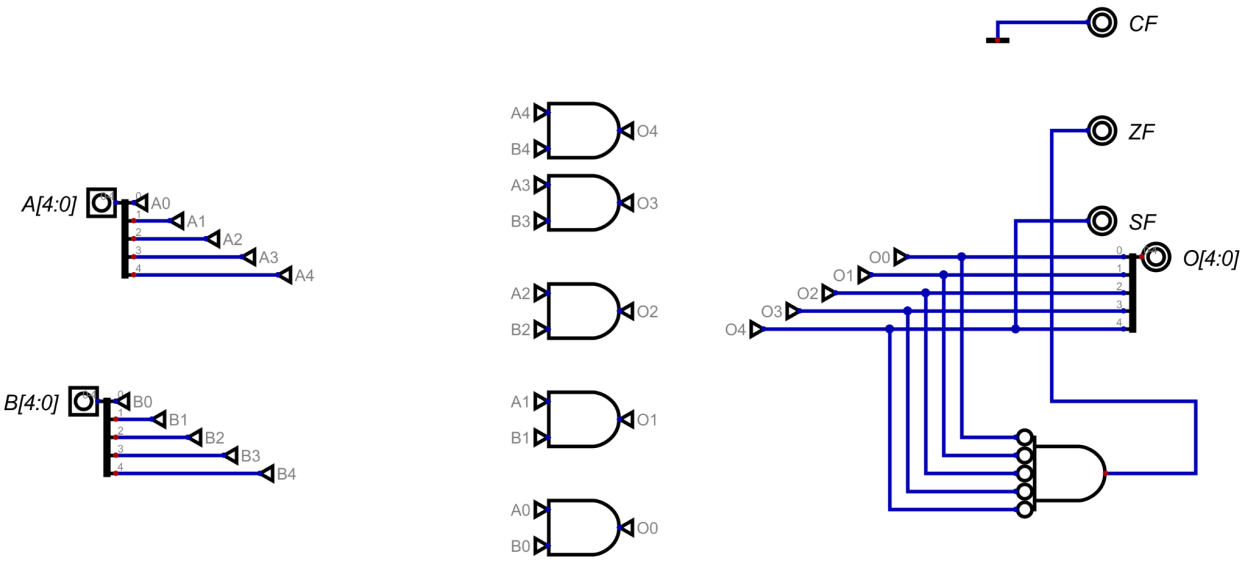
CSE 3203 CT 4 Assignment
Roll No: 1803109
Instruction: Covert this doc to PDF while uploading.

Assignment Problem:
Build CPU based on the following requirements:

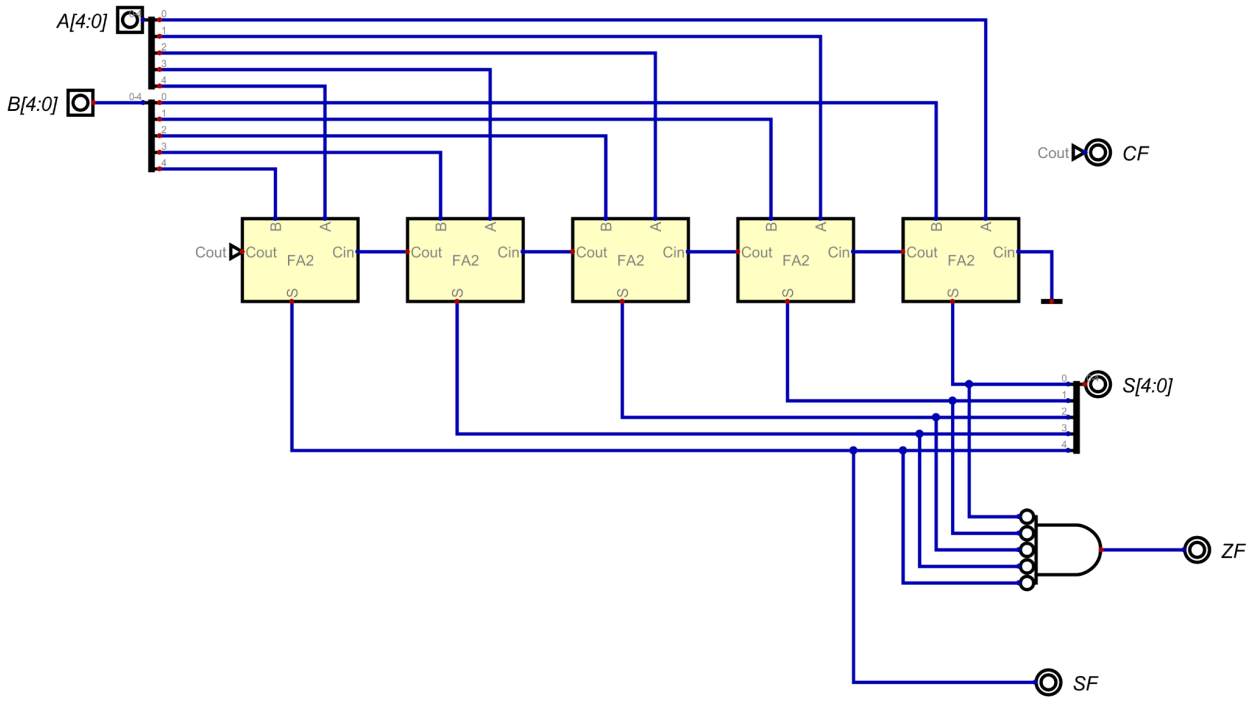
1. **Word Size of CPU = 5**
2. **ALU Operations = AND, ADD, SHR**
3. **Register Number = 4**
4. **Size of RAM = 7**
5. **Word size of ISA and RAM = 17**
6. **CPU Instructions = Register Mode, Immediate Mode, JMP, JG**

Solution:
Simulator Design:

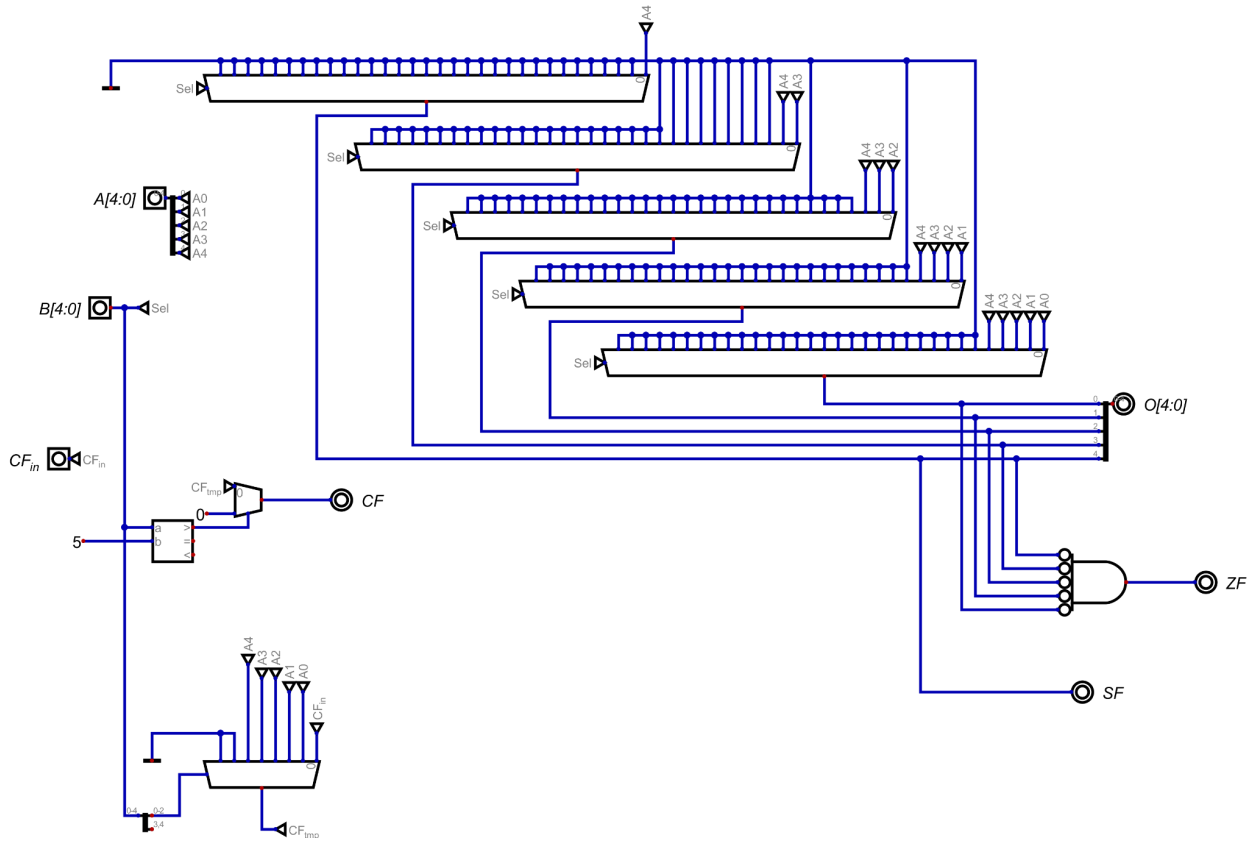
1. **ALU Circuit (Top to Bottom all circuits):**



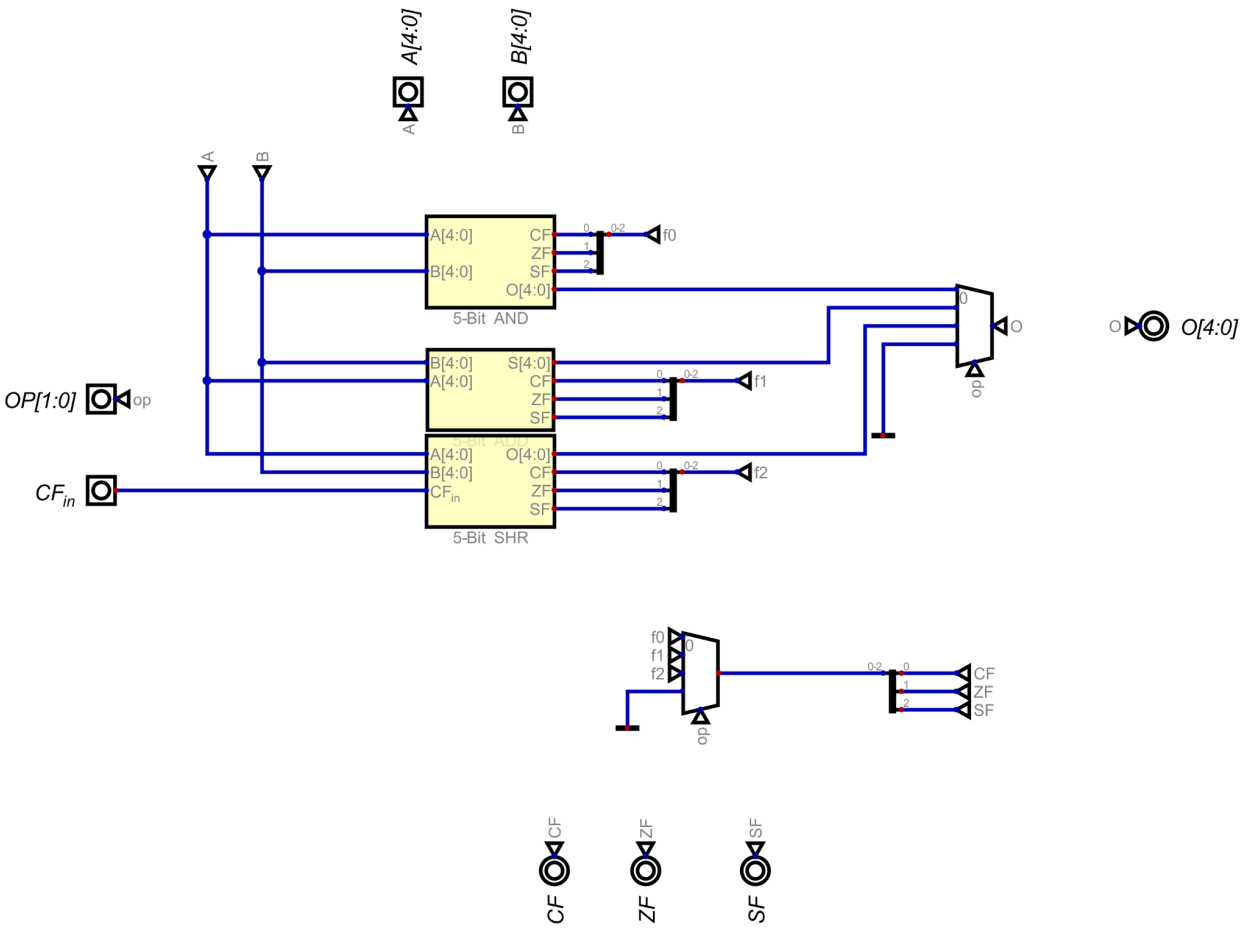
Circuit: 5-Bit AND



Circuit: 5-Bit ADD

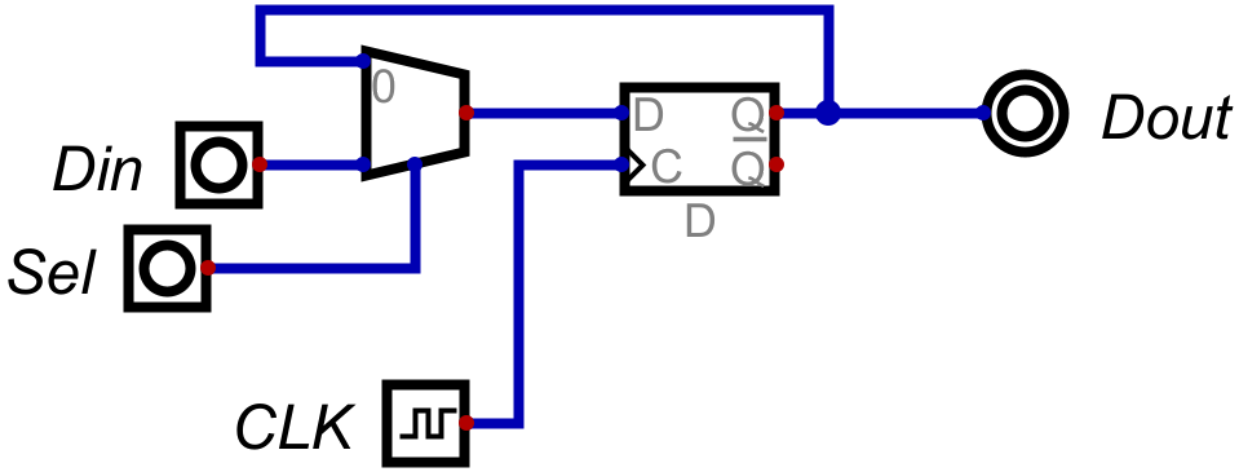


Circuit: 5-Bit SHR

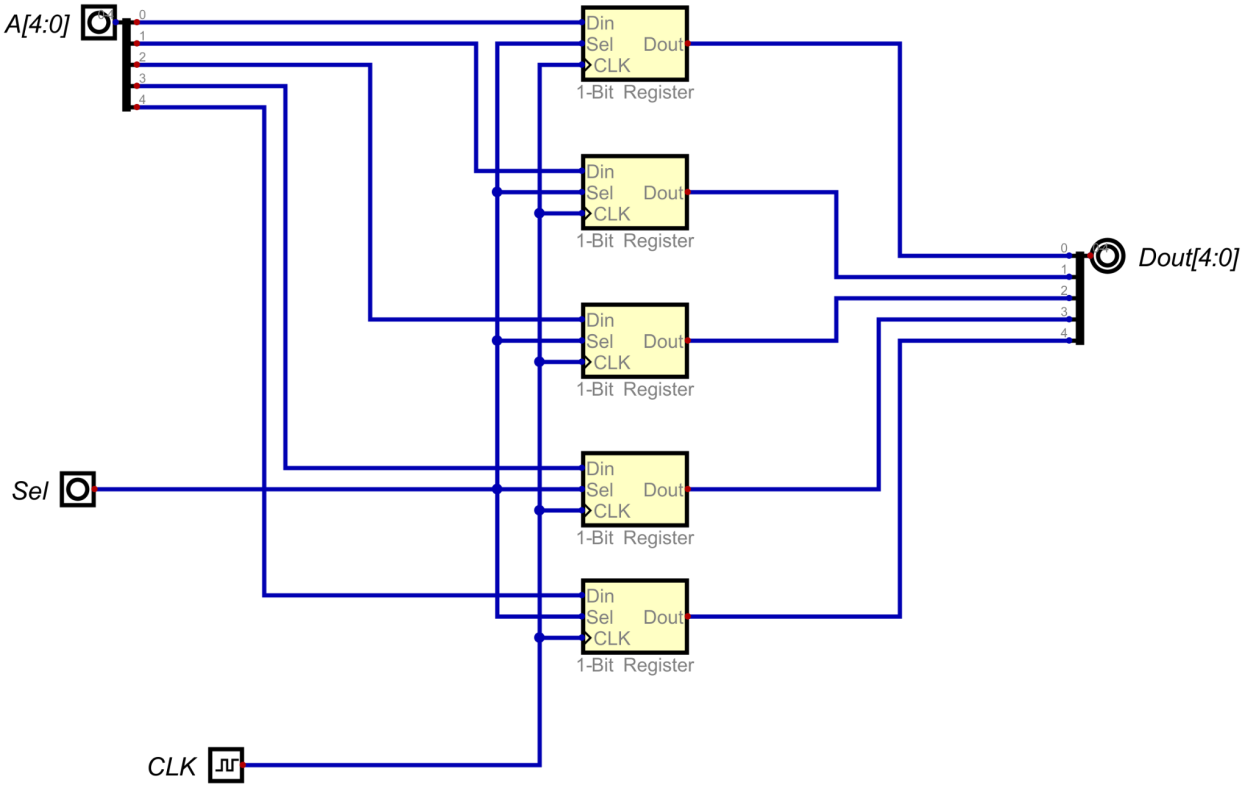


Circuit: 5 Bit ALU

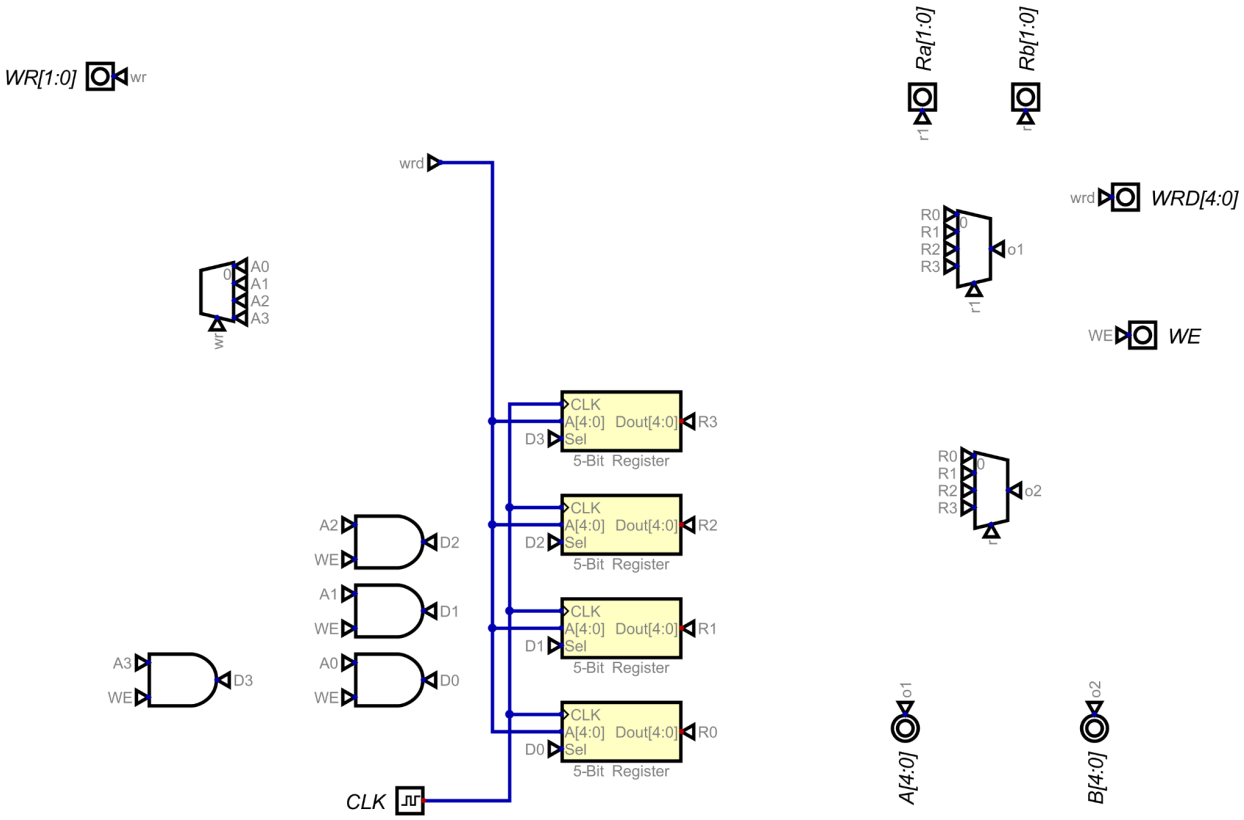
2. **Register Set Circuit (Top to Bottom all circuits):**



Circuit: 1-Bit Register

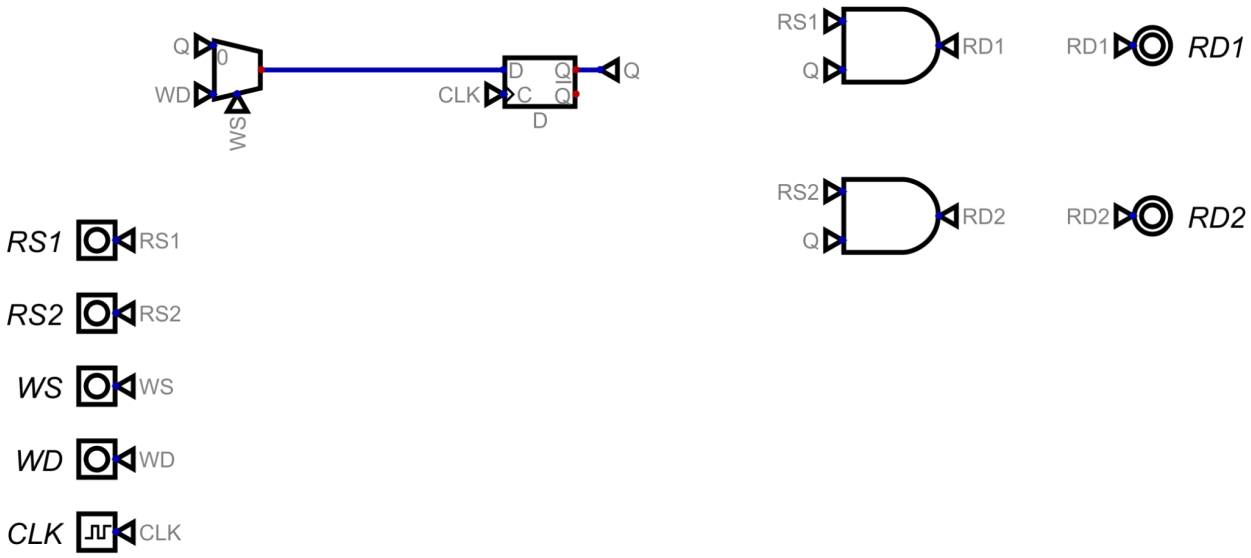


Circuit: 5-Bit Register

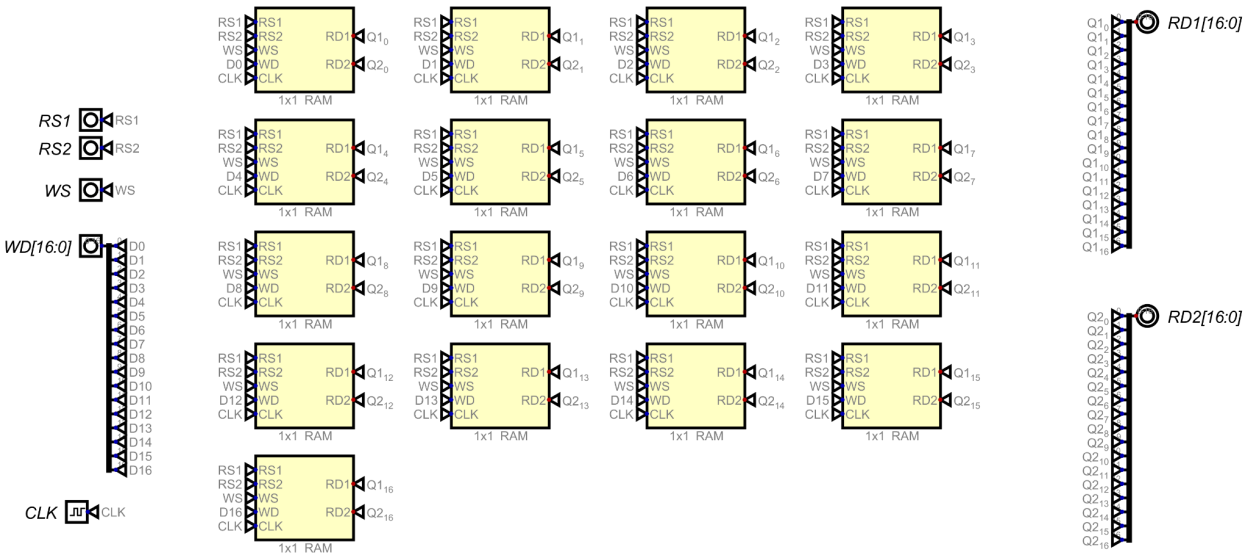


Circuit: 5-Bit Register Set with 4 Register

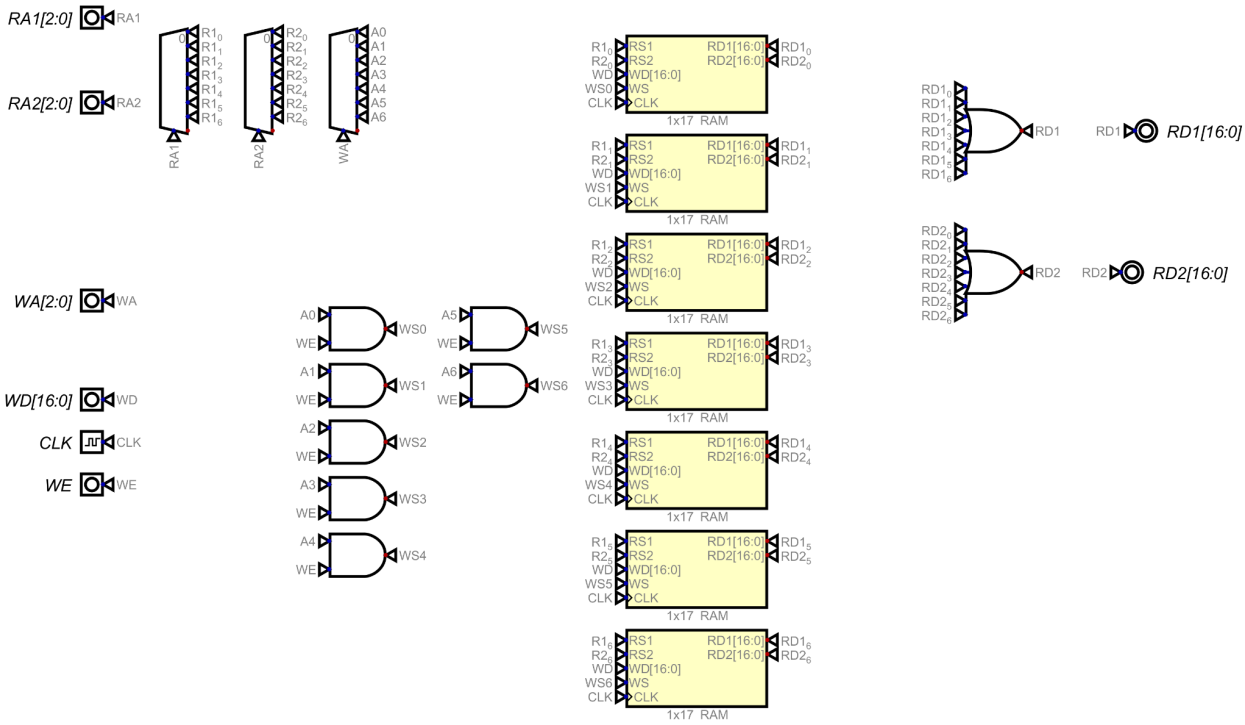
3. RAM Circuit (Top to Bottom all circuits):



Circuit: 1x1 RAM



Circuit: 1x17 RAM



Circuit: 7x17 RAM

4. ISA

17 Bits ISA FOR ASSIGNED CPU

ISA of Register Mode:

Opcode(4 bits)		Register 1	Register 2	Unused
2 bits (16-15)	2bit(14-13)	2 bits(12-11)	2 bits(10-9)	9 bits(8-0)
Types of operation(00)	Operations ALU selection line	Ra (00-11)	Rb (00-11)	X

Opcode(4 bits)		Register 1	Register 2	Unused
2 bits (16-15)	2 bit(14-13)	2 bits	2 bits	7 bits
00	00 (AND)	00-11	00-11	X
01	01 (ADD)	00-11	00-11	X
10	10(SHR)	00-11	00-11	X

ISA of Immediate Mode:

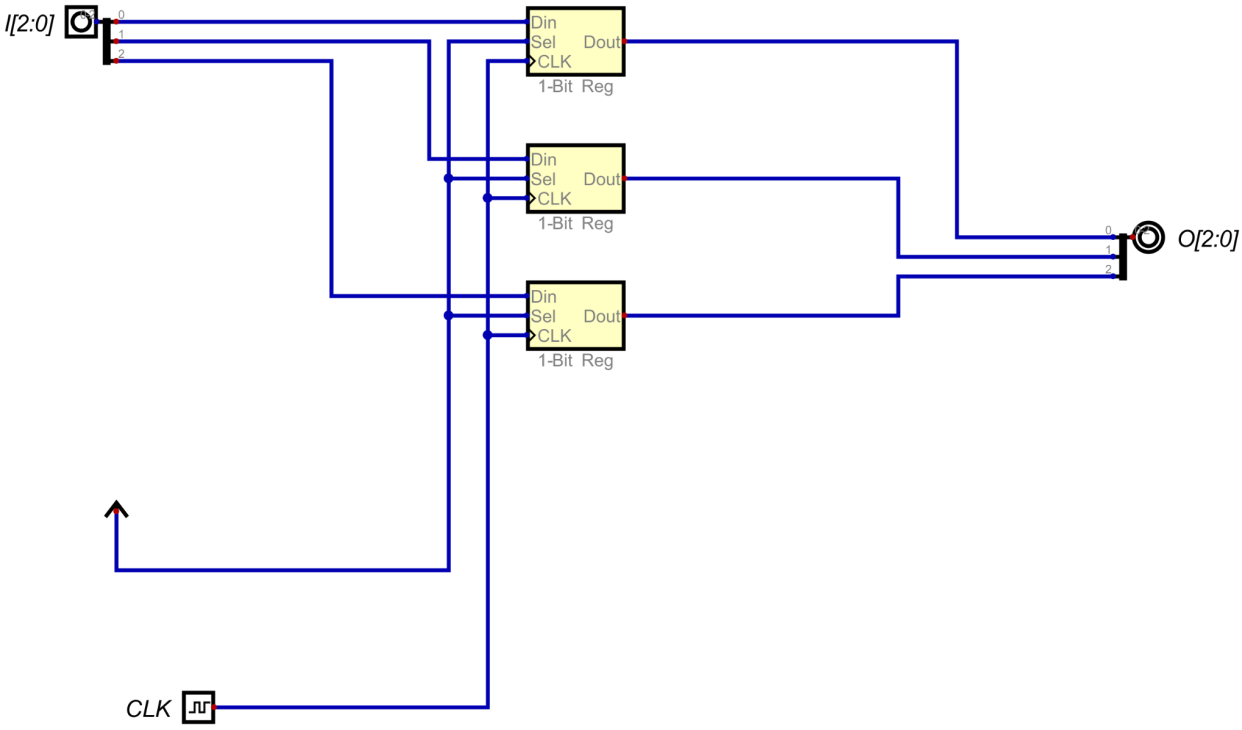
Opcode(4 bits)		Register 1	Constant	Unused
2 bits (16-15)	2 bit(14-13)	2 bits(12-11)	5 bits(10-6)	6 bits(5-0)
Types of operation (01)	Operations ALU selection line	Ra (00-11)	value (00000-11111)	X

Opcode(4 bits)		Register 1	Constant	Unused
2 bits (16-15)	2 bit(14-13)	2 bits	5 bits	6 bits
01	00(AND)	00-11	00000-11111	X
01	01 (ADD)	00-11	00000-11111	X
01	10(SHR)	00-11	00000-11111	X

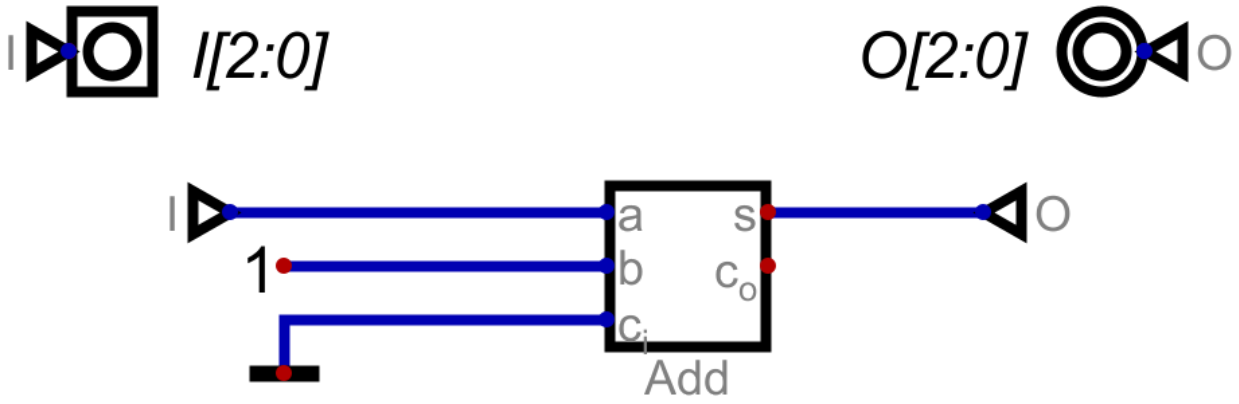
ISA for Branching Mode:

Opcode(4 bits)		Address	Unused
2 bits (15-14)	2 bit(14-13)	3 bits(12-10)	10 bits(9-0)
Types of operation (10)	Operations ALU selection line	000-111 (RAM size=7)	X
10	00 (JMP)	000-111	X
10	01(JG)	000-111	X

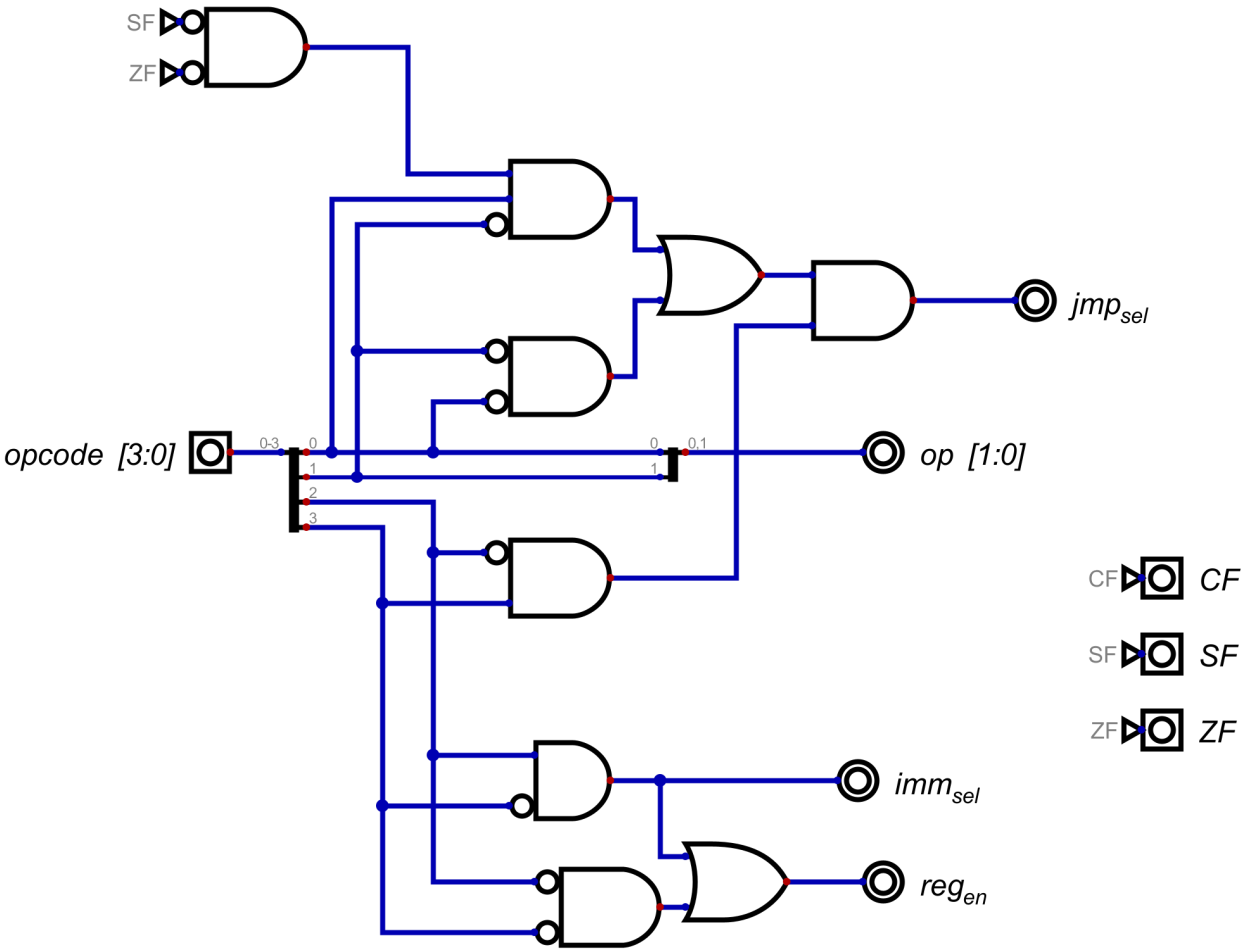
5. CPU (Top to Bottom all circuits):



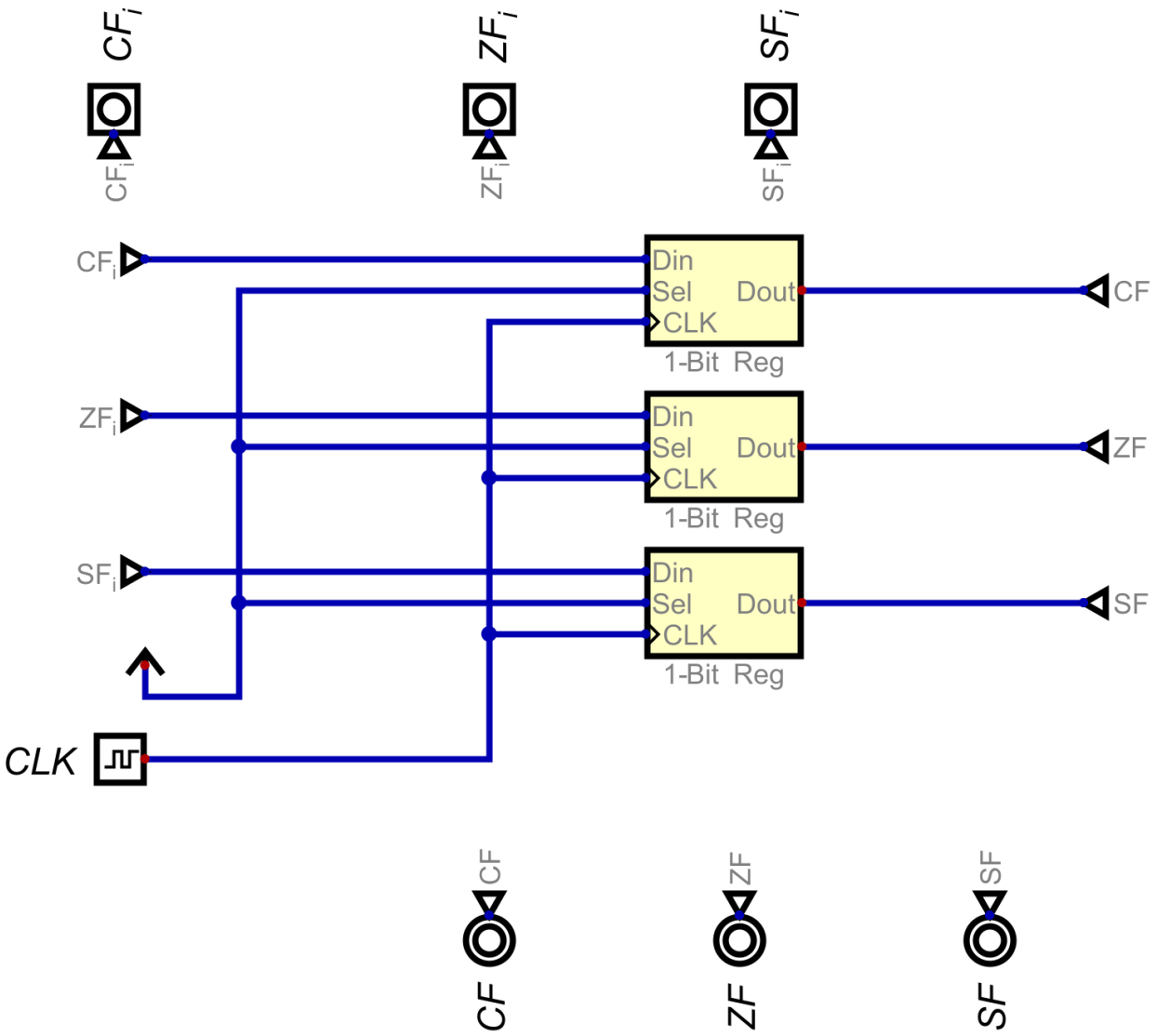
Circuit: 3-Bit Program Register



Circuit: 3_Bit_Pc_Adder



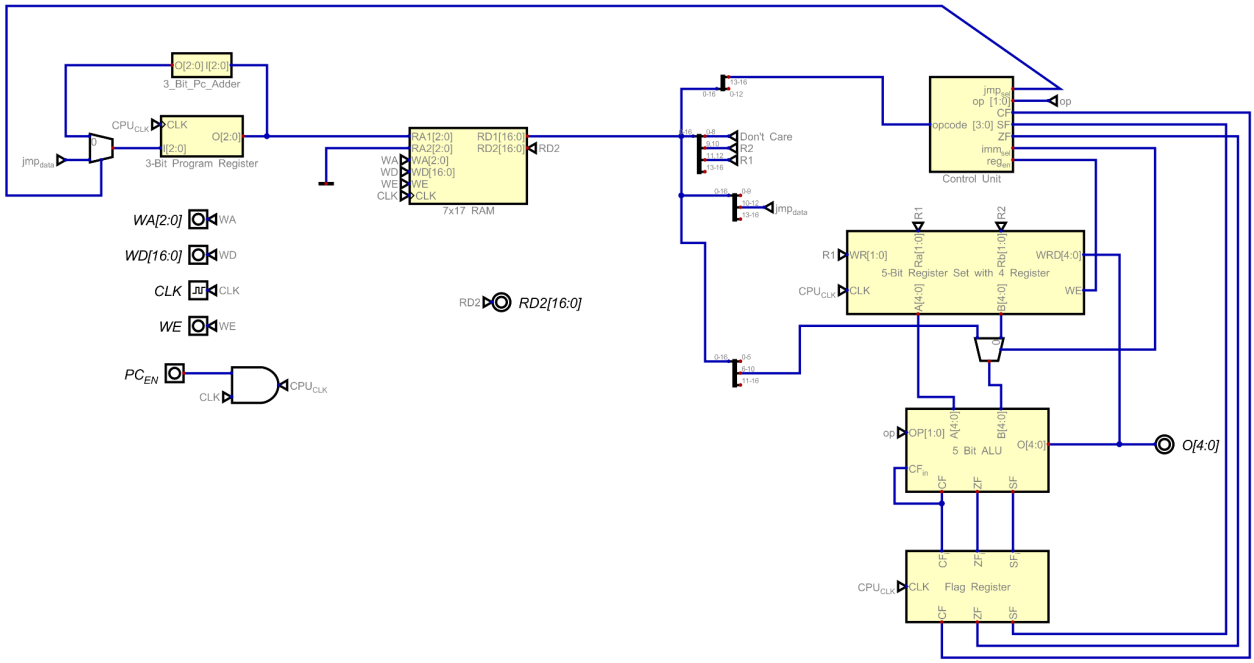
Circuit: Control Unit



Circuit: Flag Register

Register Mode (Type of op=00)=2bit(Type of op)+ 2bit(no of op)+ 2bit(Reg1) + 2bit(Reg2) + 9bit(don't care)
Immediate Mode (Type of op=01)=2bit(Type of op)+ 2bit(no of op)+ 2bit(Reg1) + 5bit(Imm value)+ 6 bit(don't care)
AND(op=00), ADD(op=01), SHR(op=10)
JMP(op=00) JG(op=01)

AND R1,R1 -> 00000101000000000
ADD R1,5 -> 01010100101000000
ADD R2,4 -> 01011000100000000
ADD R1,R2 -> 00010110000000000
JMP 0 -> 10000000000000000
SHR R1,4 -> 01100100100000000
ADD R1,R1-> 00010101000000000



Circuit: 5_Bit_CPU

Verilog Code:

1. ALU Circuit (Top to Bottom all circuits):

and1bit.v

```
module and1bit(
    input a,
    input b,
    output r
);
assign r=a&b;
endmodule
```

and1bit_tb.v

```
`timescale 1ns/1ps
module and1bit_tb;
    reg a;
    reg b;
    wire c;
    and1bit uut(.x(a), .y(b), .z(c));
    initial begin
        a = 0; b = 0; #10;
        a = 0; b = 1; #10;
        a = 1; b = 0; #10;
        a = 1; b = 1; #10;
        $finish;
    end
    initial begin
        $dumpfile("and1bit.vcd");
        $dumpvars(0, and1bit_tb);
        $monitor("a=%b   b=%b    c=%b", a,b,c);
    end
endmodule
```

and5bit.v

```
module and5bit(
    input wire [4:0] a,
    input wire [4:0] b,

    output wire [4:0] r,
    output wire cf,
    output wire zf,
    output wire sf
);
and1bit uut0(
    .a(a[0]),
    .b(b[0]),
    .r(r[0])
    );
```

```
and1bit uut1(
    .a(a[1]),
    .b(b[1]),
    .r(r[1])
    );
and1bit uut2(
    .a(a[2]),
    .b(b[2]),
    .r(r[2])
    );
and1bit uut3(
    .a(a[3]),
    .b(b[3]),
    .r(r[3])
    );
and1bit uut4(
    .a(a[4]),
    .b(b[4]),
    .r(r[4])
    );
assign sf = r[4];
assign zf = (r == 5'b0);
assign cf = 1'b0;
endmodule
```

and5bit_tb.v

```
`timescale 1ns/1ps
module and5bit_tb;
    reg [4:0] a;
    reg [4:0] b;

    wire [4:0] r;
    wire cf;
    wire zf;
    wire sf;

    and5bit uut(a, b, r, cf, zf, sf);
```

```
    initial begin
        a <= 5'b10001;
        b <= 5'b00011;
        #20;
        a <= 5'b11111;
        b <= 5'b00001;
        #20;
        a <= 5'b11111;
        b <= 5'b10001;
        #20;
        $finish;
    end
    initial begin
        $dumpfile("and5bit.vcd");
        $dumpvars(0, and5bit_tb);

        $monitor("a=%b b=%b r=%b cf=%b sf=%b zf=%b", a, b, r,
cf, sf, zf);
    end
endmodule
```

add1bit.v

```
module add1bit(
    input wire A,
    input wire B,
    input wire Cin,
    output wire S,
    output wire Cout
);

assign S=A^B^Cin;
assign Cout=A&B | B&Cin |Cin&A;

endmodule
```

add1bit_tb.v

```
`timescale 1ns/1ps

module add1bit_tb;
    reg A;
    reg B;
    reg Cin;
    wire S;
    wire Cout;

add1bit uut(
    .A(A),
    .B(B),
    .Cin(Cin),
    .S(S),
    .Cout(Cout)
);

initial begin
    $dumpfile("add1bit.vcd");
    $dumpvars(0,add1bit_tb);

    A=0;
    B=0;
    Cin=0;

    #20;
    B=1;

    #20;
    A=1;

    #20;
    B=0;

    #20;
    $finish;
end
```

```
initial begin
    $monitor("A=%b B=%b Cin=%b | S=%b
Cout=%b\n",A,B,Cin,S,Cout);
end

endmodule
```

adder.v

```
module adder(
    input wire [4:0] a,
    input wire [4:0] b,

    output wire [4:0] s,
    output wire cf,
    output wire zf,
    output wire sf
);
wire c1, c2, c3, c4;
add1bit uut0(
    .A(a[0]),
    .B(b[0]),
    .Cin(1'b0),
    .S(s[0]),
    .Cout(c1)
    );
add1bit uut1(
    .A(a[1]),
    .B(b[1]),
    .Cin(c1),
    .S(s[1]),
    .Cout(c2)
    );
add1bit uut2(
    .A(a[2]),
    .B(b[2]),
    .Cin(c2),
```

```
    .S(s[2]),
    .Cout(c3)
    );
add1bit uut3(
    .A(a[3]),
    .B(b[3]),
    .Cin(c3),
    .S(s[3]),
    .Cout(c4)
    );
add1bit uut4(
    .A(a[4]),
    .B(b[4]),
    .Cin(c4),
    .S(s[4]),
    .Cout(cf)
    );
assign sf = s[4];
assign zf = (s == 5'b00000);
endmodule
```

adder_tb.v

```
`timescale 1ns/1ps
module adder_tb;
    reg [4:0] a;
    reg [4:0] b;

    wire [4:0] s;
    wire cf;
    wire zf;
    wire sf;

    adder uut(a, b, s, cf, zf, sf);

    initial begin
        a <= 5'b10001;
        b <= 5'b00011;
```

```
        #20;
        a <= 5'b11111;
        b <= 5'b00001;
        #20;
        $finish;
    end
    initial begin
        $dumpfile("adder.vcd");
        $dumpvars(0, adder_tb);

        $monitor("a=%b b=%b s=%b cf=%b sf=%b zf=%b", a, b, s,
cf, sf, zf);
    end
endmodule
```

shr5bit.v

```
module shr5bit(
    input [4:0] a,
    input [4:0] b,
    input cf0,
    output [4:0] r,
    output cf,
    output sf,
    output zf
);
reg [4:0] res;
always @(*) begin
    if (b == 0 )
        res <= a;
    else if (b == 1)
        res <= {1'b0, a[4:1]};
    else if (b == 2)
        res <= {2'b0, a[4:2]};
    else if (b == 3)
        res <= {3'b0, a[4:3]};
    else if (b == 4)
        res <= {4'b0, a[4]};
```

```
    else
        res <= 5'b0;
end
reg cf_tmp;
always @(*) begin
    if (b == 0)
        cf_tmp <= cf0;
    else if (b == 1)
        cf_tmp <= a[0];
    else if (b == 2)
        cf_tmp <= a[1];
    else if (b == 3)
        cf_tmp <= a[2];
    else if (b == 4)
        cf_tmp <= a[3];
    else if (b == 5)
        cf_tmp <= a[4];
    else
        cf_tmp <= 1'b0;
end
assign r = res;

assign cf = cf_tmp;
assign sf = r[4];
assign zf = (r == 5'b0);
endmodule
```

shr5bit_tb.v

```
`timescale 1ns/1ps
module shr5bit_tb;
    reg [4:0] a;
    reg [4:0] b;
    reg cf0;

    wire [4:0] r;
    wire cf;
    wire zf;
```

```
    wire sf;

    shr5bit uut(a, b, cf0, r, cf, zf, sf);

    initial begin
        a <= 5'b11001;
        b <= 5'b00001;
        #20;
        a <= 5'b10011;
        b <= 5'b00100;
        #20;
        a <= 5'b11111;
        b <= 5'b00101;
        #20;
        $finish;
    end
    initial begin
        $dumpfile("shr5bit.vcd");
        $dumpvars(0, shr5bit_tb);

        $monitor("a=%b b=%b r=%b cf=%b sf=%b zf=%b", a, b, r,
cf, sf, zf);
    end
endmodule
```

alu.v

```
module alu(
    input [4:0] a,
    input [4:0] b,
    input [1:0] op,
    input cf0,
    output [4:0] r,
    output cf,
    output sf,
    output zf
);
    wire [4:0] res0, res1, res2;
```

```
    wire [2:0] f0, f1, f2;
    and5bit uut0(.a(a), .b(b), .r(res0), .cf(f0[0]), .sf(f0[1]),
.zf(f0[2]));
    adder uut1(.a(a), .b(b), .s(res1), .cf(f1[0]), .sf(f1[1]),
.zf(f1[2]));
    shr5bit uut2(.a(a), .b(b), .cf0(cf0), .r(res2), .cf(f2[0]),
.sf(f2[1]), .zf(f2[2]));


    reg [4:0] res;
    reg [2:0] f;
    always @(*) begin
        if (op == 0)
        begin
            res <= res0;
            f <= f0;
        end
        if (op == 1)
        begin
            res <= res1;
            f <= f1;
        end
        if (op == 2)
        begin
            res <= res2;
            f <= f2;
        end
    end

assign r = res;
assign cf = f[0];
assign sf = f[1];
assign zf = f[2];

endmodule
```

alu_tb.v

```
`timescale 1ns/1ps
module alu_tb;
    reg [4:0] a;
    reg [4:0] b;
    reg [1:0] op;
    reg cf0;
    wire [4:0] r;
    wire cf;
    wire sf;
    wire zf;

    alu uut(a, b, op, cf0, r, cf, sf, zf);

    initial begin
        a <= 5'b00101;
        b <= 5'b00100;
        op <= 2'b01;
        #20;
        a <= 5'b10101;
        b <= 5'b01010;
        op <= 2'b00;
        #20;
        a <= 5'b10010;
        b <= 5'b00010;
        op <= 2'b10;
        #20;
        a <= 5'b11111;
        b <= 5'b00001;
        op <= 2'b01;
        #20;
        $finish;
    end
    initial begin
        $dumpfile("alu.vcd");
        $dumpvars(0, alu_tb);

        $monitor("op=%b a=%b b=%b cf0=%b r=%b  cf=%b sf=%b
zf=%b", op, a, b, cf0, r, cf, sf, zf);
    end
```

```
endmodule
```

Makefile

```
fa:
    iverilog -o add1bit.vvp add1bit.v add1bit_tb.v
    vvp add1bit.vvp
gtk_fa:
    gtkwave add1bit.vcd

adder: add1bit.v adder.v adder_tb.v
    iverilog -o adder.vvp add1bit.v adder.v adder_tb.v
    vvp adder.vvp

and: and1bit.v and5bit.v and5bit_tb.v
    iverilog -o and5bit.vvp and1bit.v and5bit.v and5bit_tb.v
    vvp and5bit.vvp

shr: shr5bit.v shr5bit_tb.v
    iverilog -o shr5bit.vvp shr5bit.v shr5bit_tb.v
    vvp shr5bit.vvp

alu: and1bit.v and5bit.v add1bit.v adder.v shr5bit.v alu.v
alu_tb.v
    iverilog -o alu.vvp and1bit.v and5bit.v add1bit.v adder.v
shr5bit.v alu.v alu_tb.v
    vvp alu.vvp

clear:
    rm -r *.vcd
    rm -r *.vvp
```

2. **Register Set Circuit (Top to Bottom all circuits):**

3. **RAM Circuit (Top to Bottom all circuits):**

4. **CPU (Top to Bottom all circuits):**