KGP-RISC Documentation

Group No - 44
Abhilash Datta (19CS3001)
Rohit Raj (19CS10049)

Instruction Set Architecture

Class	Instruction	Usage	Meaning
Arithmetic	Add	add rs,rt	$rs \leftarrow (rs) + (rt)$
	Comp	comp rs,rt	$rs \leftarrow$ 2's Complement (rs)
	Add immediate	addi rs,imm	$rs \leftarrow (rs) + imm$
	Complement Immediate	compi rs,imm	$rs \leftarrow$ 2's Complement (imm)
Logic	AND	and rs,rt	$rs \leftarrow (rs) \wedge (rt)$
	XOR	xor rs,rt	$rs \leftarrow (rs) \oplus (rt)$
Shift	Shift left logical	shll rs, sh	$rs \leftarrow (rs)$ left-shifted by sh
	Shift right logical	shrl rs, sh	$rs \leftarrow (rs)$ right-shifted by sh
	Shift left logical variable	shllv rs, rt	$rs \leftarrow (rs)$ left-shifted by (rt)
	Shift right logical	shrl rs, rt	$rs \leftarrow (rs)$ right-shifted by (rt)
	Shift right arithmetic	shra rs, sh	$rs \leftarrow (rs)$ arithmetic right-shifted by sh
	Shift right arithmetic variable	shrav rs, rt	$rs \leftarrow (rs)$ right-shifted by (rt)
Memory	Load Word	lw rt,imm(rs)	$rt \leftarrow mem[(rs) + imm]$
	Store Word	sw rt,imm,(rs)	$mem[(rs) + imm] \leftarrow (rt)$
Branch	Unconditional branch	b L	goto L
	Branch Register	br rs	goto (rs)
	Branch on less than 0	bltz rs,L	if$(rs) < 0$ then goto L
	Branch on flag zero	bz rs,L	if $(rs) = 0$ then goto L
	Branch on flag not zero	bnz rs,L	if$(rs) \neq 0$ then goto L
	Branch and link	bl L	goto L; 31 $\leftarrow$ (PC)+4
	Branch on Carry	bcy L	goto L if Carry = 1
	Branch on No Carry	bncy L	goto L if Carry = 0

Instruction format and Encoding

Format of instructions

Total number of possible Opcodes = 8
Number of Opcodes used = 6

1. **Instructions** - Add, Comp, AND, XOR, shllv, shrlv, shrav
   Opcode - 0
   Encoding - 000
   Format - R-Format

Opcode(3)	rs(5)	rt(5)	Encoding(4)	Don't Care(15)

   Number of functions available - 16
   Number of functions currently used - 6

2. **Instructions** - Addi, Compi, shll, shrl, shra
   Opcode - 1
   Encoding - 001
   Format - L-Format

Opcode(3)	rs(5)	Don't Care(5)	Encoding(3)	Imm(16)

   Number of functions available - 8
   Number of functions currently used - 4

3. **Instructions** - lw, sw
   Opcode - 2
   Encoding - 010
   Format - Load/Store

Opcode(3)	rs(5)	rt(5)	Encoding(3)	Imm(16)

   Number of functions available - 8
   Number of functions currently used - 2

4. **Instructions** - b, bl, bcy, bncy
   Opcode - 3
   Encoding - 011
   Format - Branching

Opcode(3)	L(26)	Encoding(3)

Number of functions available - 8
Number of functions currently used - 4

5. **Instructions** - bltz, bz, bnz
   Opcode - 4
   Encoding - 100
   Format - Branching

Opcode(3)	rs(5)	Encoding(2)	L(22)

Number of functions available - 4
Number of functions currently used - 3

6. **Instructions** - br
   Opcode - 5
   Encoding - 101
   Format - Branching

Opcode(3)	rs(5)	Don’t Care(5)	Encoding(3)	Don’t Care(16)

Number of functions available - 8
Number of functions currently used - 1

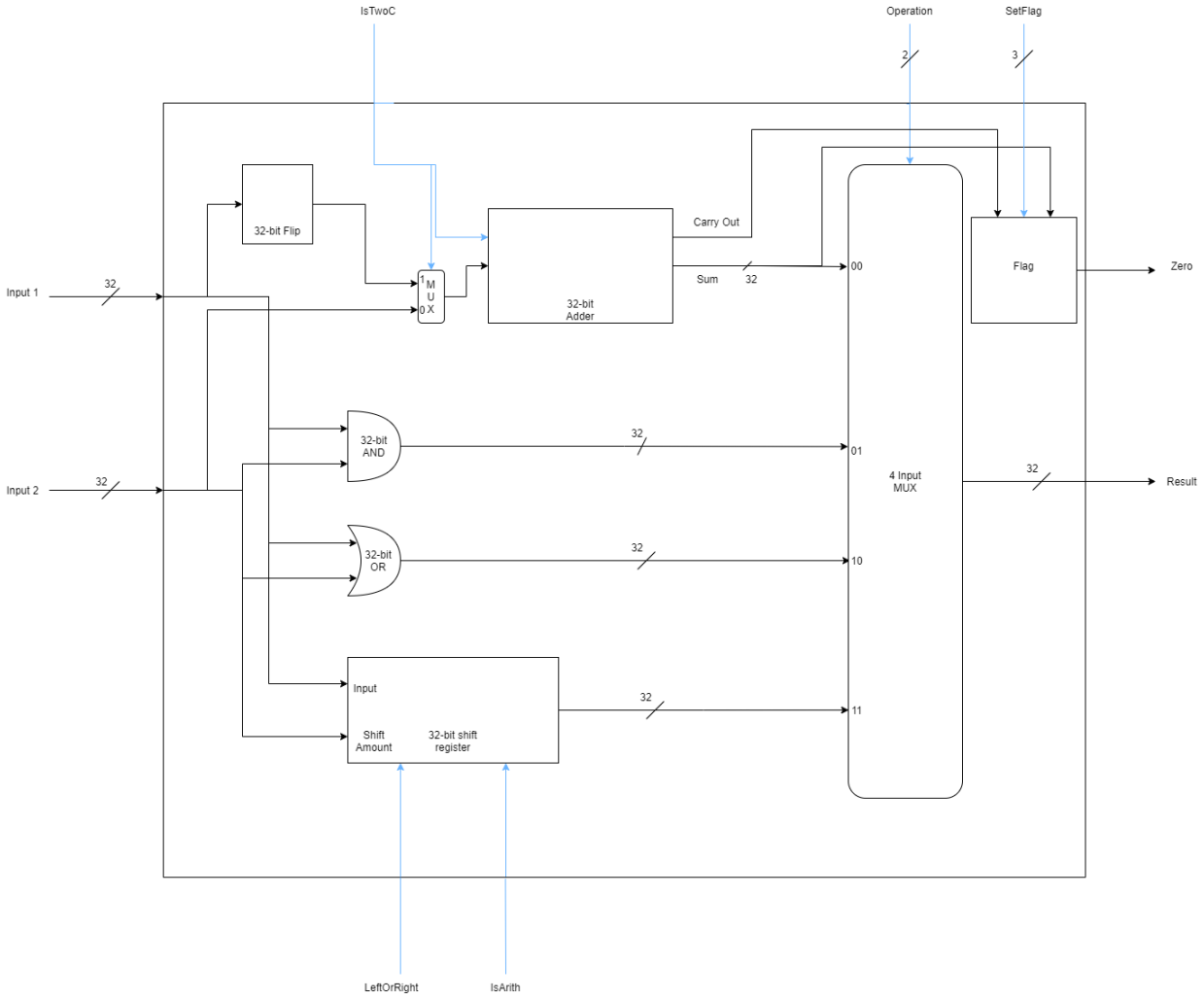
Encoding of operations -

Opcode	Operations	Function Code	Encoding
000	Add	0	0000
	Comp	1	0001
	AND	2	0010
	XOR	3	0011
	shllv	4	0100
	shrlv	5	0101
	shrav	6	0110
001	Addi	0	000
	Compi	1	001
	shll	2	010
	shrl	3	011
	shra	4	100
010	lw	0	000
	sw	1	001
011	b	0	000
	bl	1	001
	bcy	2	010
	bncy	3	011

100	bltz	0	00
	bz	1	01
	bnz	2	10

101	br	0	000

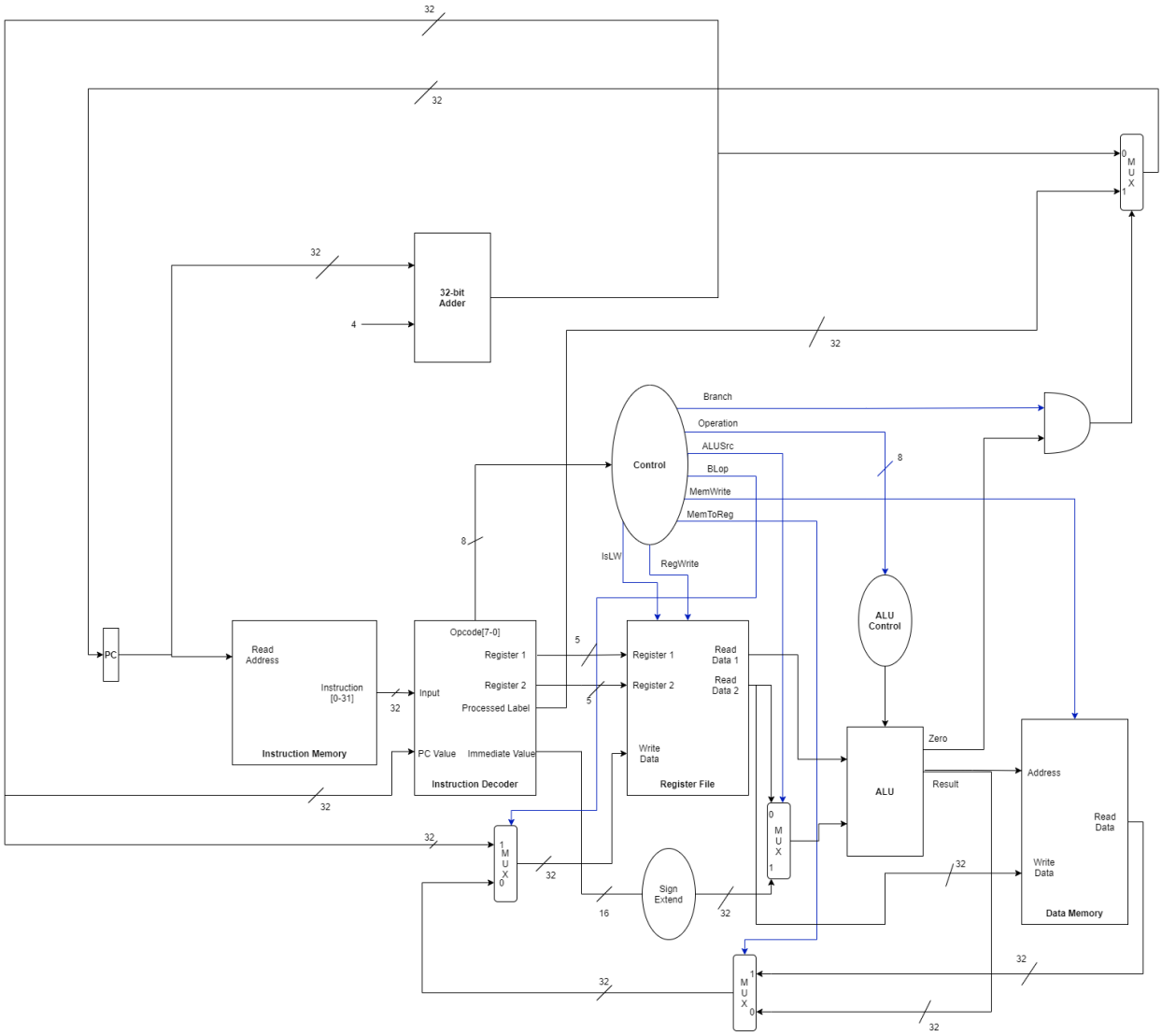
ALU DESIGN



Modules Used -

- 32-bit shift register - This module is for carrying out 32-bit shifting operations in both directions. It also supports arithmetic shifting triggered by the IsArith line.
- 4-Input MUX - Multiplexer which has 4 inputs and a single output.
- 32-bit Flip - Module to flip all 32 bits of a number.
- 32-bit Adder - Module to add two 32 bit numbers.
- Flag - Module to set a flag for branching based on carry value, result, and type of branching operation.
- 32-bit AND - Module to find logical AND of two 32 bit numbers.
- 32-bit OR - Module to find logical OR of two 32 bit numbers.

Data Path Design



ALU Control Output

Output Lines - IsArith(1 bit), IsTwoC (1 bit), LeftOrRight (1 bit), Operation (2 bits), SetFlag (3 bit)

Instruction	IsArith	isTwoC	LeftOrRight	Operation	SetFlag
Addition	0	0	0	00	111
Complement	0	1	0	00	000
And	0	0	0	01	000
Or	0	0	0	10	000
Left Shift	0	0	1	11	000
Right Shift	0	0	0	11	000
Right Shift Arit.	1	0	0	11	000
sw, lw	0	0	0	00	000
b, br, bl	0	0	0	00	001
bltz	0	0	0	00	010
bz	0	0	0	00	011
bnz	0	0	0	00	100
bcy	0	0	0	0	101
bncy	0	0	0	0	110

Control Input

Input Lines - Opcode (8 bit)

Instruction	Opcode
Add	00000000
Comp	00000001
AND	00000010
XOR	00000011
shllv	00000100
shrlv	00000101
shrav	00000110
Addi	00100000
Compi	00100001
shll	00100010
shrl	00100011
shra	00100100
lw	01000000
sw	01000001
b	01100000
bl	01100001
bcy	01100010
bncy	01100011

bltz	10000000
bz	10000001
bnz	10000010

br	10100000

Control Output

Output Lines - Branch(1 bit), ALUOp(8 bit), ALUSrc(1 bit), BLop(1 bit), Memwrite(1 bit), MemToReg(1 bit), RegWrite(1 bit), IsB(1 bit), IsLW(1 bit)

Instruction	Branch	ALUOp	ALUSrc	BLop	MemWrite	MemToReg	IsB	IsLW	RegWrite
Add	0	00000111	0	0	0	0	0	0	1
Add Imm	0	00000111	1	0	0	0	0	0	1
Complement	0	01000000	0	0	0	0	0	0	1
Comp Imm	0	01000000	1	0	0	0	0	0	1
And	0	00001000	0	0	0	0	0	0	1
Or	0	00010000	0	0	0	0	0	0	1
shll	0	00111000	1	0	0	0	0	0	1
shrl	0	00011000	1	0	0	0	0	0	1
shllv	0	00111000	0	0	0	0	0	0	1
shrlv	0	00011000	0	0	0	0	0	0	1
shra	0	10011000	1	0	0	0	0	0	1
shrav	0	10011000	0	0	0	0	0	0	1

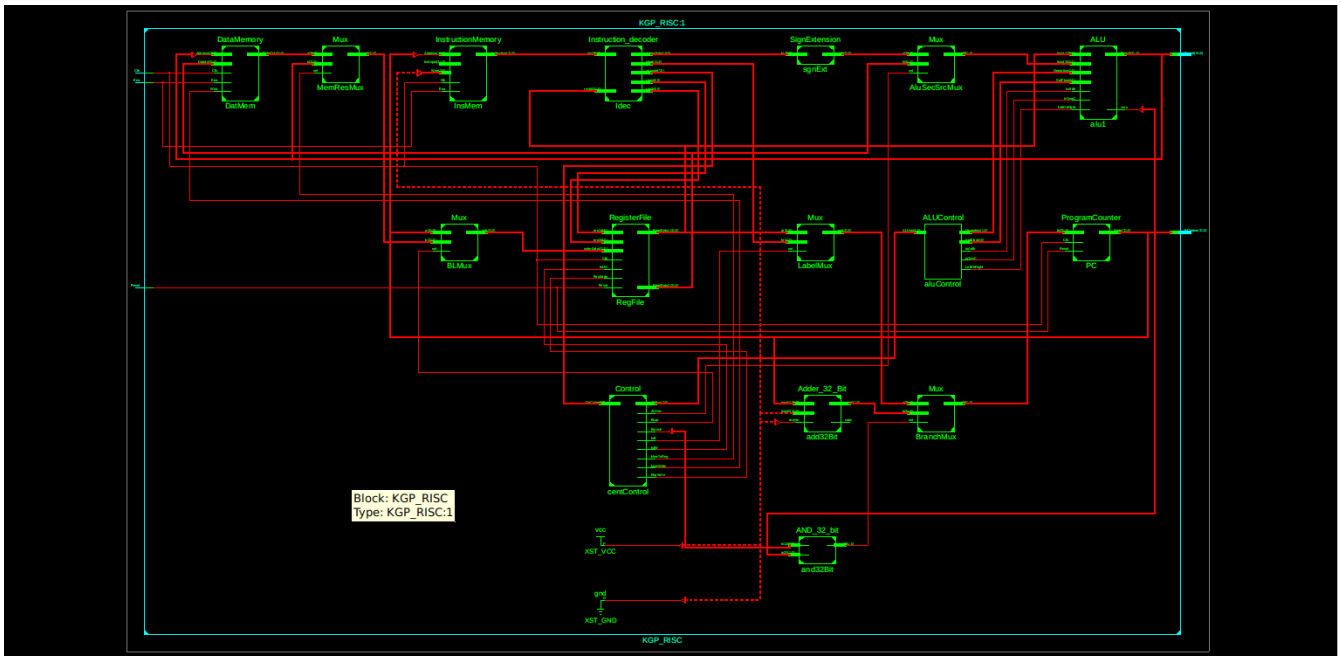
sw	0	00000000	1	0	1	0	0	0	0
lw	0	00000000	1	0	0	1	0	1	1

b	1	00000001	0	0	0	0	0	0	0
br	1	00000001	0	0	0	0	1	0	0
bltz	1	00000010	0	0	0	0	0	0	0
bz	1	00000011	0	0	0	0	0	0	0
bnz	1	00000100	0	0	0	0	0	0	0
bl	1	00000001	0	1	0	0	0	0	1
bcy	1	00000101	0	0	0	0	0	0	0
bncy	1	00000110	0	0	0	0	0	0	0

IMPLEMENTATION && SIMULATION

The above design has been implemented using Verilog in Xilinx and has been simulated on code to calculate the GCD of 2 numbers using an iterative algorithm.

Design implemented -



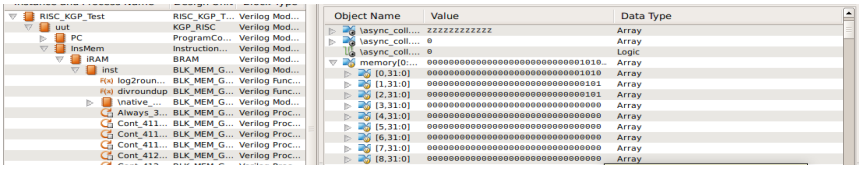
Code Simulated -

```
main:
    addi $t1, 10
    addi $t2, 5
    sw $t1, 0($zero)
    sw $t2, 1($zero)
    lw $t1, 0($zero)
    lw $t2, 1($zero)
Loop:
    comp $t3, $zero
    add $t3, $t1
    comp $t4, $zero
    comp $t4, $t2
    add $t3, $t4
    bz $t3,  EXIT
    bltz $t3, Loop1
    comp $t1, $zero
    add $t1, $t3
    b Loop
Loop1:
    comp $t2, $t3
    b Loop
EXIT:
    sw $t2, 2($zero)
    add $zero, $zero
```

```
00101001000000000000000000001010
00101010000000000000000000000101
01001001000000010000000000000000
01001010000000010000000000000001
01001001000000000000000000000000
01001010000000000000000000000001
00001011000000001000000000000000
00001011010010000000000000000000
00001100000000001000000000000000
00001100010100001000000000000000
00001011011000000000000000000000
10001011010000000000000000010110
00000000000000000000000000000000
10001011000000000000000000010011
00000000000000000000000000000000
00001001000000001000000000000000
00001001010111000000000000000000
01100000000000000000000000110000
00000000000000000000000000000000
00001010010110001000000000000000
01100000000000000000000000110000
00000000000000000000000000000000
01001010000000010000000000000010
00000000000000000000000000000000
```

Here, on the left, we showed the code written in our Instruction set Architecture (ISA), and on the right corresponding binary codes are shown which are encoded according to the proper format.

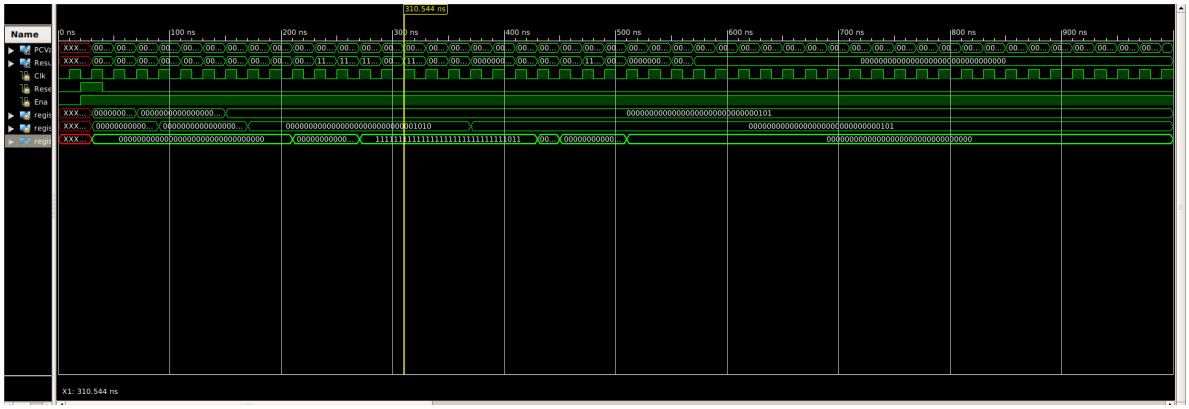
Results Generated -



This shows the values stored in data memory, where-

- 0th index has the first operand
- 1st index has the second operand
- 2nd index has the final result

Waveform Generated -



Here last three waveforms are of register $t1, $t2, $t3 which are use by our code.

Synthesis Report -

```
Timing Summary:
---------------
Speed Grade: -1

   Minimum period: 9.332ns (Maximum Frequency: 107.156MHz)
   Minimum input arrival time before clock: 1.173ns
   Maximum output required time after clock: 40.821ns
   Maximum combinational path delay: No path found

Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'Clk'
  Clock period: 9.332ns (frequency: 107.156MHz)
  Total number of paths / destination ports: 15285829 / 1088
-------------------------------------------------------------------------
Delay:               9.332ns (Levels of Logic = 12)
  Source:            RegFile/registers_0_834 (FF)
  Destination:       PC/State_30 (FF)
  Source Clock:      Clk rising
  Destination Clock: Clk rising

  Data Path: RegFile/registers_0_834 to PC/State_30
                            Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDRE:C->Q             3   0.478   0.790  RegFile/registers_0_834 (RegFile/registers_0_834)
     LUT6:I2->O            1   0.124   0.776  RegFile/Mmux_ReadData1_867 (RegFile/Mmux_ReadData1_867)
     LUT6:I2->O            1   0.124   0.000  RegFile/Mmux_ReadData1_322 (RegFile/Mmux_ReadData1_322)
     MUXF7:I1->O          20   0.368   0.841  RegFile/Mmux_ReadData1_2_f7_21 (ReadData1<2>)
     LUT6:I3->O            1   0.124   0.776  alu1/addr/add1/lcu/C<0>2_SW1 (N95)
     LUT6:I2->O            2   0.124   0.427  alu1/addr/add1/lcu/C<0>2 (alu1/addr/add1/lcu/C<0>1)
     LUT5:I4->O            4   0.124   0.441  alu1/addr/add1/lcu/C<0>3 (alu1/addr/add1/C<1>)
     LUT5:I4->O           14   0.124   0.506  alu1/addr/add1/lcu/C<1><1>4 (alu1/addr/add1/C<2>)
     LUT6:I5->O            1   0.124   0.421  alu1/addr/add2/lcu/C<1><1>4_SW4 (N282)
     LUT6:I5->O            3   0.124   0.953  alu1/addr/add2/cla3/Mxor_sum<2>_xo<0>1 (alu1/adderOut<26>)
     LUT6:I0->O            9   0.124   0.589  alu1/fl/sum[31]_GND_246_o_equal_5_o<31>6_SW0 (N263)
     LUT6:I4->O           26   0.124   0.572  and32Bit/out<0>2 (n0021<0>)
     LUT6:I5->O            1   0.124   0.000  BranchMux/Mmux_out121 (FinAddr<1>)
     FDR:D                     0.030          PC/State_1
    ----------------------------------------
    Total                      9.332ns (2.240ns logic, 7.092ns route)
                               (24.0% logic, 76.0% route)
```

This shows the Timing summary of the processor.

```
Device utilization summary:
---------------------------

Selected Device : 7a100tcsg324-1


Slice Logic Utilization:
 Number of Slice Registers:           1137  out of  126800     0%
 Number of Slice LUTs:                2605  out of  63400     4%
    Number used as Logic:             2605  out of  63400     4%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:   2608
   Number with an unused Flip Flop:    1471  out of   2608    56%
   Number with an unused LUT:             3  out of   2608     0%
   Number of fully used LUT-FF pairs:  1134  out of   2608    43%
   Number of unique control sets:        22

IO Utilization:
 Number of IOs:                         67
 Number of bonded IOBs:                 67  out of    210    31%

Specific Feature Utilization:
 Number of Block RAM/FIFO:               8  out of    135     5%
    Number using Block RAM only:         8
 Number of BUFG/BUFGCTRLs:               2  out of     32     6%

---------------------------
```

This shows FPGA resources used by the processor

How to simulate it?

1. Simulate the test bench.
2. In the ISE simulator, see the value stored in the 0th, 1st, and 2nd index of data memory. As mentioned above, they should be first operand, second operand, and result respectively.