**MEL G421- CAD FOR IC DESIGN**

**PROJECT REPORT**

**MIN ARCHITECTURE DESIGN - CISC PROCESSOR**



**Submitted By**

**NIKHIL GUPTA 2022H123P**

**AMURT PRAKASH 2022H123P**

**AATIB MOHAMMAD 2022H1230239P**

**AVADH RAJESH HARKISHANKA 2018HS230322P**

**Submitted To**

Dr. Abhijit Asati

Electrical and Electronics Department, BITS Pilani (Pilani Campus)

1. **MIN - ARCHITECTURE DESIGN - CISC PROCESSOR**
   * + 1. **INTRODUCTION**

CISC Processors use variable size instructions and a variety of addressing modes to access data and do operations on them. This flexibility on the side of the programmer when implemented on hardware needs variable cycles for implementation. The various addressing modes in order to be implemented on the hardware need a controller design using micro-coded implementation. MIN Architecture is one such implementation which covers some basic set of instructions of CISC processors.

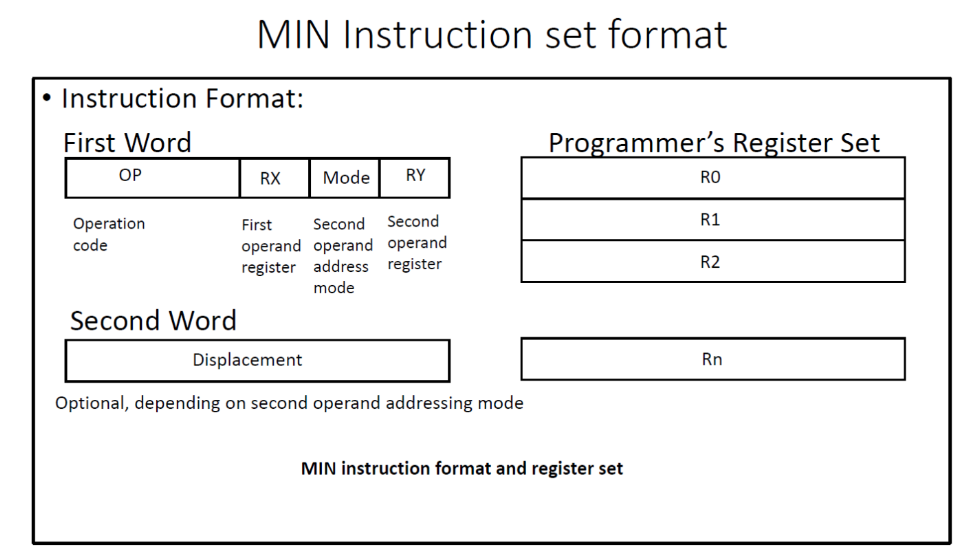
The instructions are firstly broken down into RTL steps. These are noted down as hardware flowcharts and then each RTL step is mapped to a micro-coded instruction.

These steps are implemented using the execution unit which takes in the micro-coded instruction as input giving us the control signals.

Each micro-coded instruction maps to a set of control bits.

The processor is designed using behavioral modeling and is simulated on Xilinx Vivado.

MIN instruction set format is:

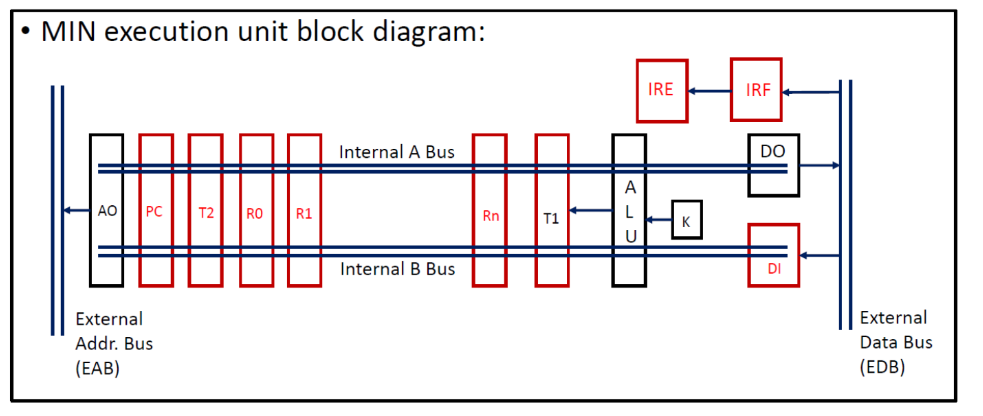


The microcoded control implementation block diagram for MIN architecture is shown:

A picture containing text, diagram, plan, font

Description automatically generated

The execution unit block is implemented as:



The following implementation can be seen on Xilinx Vivado:

A picture containing sketch, diagram, plan, technical drawing

Description automatically generated

1. VERILOG DESIGN AND SIMULATION

**Testbench**

`timescale 1ns / 1ps

module tb;

reg signed [15:0] mem [0:15];

reg clk,rst;

reg signed [15:0] edb;

wire signed [15:0] eab;

wire signed [17:0] cntrl;

wire signed [15:0] pc,t1,t2,di,do,irf,ire;

wire zero,sign,carry,overflow;

wire [4:0] ib,sb,bc;

initial

begin

clk=0;

forever #5 clk=~clk;

end

initial

begin

mem[0]=16'd4;

mem[1]=16'h3197;

mem[6]=16'h0055;

end

always@(eab)

begin

edb<=mem[eab];

end

cisc\_proc DUT(clk,rst,cntrl,edb,eab,zero,overflow,sign,carry,pc,t1,t2,di,do,irf,ire,ib,sb,bc);

initial

begin

rst=1'b1;

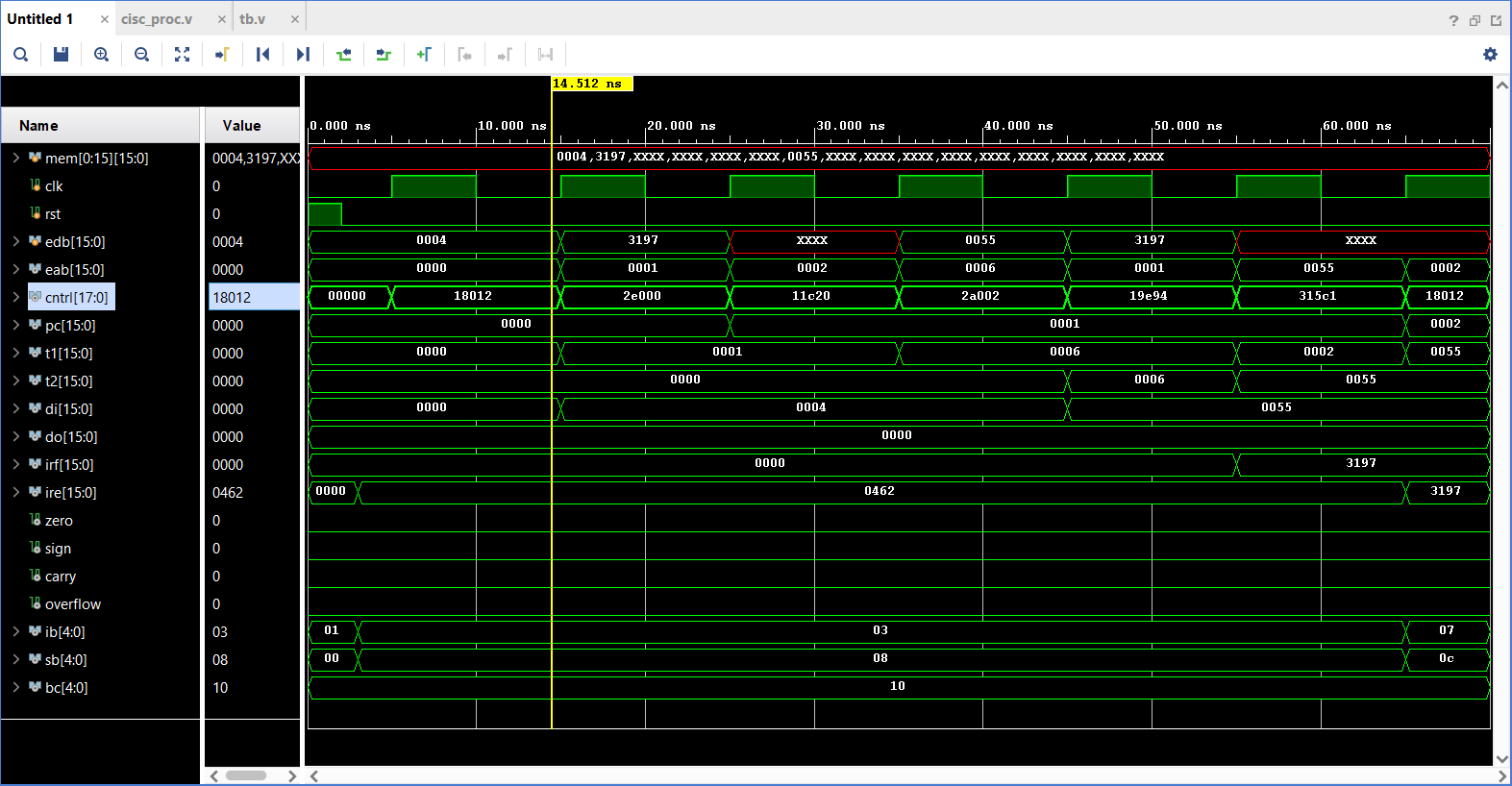
#2 rst=1'b0;

#68 $finish;

end

endmodule

**Simulation Result**

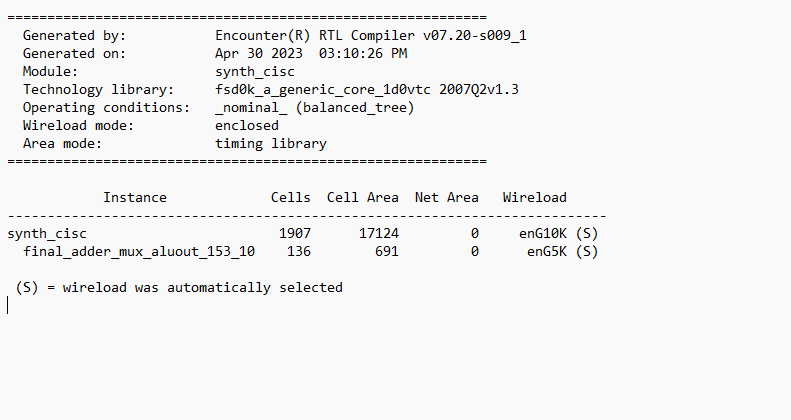


1. RTL Compiler

* RTL Compiler is a tool that is commonly used in digital design to synthesize register transfer level (RTL) code to gate-level netlists. The RTL code is a high-level description of the digital circuit behavior that is expressed in a hardware description language (HDL) such as Verilog or VHDL.
* The RTL Compiler takes the RTL code as input and performs a series of optimizations to produce an optimized gate-level netlist that can be used to generate the physical layout of the circuit. These optimizations include logic optimization, mapping to a library of standard cells, timing optimization, and placement and routing.
* The output of the RTL Compiler is a gate-level netlist, which is a description of the circuit in terms of gates, flip-flops, and interconnects. This netlist can then be used by other tools in the design flow, such as the place and route tool, to produce the final physical layout of the circuit.
* The use of RTL Compiler is an important step in the digital design flow, as it allows designers to optimize their RTL code for area, power, and timing. By using a tool such as RTL Compiler, designers can achieve better performance and lower power consumption in their digital circuits.

1. RTL Compiler

* Area report



* Gate Area Report

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

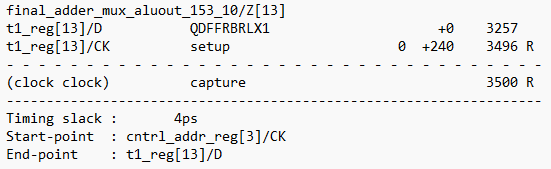
* Power Report

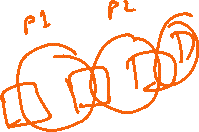
A screenshot of a computer program

Description automatically generated with medium confidence

* Timing report

Path 1





Path 2



A screenshot of a computer program

Description automatically generated with low confidence



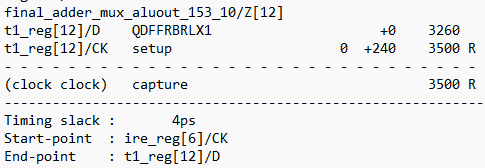
Path 3

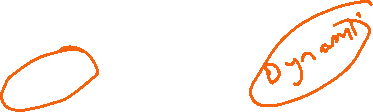
A screenshot of a computer program

Description automatically generated with low confidence

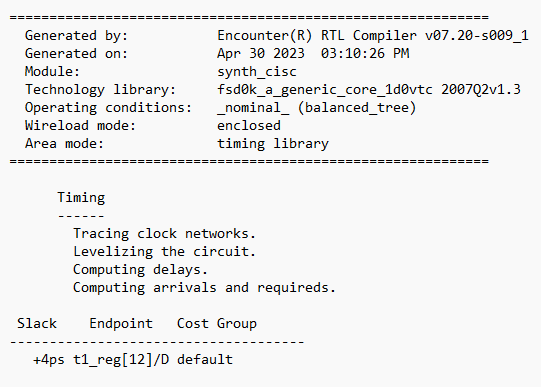
Path 4







* Report Summary



A screenshot of a computer

Description automatically generated with medium confidence

1. SOC Encounter

* It is a complete end-to-end physical design tool from Cadence Design Systems that is used to design and implement complex system-on-chip (SoC) designs. It is a powerful tool that supports all aspects of physical design, including floorplanning, placement, routing, and verification.
* It provides a comprehensive set of features for physical design, including optimization engines, detailed analysis tools, and automated scripting capabilities. The tool is designed to handle large-scale designs and can optimize the design for various goals, such as power, performance, and area.
* The tool is integrated with other tools in the Cadence design flow, such as the RTL Compiler and Innovus Implementation System, which enables designers to use the output of these tools as input to SOC Encounter. This tight integration allows for a smooth design flow and reduces the time-to-market for complex SoC designs.
* It is widely used in the semiconductor industry for designing high-performance, low-power SoCs for a range of applications, such as mobile devices, networking equipment, and automotive systems. It is known for its scalability, speed, and accuracy, which make it an ideal tool for designing complex SoCs with millions of gates.
* Placement check

A screenshot of a computer

Description automatically generated with medium confidence

* Timing Report (pre CTS)

A screenshot of a computer

Description automatically generated with medium confidence

* Optimizing Design

A picture containing text, font, number, diagram

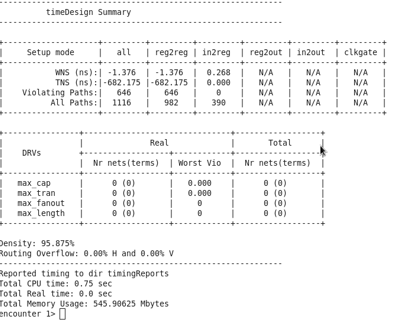
Description automatically generated

* Timing report (After Optimization)

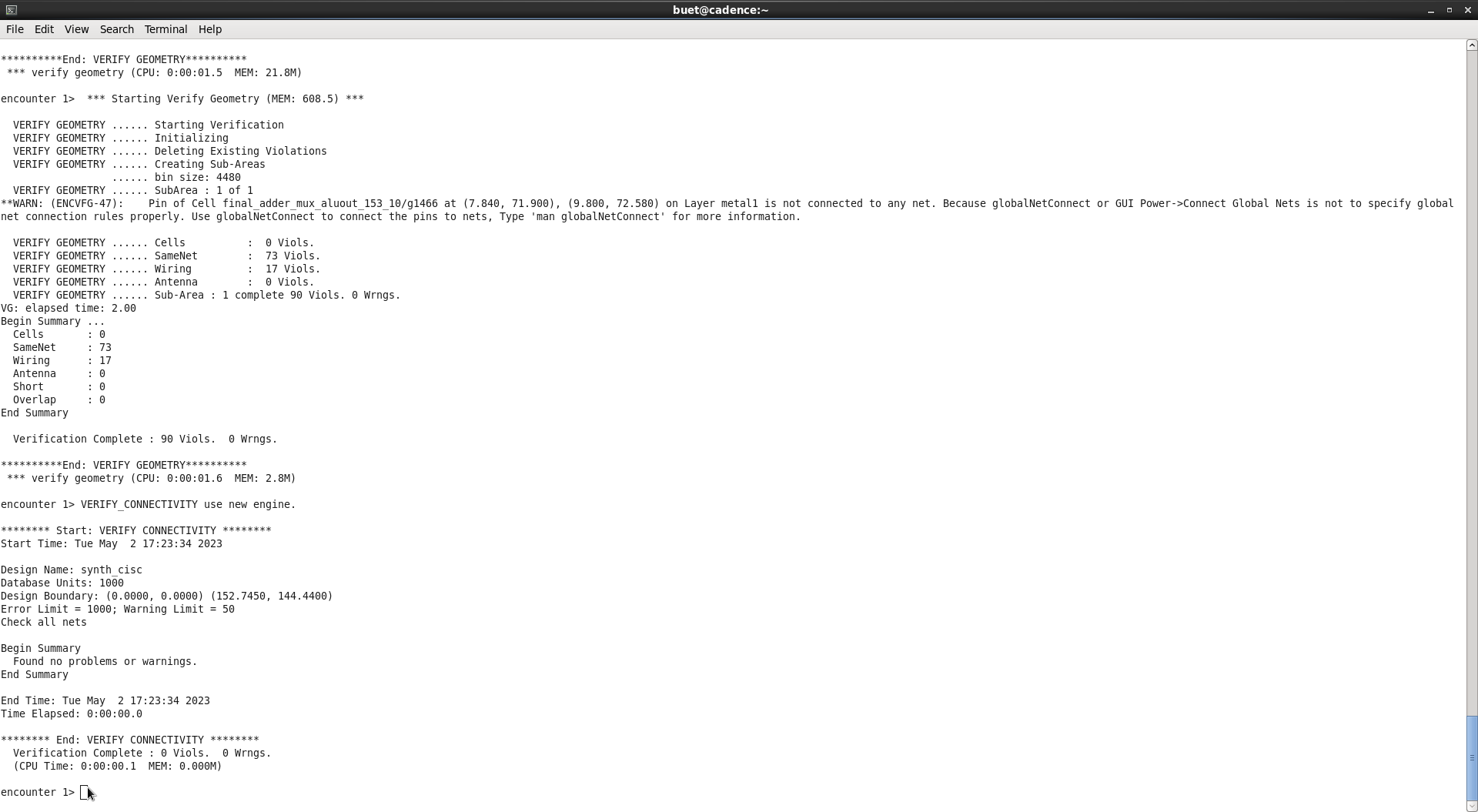
A screenshot of a computer

Description automatically generated with medium confidence

* Timing report (post CTS)



* Verifying connectivity

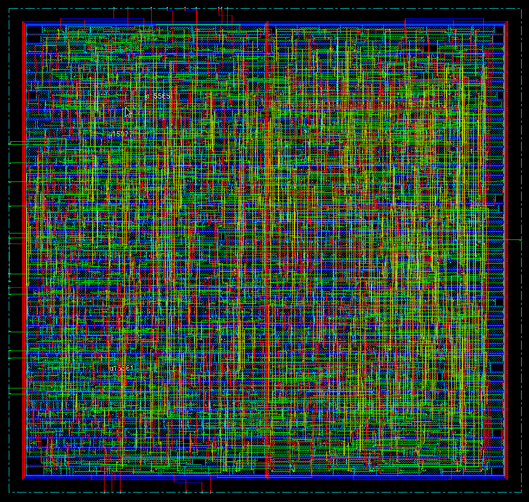


* Power Analysis

A screenshot of a computer

Description automatically generated with medium confidence

* Final Layout



**REFERENCES**

* XLINX official Website
* verilog-hdl-samir-palnitkar-2nd-edition
* handbook-of-hardware-software-codesign-soonhoi-ha
* <https://xilinxprod-catalog.netexam.com/Certification/47226/designing-with-versal-ai-engine-2-graph-programming-with-ai-engine-kernels>
* https://www.abebooks.fr/9780792376446/Advanced-ASIC-Chip-Synthesis-Using-0792376447/plp