R-type DLX Processor

Aaryan Darad (21110001)
Abhay Kumar Upparwal (21110004)
Vaibhavi Sharma (21110231)

1) Introduction:

This report presents the design and implementation of an R-Type Triadic Instruction Processor in Verilog. The instruction set of the processor is based on the RISC architecture, which is a reduced instruction set computer. The R-Type Triadic Instruction Processor is capable of performing arithmetic and logical operations on three operands. The processor has been designed using the Verilog hardware description language and has been verified through simulation.

2) Design:

The R-Type Triadic Instruction Processor consists of three main components: the ALU, reg file, and the top module.

ALU consists of all the operation such as ADD, SUB, AND, OR, XOR, SLL (shift left logical), SRL (shift right logical), SRA (shift right arithmetic), ROL (rotate left), ROR (rotate right), SLT (signed less than comparison), SGT (signed greater than), SLE (signed less than or equal to comparison), SGE, UGT, ULT, ULE, UGE.

The register file is typically accessed by two inputs: the register address, which specifies the register to be read or written, and the register data, which specifies the value to be written to the register or the value to be read from the register.

Top module integrates all the components and it receives a 32 bit instruction and gives a 32 bit output value.

The R-Type Triadic Instruction Processor has been verified using simulation. A testbench was created to verify the functionality of the processor. The testbench consists of a set of test cases that cover all the possible scenarios of the processor. The test cases include simple arithmetic and logical operations.
The simulation results show that the processor is able to execute all the instructions correctly. The processor was able to perform the operations on the operands as specified by the instructions. The register file and the ALU were able to store and perform the operations correctly.

3) Work Distribution:

1. Aaryan Darad:

- Coded Arithmetic and Comparator operations of ALU
- Coded the Constraint File and Ran the Implementation on FPGA, checked for errors in the codes

2. Abhay Kumar Upparwal:

- Coded the ALU and the logical operations
- Coded the Top Module

3. Vaibhavi Sharma:

- Coded the Register File and Top Module
- Responsible for Debugging in ALU

4) Verilog Codes

Top Module:

```
module topModule(input [7:0]IR,input p1,input p2, input p3, input
p4, output reg [15:0]result, input reset, input clk, input display);

wire   [31:0]D1out;
wire   [31:0]D2out;
reg [31:0]Instruction;
wire [31:0]Din;
reg_file M1(1'b1,Din,Instruction[15:11],Instruction[25:21],
Instruction [20:16],  D1out, D2out, reset, clk);
ALU M2(reset,Instruction [5:0], D1out, D2out, Din) ;
always@(reset,clk) begin
   if (reset) begin
   Instruction=0;
   result=0;
   end
   else
   if(p1) Instruction[31:24]= IR[7:0];
   else if(p2) Instruction[23:16]= IR[7:0];
   else if(p3) Instruction[15:8]= IR[7:0];
   else if (p4) Instruction[7:0]= IR[7:0];
   if (display) result=Din[15:0];
   end
endmodule
```

Register File:

```
module reg_file( input WE, input [31:0]Din, input [4:0] RD, RS1,
RS2,  output [31:0]D1out, output [31:0]D2out,input reset, input clk
);
    integer i;
    reg [31:0] RegFile [31:0];
    assign D1out=RegFile[RS1];
    assign D2out=RegFile[RS2];

    always @(posedge clk) begin
    if(reset) begin
```

```
        for (i=0; i<32;i=i+1) begin
            RegFile[i]=i;
        end
    end
        if (WE) RegFile [RD]<= Din;
   end
endmodule
```

ALU:

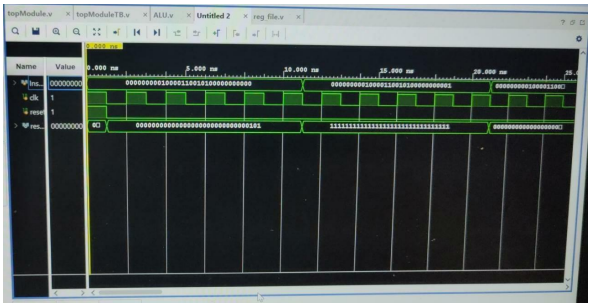
```
module ALU(input reset,input [5:0] opcode, input [31:0] rs1, rs2,
output reg [31:0] rd);

reg [32:0]w;
//reg func_code;


parameter ADD = 6'b000000;
parameter SUB = 6'b000001;
parameter AND = 6'b000010;
parameter OR  = 6'b000011;
parameter XOR = 6'b000100;
parameter SLL = 6'b000101;
parameter SRL = 6'b000110;
parameter ROL = 6'b000111;
parameter ROR = 6'b001000;
parameter SLT = 6'b001001;
parameter SGT = 6'b001010;
parameter SLE = 6'b001011;
parameter SGE = 6'b001100;
parameter UGT = 6'b001101;
parameter ULT = 6'b001110;
parameter ULE = 6'b001111;
parameter UGE = 6'b010000;
parameter SRA = 6'b010001;

always@(*) begin
//func_code = opcode
    if (reset)
        rd=0;
   else if(opcode == ADD)
        begin
```

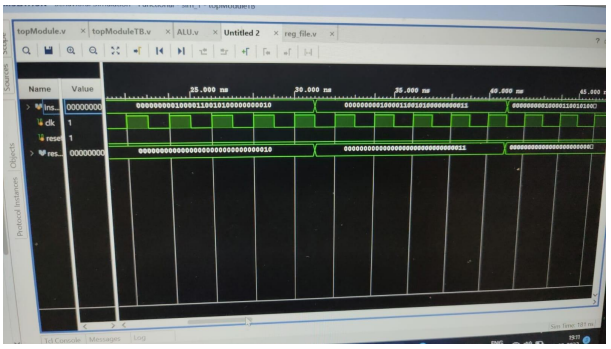
```
       rd <= rs1 + rs2;

        end
    else if(opcode == SUB)
        begin
        rd <= rs1 - rs2;

        end
    else if(opcode == AND)
        begin
        rd <= rs1 & rs2;
        end
    else if(opcode == OR)
        begin
        rd <= rs1 | rs2;
        end
    else if(opcode == XOR)
        begin
        rd<= rs1^rs2;
        end
    else if(opcode == SLL)
        begin
        if(rs2<32)
            rd <= rs1 << rs2;
        else
            rd<=32'b0;
        end
    else if(opcode == SRL)
        begin
        if(rs2<32)
            rd <= rs1 >> rs2;
        else
            rd<=32'b0;
        end
    else if(opcode == SRA)
        if(rs2<32)
                rd <= rs1 >>> rs2;
        else
             if(rs1[31]==1)
                 rd<='hFFFFFFFF;
             else
                 rd<='h00000000;
```

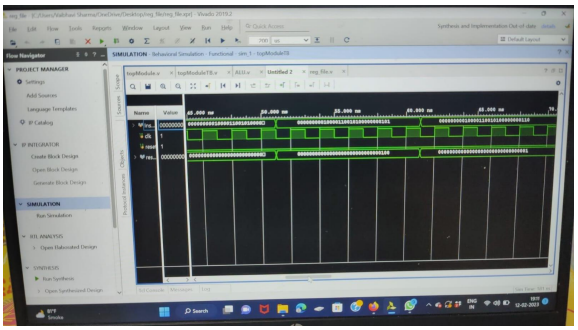
```
    else if(opcode == ROR)
        begin
       rd = {rs1[0], rs1[31:1]} ;
        end
    else if(opcode == ROL)
        begin
                if(rs1[31]==0)
                    rd <= rs1 << 1;
                else
                    rd <= rs1 <<< 1;
        end
    else if(opcode == SLT) begin
        w=rs1-rs2;
        if (rs1[31]^rs2[31]==1)
          if(rs1[31]==1)
          rd = 'hffffffff;
          else
          rd = 'h00000000;
        else
            if(w[32]==1) rd='hffffffff;
            else rd=1'b0;
   end
    else if(opcode == SGT) begin
        w=rs1-rs2;
        if (rs1[31]^rs2[31]==1)
          if(rs1[31]==1)
          rd = 1'b0;
          else
          rd ='hffffffff;
        else
            if(w[32]==1|| w==0) rd=1'b0;
            else rd='hffffffff;
    end
    else if(opcode == SLE) begin
        w=rs1-rs2;
        if (rs1[31]^rs2[31]==1)
          if(rs1[31]==1)
          rd = 'hffffffff;
          else
          rd = 1'b0;
        else
            if(w[32]==1|| w==0) rd='hffffffff;
            else rd=1'b0;
```

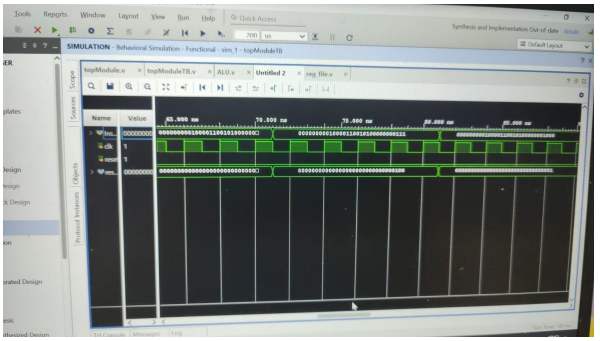
```
    end
    else if(opcode == SGE)
begin
        w=rs1-rs2;
        if (rs1[31]^rs2[31]==1)
          if(rs1[31]==1)
          rd = 1'b0;
          else
          rd = 1'b1;
        else if(rs1[31]==0)
            if(w[32]==1) rd=1'b0;
            else rd='hffffffff;
   end
    else if(opcode == UGT) begin
        w=rs1-rs2;
        if(w[32]==1||w==0) rd=1'b0;
        else rd='hffffffff;
        end
    else if(opcode == ULT)
        begin
        w=rs1-rs2;
        if(w[32]==1) rd='hffffffff;
            else rd=1'b0;
        end
    else if(opcode == ULE) begin
        w=rs1-rs2;
        if(w[32]==1||w==0) rd='hffffffff;
        else rd=1'b0;
        end
    else if(opcode == UGE)
        begin
        w=rs1-rs2;
        if(w[32]==1) rd=1'b0;
            else rd='hffffffff;
        end
end
endmodule
```

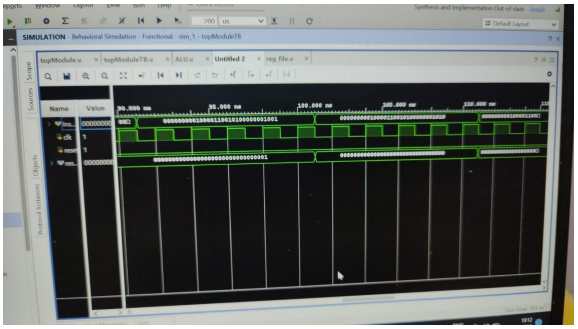
5) Simulation Waveforms:

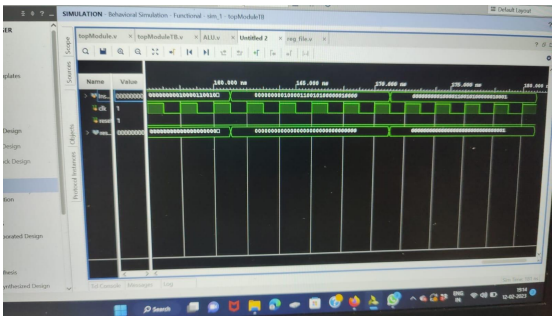


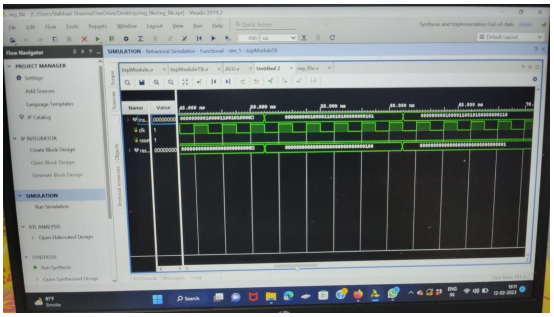


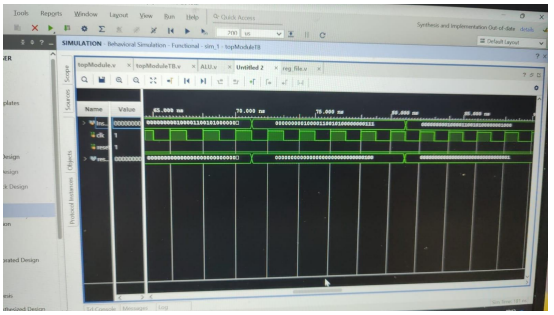


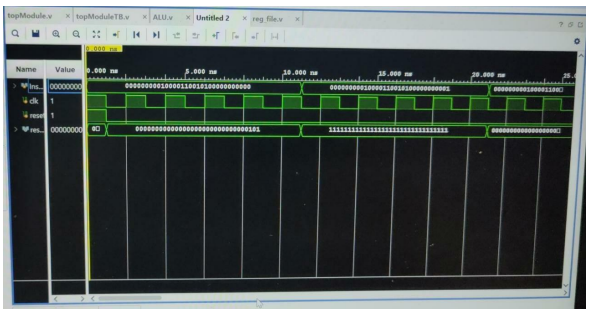
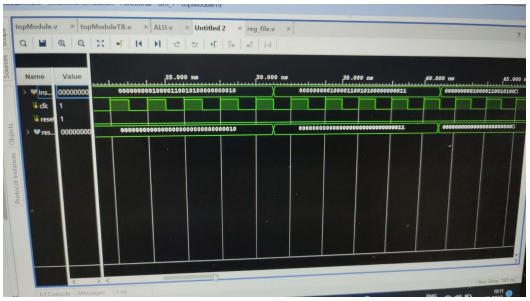
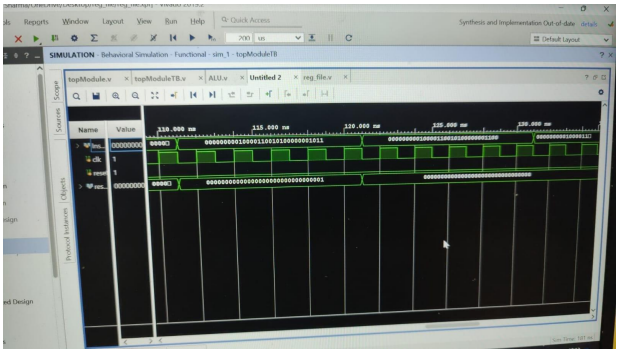
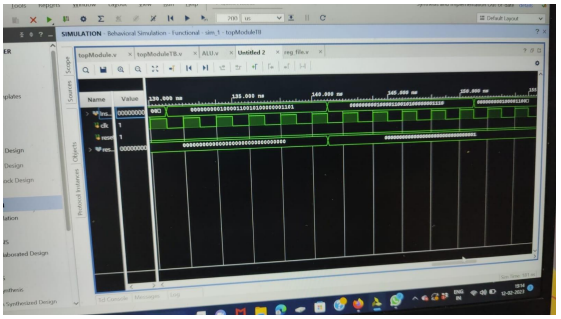
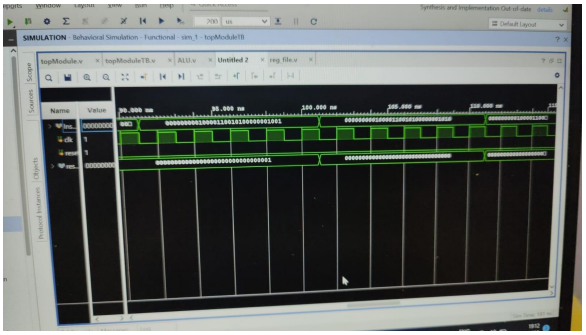
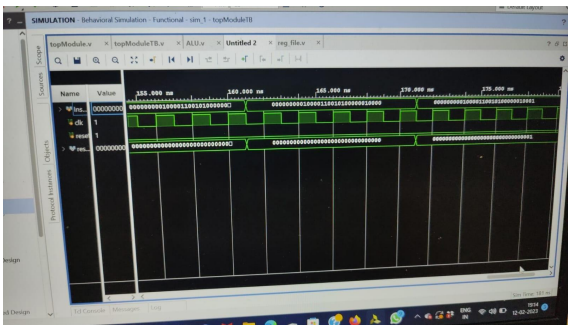




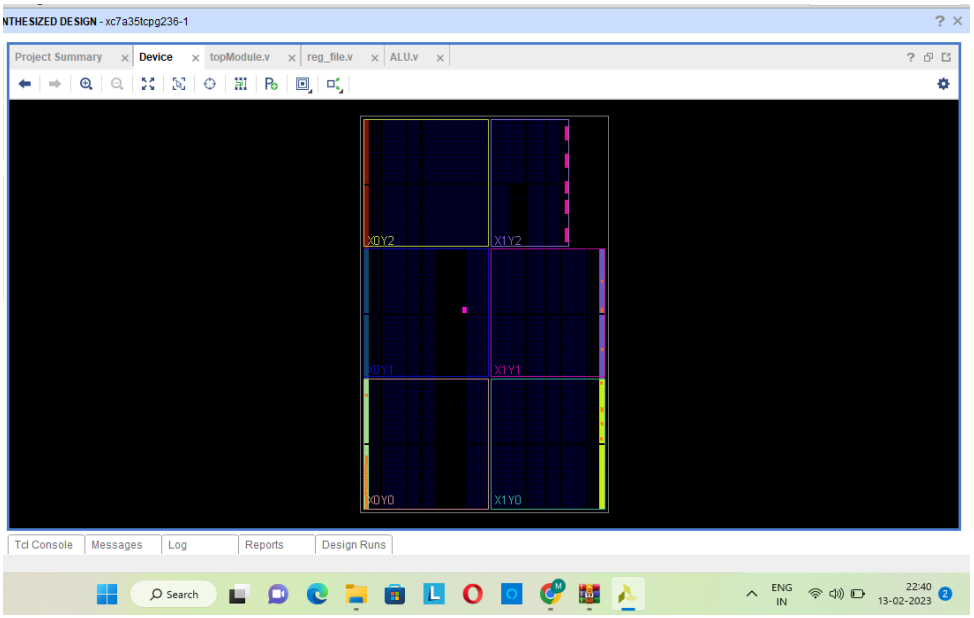






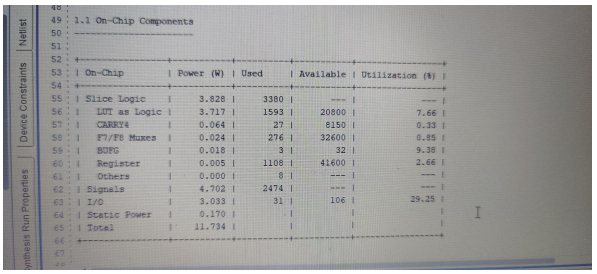







6) Synthesis


NTHESIZED DESIGN - xc7a35tcpg236-1
Project Summary
Device
topModule.v
reg_file.v
ALU.v
X0Y2
X1Y2
X0Y1
X1Y1
X0Y0
X1Y0
Tcl Console
Messages
Log
Reports
Design Runs
Search
ENG
IN
22:40
13-02-2023

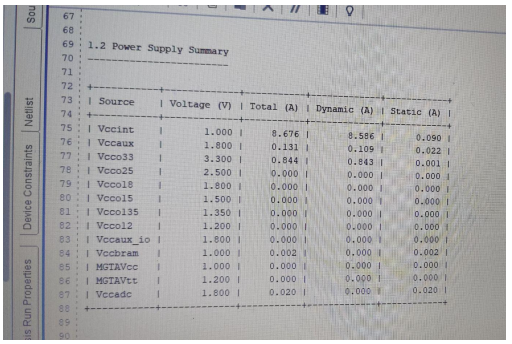
7) Implementation

8) Report



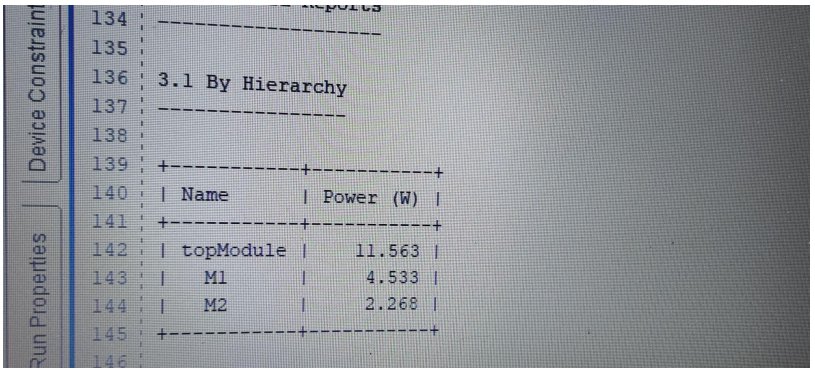
1.1 On-Chip Components

On-Chip	Power (W)	Used	Available	Utilization (%)
Slice Logic	3.828	3380	---	---
LUT as Logic	3.717	1593	20800	7.66
CARRY4	0.064	27	8150	0.33
F7/F8 Muxes	0.024	276	32600	0.85
BUFG	0.018	3	32	9.38
Register	0.005	1108	41600	2.66
Others	0.000	8	---	---
Signals	4.702	2474	---	---
I/O	3.033	31	106	29.25
Static Power	0.170			
Total	11.734			



1.2 Power Supply Summary

Source	Voltage (V)	Total (A)	Dynamic (A)	Static (A)
Vccint	1.000	8.676	8.586	0.090
Vccaux	1.800	0.131	0.109	0.022
Vcco33	3.300	0.844	0.843	0.001
Vcco25	2.500	0.000	0.000	0.000
Vcco18	1.800	0.000	0.000	0.000
Vcco15	1.500	0.000	0.000	0.000
Vcco135	1.350	0.000	0.000	0.000
Vcco12	1.200	0.000	0.000	0.000
Vccaux_io	1.800	0.000	0.000	0.000
Vccbram	1.000	0.002	0.000	0.002
MGTAVcc	1.000	0.000	0.000	0.000
MGTAVtt	1.200	0.000	0.000	0.000
Vccadc	1.800	0.020	0.000	0.020



3.1 By Hierarchy

Name	Power (W)
topModule	11.563
M1	4.533
M2	2.268

9) Summary

Overall, it was a great learning experience for us, when we implemented all the various types of instructions that we studied in class for real on FPGA. It felt real and not just what we studied in a book. We're excited to learn more about Computer Architecture and Organization further.