CO224 Computer Architecture

Lab 5 Building a Simple Processor

Part 5 : Extended ISA

Report

Group 12 :

E/19/455 YASHAN W.V.

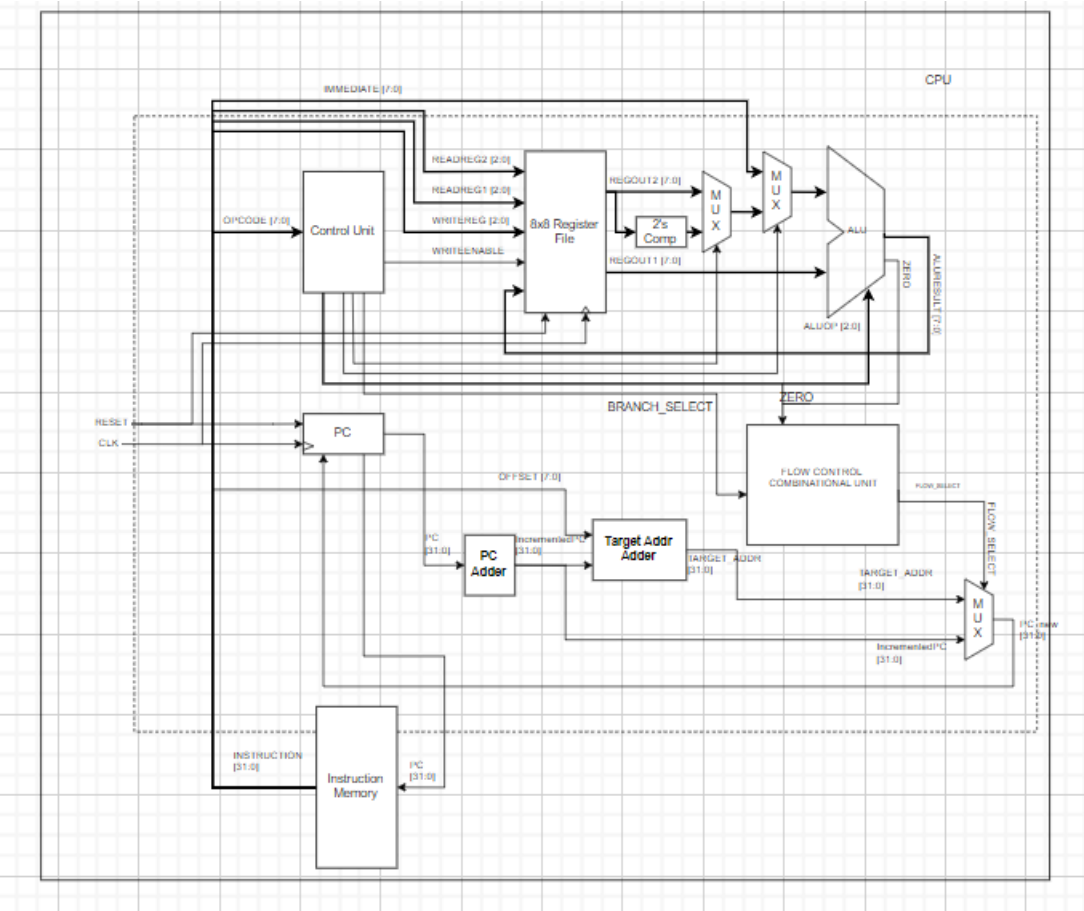
E/19/124 GUNASEKARA M.H.

The purpose of this report is to provide a comprehensive overview of the recent modifications made to the CPU hardware and the corresponding changes introduced in the assembler. These modifications were implemented to extend the instruction set of the CPU, incorporating 6 new instructions:

mult <r4> <r2> <r1>	: Multiplies the values in r1 and r2 and stores result in r4
sll <r4> <r1> <offset>	: Logically left shifts r1 by offset value and stores result in r4
srl <r4> <r1> <offset>	: Logically right shifts r1 by offset value and stores result in r4
sra <r4> <r1> <offset>	: Arithmetically right shifts r1 by offset value and stores result in r4
ror <r4> <r1> <offset>	: Rotates r1 right by offset value and stores result in r4
bne <offset> <r1> <r2>	: Branches based on offset if values in r1 and r2 are not equal

To facilitate these 6 new instructions, modification was made. And some additional functional units were implemented within the ALU of the CPU. These changes are reflected in the CPU block diagram and the ALU functions table given below.

CPU block diagram



ALU Functions

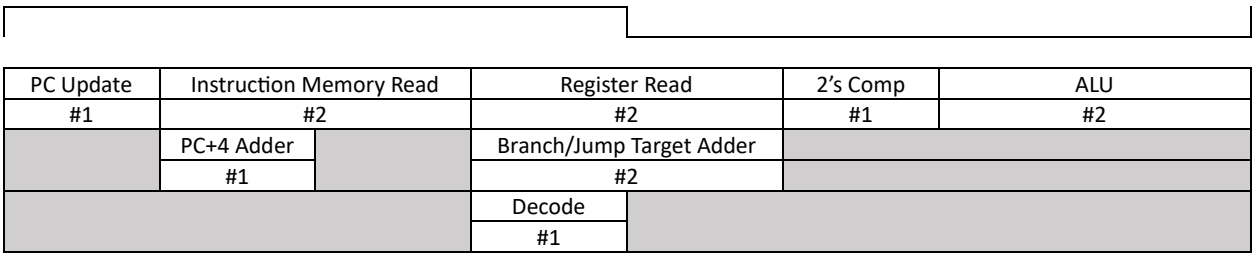
SELECT	Function	Description	Supported Instructions	Unit's Delay
000	FORWARD	DATA2→RESULT	loadi, mov	#1
001	ADD	DATA1+DATA2→RESULT	add, sub, beq, **bne**	#2
010	AND	DATA1&DATA2→RESULT	and	#1
011	OR	DATA1|DATA2→RESULT	or	#1
100	**MULT**	**DATA1*DATA2→RESULT**	**mult**	**#3**
101	**SHIFT_ROTATE**	**(Logical Shift) DATA1>>DATA2→RESULT Or (Arithmetic Shift) DATA1>>>DATA2→RESULT Or Rotate Right DATA1 by DATA2 times (Operation chosen depends on first two MSB bits of DATA2)**	**sll, srl, sra, ror**	**#2**

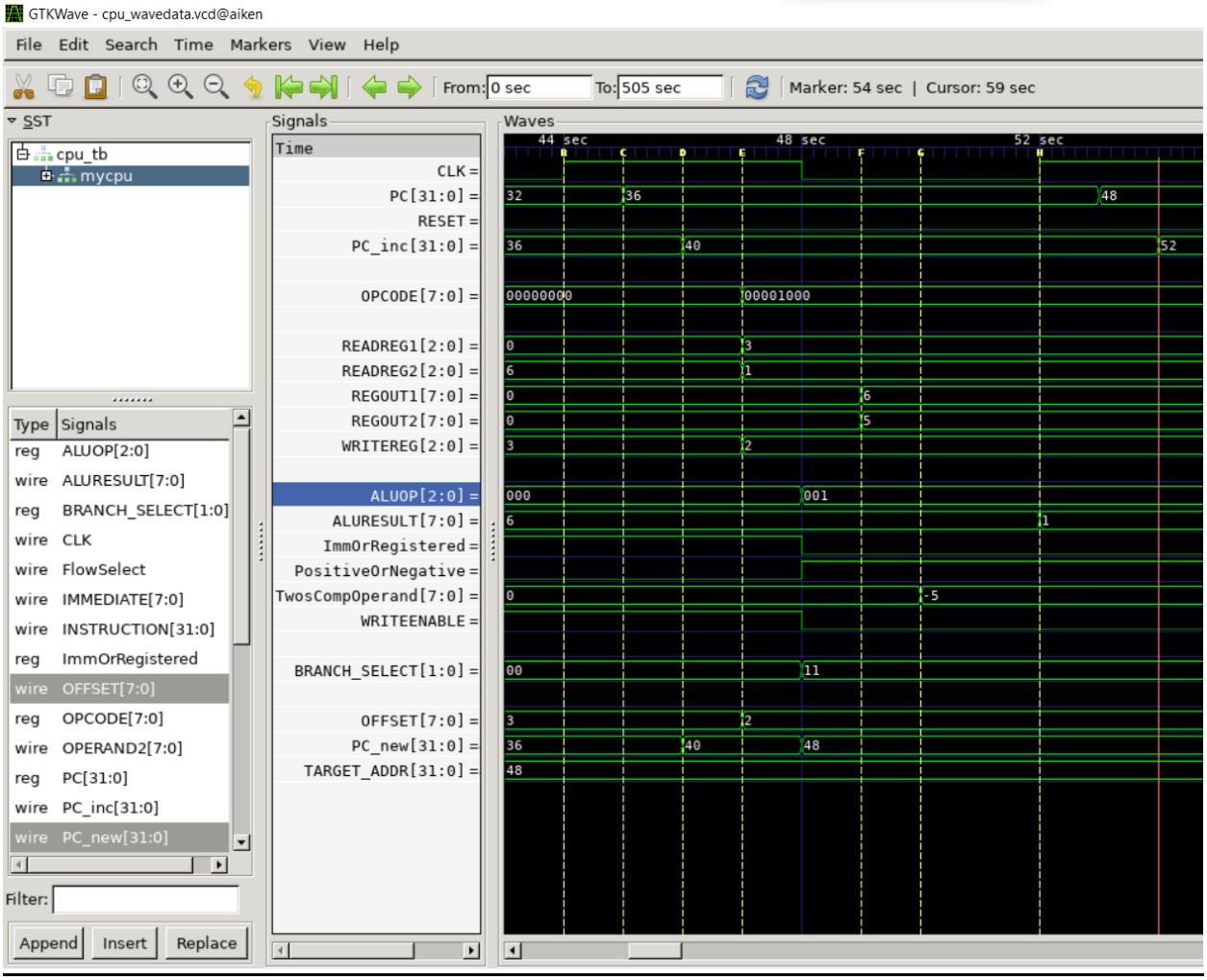
BNE Instruction

CPU's Flow Control Unit was modified to incorporate the branch condition check. The control unit was enhanced to compare the values in two registers (r1 and r2) and branch to a specified offset address if the values are not equal.

The modification involved adding a comparator circuit to the control unit that compares the values in the specified registers and generates the appropriate control signals to perform the branch operation.

BRANCH_SELECT	FLOW_SELECT	Instruction
00	0	
01	0	jump
10	1	beq
11	1	bne

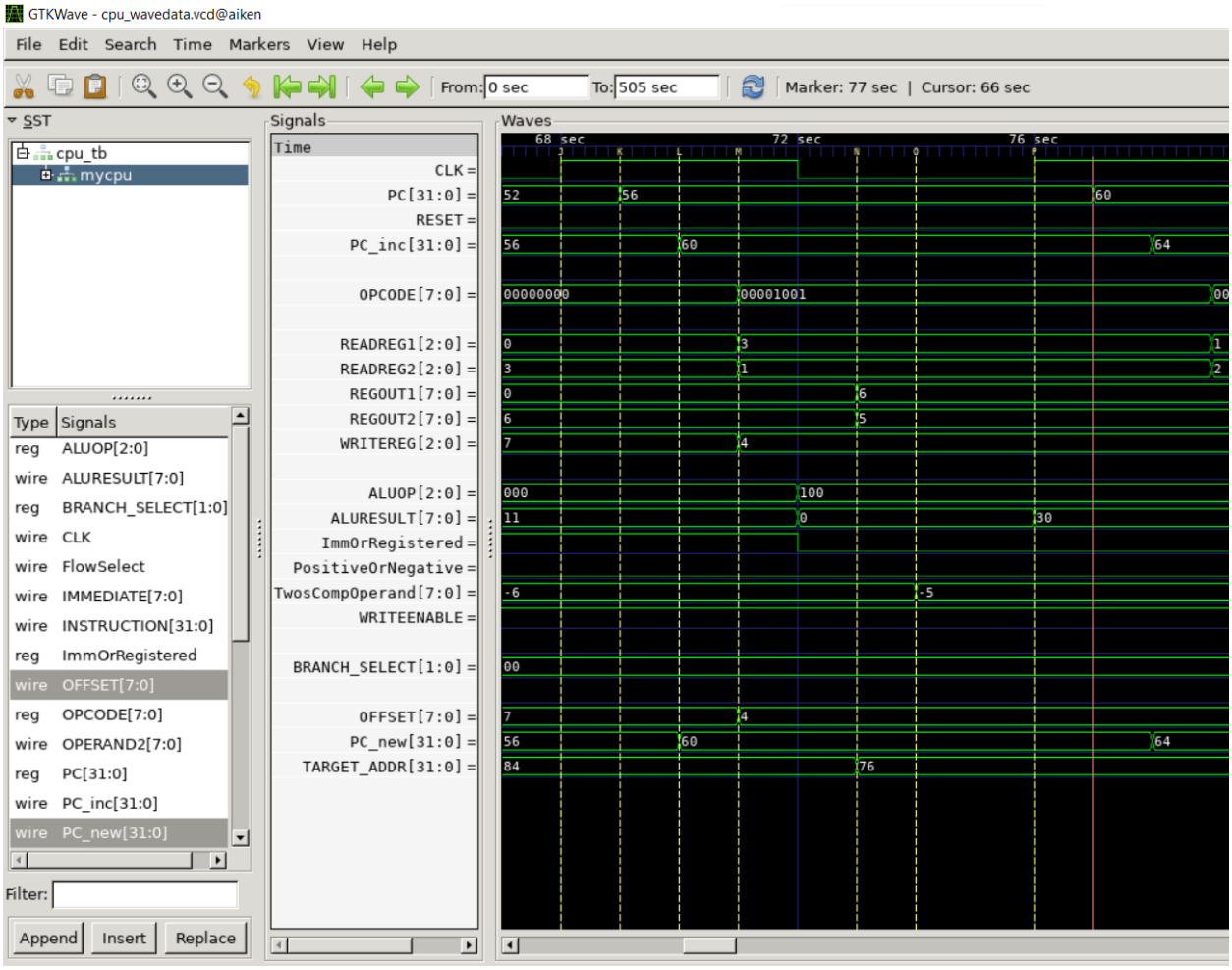



GTKWave - cpu_wavedata.vcd@aiken
File Edit Search Time Markers View Help
From: 0 sec
To: 505 sec
Marker: 54 sec | Cursor: 59 sec
SST
cpu_tb
mycpu
Type Signals
reg ALUOP[2:0]
wire ALURESULT[7:0]
reg BRANCH_SELECT[1:0]
wire CLK
wire FlowSelect
wire IMMEDIATE[7:0]
wire INSTRUCTION[31:0]
reg ImmOrRegistered
wire OFFSET[7:0]
reg OPCODE[7:0]
wire OPERAND2[7:0]
reg PC[31:0]
wire PC_inc[31:0]
wire PC_new[31:0]
Filter:
Append
Insert
Replace
Signals
Time
CLK
PC[31:0]
RESET
PC_inc[31:0]
OPCODE[7:0]
READREG1[2:0]
READREG2[2:0]
REGOUT1[7:0]
REGOUT2[7:0]
WRITEREG[2:0]
ALUOP[2:0]
ALURESULT[7:0]
ImmOrRegistered
PositiveOrNegative
TwosCompOperand[7:0]
WRITEENABLE
BRANCH_SELECT[1:0]
OFFSET[7:0]
PC_new[31:0]
TARGET_ADDR[31:0]
Waves
44 sec
48 sec
52 sec

MULT Instruction

the MULT instruction in the ALU utilizes a separate multiplier unit consisting of an array of Full Adders. The result is limited to 8 bits, and any result exceeding 255 will be inaccurate. The multiplication operation takes more time compared to other ALU units, with a simulation delay of #3 time units incorporated to reflect this.

PC Update	Instruction Memory Read		Register Read	ALU
#1	#2		#2	#3
	PC+4 Adder		Decode	
	#1		#1	
Register Write				
#1				



Logical Left Shift Instruction

The LshiftComb module was added to the ALU to enable bitwise left shift operations. The shifter supports shifts up to 8 positions, Shifting more than 8 positions would give the same results

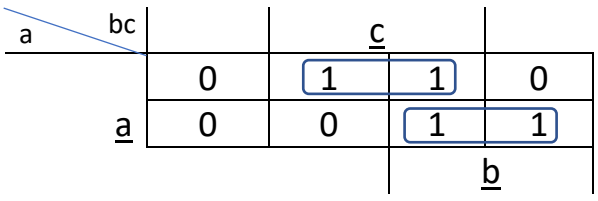
LshiftComb module takes three inputs (a, b, and c) and performs a left shift operation using combinational logic. The result is obtained by ANDing the negation of a with c and ANDing a with b, and then ORing the two intermediate results to produce the final output out.

A simulation delay of #2 time units is included to represent the time taken for the shift operation.

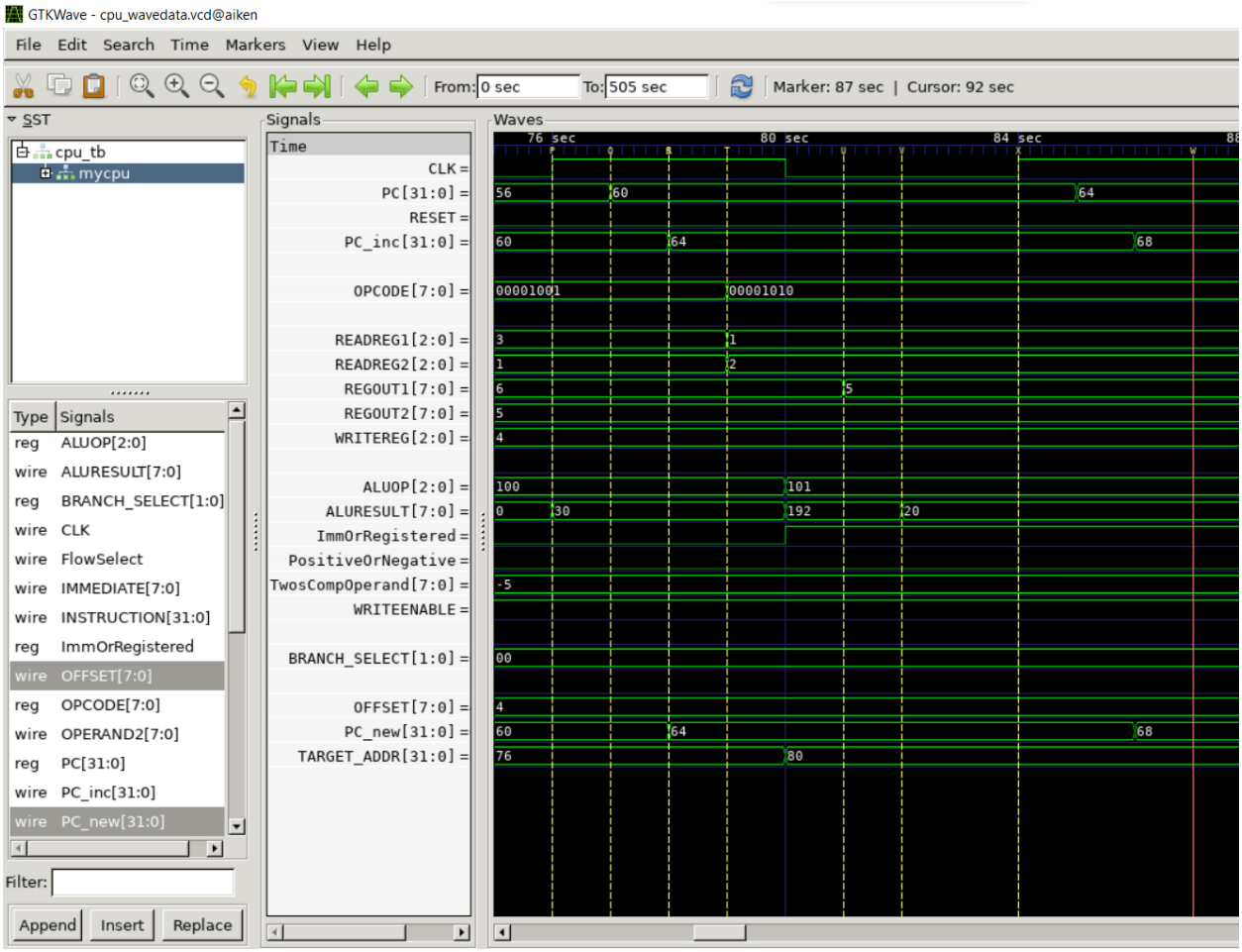
a : Bit in the relevant position of OPERAND2
b : Bit if shifted
c : Bit if not shifted

a	b	c	out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Karnaugh Map



Out = a'c + ab


GTKWave - cpu_wavedata.vcd@aiken
File Edit Search Time Markers View Help
From: 0 sec
To: 505 sec
Marker: 87 sec | Cursor: 92 sec
SST
cpu_tb
mycpu
Type Signals
reg ALUOP[2:0]
wire ALURESULT[7:0]
reg BRANCH_SELECT[1:0]
wire CLK
wire FlowSelect
wire IMMEDIATE[7:0]
wire INSTRUCTION[31:0]
reg ImmOrRegistered
wire OFFSET[7:0]
reg OPCODE[7:0]
wire OPERAND2[7:0]
reg PC[31:0]
wire PC_inc[31:0]
wire PC_new[31:0]
Filter:
Append
Insert
Replace
Signals
Time
CLK
PC[31:0]
RESET
PC_inc[31:0]
OPCODE[7:0]
READREG1[2:0]
READREG2[2:0]
REGOUT1[7:0]
REGOUT2[7:0]
WRITEREG[2:0]
ALUOP[2:0]
ALURESULT[7:0]
ImmOrRegistered
PositiveOrNegative
TwosCompOperand[7:0]
WRITEENABLE
BRANCH_SELECT[1:0]
OFFSET[7:0]
PC_new[31:0]
TARGET_ADDR[31:0]
Waves
76 sec
80 sec
84 sec
56
60
64
68
00001001
00001010
3
1
2
6
5
4
100
101
0
30
192
20
-5
00
4
60
64
68
76
80

Logical Right Shift/Arithmetic Shift right/Rotate Right Instruction

The RshiftComb module was added to the ALU to enable bitwise left shift operations.

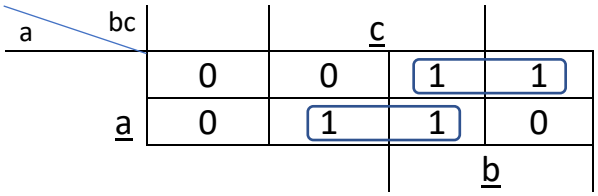
The RshiftComb module takes three inputs (a, b, and c) and performs a right shift operation using combinational logic. The result is obtained by ANDing the negation of a with b and ANDing a with c, and then ORing the two intermediate results to produce the final output out.

A simulation delay of #2 time units is included to represent the time taken for the shift operation.

a : Bit in the relevant position of OPERAND2
b : Bit if not shifted
c : Bit if shifted

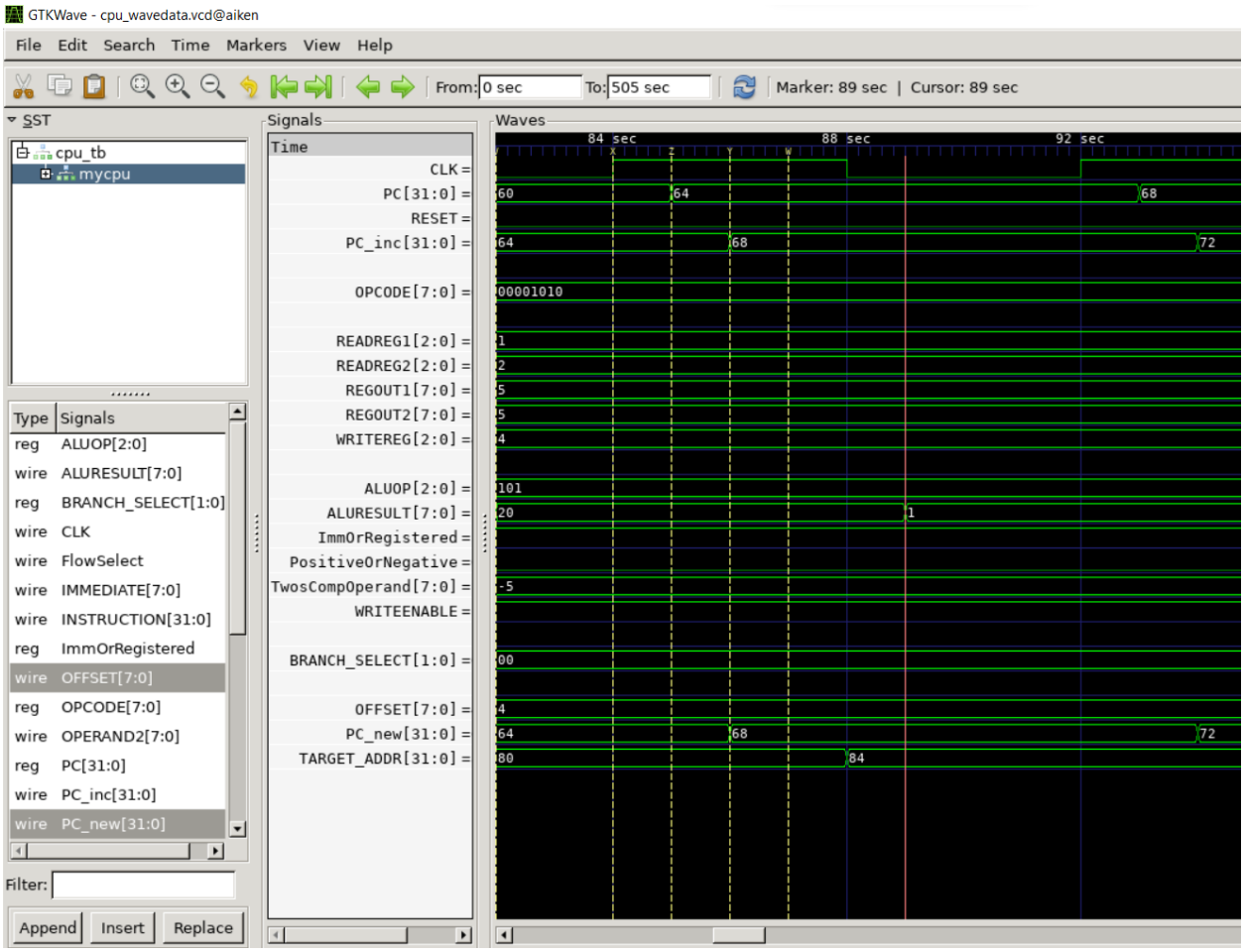
a	b	c	out
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Karnaugh Map

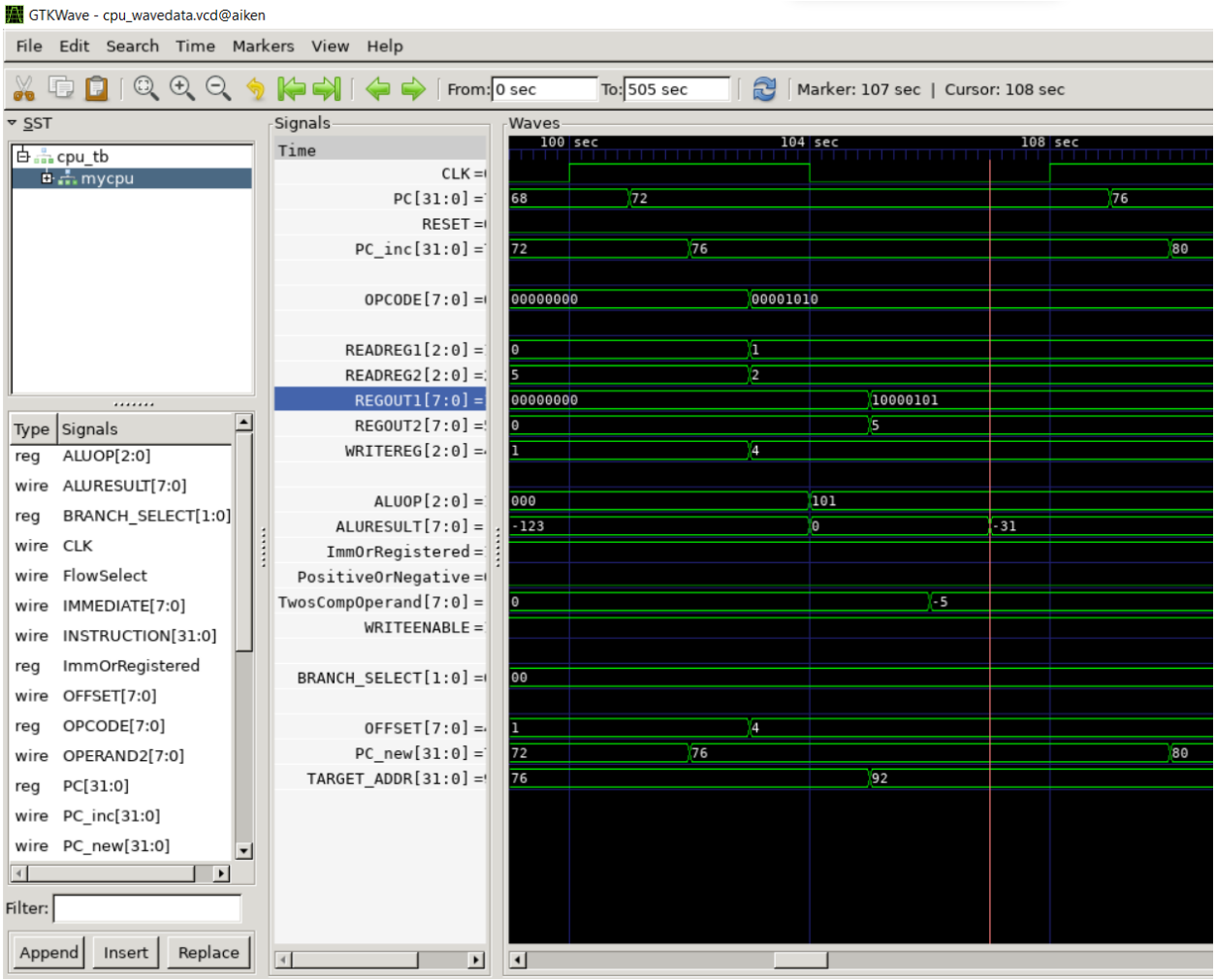


Out = a'b + ac

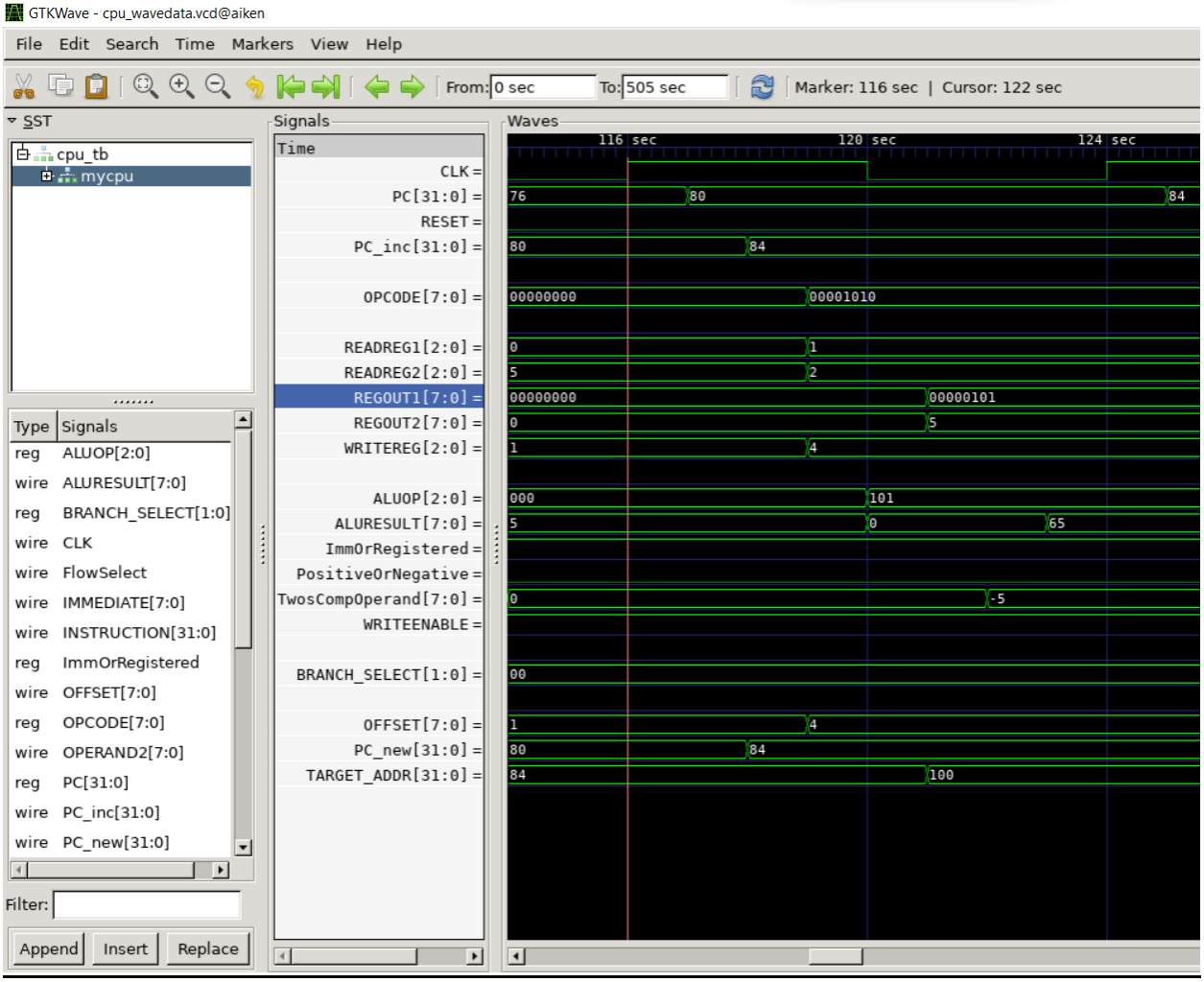
LOGICAL SHIFT RIGHT



ARITHMETIC SHIFT RIGHT



ROTATE RIGHT



SHIFT_ROTATE Module

This module performs various shift and rotate operations on 8-bit operands based on the control input OPERAND2. The selected operation is performed in multiple stages using shift and rotate modules, and the final result is assigned to the output RESULT.

Because of,

Logical shifting a 8bit operand more than 7 times always gives the 0

Arithmetic shifting a 8bit operand more than 7 times always gives the sign bit of that operand

Rotating right more than 7 times gives the same result when rotating 7 times

The only first four least significand bits of OPERAND 2 is usable.

So, we used two most significand bits of OPERAND 2 as selector for sll, srl, sra and ror instructions for using less functional units of ALU.

00	Shift Left Logical
01	Shift Right Logical
10	Shift Rotate Arithmetic
11	Rotate right

CPU OPCODEs for Decoding Instructions

Instruction	OPCODE
loadi	00000000
mov	00000001
add	00000010
sub	00000011
and	00000100
or	00000101
jump	00000110
beq	00000111
bne	00001000

mult	00001001
sll	00001010
srl	00001010
sra	00001010
ror	00001010

We Machine coded these lines of instructions in instr_mem.mem file

10. bne 0x02 3 1

15. mult 4 3 1

16. sll 4 1 0x02

17. srl 4 1 0x02

19. sra 4 1 0x02

21. ror 4 1 0x02