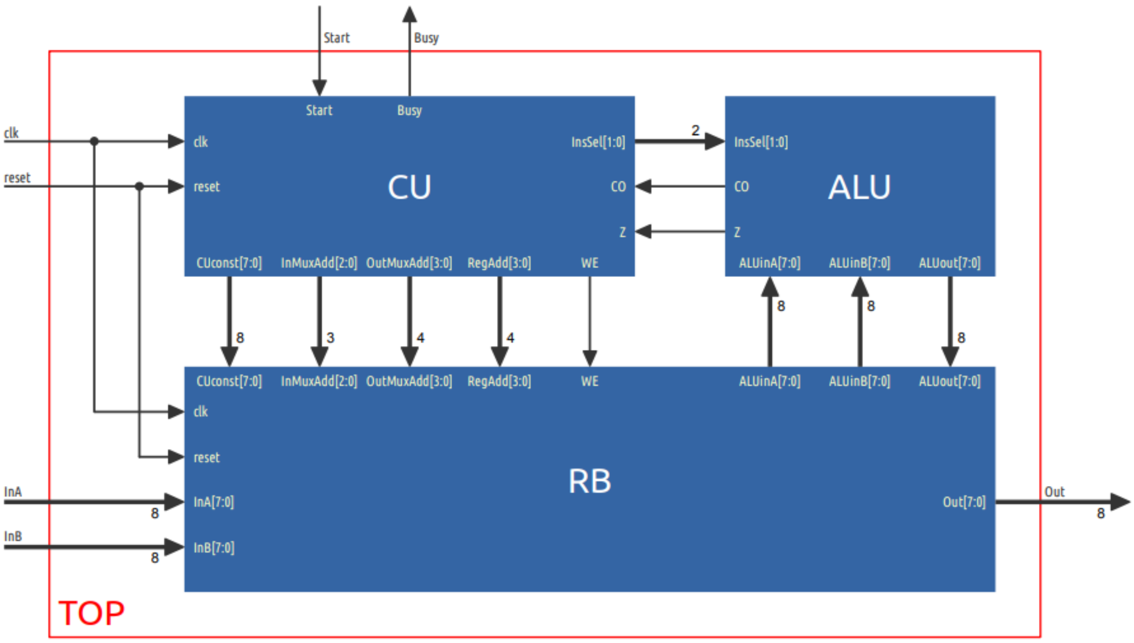
```
                else if(InB == 8'd3) begin // B==11 => sonuc = 1000
                    if (durum2 == 4'b0) begin //reg1 <=100
                        InsSel <= 2'd3;
                        InMuxAdd <= 3'd3;
                        RegAdd <= 3'd1;
                        WE <= 1'b1;
                        durum2 = 4'b1;
                    end
                    else if (durum2 == 4'b1) begin //sonuc = reg1 << 1;
                        InsSel <= 2'd3;
                        InMuxAdd <= 3'd3;
                        RegAdd <= 3'd0;
                        WE <= 1'b1;
                        Busy <= 0;
                    end
                end
            end
        end
        else if (InA == 8'd3)begin // A == 11
            if (durum == 4'd0) begin // reg1 <= A
                InMuxAdd <=3'd0;
                RegAdd <= 3'd1;
                WE <= 1'b1;
                durum <= 4'd1;
            end
            else if (durum == 4'd1) begin // reg2 <= B
                InMuxAdd <=3'd1;
                RegAdd <= 3'd2;
                WE <= 1'b1;
                durum <= 4'd2;
            end
            else if (durum == 4'd2) begin // sonuc = A << B
                if(InB == 8'd0) begin // B==0 => sonuc = 1
                    InMuxAdd <=3'd2;
                    RegAdd <= 3'd0;
                    WE <= 1'b1;
                    CUconst <= 8'd1;
                    Busy <= 0;
                end
                else if(InB == 8'd1) begin // B==1 => sonuc = 11
                    InMuxAdd <=3'd0;
                    RegAdd <= 3'd0;
                    WE <= 1'b1;
                    Busy <= 0;
                end
                else if(InB == 8'd2) begin // B==10 => sonuc = 1001
                    if (durum2 == 4'd0) begin //reg2 <=110
                        InsSel <= 2'd3;
                        InMuxAdd <= 3'd3;
                        RegAdd <= 3'd2;
                        WE <= 1'b1;
                        durum2 = 4'b1;
                    end
                    else if (durum2 == 4'd1) begin //sonuc = reg1 + reg2
                        InsSel <= 2'd2;
                        InMuxAdd <= 3'd3;
                        RegAdd <= 3'd0;
                        WE <= 1'b1;
                        Busy <= 0;
                    end
                end
                else if(InB == 8'd3) begin // B==11 => sonuc = 11011
                    if (durum2 == 4'b0) begin //reg1 <=110
                        InsSel <= 2'd3;
                        InMuxAdd <= 3'd3;
                        RegAdd <= 3'd1;
                        WE <= 1'b1;
                        durum2 = 4'b1;
                    end
                    else if (durum2 == 4'b1) begin //reg1 <=1100
                        InsSel <= 2'd3;
                        InMuxAdd <= 3'd3;
                        RegAdd <= 3'd1;
                        WE <= 1'b1;
                        durum2 = 4'd2;
                    end
                    else if (durum2 == 4'd2) begin //reg1 <=11000
                        InsSel <= 2'd3;
                        InMuxAdd <= 3'd3;
                        RegAdd <= 3'd1;
                        WE <= 1'b1;
                        durum2 = 4'd3;
                    end
                    else if (durum2 == 4'd3) begin //sonuc = reg1+reg2
                        InsSel <= 2'd2;
                        InMuxAdd <= 3'd3;
                        RegAdd <= 3'd0;
                        WE <= 1'b1;
                        Busy <=0;
                    end
                end
            end
        end
    end
end
endmodule
```

RTL Schematic



5. Top Module

All the blocks described above are connected to this module. As a result of the connected blocks, it has been confirmed by the simulation results that our circuit works correctly. More detailed information is given with the following results.

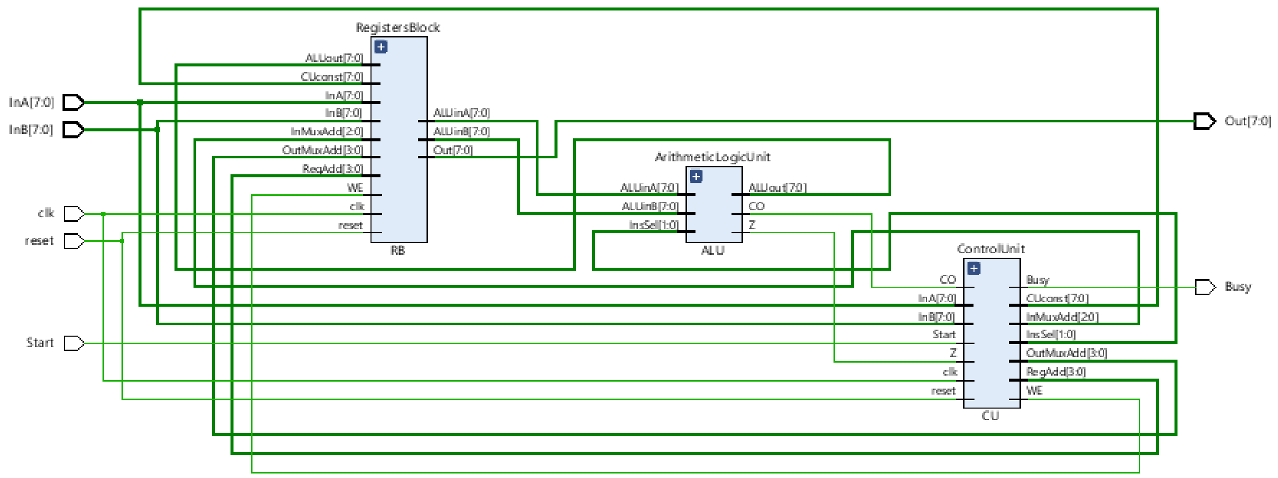


Verilog Code

```
`timescale 1ns / 1ps

module TOP(
    input clk, reset, Start,
    input [7:0] InA, InB,
    output Busy,
    output [7:0]Out
    );

    wire we, co, z;
    wire [1:0] inssel;
    wire [2:0] inmuxadd;
    wire [3:0] outmuxadd, regadd;
    wire [7:0] cuconst, aluina, aluinb, aluout;

    ALU ArithmeticLogicUnit (
        .InsSel(inssel),
        .ALUinA(aluina),
        .ALUinB(aluinb),
        .Z(z),
        .CO(co),
        .ALUout(aluout)
        );

    RB RegistersBlock (
        .clk(clk),
        .reset(reset),
        .InA(InA),
        .InB(InB),
        .WE(we),
        .CUconst(cuconst),
        .InMuxAdd(inmuxadd),
        .OutMuxAdd(outmuxadd),
        .RegAdd(regadd),
        .ALUinA(aluina),
        .ALUinB(aluinb),
        .ALUout(aluout),
        .Out(Out)
        );

     CU ControlUnit(
        .clk(clk),
        .reset(reset),
        .Start(Start),
        .InA(InA),
        .InB(InB),
        .CO(co),
        .Z(z),
        .Busy(Busy),
        .CUconst(cuconst),
        .InMuxAdd(inmuxadd),
        .OutMuxAdd(outmuxadd),
        .RegAdd(regadd),
        .WE(we),
        .InsSel(inssel)
        );

endmodule
```

RTL Schematic

As requested, the blocks here are structurally designed. In this way, it has been seen that the circuit diagram given with RTL Schematic is similar to each other.

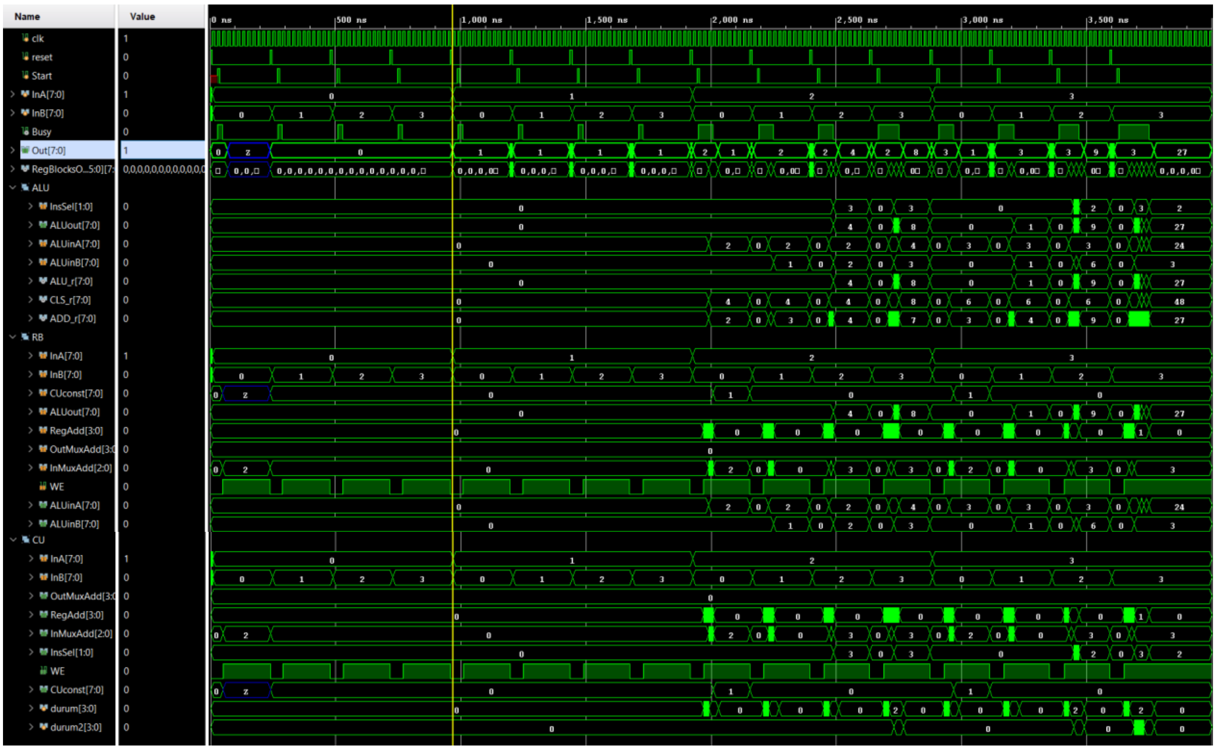


Test Bench Code

```
`timescale 1ns / 1ps

module TOP_tb( );
    reg clk, reset, Start;
    reg [7:0] InA, InB;
    wire Busy;
    wire [7:0]Out;

    TOP DUT (
        .clk(clk),
        .reset(reset),
        .Start(Start),
        .InA(InA),
        .InB(InB),
        .Busy(Busy),
        .Out(Out)
        );

    always
    begin
        clk = ~clk; #10;
    end

    initial begin
        clk = 0; reset = 1; #10; reset = 0;
        InA = 8'd0; InB = 8'd0; #19;
        Start = 1; #10 Start = 0; #200;

        reset = 1; #10 reset = 0;
        InA = 8'd0; InB = 8'd1; #20;
        Start = 1; #10Start = 0; #200;

        reset = 1; #10; reset = 0;
        InA = 8'd0; InB = 8'd2; #19;
        Start = 1; #10 Start = 0; #200;

        reset = 1; #10 reset = 0;
        InA = 8'd0; InB = 8'd3; #20;
        Start = 1; #10Start = 0; #200;

        reset = 1; #10 reset = 0;
        InA = 8'd1; InB = 8'd0; #19;
        Start = 1; #10 Start = 0; #200;

        reset = 1; #10 reset = 0;
        InA = 8'd1; InB = 8'd1; #20;
        Start = 1; #10 Start = 0; #200

        reset = 1; #10; reset = 0;
        InA = 8'd1; InB = 8'd2; #19;
        Start = 1; #10 Start = 0; #200

        reset = 1; #10 reset = 0;
        InA = 8'd1; InB = 8'd3; #20;
        Start = 1; #10 Start = 0; #200

        reset = 1; #10 reset = 0;
        InA = 8'd2; InB = 8'd0; #19;
        Start = 1; #10 Start = 0; #200

        reset = 1; #10 reset = 0;
        InA = 8'd2; InB = 8'd1; #20;
        Start = 1; #10 Start = 0; #200

        reset = 1; #10; reset = 0;
        InA = 8'd2; InB = 8'd2; #19;
        Start = 1; #10 Start = 0; #200

        reset = 1; #10 reset = 0;
        InA = 8'd2; InB = 8'd3; #20;
        Start = 1; #10Start = 0; #200


        reset = 1; #10 reset = 0;
        InA = 8'd3; InB = 8'd0; #19;
        Start = 1; #10 Start = 0; #200

        reset = 1; #10 reset = 0;
        InA = 8'd3; InB = 8'd1; #20;
        Start = 1; #10 Start = 0; #200

        reset = 1; #10; reset = 0;
        InA = 8'd3; InB = 8'd2; #19;
        Start = 1; #10 Start = 0; #200

        reset = 1; #10 reset = 0;
        InA = 8'd3; InB = 8'd3; #20;
        Start = 1; #10Start = 0; #200;
    end
endmodule
```

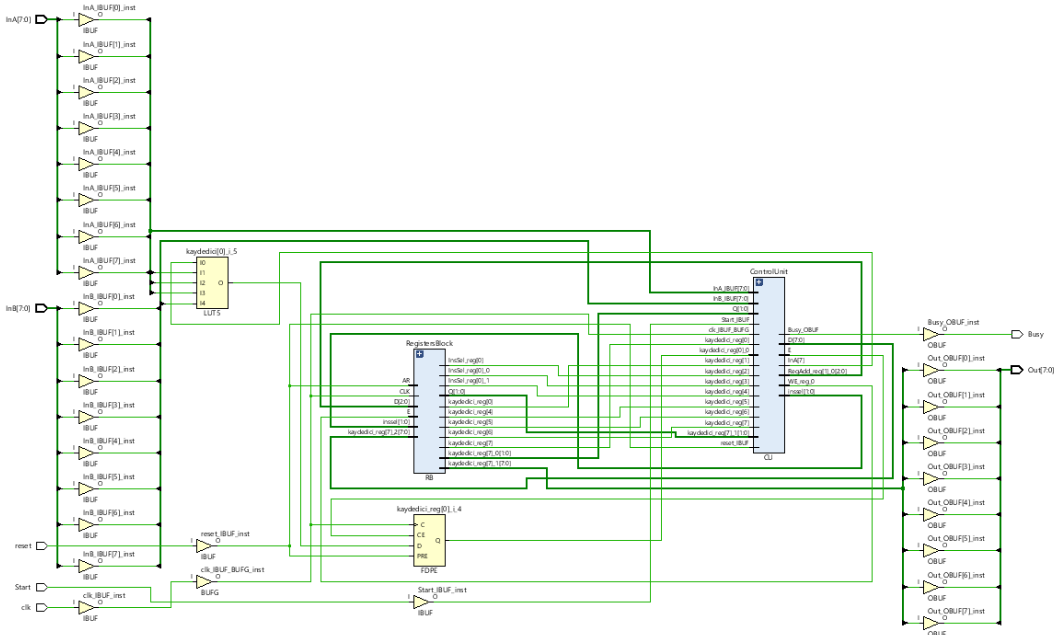
Behavioral Simulation Results

Below are the simulation results of the Top Module. When Reset = 1, all Register Blocks and Control Unit states are reset. In cases where Start = 1, the Busy signal is activated. As long as this signal is active, Control Unit, ALU and Register Block control the situation by transmitting the necessary signals in line with the given inputs. Our outputs appear in the OUT signal. State 0^0 is entered as undefined. All other states are transferred to OUT after 1 clock cycle from the negedge part of the BUSY signal. As seen in the simulation, 16 cases are also specified. The minimum number that can come as a result of exponential numbers is 0 and the maximum number is 27.



6. Circuit Analysis Summary (Top Module)

Technology Schematic



Setup Delays

Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	De
Path 10	∞	4	4	13	InB[6]	ControlUnit/...dd_reg[1]/CE	5.323	1.556	3.767	∞	input port clock	
Path 8	∞	4	4	13	InB[6]	ControlUnit/...dd_reg[0]/CE	5.362	1.556	3.806	∞	input port clock	
Path 9	∞	4	4	13	InB[6]	ControlUnit/WE_reg/CE	5.362	1.556	3.806	∞	input port clock	
Path 7	∞	5	5	3	RegistersBlo...ici_reg[1]/C	RegistersBlo...ici_reg[5]/D	5.400	1.210	4.190	∞		
Path 5	∞	4	4	13	InB[6]	ControlUnit/...dd_reg[0]/CE	5.466	1.556	3.910	∞	input port clock	
Path 6	∞	4	4	13	InB[6]	ControlUnit/...dd_reg[1]/CE	5.466	1.556	3.910	∞	input port clock	
Path 4	∞	5	4	7	InA[3]	ControlUnit/...m2_reg[1]/D	5.475	1.691	3.785	∞	input port clock	
Path 3	∞	5	4	13	InB[6]	ControlUnit/Busy_reg/D	5.496	1.450	4.046	∞	input port clock	
Path 2	∞	5	4	7	InA[3]	ControlUnit/I...Sel_reg[0]/D	5.718	1.715	4.004	∞	input port clock	
Path 1	∞	5	4	7	InA[3]	ControlUnit/I...Sel_reg[1]/D	5.854	1.743	4.112	∞	input port clock	

Hold Delays

Unconstrained Paths - NONE - NONE - Hold

Name	Slack	Levels	Routes	High Fanout	From	To	Total ...	Logic Delay	Net Delay	Requirement	Sourc
Path 11	∞	2	1	18	ControlUnit/...rum_reg[1]/C	ControlUnit/...rum_reg[0]/D	0.334	0.186	0.148	-∞	
Path 12	∞	2	1	8	ControlUnit/...m2_reg[0]/C	ControlUnit/I...Sel_reg[0]/D	0.334	0.186	0.148	-∞	
Path 13	∞	1	1	8	RegistersBlo...out_reg[2]/G	RegistersBloc...ci_reg[5]/CE	0.335	0.212	0.123	-∞	
Path 14	∞	1	1	8	RegistersBlo...out_reg[2]/G	RegistersBloc...ci_reg[6]/CE	0.335	0.212	0.123	-∞	
Path 15	∞	1	1	8	RegistersBlo...out_reg[2]/G	RegistersBloc...ci_reg[7]/CE	0.335	0.212	0.123	-∞	
Path 16	∞	1	1	9	ControlUnit/Busy_reg/C	ControlUnit/WE_reg/D	0.336	0.164	0.172	-∞	
Path 17	∞	2	1	18	ControlUnit/...rum_reg[1]/C	ControlUnit/...rum_reg[1]/D	0.337	0.186	0.151	-∞	
Path 18	∞	1	1	8	RegistersBlo...out_reg[1]/G	RegistersBloc...ci_reg[7]/CE	0.346	0.212	0.134	-∞	
Path 19	∞	1	1	8	RegistersBlo...out_reg[2]/G	RegistersBloc...ci_reg[1]/CE	0.378	0.212	0.166	-∞	
Path 20	∞	1	1	8	RegistersBlo...out_reg[2]/G	RegistersBloc...ci_reg[2]/CE	0.378	0.212	0.166	-∞	

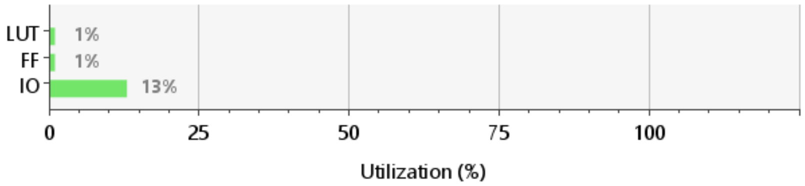
As seen above, the maximum delay of our circuit is 5.854ns. To find the maximum frequency, we find 170.82 Mhz using the equation 1/T = f (hz)

Utilization Summary

As seen in the circuit, 65 LUTs, 41 FF's and 28 IO are used..

Summary

Resource	Utilization	Available	Utilization %
LUT	65	32600	0.20
FF	41	65200	0.06
IO	28	210	13.33



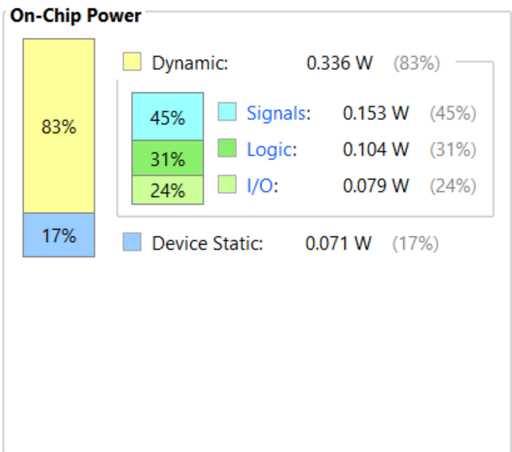
As seen below, the total chip on power value is 0,406W. The individual power consumptions are indicated on the right. Most of the power consumption is Dynamic Power. Although signal and logic consume approximately 55% power, I/O value accounts for all of the power consumption with 45% value.

Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.406 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26,9°C**
Thermal Margin:	58,1°C (12,1 W)
Effective ϑJA:	4,8°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity



7. Work Package

NAME SURNAME	Assignment Research	Verilog Code	ASM Diagram	Report	Vivado Outputs
Yiğit Bektaş GÜRSOY	X	TOP.v ALU.v RB.v CU.v		X	X
Muhammed Velihan BAĞCI	X	TOP.v ALU.v RB.v CU.v	X	X	