UNIVERSITY OF MORATUWA, SRI LANKA

Faculty of Engineering

Department of Electronic and Telecommunication Engineering

Semester 5 (Intake 2020)

EN3021 - Digital System Design

Non-pipelined Single Stage (Cycle) CPU Design

Individual Project

W. A. S. S. Wanigathunga

200693D

October 16 , 2023

Contents

Abstract

This report presents the design and implementation of a 32-bit non-pipelined RISC-V processor on a FPGA. The project consists of three stages, and this report focuses on the first stage, which is the individual project.

The processor uses microprogramming with a three-bus structure and supports the RV32I instruction set. The processor can execute all computational, memory access, and control flow instructions covered by the R and I, S, and SB types respectively. In addition, the processor can handle two new instructions: MEMCOPY and MUL. The MEMCOPY instruction copies an array of size N from one source location to another destination location in memory. The MUL instruction performs unsigned multiplication of two operands.

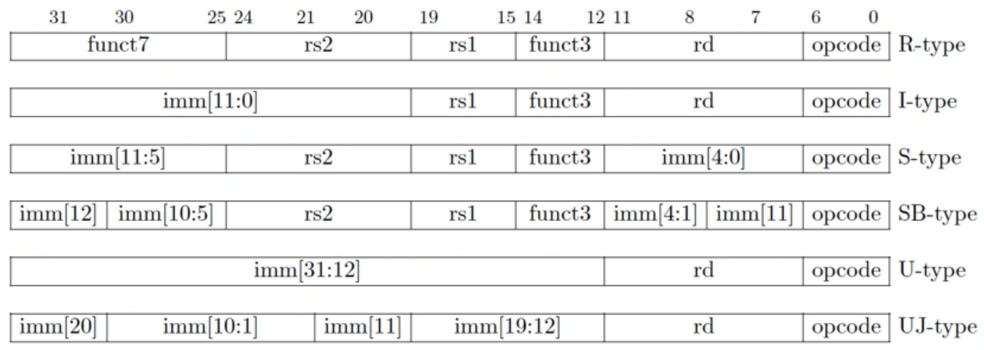
The report discusses the design choices, implementation details, and testing results of the processor. It also analyzes the limitations and challenges of the new instructions, such as the maximum array size for MEMCOPY and the operand range for MUL.

GitHub Link : https://github.com/SasiniWanigathunga/Single_Cycle_RISCV_Processor.git

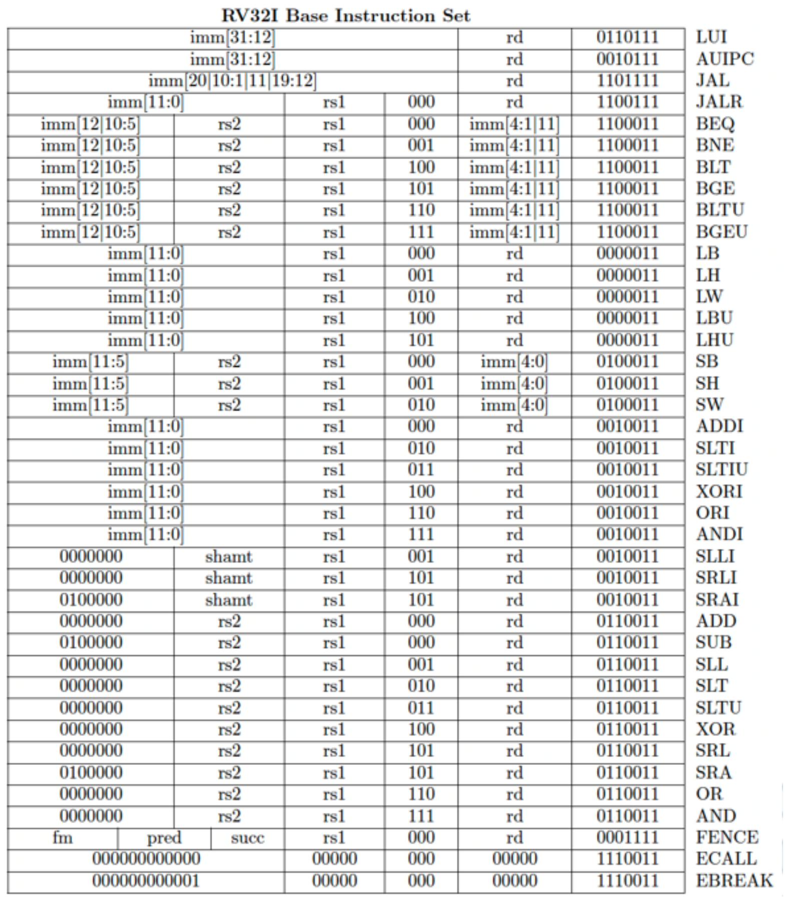
Design of Non-pipelined Single Stage (Cycle) RISC-V Processor

Instruction Set Architecture

There are 6 main types of instructions in RISC-V Instruction Set Architecture. But only R, I, S and SB type instructions are implemented in this project. Each instruction has a length of 32 bits.



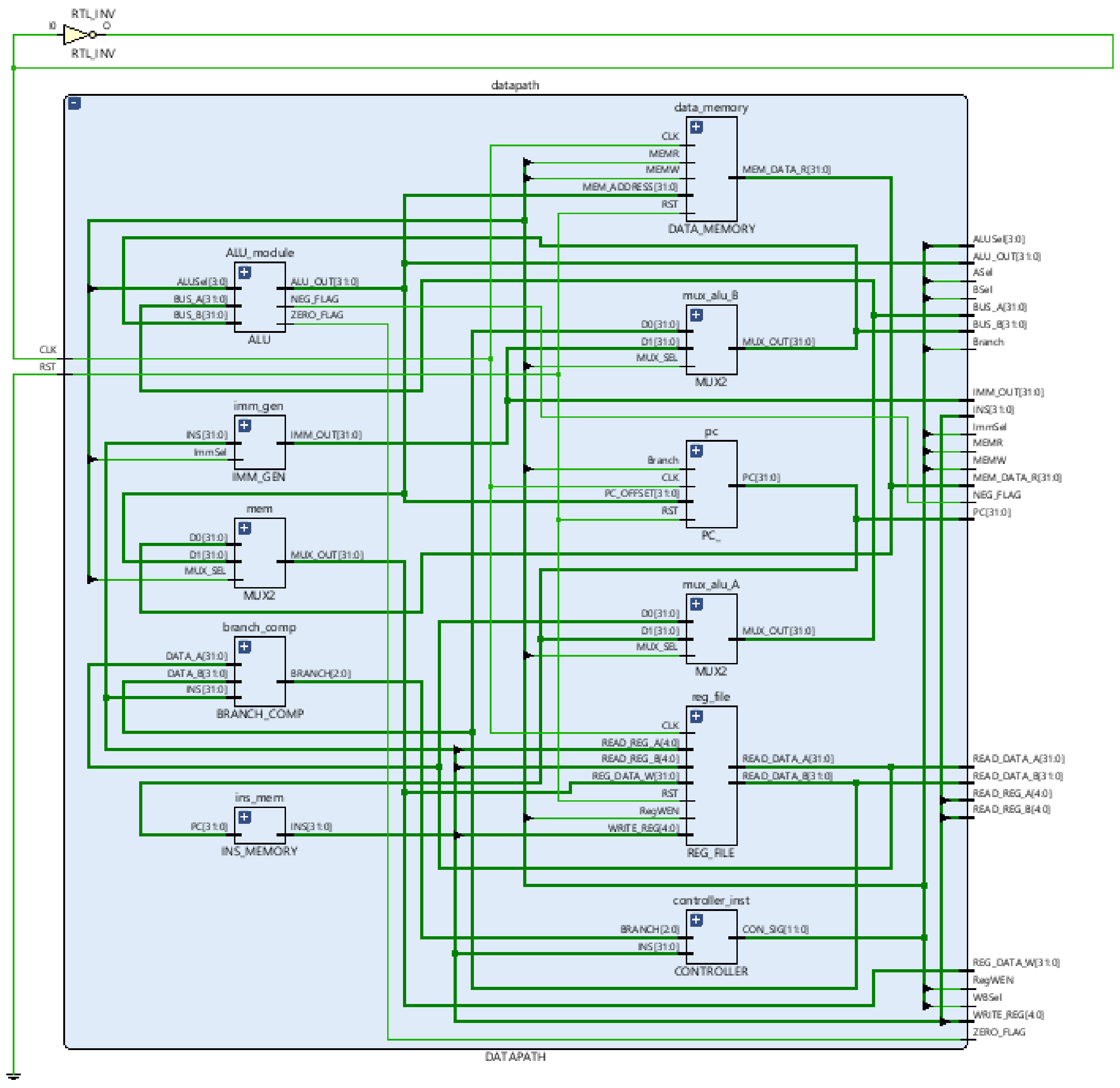
31	30 25	24 21	20	19 15	14 12	11 8	7	6 0	
funct7		rs2		rs1	funct3	rd		opcode	R-type
imm[11:0]				rs1	funct3	rd		opcode	I-type
imm[11:5]		rs2		rs1	funct3	imm[4:0]		opcode	S-type
imm[12]	imm[10:5]	rs2		rs1	funct3	imm[4:1]	imm[11]	opcode	SB-type
imm[31:12]						rd		opcode	U-type
imm[20]	imm[10:1]		imm[11]	imm[19:12]		rd		opcode	UJ-type



RV32I Base Instruction Set

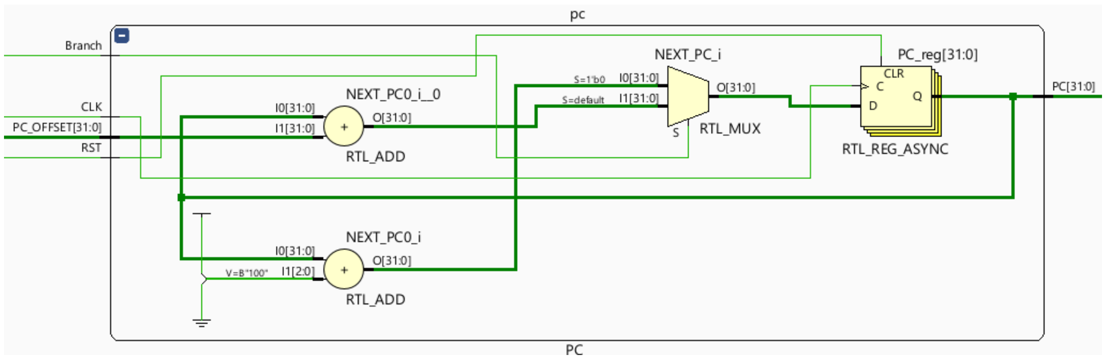
imm[31:12]				rd	0110111	LUI
imm[31:12]				rd	0010111	AUIPC
imm[20|10:1|11|19:12]				rd	1101111	JAL
imm[11:0]		rs1	000	rd	1100111	JALR
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW
imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND
fm pred	succ	rs1	000	rd	0001111	FENCE
000000000000		00000	000	00000	1110011	ECALL
000000000001		00000	000	00000	1110011	EBREAK

Datapath

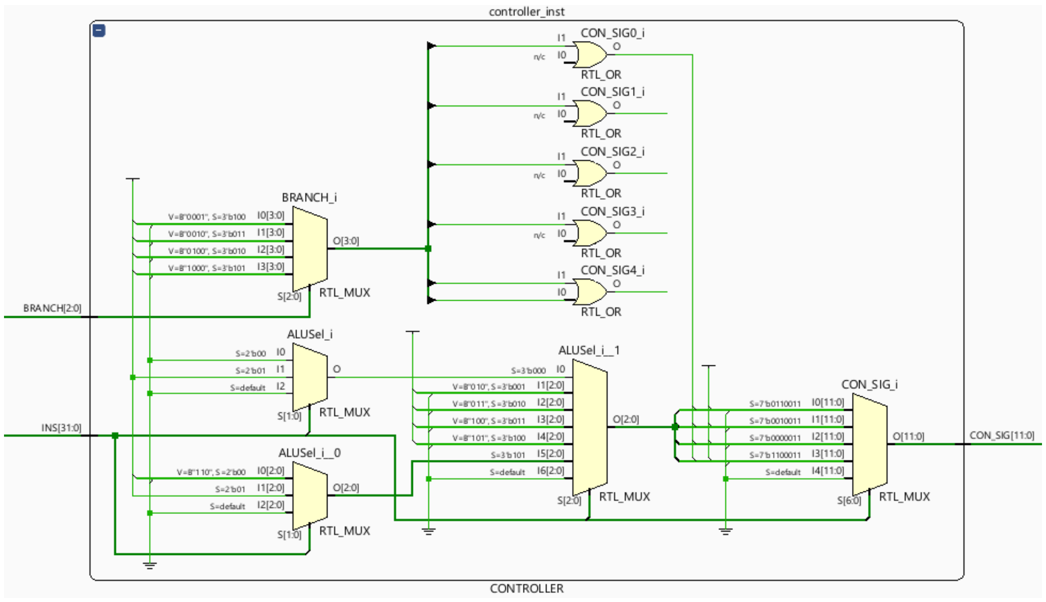

RTL_INV
I0
O
RTL_INV
datapath
data_memory
CLK
MEMR
MEMW
MEM_ADDRESS[31:0]
RST
MEM_DATA_R[31:0]
DATA_MEMORY
ALU_module
ALUSel[3:0]
BUS_A[31:0]
BUS_B[31:0]
ALU_OUT[31:0]
NEG_FLAG
ZERO_FLAG
ALU
mux_alu_B
D0[31:0]
D1[31:0]
MUX_SEL
MUX_OUT[31:0]
MUX2
CLK
RST
imm_gen
INS[31:0]
ImmSel
IMM_OUT[31:0]
IMM_GEN
pc
Branch
CLK
PC_OFFSET[31:0]
RST
PC[31:0]
PC_
mem
D0[31:0]
D1[31:0]
MUX_SEL
MUX_OUT[31:0]
MUX2
mux_alu_A
D0[31:0]
D1[31:0]
MUX_SEL
MUX_OUT[31:0]
MUX2
branch_comp
DATA_A[31:0]
DATA_B[31:0]
INS[31:0]
BRANCH[2:0]
BRANCH_COMP
reg_file
CLK
READ_REG_A[4:0]
READ_REG_B[4:0]
REG_DATA_W[31:0]
RST
RegWEN
WRITE_REG[4:0]
READ_DATA_A[31:0]
READ_DATA_B[31:0]
REG_FILE
ins_mem
PC[31:0]
INS[31:0]
INS_MEMORY
controller_inst
BRANCH[2:0]
INS[31:0]
CON_SIG[11:0]
CONTROLLER
DATAPATH
ALUSel[3:0]
ALU_OUT[31:0]
ASel
BSel
BUS_A[31:0]
BUS_B[31:0]
Branch
IMM_OUT[31:0]
INS[31:0]
ImmSel
MEMR
MEMW
MEM_DATA_R[31:0]
NEG_FLAG
PC[31:0]
READ_DATA_A[31:0]
READ_DATA_B[31:0]
READ_REG_A[4:0]
READ_REG_B[4:0]
REG_DATA_W[31:0]
RegWEN
WBSel
WRITE_REG[4:0]
ZERO_FLAG

Modules and Components

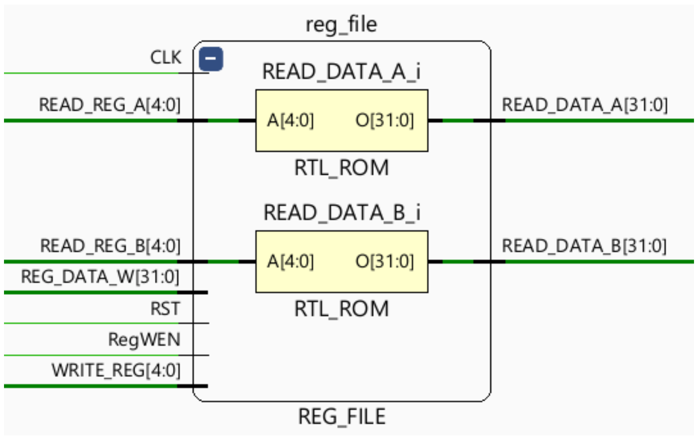
Program Counter



Control Store



Register File



Control Unit

The derivation of the control logic

TYPE_R	ADD		0000000	rs2	rs1	000	rd	0110011
	SUB		0100000	rs2	rs1	000	rd	0110011
	SLL	shift left logical	0000000	rs2	rs1	001	rd	0110011
	SLT	set if less than , 2's complement	0000000	rs2	rs1	010	rd	0110011
	SLTU	set if less than , unsigned	0000000	rs2	rs1	011	rd	0110011
	XOR	bitwise xor	0000000	rs2	rs1	100	rd	0110011
	SRL	shift right logical	0000000	rs2	rs1	101	rd	0110011
	SRA	shift right arithmetic	0100000	rs2	rs1	101	rd	0110011
	OR	bitwise or	0000000	rs2	rs1	110	rd	0110011
	AND	bitwise and	0000000	rs2	rs1	111	rd	0110011
TYPE_I_COMP	ADDI		imm[11:0]		rs2	000	rd	0010011
	SLTI		imm[11:0]		rs3	010	rd	0010011
	SLTIU		imm[11:0]		rs4	011	rd	0010011
	XORI		imm[11:0]		rs5	100	rd	0010011
	ORI		imm[11:0]		rs6	110	rd	0010011
	ANDI		imm[11:0]		rs7	111	rd	0010011
	SLLI		0000000	shamt	rs1	001	rd	0010011
	SRLI		0000000	shamt	rs1	101	rd	0010011
	SRAI		0100000	shamt	rs1	101	rd	0010011
TYPE_I_JALR	JALR		imm[11:0]		rs1	000	rd	1100111
TYPE_I_LO	LB		imm[11:0]		rs1	000	rd	0000011

AD								
	LH		imm[11:0]		rs1	001	rd	0000011
	LW		imm[11:0]		rs1	010	rd	0000011
	LBU		imm[11:0]		rs1	100	rd	0000011
	LHU		imm[11:0]		rs1	101	rd	0000011
TYPE_S	SB		imm[11:5]	rs2	rs1	000	imm[4:0]	0100011
	SH		imm[11:5]	rs2	rs1	001	imm[4:0]	0100011
	SW		imm[11:5]	rs2	rs1	010	imm[4:0]	0100011
TYPE_SB	BEQ		imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011
	BNE		imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011
	BLT		imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011
	BGE		imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011
	BLTU		imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011
	BGEU		imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011

Control Store

			[31:25]	[24:20]	[19:15]	[14:12]	[11:7]	[6:0]		PCSel	ImmSel	RegWEn	ASel	BSel	ALUSel	MEMR	MEMW	LS	Unsigned	WBSel	BrUn	Branch	BrEq	BrLT	BrGT
TYPE_R	ADD		0000000	rs2	rs1	000	rd	0110011		0	0	1	0	0	0000	0	0	00	0	01	0	000	0	0	0
	SUB		0100000	rs2	rs1	000	rd	0110011		0	0	1	0	0	0001	0	0	00	0	01	0	000	0	0	0

	SLL	shift left logical	0000000	rs2	rs1	001	rd	0110011		0	0	1	0	0	0010	0	0	00	0	01	0	000	0	0	0
	SLT	set if less than , 2's complement	0000000	rs2	rs1	010	rd	0110011		0	0	1	0	0	0011	0	0	00	0	01	0	000	0	0	0
	SLTU	set if less than , unsigned	0000000	rs2	rs1	011	rd	0110011		0	0	1	0	0	0100	0	0	00	0	01	0	000	0	0	0
	XOR	bitwise xor	0000000	rs2	rs1	100	rd	0110011		0	0	1	0	0	0101	0	0	00	0	01	0	000	0	0	0
	SRL	shift right logical	0000000	rs2	rs1	101	rd	0110011		0	0	1	0	0	0110	0	0	00	0	01	0	000	0	0	0
	SRA	shift right arithmetic	0100000	rs2	rs1	101	rd	0110011		0	0	1	0	0	0111	0	0	00	0	01	0	000	0	0	0
	OR	bitwise or	0000000	rs2	rs1	110	rd	0110011		0	0	1	0	0	1000	0	0	00	0	01	0	000	0	0	0
	AND	bitwise and	0000000	rs2	rs1	111	rd	0110011		0	0	1	0	0	1001	0	0	00	0	01	0	000	0	0	0
TYPE_I_COMP	ADDI		imm[11:0]		rs2	000	rd	0010011		0	1	1	0	1	0000	0	0	00	0	01	0	000	0	0	0
	SLTI		imm[11:0]		rs3	010	rd	0010011		0	1	1	0	1	0011	0	0	00	0	01	0	000	0	0	0
	SLTIU		imm[11:0]		rs4	011	rd	0010011		0	1	1	0	1	0100	0	0	00	0	01	0	000	0	0	0
	XORI		imm[11:0]		rs5	100	rd	0010011		0	1	1	0	1	0101	0	0	00	0	01	0	000	0	0	0
	ORI		imm[11:0]		rs6	110	rd	0010011		0	1	1	0	1	1000	0	0	00	0	01	0	000	0	0	0
	ANDI		imm[11:0]		rs7	111	rd	0010011		0	1	1	0	1	1001	0	0	00	0	01	0	000	0	0	0

	SLLI		0000000	shamt	rs1	001	rd	0010011		0	1	1	0	1	0010	0	0	00	0	01	0	000	0	0	0
	SRLI		0000000	shamt	rs1	101	rd	0010011		0	1	1	0	1	0110	0	0	00	0	01	0	000	0	0	0
	SRAI		0100000	shamt	rs1	101	rd	0010011		0	1	1	0	1	0111	0	0	00	0	01	0	000	0	0	0
TYPE_I_JALR	JALR		imm[11:0]		rs1	000	rd	1100111		Branch	1	1	0	1	0000	1	0	00	0	10	0	000	0	0	0
TYPE_I_LOAD	LB		imm[11:0]		rs1	000	rd	0000011		0	1	1	0	1	0000	1	0	01	0	00	0	000	0	0	0
	LH		imm[11:0]		rs1	001	rd	0000011		0	1	1	0	1	0000	1	0	10	0	00	0	000	0	0	0
	LW		imm[11:0]		rs1	010	rd	0000011		0	1	1	0	1	0000	1	0	11	0	00	0	000	0	0	0
	LBU		imm[11:0]		rs1	100	rd	0000011		0	1	1	0	1	0000	1	0	01	1	00	0	000	0	0	0
	LHU		imm[11:0]		rs1	101	rd	0000011		0	1	1	0	1	0000	1	0	10	1	00	0	000	0	0	0
TYPE_S	SB		imm[11:5]	rs2	rs1	000	imm[4:0]	0100011		0	1	0	0	1	0000	0	1	01	0	00	0	000	0	0	0
	SH		imm[11:5]	rs2	rs1	001	imm[4:0]	0100011		0	1	0	0	1	0000	0	1	10	0	00	0	000	0	0	0
	SW		imm[11:5]	rs2	rs1	010	imm[4:0]	0100011		0	1	0	0	1	0000	0	1	11	0	00	0	000	0	0	0
TYPE_SB	BEQ		imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011		Branch	1	0	1	1	0001	0	0	00	0	00	0	100	1	0	0
	BNE		imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011		Branch	1	0	1	1	0001	0	0	00	0	00	0	011	0	1	1
	BLT		imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011		Branch	1	0	1	1	0011	0	0	00	0	00	0	010	0	1	0

	BGE		imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011		Branch	1	0	1	1	0011	0	0	00	0	00	0	101	1	0	1
	BLTU		imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011		Branch	1	0	1	1	0100	0	0	00	0	00	1	010	0	1	0
	BGEU		imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011		Branch	1	0	1	1	0100	0	0	00	0	00	1	101	1	0	1

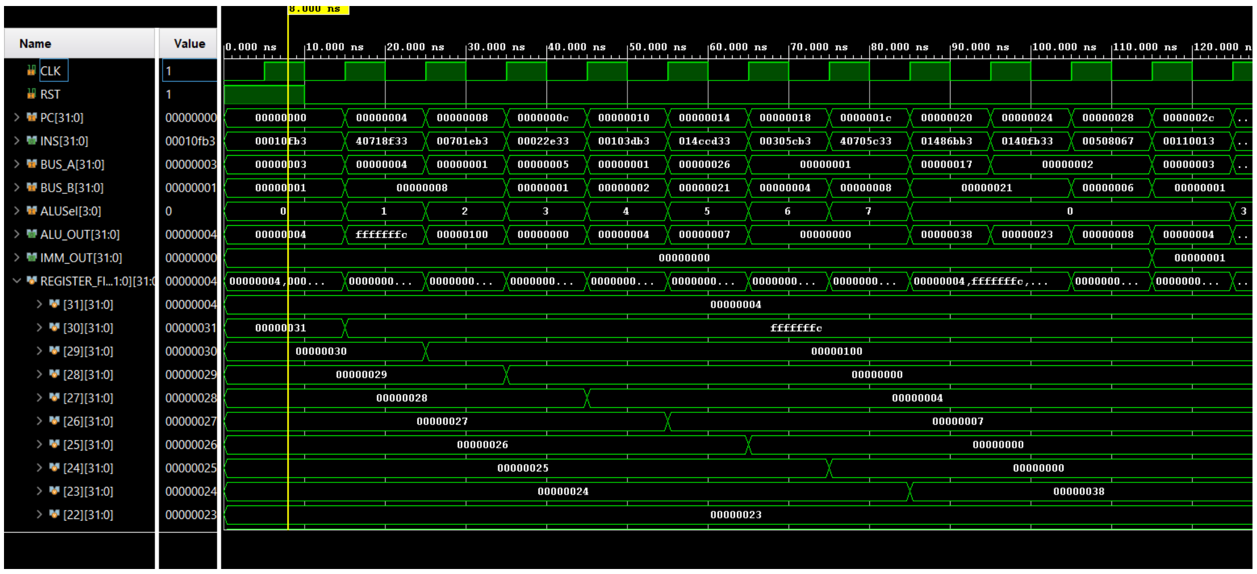
Verilog Implementation

RTL View of the complete processor

Implementation Results of R and I Type Computational instructions

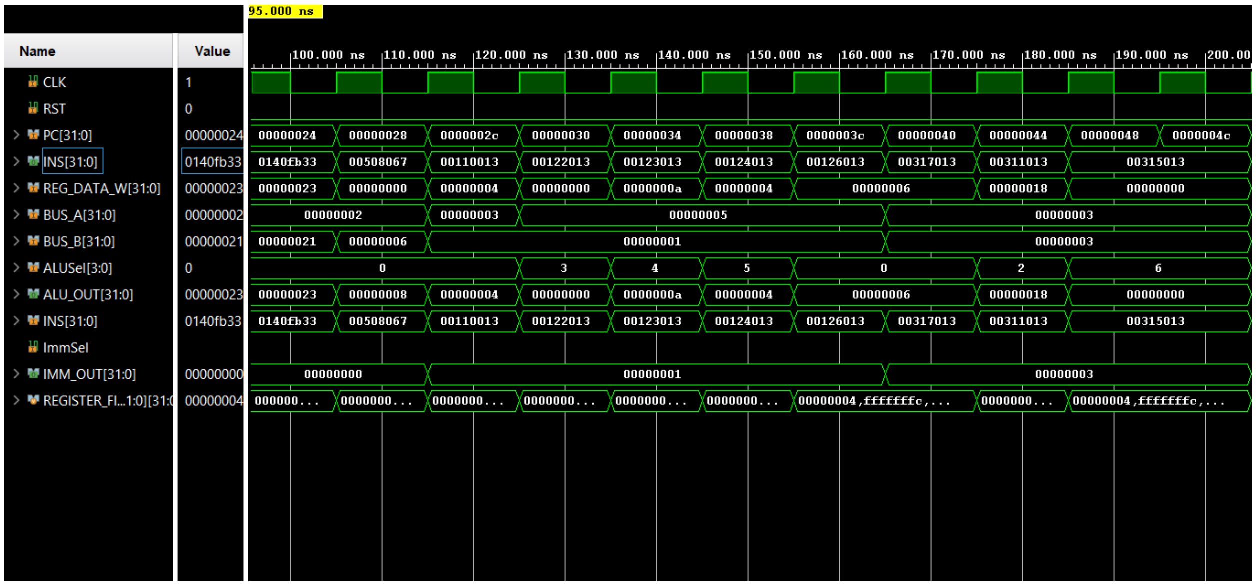
R Type

```
assign INS_MEM[0]   = 32'b0000000_00000_00010_000_11111_0110011; // reg0,reg4,reg31,add
assign INS_MEM[1]   = 32'b0100000_00111_00011_000_11110_0110011; // reg7,reg3,reg30,sub
assign INS_MEM[2]   = 32'b0000000_00111_00000_001_11101_0110011; // reg7,reg0,reg29, SLL
assign INS_MEM[3]   = 32'b0000000_00000_00100_010_11100_0110011; // or $s0,reg28, SLT
assign INS_MEM[4]   = 32'b0000000_00001_00000_011_11011_0110011; // sw $s0,reg27,SLTU
assign INS_MEM[5]   = 32'b0000000_10100_11001_100_11010_0110011; // sw $t0,reg26 ,XOR
assign INS_MEM[6]   = 32'b0000000_00011_00000_101_11001_0110011; // add $s1,reg25, SRL
assign INS_MEM[7]   = 32'b0100000_00111_00000_101_11000_0110011; // sub $s2,reg24, SRA
assign INS_MEM[8]   = 32'b0000000_10100_10000_110_10111_0110011; // beq $s1,reg23, OR
assign INS_MEM[9]   = 32'b0000000_10100_00001_111_10110_0110011; // lw $s1,reg22,AND
```

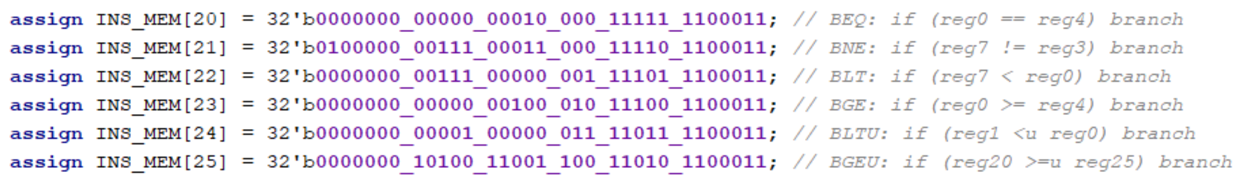


I Type Computational

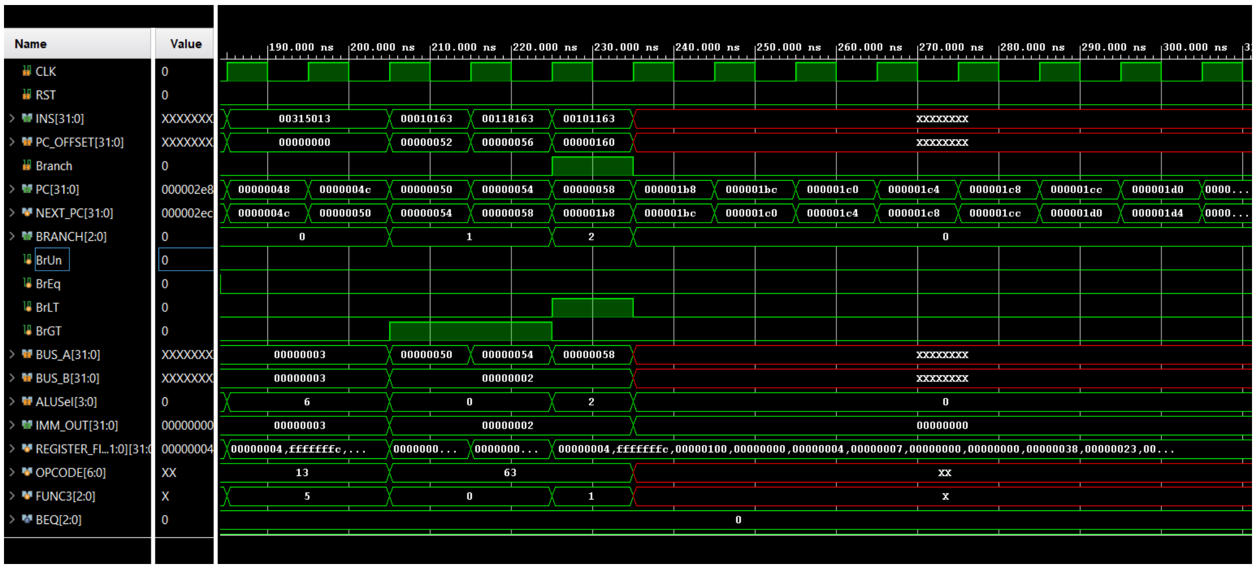
```
assign INS_MEM[11]  = 32'b000000000001_00010_000_00000_0010011; // ,addi
assign INS_MEM[12]  = 32'b000000000001_00100_010_00000_0010011; // ,SLTi
assign INS_MEM[13]  = 32'b000000000001_00100_011_00000_0010011; // ,SLTiu
assign INS_MEM[14]  = 32'b000000000001_00100_100_00000_0010011; // ,XORi
assign INS_MEM[15]  = 32'b000000000001_00100_110_00000_0010011; // ,ORi
assign INS_MEM[16]  = 32'b000000000011_00010_111_00000_0010011; // ,ANDi
assign INS_MEM[17]  = 32'b0000000_00011_00010_001_00000_0010011; // ,SLLi
assign INS_MEM[18]  = 32'b0000000_00011_00010_101_00000_0010011; // ,SRLi
assign INS_MEM[19]  = 32'b0000000_00011_00010_101_00000_0010011; // ,SRAi
```



Implementation Results SB Type Instructions



```
assign INS_MEM[20] = 32'b0000000_00000_00010_000_11111_1100011; // BEQ: if (reg0 == reg4) branch
assign INS_MEM[21] = 32'b0100000_00111_00011_000_11110_1100011; // BNE: if (reg7 != reg3) branch
assign INS_MEM[22] = 32'b0000000_00111_00000_001_11101_1100011; // BLT: if (reg7 < reg0) branch
assign INS_MEM[23] = 32'b0000000_00000_00100_010_11100_1100011; // BGE: if (reg0 >= reg4) branch
assign INS_MEM[24] = 32'b0000000_00001_00000_011_11011_1100011; // BLTU: if (reg1 <u reg0) branch
assign INS_MEM[25] = 32'b0000000_10100_11001_100_11010_1100011; // BGEU: if (reg20 >=u reg25) branch
```



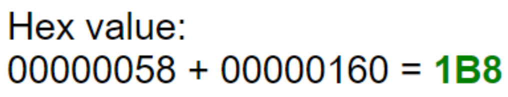
For instruction 32’h00101163 branch has happened

BRANCH = 3’b010 & BrUn=0

Branch type = BLT

PC = 00000058

PC_OFFSET = 00000160



Hex value:

00000058 + 00000160 = **1B8**

(**But for address 32’h000001B8,instruction memory is empty, that’s the reason for the unknown value(red) in the above.)

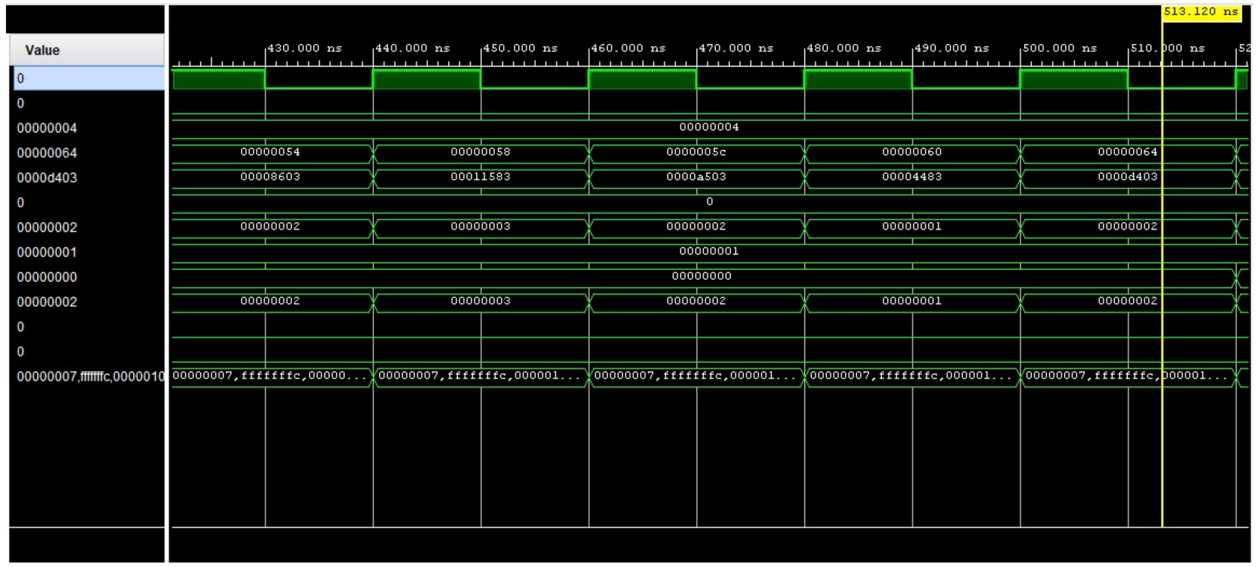
Implementation Results I and S Type Memory Access Instructions

TYPE_I_LOAD	LB		imm[11:0]	rs1	000	rd	0000011

	LH		imm[11:0]		rs1	001	rd	0000011
	LW		imm[11:0]		rs1	010	rd	0000011
	LBU		imm[11:0]		rs1	100	rd	0000011
	LHU		imm[11:0]		rs1	101	rd	0000011
TYPE_S	SB		imm[11:5]	rs2	rs1	000	imm[4:0]	0100011
	SH		imm[11:5]	rs2	rs1	001	imm[4:0]	0100011
	SW		imm[11:5]	rs2	rs1	010	imm[4:0]	0100011

Load Instructions

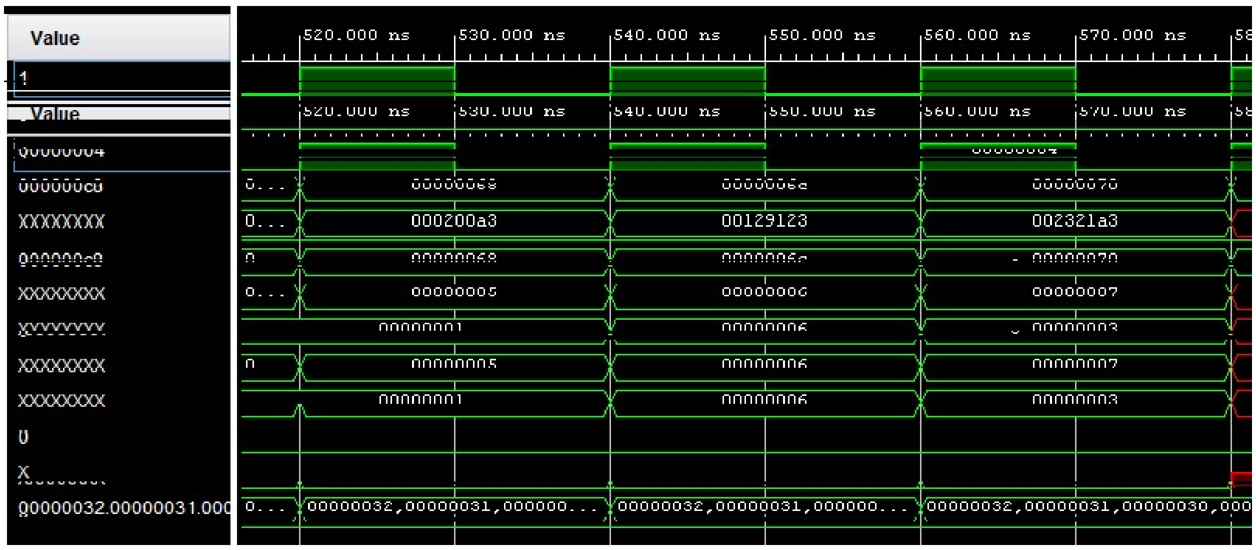
```
assign INS_MEM[32] = 32'b000000000000_00001_000_01100_0000011; // ,LB
assign INS_MEM[33] = 32'b000000000000_00010_001_01011_0000011; // ,LH
assign INS_MEM[34] = 32'b000000000000_00001_010_01010_0000011; // ,LW
assign INS_MEM[35] = 32'b000000000000_00000_100_01001_0000011; // ,LBU
assign INS_MEM[36] = 32'b000000000000_00001_101_01000_0000011; // ,LHU
```



```
Value
0
0
00000004
00000064
0000d403
0
00000002
00000001
00000000
00000002
0
0
00000007,fffffffc,0000010
```

Store Instructions

```
assign INS_MEM[37] = 32'b0000000_00000_00100_000_00001_0100011; //SB
assign INS_MEM[38] = 32'b0000000_00001_00101_001_00010_0100011; //SH
assign INS_MEM[39] = 32'b0000000_00010_00110_010_00011_0100011; //SW
```



Synthesis Report

```
Detailed RTL Component Info :
+---Adders :
	   2 Input     32 Bit        Adders := 1
	   3 Input     32 Bit        Adders := 1
+---XORs :
	   2 Input      32 Bit          XORs := 1
+---Registers :
	                32 Bit    Registers := 64
+---Muxes :
	   2 Input     32 Bit         Muxes := 6
	   5 Input     32 Bit         Muxes := 1
	   5 Input     12 Bit         Muxes := 1
	   9 Input      3 Bit         Muxes := 1
	   8 Input      3 Bit         Muxes := 1
	   2 Input      1 Bit         Muxes := 43
```

Resource utilization

```
Copyright 1986-2022 Xilinx, Inc. All Rights Reserved. Copyright 2022-2023 Advanced Micro Devices, Inc. All Rights
Reserved.
---------------------------------------------------------------------------------------------------------------------------------------------
| Tool Version : Vivado v.2023.1 (win64) Build 3865809 Sun May  7 15:05:29 MDT 2023
| Date         : Mon Oct 16 23:12:54 2023
| Host         : LAPTOP-P7TAB0BF running 64-bit major release  (build 9200)
| Command      : report_utilization -file TESTBENCH_utilization_synth.rpt -pb TESTBENCH_utilization_synth.pb
| Design       : TESTBENCH
| Device       : xc7z010iclg225-1L
| Speed File   : -1L
| Design State : Synthesized
---------------------------------------------------------------------------------------------------------------------------------------------

Utilization Design Information

Table of Contents
-----------------
1. Slice Logic
1.1 Summary of Registers by Type
2. Slice Logic Distribution
3. Memory
4. DSP
5. IO and GT Specific
6. Clocking
7. Specific Feature
8. Primitives
9. Black Boxes
10. Instantiated Netlists

1. Slice Logic
--------------

+-------------------------+------+-------+------------+-----------+-------+
|        Site Type        | Used | Fixed | Prohibited | Available | Util% |
+-------------------------+------+-------+------------+-----------+-------+
| Slice LUTs              |    0 |     0 |          0 |     17600 |  0.00 |
|   LUT as Logic          |    0 |     0 |          0 |     17600 |  0.00 |
|   LUT as Memory         |    0 |     0 |          0 |      6000 |  0.00 |
| Slice Registers         |    0 |     0 |          0 |     35200 |  0.00 |
|   Register as Flip Flop |    0 |     0 |          0 |     35200 |  0.00 |
|   Register as Latch     |    0 |     0 |          0 |     35200 |  0.00 |
| F7 Muxes                |    0 |     0 |          0 |      8800 |  0.00 |
| F8 Muxes                |    0 |     0 |          0 |      4400 |  0.00 |
+-------------------------+------+-------+------------+-----------+-------+
* Warning! LUT value is adjusted to account for LUT combining.


1.1 Summary of Registers by Type
--------------------------------

+-------+--------------+-------------+--------------+
| Total | Clock Enable | Synchronous | Asynchronous |
+-------+--------------+-------------+--------------+
| 0     |            _ |           - |            - |
| 0     |            _ |           - |          Set |
| 0     |            _ |           - |        Reset |
| 0     |            _ |         Set |            - |
| 0     |            _ |       Reset |            - |
| 0     |          Yes |           - |            - |
| 0     |          Yes |           - |          Set |
| 0     |          Yes |           - |        Reset |
| 0     |          Yes |         Set |            - |
| 0     |          Yes |       Reset |            - |
+-------+--------------+-------------+--------------+


2. Slice Logic Distribution
---------------------------

+--------------------------------------------+------+-------+------------+-----------+-------+
|                  Site Type                 | Used | Fixed | Prohibited | Available | Util% |
+--------------------------------------------+------+-------+------------+-----------+-------+
| Slice                                      |    0 |     0 |          0 |      4400 |  0.00 |
|   SLICEL                                   |    0 |     0 |            |           |       |
|   SLICEM                                   |    0 |     0 |            |           |       |
| LUT as Logic                               |    0 |     0 |          0 |     17600 |  0.00 |
| LUT as Memory                              |    0 |     0 |          0 |      6000 |  0.00 |
|   LUT as Distributed RAM                   |    0 |     0 |            |           |       |
|   LUT as Shift Register                    |    0 |     0 |            |           |       |
| Slice Registers                            |    0 |     0 |          0 |     35200 |  0.00 |
|   Register driven from within the Slice    |    0 |       |            |           |       |
|   Register driven from outside the Slice   |    0 |       |            |           |       |
| Unique Control Sets                    |    0 |       |          0 |      4400 |  0.00 |
+----------------------------------------+------+-------+------------+-----------+-------+
* * Note: Available Control Sets calculated as Slice * 1, Review the Control Sets Report for more information regarding control sets.


3. Memory
---------

+----------------+------+-------+------------+-----------+-------+
|    Site Type   | Used | Fixed | Prohibited | Available | Util% |
+----------------+------+-------+------------+-----------+-------+
| Block RAM Tile |    0 |     0 |          0 |        60 |  0.00 |
|   RAMB36/FIFO* |    0 |     0 |          0 |        60 |  0.00 |
|   RAMB18       |    0 |     0 |          0 |       120 |  0.00 |
+----------------+------+-------+------------+-----------+-------+
* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E1


4. DSP
------

+-----------+------+-------+------------+-----------+-------+
| Site Type | Used | Fixed | Prohibited | Available | Util% |
+-----------+------+-------+------------+-----------+-------+
| DSPs      |    0 |     0 |          0 |        80 |  0.00 |
+-----------+------+-------+------------+-----------+-------+


5. IO and GT Specific
---------------------

+-----------------------------+------+-------+------------+-----------+-------+
|          Site Type          | Used | Fixed | Prohibited | Available | Util% |
+-----------------------------+------+-------+------------+-----------+-------+
| Bonded IOB                  |    0 |     0 |          0 |        54 |  0.00 |
| Bonded IPADs                |    0 |     0 |          0 |         2 |  0.00 |
| Bonded IOPADs               |    0 |     0 |          0 |       130 |  0.00 |
| PHY_CONTROL                 |    0 |     0 |          0 |         2 |  0.00 |
| PHASER_REF                  |    0 |     0 |          0 |         2 |  0.00 |
| OUT_FIFO                    |    0 |     0 |          0 |         8 |  0.00 |
| IN_FIFO                     |    0 |     0 |          0 |         8 |  0.00 |
| IDELAYCTRL               |    0 |     0 |          0 |         2 |  0.00 |
| IBUFDS                   |    0 |     0 |          0 |        54 |  0.00 |
| PHASER_OUT/PHASER_OUT_PHY |    0 |     0 |          0 |         8 |  0.00 |
| PHASER_IN/PHASER_IN_PHY   |    0 |     0 |          0 |         8 |  0.00 |
| IDELAYE2/IDELAYE2_FINEDELAY |    0 |     0 |          0 |       100 |  0.00 |
| ILOGIC                   |    0 |     0 |          0 |        54 |  0.00 |
| OLOGIC                   |    0 |     0 |          0 |        54 |  0.00 |
+-----------------------------+------+-------+------------+-----------+-------+


6. Clocking
-----------

+------------+------+-------+------------+-----------+-------+
|  Site Type | Used | Fixed | Prohibited | Available | Util% |
+------------+------+-------+------------+-----------+-------+
| BUFGCTRL   |    0 |     0 |          0 |        32 |  0.00 |
| BUFIO      |    0 |     0 |          0 |         8 |  0.00 |
| MMCME2_ADV |    0 |     0 |          0 |         2 |  0.00 |
| PLLE2_ADV  |    0 |     0 |          0 |         2 |  0.00 |
| BUFMRCE    |    0 |     0 |          0 |         4 |  0.00 |
| BUFHCE     |    0 |     0 |          0 |        48 |  0.00 |
| BUFR       |    0 |     0 |          0 |         8 |  0.00 |
+------------+------+-------+------------+-----------+-------+


7. Specific Feature
-------------------

+-------------+------+-------+------------+-----------+-------+
|  Site Type  | Used | Fixed | Prohibited | Available | Util% |
+-------------+------+-------+------------+-----------+-------+
| BSCANE2     |    0 |     0 |          0 |         4 |  0.00 |
| CAPTUREE2   |    0 |     0 |          0 |         1 |  0.00 |
| DNA_PORT    |    0 |     0 |          0 |         1 |  0.00 |
| EFUSE_USR   |    0 |     0 |          0 |         1 |  0.00 |
| FRAME_ECCE2 |    0 |     0 |          0 |         1 |  0.00 |
| ICAPE2      |    0 |     0 |          0 |         2 |  0.00 |
| STARTUPE2   |    0 |     0 |          0 |         1 |  0.00 |
| XADC        |    0 |     0 |          0 |         1 |  0.00 |
+-------------+------+-------+------------+-----------+-------+
8. Primitives
-------------

+----------+------+---------------------+
| Ref Name | Used | Functional Category |
+----------+------+---------------------+


9. Black Boxes
--------------

+----------+------+
| Ref Name | Used |
+----------+------+


10. Instantiated Netlists
-------------------------

+----------+------+
| Ref Name | Used |
+----------+------+
```

**File is attached herewith