EE446 Laboratory Work 4
ISA & Datapath Design for Multi-Cycle CPU

Author: Ali Necat Karakuloğlu

ID: 2166742

Date: 27/04/2020

Introduction

In this report, designed instruction set architecture will be explained. Based on ISA, the datapath design will be explained and the test results of the datapath will be provided.

Instruction Set Architecture

This ISA is designed as 16-bit words that can support direct addressing, immediate addressing and indirect addressing modes according to the instructions. In this ISA, 8 registers are used. Furthermore, a program counter and a link register is used to achieve the requirements.

Instructions

As stated in the manual, there are 5 types of operation. Arithmetic, Logic, Shift, Branch and Memory. Rd denotes destination register, Rn denotes 1st operand and Rm denotes 2nd operand. Also, $Mem denotes a data in a memory address.

1.Arithmetic Operations

MNEMONIC	OPERATION	DESCRIPTION
ADD Rd, Rn, Rm	Rd = Rn + Rm	Register Addition
ADDI Rd, Rn, $Mem	Rd = Rn + $Mem	Indirect Addition with 4 bit Memory Addressing
SUB Rd, Rn, Rm	Rd = Rn - Rm	Register Addition
SUBI Rd, Rn, $Mem	Rd = Rn - $Mem	Indirect Addition with 4 bit Memory Addressing

2.Logic Operations

MNEMONIC	OPERATION	DESCRIPTION
AND Rd, Rn, Rm	Rd = Rn & Rm	Bitwise AND
ORR Rd, Rn, Rm	Rd = Rn | Rm	Bitwise OR
XOR Rd, Rn, Rm	Rd = Rn ^ Rm	Bitwise XOR
CLR Rd	Rd = 0	Clear

3.Shift Operations

MNEMONIC	OPERATION	DESCRIPTION
ROL Rd, Rn, Rm	Rd = {Rn[15-(Rm):(Rm)] , Rn[(Rm)-1:0]}	Rotate Circular Left
ROR Rd, Rn, Rm	Rd = {Rn[(Rm)-1:0] , Rn[15:(Rm)]}	Rotate Circular Right
LSL Rd, Rn, Rm	Rd = Rn << Rm	Logical Shift Left
ASR Rd, Rn, Rm	Rd = Rn <<< Rm	Arithmetic Shift Right
LSR Rd, Rn, Rm	Rd = Rn << Rm	Logical Shift Right

**(Rm) denotes value in Rm*

4.Branch Operations

MNEMONIC	OPERATION	DESCRIPTION
B Label	PC <= Label	Branch Uncond.
BL Label	PC <= Label , LR <= PC	Branch Uncond. With Link
BIL $Mem	PC <= $Mem , LR <= PC	Branch Indirect With Link
BIZ Label	PC <= Label	Branch If Zero
BNZ Label	PC <= Label	Branch If Not Zero
BIC Label	PC <= Label	Branch If Carry Set
BNC Label	PC <= Label	Branch If Carry Not Set

5.Logic Operations

MNEMONIC	OPERATION	DESCRIPTION
LDR Rd, $Mem	Rd = $Mem	Load to Register from Memory
MDR Rn, DATA	Rn = Data	Load Immediate Data to Register
SDM Rn, $Mem	$Mem = Rn	Store from Register to Memory

Instruction Format

In this instruction format op represents the operations i.e. arithmetic, logic etc. and ops field represents instructions in an operation. Extension field is clarified in the following part of this section.

Op(3 Bit)	Ops(3 Bit)	Extension(10 Bit)

OP	OPS	INSTRUCTION	EXTENSION
000(ARITHMETIC)	000	ADD	Rd(3-bit), Rn(3-bit), Rm(3-bit)
000(ARITHMETIC)	001	ADDI	Rd(3-bit),Rn(3-bit), $Mem(4-bit)
000(ARITHMETIC)	010	SUB	Rd(3-bit), Rn(3-bit), Rm(3-bit)
000(ARITHMETIC)	011	SUBI	Rd(3-bit),Rn(3-bit), $Mem(4-bit)
001(LOGIC)	000	AND	Rd(3-bit), Rn(3-bit), Rm(3-bit)
001(LOGIC)	001	OR	Rd(3-bit), Rn(3-bit), Rm(3-bit)
001(LOGIC)	010	XOR	Rd(3-bit), Rn(3-bit), Rm(3-bit)
001(LOGIC)	011	CLR	Rd(3-bit), Rn(3-bit), Rm(3-bit)
010(SHIFT)	000	ROL	Rd(3-bit), Rn(3-bit), Rm(3-bit)
010(SHIFT)	001	ROR	Rd(3-bit), Rn(3-bit), Rm(3-bit)
010(SHIFT)	010	LSL	Rd(3-bit), Rn(3-bit), Rm(3-bit)
010(SHIFT)	011	ASR	Rd(3-bit), Rn(3-bit), Rm(3-bit)
010(SHIFT)	100	LSR	Rd(3-bit), Rn(3-bit), Rm(3-bit)
011(BRANCH)	000	B	Label(10-bit)
011(BRANCH)	001	BL	Label(10-bit)
011(BRANCH)	010	BIL	Label(10-bit)
011(BRANCH)	011	BIZ	Label(10-bit)
011(BRANCH)	100	BNZ	Label(10-bit)
011(BRANCH)	101	BIC	Label(10-bit)
011(BRANCH)	110	BNC	Label(10-bit)
100(MEMORY)	000	LDR	Rd(3-bit), $Mem(7-bit)
100(MEMORY)	001	MDR	Rd(3-bit), DATA(7-bit)
100(MEMORY)	010	SDM	Rn(3-bit), $Mem(7-bit)

Registers

This ISA supports 8 registers that stores 16-bit data. Since there are 8 registers, 3-bit register address is used. In addition, a program counter and a link register is used for the control of the system. Note that, PC and LR are also 16-bit. Meaning that 16-bit memory address can be used.

Addressing Modes

The architecture supports indirect addressing, register addressing and immediate addressing. However, supported modes varies according to the instruction.

Memory Utilization

First 16 address is reserved for indirect address accessible data and first 128 address is reserved for memory type instructions. Furthermore, branch target addresses are represented with 10-bits. Thus, 1024 – 128 = 896 instructions can be written to the address space with Label addresses.

Datapath Design

In the design procedure of the datapath, indirect memory read has a significant importance. The first reason of this importance is the memory is unified and the second is indirect reads require multiple reads from the memory (Instruction and Data). Those reasons add complexity to the datapath structure. For example, indirect addition instruction has its effective address in the instruction. Therefore, we need to read the memory twice to obtain the data. In this section of the report, datapath design is separated into three sections. Those sections are Fetch, Decode and Execute in general. Note that, those are not the actual stages but, an abstract of actual states that will be used in the controller.

Fetch

This stage starts with a read from memory. By using the program counter, the instruction is read and, it is prepared to be written to the instruction register. Furthermore, the next PC address is calculated but not written to PC. In this stage, PC is used to get the instruction address, memory is used for obtaining the instruction, ALU is used for calculating the next address and instruction register is used for storage of the instruction since, another memory read are needed. Besides, some multiplexers used for the input of the PC and address input of the memory. In Figure 1, the dataflow of fetch process can be seen. The first blue arrow represents the memory address of the instruction and the second arrow represents the fetched instruction. Increment in the PC will be shown in execute part because, the schematic is too large to fit in this report.

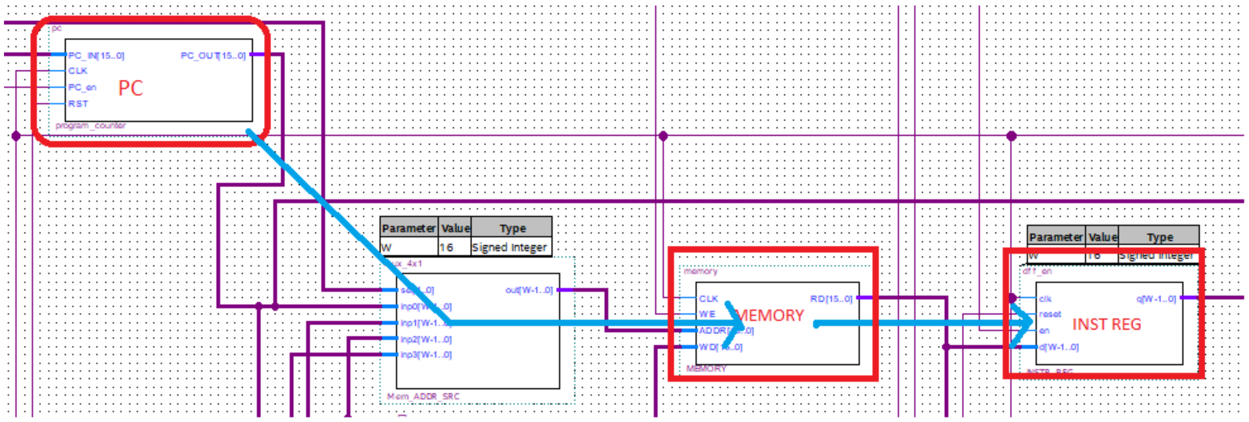


Figure 1: Fetch Process in Datapath.

Decode

This stage starts with the write to the instruction register. After the instruction is read, the data to be processed is obtained in this stage. Register file is read if necessary or another memory read is made by using a memory data register. Note that, the second one occurs when an indirect instruction is fetched from the memory. In general, the data for execution stage is obtained from the memory or register file in this stage. In branch instructions, the same is done. Branch target address is obtained from memory.

In this stage, memory, register file and memory data register have effective role. According to the instruction memory is read or register file is read. In Figure 2 the dataflow in decode instruction can be seen. In this figure, arrow 1 represents the instruction fetched from the memory. Arrow 2 represents the effective memory address which is obtained from the memory. Note that, instruction register is also connected to the register file. Therefore, simultaneously with the memory read, the register file is read as can be seen from arrow 4. The memory read is represented with arrow 3. In decode cycle, register file buffers (arrow 4) and memory data register is written. Thus, the data for execute cycle is obtained.

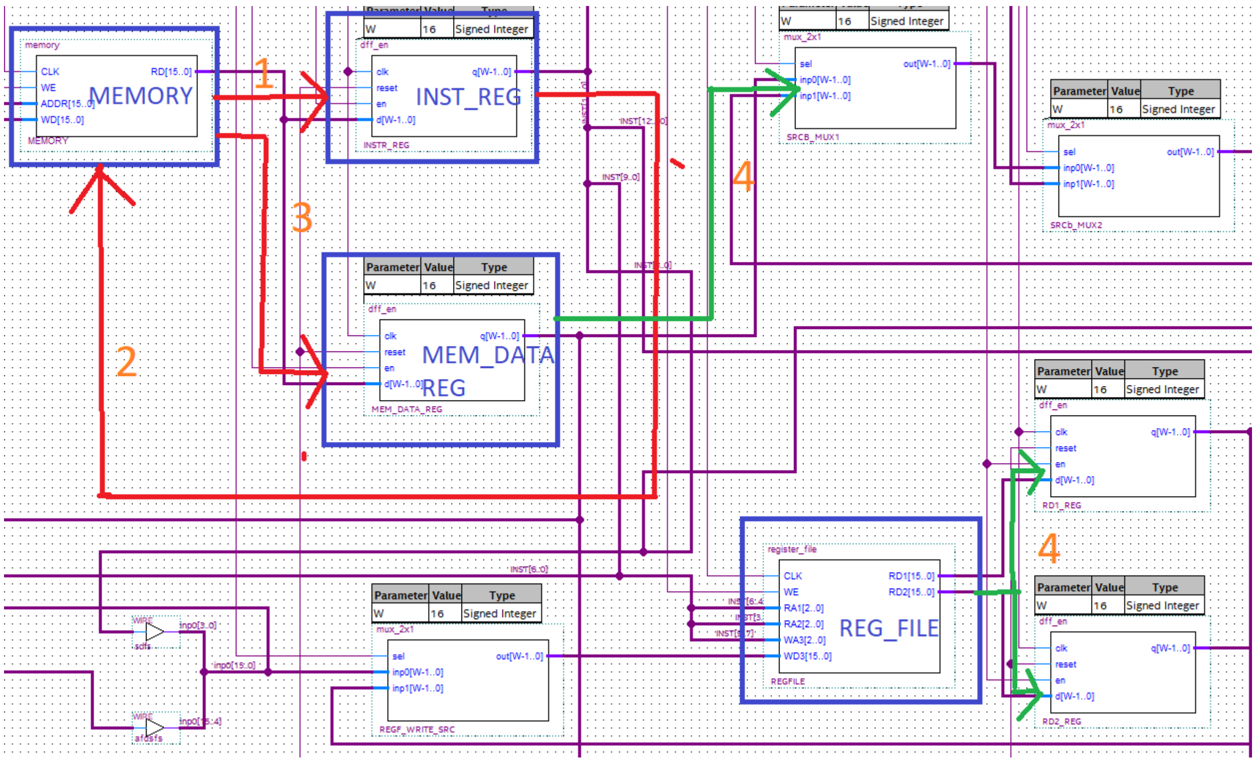


Figure 2: Decode Process in Datapath.

Execute

In execute part of the datapath, arithmetic-logic and shift operations take place. To serve that purpose, an ALU and a shifter takes place in the center of the cycle. Other hardware is used to choose between operators and results. In other words, multiplexers are used to choose between operands. ALU and shifter are connected separately (Shifter output doesn't connect to ALU as in ARM). The result is chosen according to the instruction and prepared to be loaded to the result register. As mentioned in fetch part, ALU is responsible from calculating the next PC address. When the processor is in fetch cycle, execute hardware of the datapath is idle. With a direct connection from the result of the ALU. PC address is calculated.

In Figure 3, dataflow is shown on the schematic. Shifter is directly connected to RD1 and instruction register. Therefore, it doesn't require multiplexers. However, ALU has multiple input options. SRCA (source A) can be the constant to increment the PC or it can be RD1. Whereas, SRCB (source B) can be RD2, memory data or program counter. Thus, 2 multiplexers are needed for SRCB. After the result is obtained, depending on the instruction type, ALU result or shifter result is chosen. Then, ALU_DATA register is written. The arrows represent the dataflow in this stage.

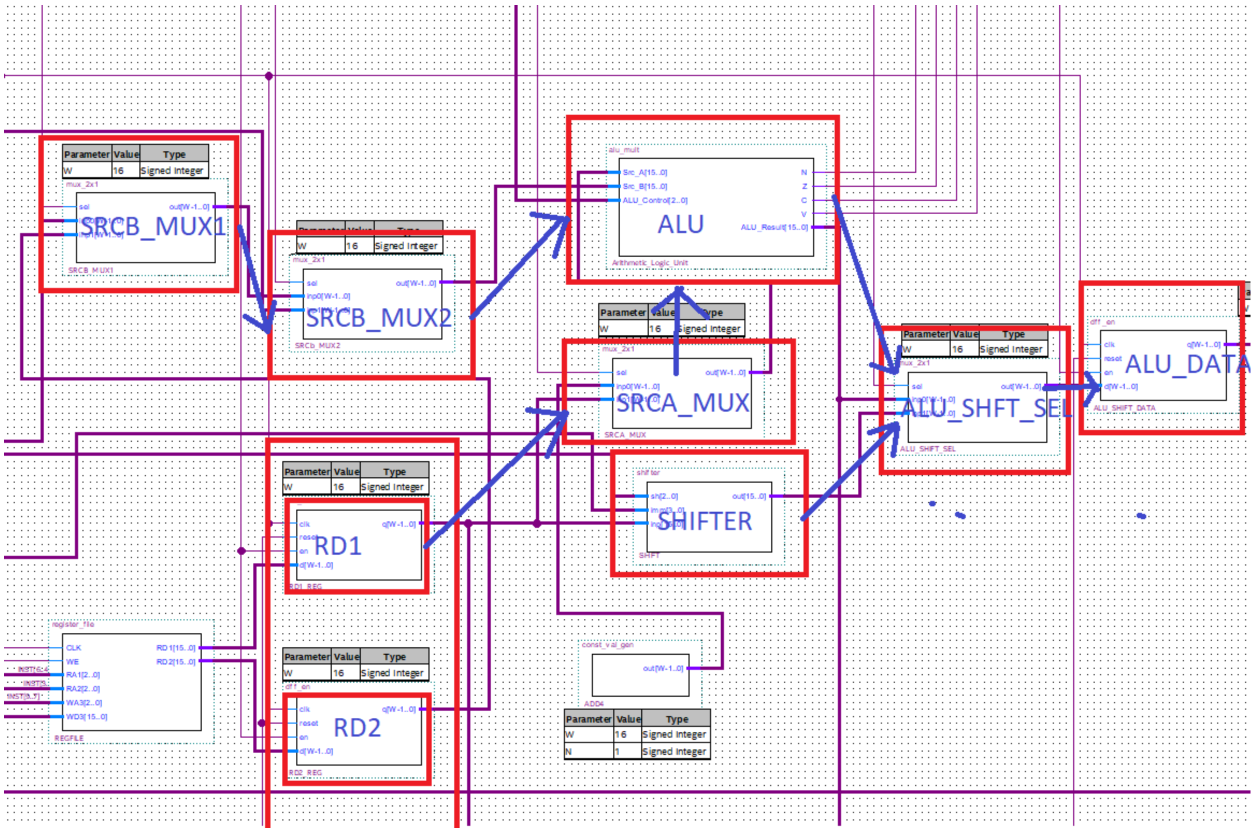


Figure 3: Execute Process in Datapath.

Note that, another writeback stage is required to complete the datapath. However, it is a simpler mechanism than the other three. When the result is calculated in execute stage, it is written to register file or memory. The next phase of this section illuminates the special care given to some operation types. Where the datapath becomes more than an arithmetic logic processor.

Further Analysis on Operation Types and Datapath Design

Until now, an arithmetic logic machine with some storage elements are demonstrated. In this section, datapath will be examined thoroughly. The point of view will be instruction families. Namely, operation types. As stated previously, there are 5 types of instructions. Firstly, arithmetic operations, logic operations and shift operations are the subjects to be studied. The reason for this grouping is, they have the very same mechanisms but some different operations in ALU. Arithmetic and logic operations differ only on the selection mechanism. Note that fetch operations are the same for all instructions.

The first group starts with decode. They read the memory for instruction and read register file for operands. However, in indirect operations the memory read is made twice. One for data and the other one is for instruction. After the operands are obtained, they are send to SRCA and SRCB.

Simultaneously, the controller sends ALU control signals that are determined by instruction. Then the result is first written to ALU_DATA register and written to register file.

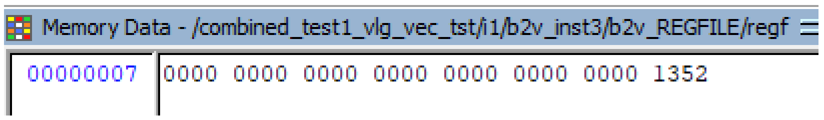
The second type of operations are branch operations, those operations are also separated. However, the difference is on the link register. Thus, only difference is to write LR or not to write. Starting from decode stage, instruction is decoded and a decision is made for indirect branch operation. If the instruction requires read from memory, the branch address is obtained from the MEM_DATA register. Otherwise, the BTA is instruction. The PC is able to choose between those two options. Therefore, PC can be chosen from the MEM_READ register, ALU, instruction register and link register. In the next cycle, PC is written if needed or decided link register is also written and link register is written with the previous value of the PC.

Third operation type is memory. They use the ALU to transfer the memory data. Therefore, their control is more complicated than the others. For example, load data to register(LDR) makes a read from the register file. Then, sends the data to the ALU. In ALU, there is a transfer option. The rest is similar to arithmetic operations. In the end, the data from memory is written to the register file.

Test Procedure

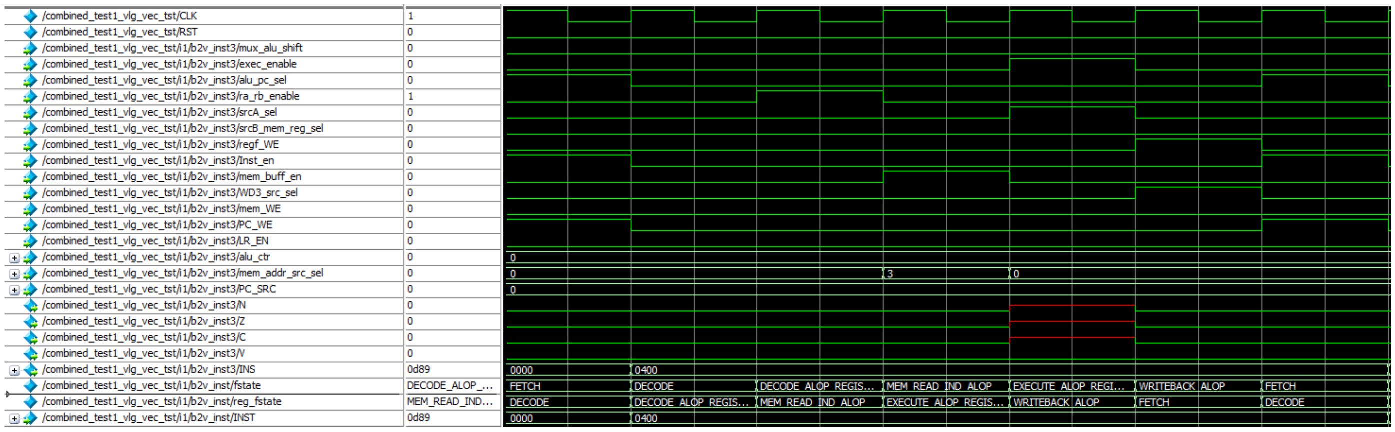
Because of the number of the instructions, they will be demonstrated separately but, some conditional branch instructions are demonstrated with arithmetic instructions. The purpose is to set carry or zero flag. Since there is a controller, the testbench will only provide the reset and clock signals. ASM of the controller is provided for demonstration. With the ASM chart and simulation results, the parts that are used for the specific instruction will be explained and operation of the datapath will be verified. To ease the work of reader, ASM chart contains the driving signals. Thus, activated hardware in datapath. Since, this is a multicycle design, other parts are don't care. Signals are also shown on simulation results. To check the results please take "fstate" signal as present state. One can easily follow the ASM chart by using "fstate" signal.

1) As can be seen from Figure 5, "0x0400" instruction is fetched. This instruction is ADDI instruction that reads the $0 which stores the data "0x1352". And the instruction loads the data to R0. The register file can be seen in Figure 4.



```
Memory Data - /combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_REGFILE/regf
00000007 0000 0000 0000 0000 0000 0000 0000 1352
```

Figure 4: Register File After The 1st Instruction.

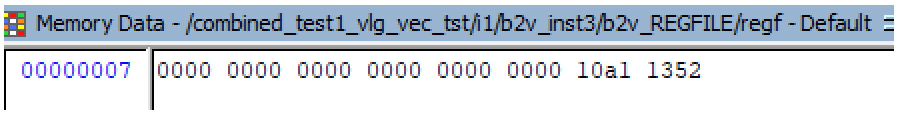


```
/combined_test1_vlg_vec_tst/CLK                             1
/combined_test1_vlg_vec_tst/RST                             0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mux_alu_shift      0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/exec_enable        0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/alu_pc_sel         0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/ra_rb_enable       1
/combined_test1_vlg_vec_tst/i1/b2v_inst3/srcA_sel           0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/srcB_mem_reg_sel   0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/regf_WE            0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/Inst_en            0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_buff_en        0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/WD3_src_sel        0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_WE             0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/PC_WE              0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/LR_EN              0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/alu_ctr            0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_addr_src_sel   0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/PC_SRC             0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/N                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/Z                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/C                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/V                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/INS                0d89
/combined_test1_vlg_vec_tst/i1/b2v_inst/fstate              DECODE_ALOP_...
/combined_test1_vlg_vec_tst/i1/b2v_inst/reg_fstate          MEM_READ_IND...
/combined_test1_vlg_vec_tst/i1/b2v_inst/INST                0d89
```

0
0 | 3 | 0
0
0000 | 0400
FETCH | DECODE | DECODE_ALOP_REGIS... | MEM_READ_IND_ALOP | EXECUTE_ALOP_REGI... | WRITEBACK_ALOP | FETCH
DECODE | DECODE_ALOP_REGIS... | MEM_READ_IND_ALOP | EXECUTE_ALOP_REGIS... | WRITEBACK_ALOP | FETCH | DECODE
0000 | 0400

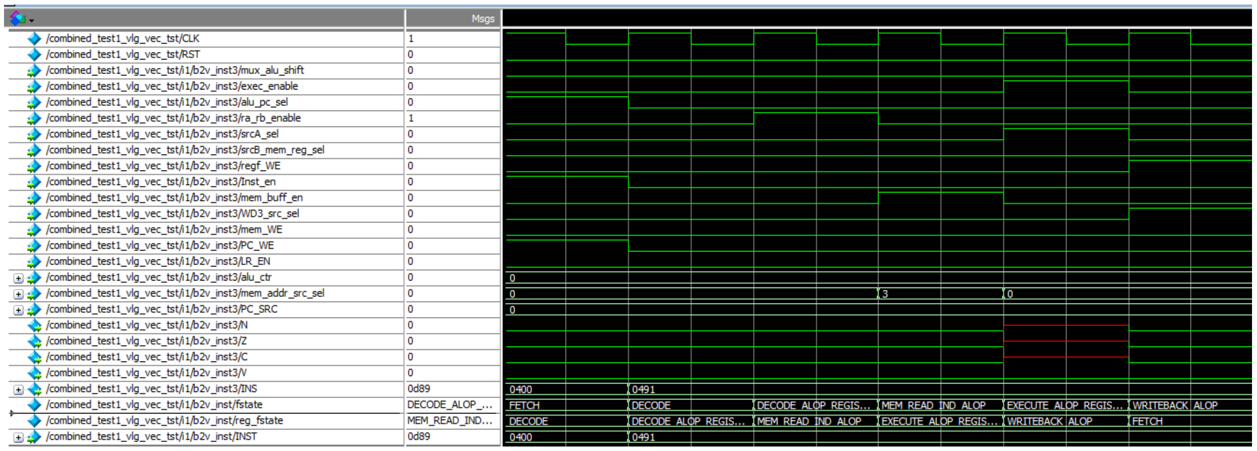
Figure 5: Simulation Result of ADDI R0,R0,$0 Operation.

2) The same instruction is used to load 16-bit data to another register. The results can be seen from Figure 6 and Figure 7. "0x0491" instruction corresponds to ADDI R1,R1,$1.



```
Memory Data - /combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_REGFILE/regf - Default
00000007  0000 0000 0000 0000 0000 0000 10a1 1352
```

Figure 6: Register File After The 2nd Instruction.

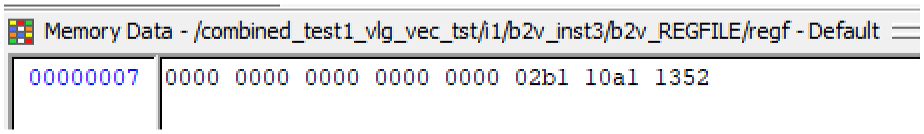


```
                                                            Msgs
/combined_test1_vlg_vec_tst/CLK                             1
/combined_test1_vlg_vec_tst/RST                             0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mux_alu_shift      0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/exec_enable        0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/alu_pc_sel         0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/ra_rb_enable       1
/combined_test1_vlg_vec_tst/i1/b2v_inst3/srcA_sel           0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/srcB_mem_reg_sel   0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/regf_WE            0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/Inst_en            0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_buff_en        0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/WD3_src_sel        0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_WE             0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/PC_WE              0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/LR_EN              0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/alu_ctr            0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_addr_src_sel   0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/PC_SRC             0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/N                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/Z                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/C                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/V                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/INS                0d89
/combined_test1_vlg_vec_tst/i1/b2v_inst/fstate              DECODE_ALOP_...
/combined_test1_vlg_vec_tst/i1/b2v_inst/reg_fstate          MEM_READ_IND...
/combined_test1_vlg_vec_tst/i1/b2v_inst/INST                0d89
```

0
0 | 3 | 0
0
0400 | 0491
FETCH | DECODE | DECODE_ALOP_REGIS... | MEM_READ_IND_ALOP | EXECUTE_ALOP_REGIS... | WRITEBACK_ALOP
DECODE | DECODE_ALOP_REGIS... | MEM_READ_IND_ALOP | EXECUTE_ALOP_REGIS... | WRITEBACK_ALOP | FETCH
0400 | 0491

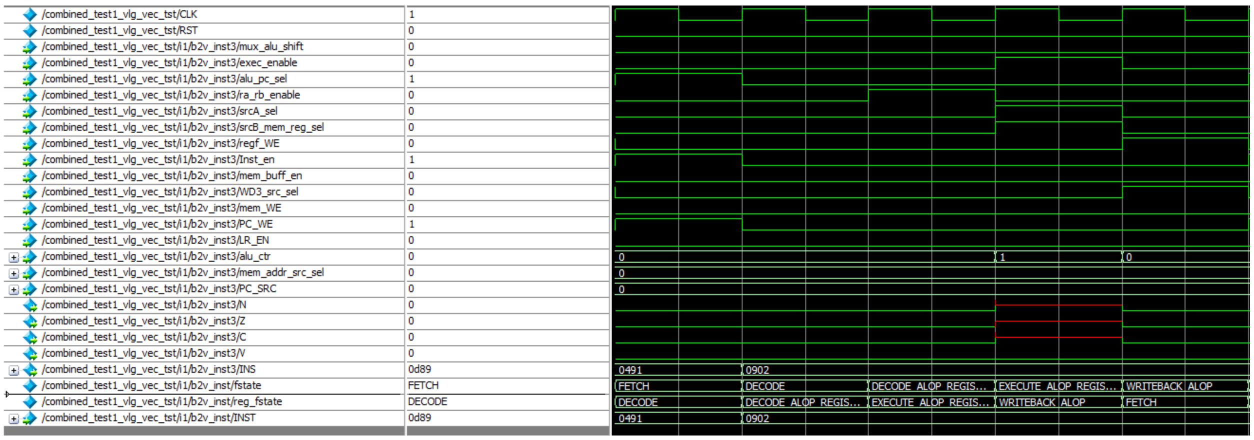
Figure 7: Simulation Result of ADDI R1,R1,$1 Operation.

3) The results of SUB operation can be seen from Figure 8 and 9. As mentioned in the previous part of the report, arithmetic and logic operations differ only in the ALU operation (add, subtract, or, and, etc.). 0x02b1 = 0x1352 - 0x10a1



```
Memory Data - /combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_REGFILE/regf - Default
00000007  0000 0000 0000 0000 0000 02b1 10a1 1352
```

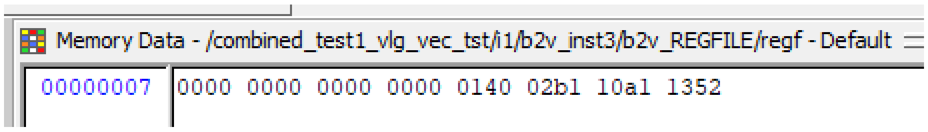
Figure 8: Register File After The 3rd Instruction.



/combined_test1_vlg_vec_tst/CLK	1
/combined_test1_vlg_vec_tst/RST	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mux_alu_shift	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/exec_enable	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/alu_pc_sel	1
/combined_test1_vlg_vec_tst/i1/b2v_inst3/ra_rb_enable	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/srcA_sel	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/srcB_mem_reg_sel	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/regf_WE	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/Inst_en	1
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_buff_en	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/WD3_src_sel	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_WE	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/PC_WE	1
/combined_test1_vlg_vec_tst/i1/b2v_inst3/LR_EN	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/alu_ctr	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_addr_src_sel	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/PC_SRC	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/N	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/Z	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/C	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/V	0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/INS	0d89
/combined_test1_vlg_vec_tst/i1/b2v_inst/fstate	FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/reg_fstate	DECODE
/combined_test1_vlg_vec_tst/i1/b2v_inst/INST	0d89

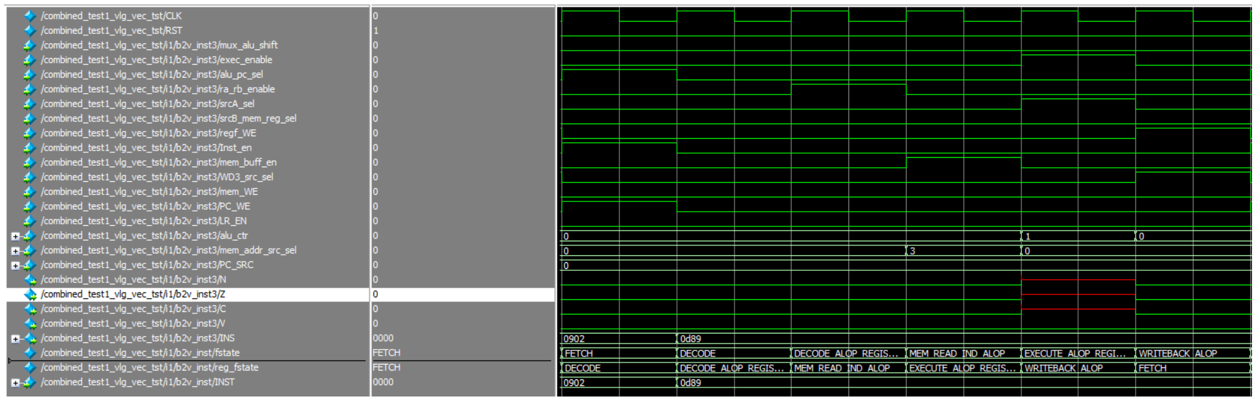
Figure 9: Simulation Result of SUB R2, R0, R1 Operation.

4) Similar to ADDI, SUBI differs only on the ALU operation. In indirect operations another memory read is made. The results can be seen from Figures 10 and 11. 0x0140 = 0x1352 – 0x1212



```
Memory Data - /combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_REGFILE/regf - Default
00000007  0000 0000 0000 0000 0140 02b1 10a1 1352
```

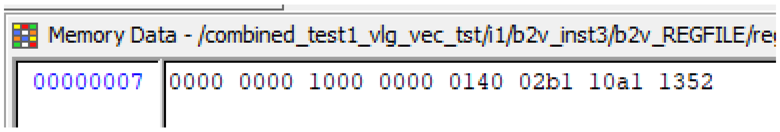
Figure 10: Register File After The 4th Instruction.



```
/combined_test1_vlg_vec_tst/CLK                          0
/combined_test1_vlg_vec_tst/RST                          1
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mux_alu_shift   0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/exec_enable     0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/alu_pc_sel      0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/ra_rb_enable    0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/srcA_sel        0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/srcB_mem_reg_sel 0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/regf_WE         0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/Inst_en         0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_buff_en     0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/WD3_src_sel     0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_WE          0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/PC_WE           0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/LR_EN           0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/alu_ctr         0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_addr_src_sel 0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/PC_SRC          0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/N               0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/Z               0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/C               0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/V               0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/INS             0000
/combined_test1_vlg_vec_tst/i1/b2v_inst/fstate           FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/reg_fstate       FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/INST             0000
```

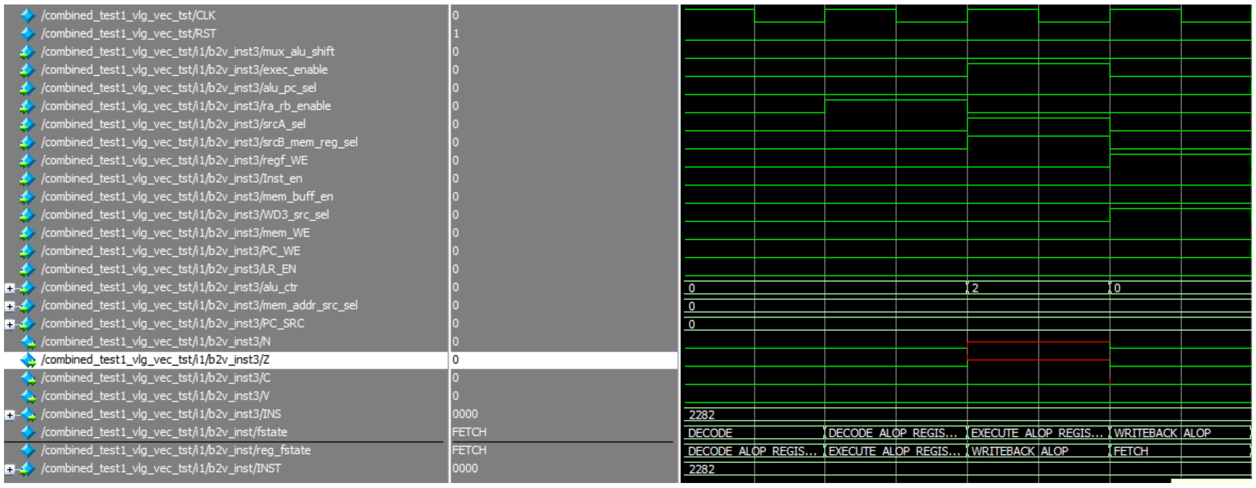
Figure 11: Simulation Result of SUBI R3, R0, $9 Operation.

5) The simulation results for AND operation is given in Figures 12 and 13.
   0x1000 = 0x1352 & 0x10a1



```
Memory Data - /combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_REGFILE/re
00000007   0000 0000 1000 0000 0140 02b1 10a1 1352
```

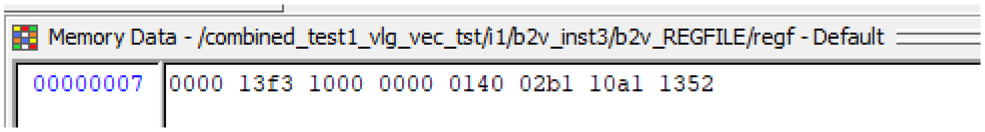
Figure 12: Register File After The 5th Instruction.



```
/combined_test1_vlg_vec_tst/CLK                          0
/combined_test1_vlg_vec_tst/RST                          1
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mux_alu_shift   0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/exec_enable     0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/alu_pc_sel      0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/ra_rb_enable    0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/srcA_sel        0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/srcB_mem_reg_sel 0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/regf_WE         0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/Inst_en         0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_buff_en     0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/WD3_src_sel     0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_WE          0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/PC_WE           0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/LR_EN           0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/alu_ctr         0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_addr_src_sel 0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/PC_SRC          0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/N               0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/Z               0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/C               0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/V               0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/INS             0000
/combined_test1_vlg_vec_tst/i1/b2v_inst/fstate           FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/reg_fstate       FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/INST             0000
```

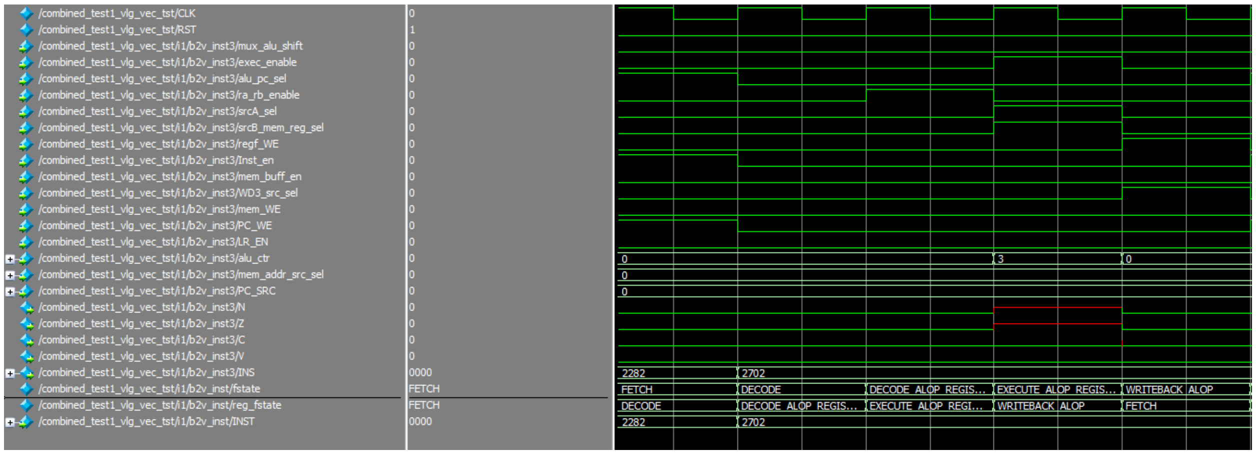
Figure 13: Simulation Result of AND R5, R0, R1 Operation.

6) The simulation results for ORR operation is given in Figures 14 and 15.
   0x13f3 = 0x1352 | 0x10a1



```
Memory Data - /combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_REGFILE/regf - Default
00000007  0000 13f3 1000 0000 0140 02b1 10a1 1352
```

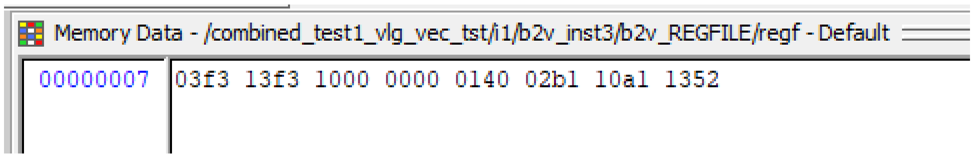
Figure 14: Register File After The 6th Instruction.



```
/combined_test1_vlg_vec_tst/CLK                          0
/combined_test1_vlg_vec_tst/RST                          1
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mux_alu_shift   0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/exec_enable     0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/alu_pc_sel      0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/ra_rb_enable    0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/srcA_sel        0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/srcB_mem_reg_sel 0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/regf_WE         0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/Inst_en         0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_buff_en     0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/WD3_src_sel     0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_WE          0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/PC_WE           0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/LR_EN           0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/alu_ctr         0      0                                3                    0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_addr_src_sel 0     0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/PC_SRC          0      0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/N               0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/Z               0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/C               0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/V               0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/INS             0000   2282   2702
/combined_test1_vlg_vec_tst/i1/b2v_inst/fstate           FETCH  FETCH  DECODE  DECODE ALOP REGIS...  EXECUTE ALOP REGIS...  WRITEBACK ALOP
/combined_test1_vlg_vec_tst/i1/b2v_inst/reg_fstate       FETCH  DECODE  DECODE ALOP REGIS...  EXECUTE ALOP REGI...  WRITEBACK ALOP  FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/INST             0000   2282   2702
```

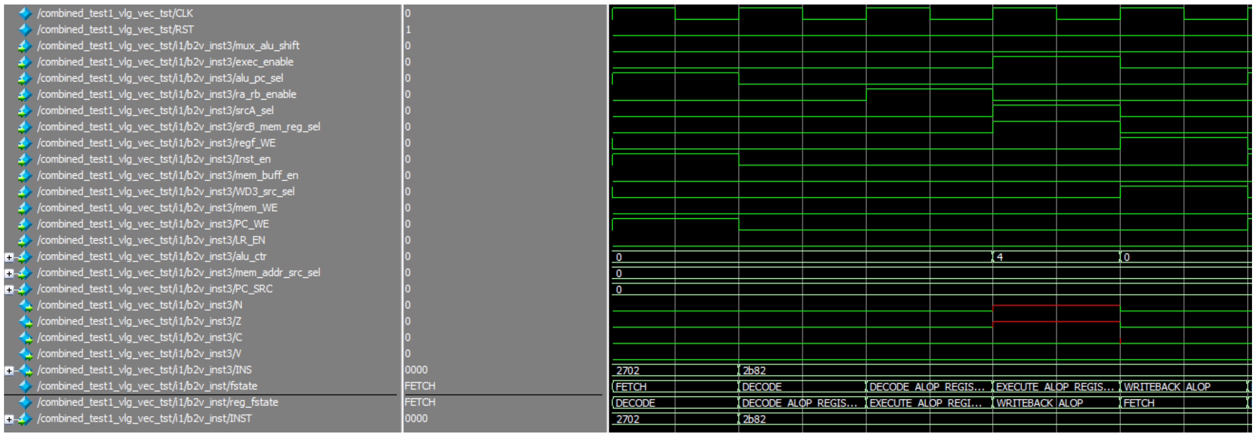
Figure 15: Simulation Result of ORR R6, R0, R1 Operation.

7) The simulation results for XOR operation is given in Figures 16 and 17.
   0x03f3 = 0x1352 | 0x10a1



```
Memory Data - /combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_REGFILE/regf - Default
00000007  03f3 13f3 1000 0000 0140 02b1 10a1 1352
```

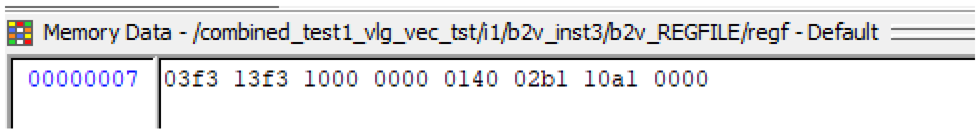
Figure 16: Register File After The 7th Instruction.



```
/combined_test1_vlg_vec_tst/CLK                                 0
/combined_test1_vlg_vec_tst/RST                                 1
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mux_alu_shift          0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/exec_enable            0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/alu_pc_sel             0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/ra_rb_enable           0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/srcA_sel               0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/srcB_mem_reg_sel       0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/regf_WE                0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/Inst_en                0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_buff_en            0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/WD3_src_sel            0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_WE                 0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/PC_WE                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/LR_EN                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/alu_ctr                0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_addr_src_sel       0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/PC_SRC                 0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/N                      0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/Z                      0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/C                      0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/V                      0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/INS                    0000
/combined_test1_vlg_vec_tst/i1/b2v_inst/fstate                  FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/reg_fstate              FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/INST                    0000
```

Figure 17: Simulation Result of XOR R7, R0, R1 Operation.

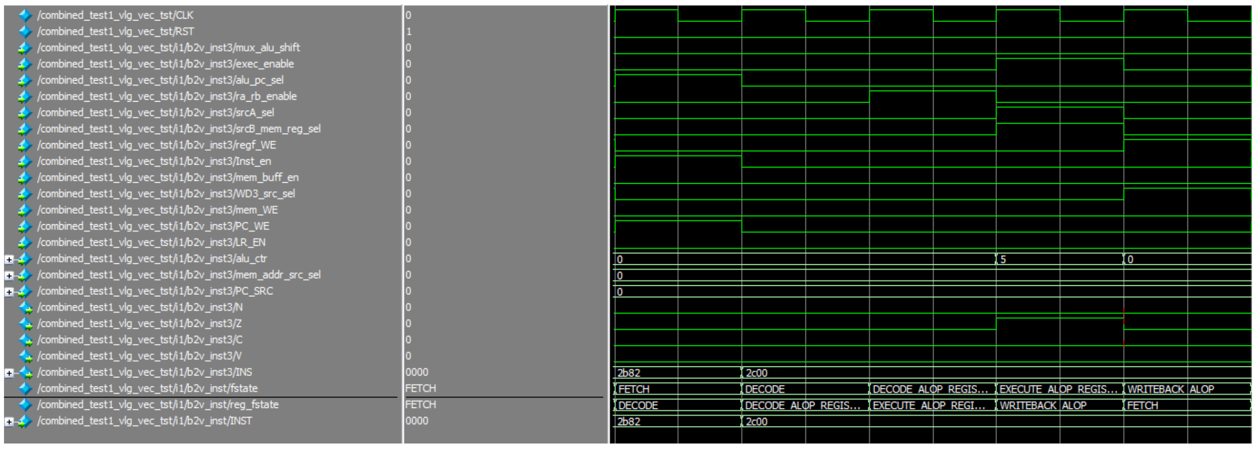
8) The simulation results for CLR operation is given in Figures 18 and 19.



```
Memory Data - /combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_REGFILE/regf - Default

00000007  03f3 13f3 1000 0000 0140 02b1 10a1 0000
```

Figure 18: Register File After The 8th Instruction.

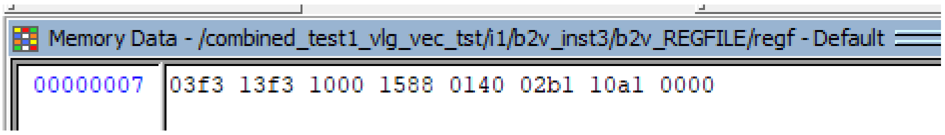


```
/combined_test1_vlg_vec_tst/CLK                                 0
/combined_test1_vlg_vec_tst/RST                                 1
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mux_alu_shift          0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/exec_enable            0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/alu_pc_sel             0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/ra_rb_enable           0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/srcA_sel               0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/srcB_mem_reg_sel       0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/regf_WE                0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/Inst_en                0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_buff_en            0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/WD3_src_sel            0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_WE                 0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/PC_WE                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/LR_EN                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/alu_ctr                0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/mem_addr_src_sel       0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/PC_SRC                 0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/N                      0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/Z                      0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/C                      0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/V                      0
/combined_test1_vlg_vec_tst/i1/b2v_inst3/INS                    0000
/combined_test1_vlg_vec_tst/i1/b2v_inst/fstate                  FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/reg_fstate              FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/INST                    0000
```

Figure 19: Simulation Result of CLR R0 Operation.

9) The simulation results for ROL operation is given in Figures 20 and 21.

0x1588 = 0x02b1 ROL #3



```
Memory Data - /combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_REGFILE/regf - Default
00000007  03f3 13f3 1000 1588 0140 02b1 10a1 0000
```

Figure 20: Register File After The 9th Instruction.

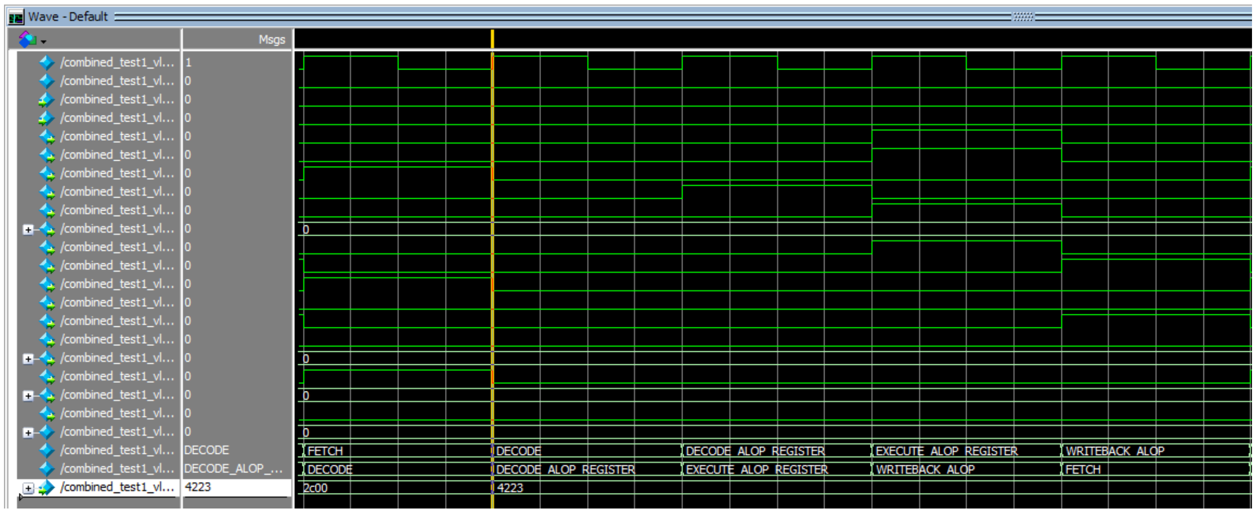
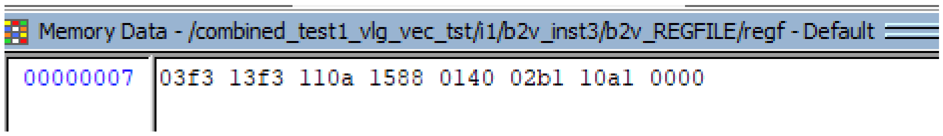


Figure 21: Simulation Result of ROL R4, R2, #3 Operation.

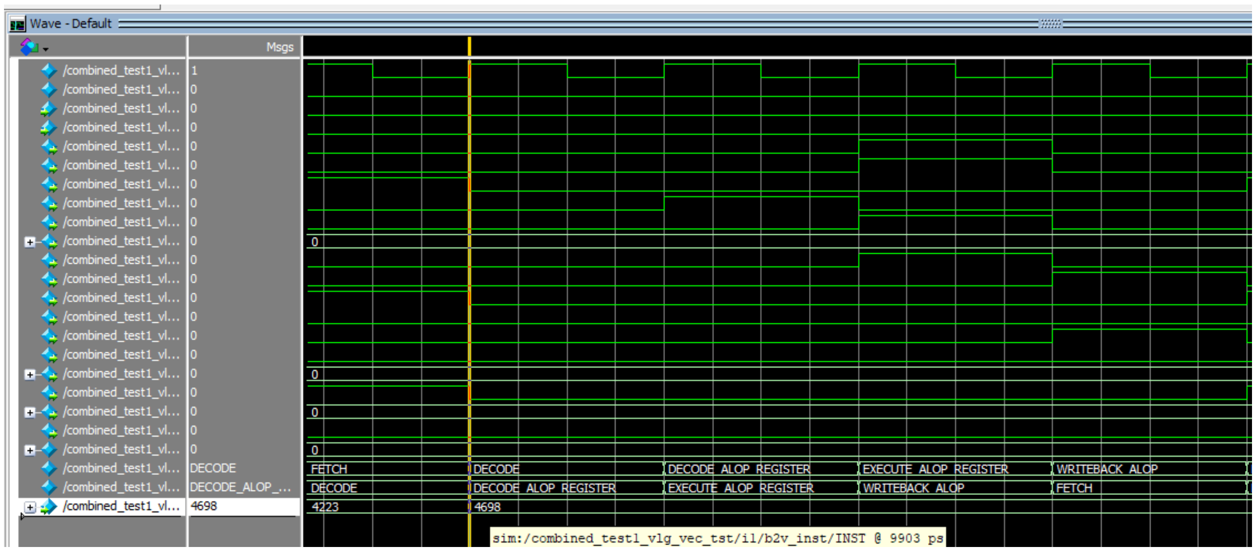
10) The simulation results for ROR operation is given in Figures 22 and 23.

0x110a = 0x10a1 ROR #4



```
Memory Data - /combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_REGFILE/regf - Default
00000007  03f3 13f3 110a 1588 0140 02b1 10a1 0000
```

Figure 22: Register File After The 10th Instruction.

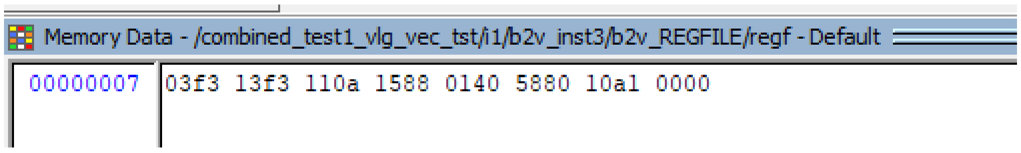


```
Wave - Default
                               Msgs
/combined_test1_vl...   1
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   DECODE
/combined_test1_vl...   DECODE_ALOP_...
/combined_test1_vl...   4698

FETCH | DECODE | DECODE_ALOP_REGISTER | EXECUTE_ALOP_REGISTER | WRITEBACK_ALOP
DECODE | DECODE_ALOP_REGISTER | EXECUTE_ALOP_REGISTER | WRITEBACK_ALOP | FETCH
4223 | 4698
sim:/combined_test1_vlg_vec_tst/i1/b2v_inst/INST @ 9903 ps
```

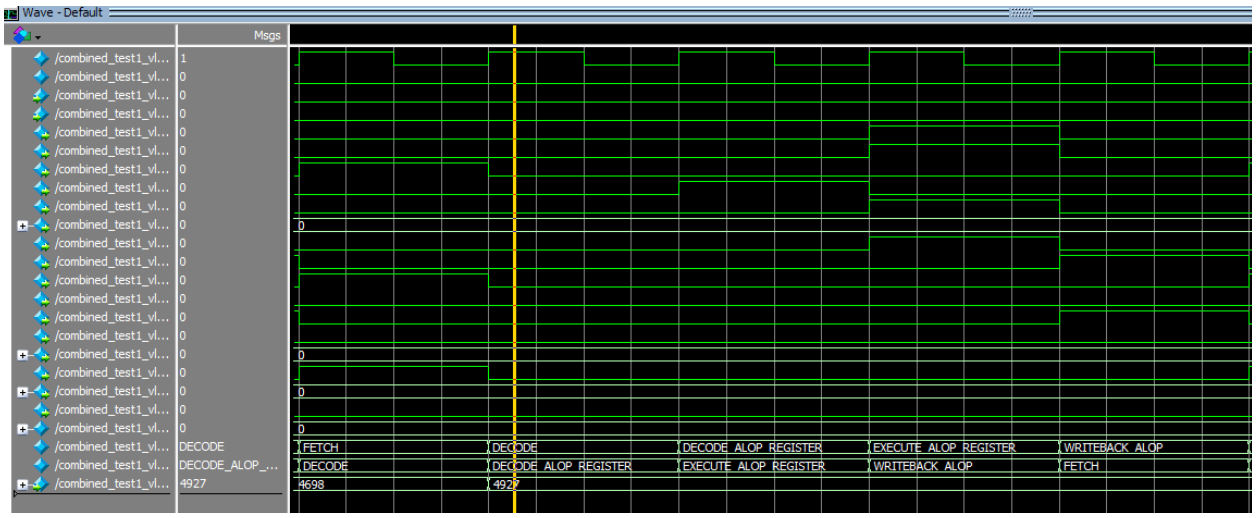
Figure 23: Simulation Result of ROR R5, R1, #4 Operation.

11) The simulation results for LSL operation is given in Figures 24 and 25.
    0x5880 = 0x02b1 >> 7



```
Memory Data - /combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_REGFILE/regf - Default
00000007  03f3 13f3 110a 1588 0140 5880 10a1 0000
```

Figure 24: Register File After The 11th Instruction.

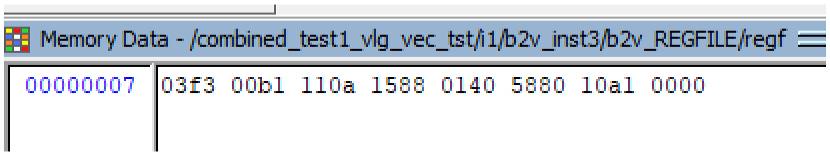


```
Wave - Default
                               Msgs
/combined_test1_vl...   1
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   DECODE
/combined_test1_vl...   DECODE_ALOP_...
/combined_test1_vl...   4927

FETCH | DECODE | DECODE_ALOP_REGISTER | EXECUTE_ALOP_REGISTER | WRITEBACK_ALOP
DECODE | DECODE_ALOP_REGISTER | EXECUTE_ALOP_REGISTER | WRITEBACK_ALOP | FETCH
4698 | 4927
```

Figure 25: Simulation Result of LSL R2, R2, #7 Operation.

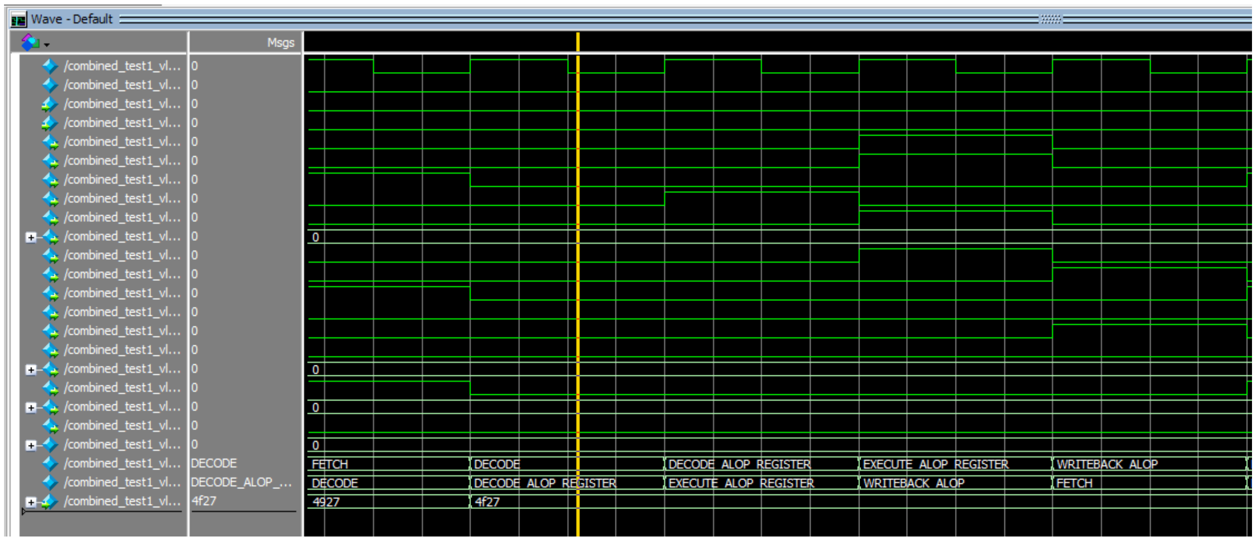
12) The simulation results for ASR operation is given in Figures 26 and 27.
0x00b1 = 0x5880 >> 7



Memory Data - /combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_REGFILE/regf

```
00000007  03f3 00b1 110a 1588 0140 5880 10a1 0000
```

Figure 26: Register File After The 12th Instruction.

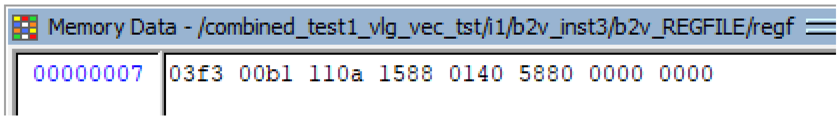


Wave - Default

```
                        Msgs
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0          0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0
/combined_test1_vl...   0          0
/combined_test1_vl...   0
/combined_test1_vl...   0          0
/combined_test1_vl...   0
/combined_test1_vl...   0          0
/combined_test1_vl...   DECODE        FETCH   DECODE                  DECODE_ALOP_REGISTER   EXECUTE_ALOP_REGISTER   WRITEBACK_ALOP
/combined_test1_vl...   DECODE_ALOP_...  DECODE  DECODE_ALOP_REGISTER  EXECUTE_ALOP_REGISTER  WRITEBACK_ALOP        FETCH
/combined_test1_vl...   4f27          4927    4f27
```

Figure 27: Simulation Result of ASR R6, R2, #7 Operation.

13) The simulation results for LSR operation is given in Figures 28 and 29.
0x0000 = 0x00b1 >> 8



Memory Data - /combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_REGFILE/regf

```
00000007  03f3 00b1 110a 1588 0140 5880 0000 0000
```

Figure 28: Register File After The 13th Instruction.

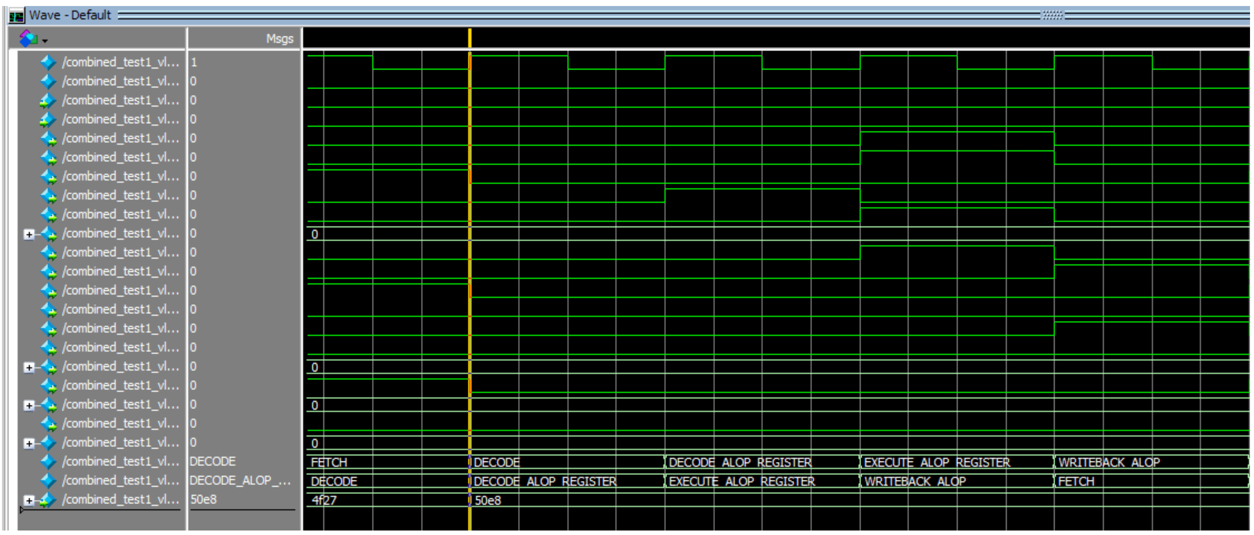
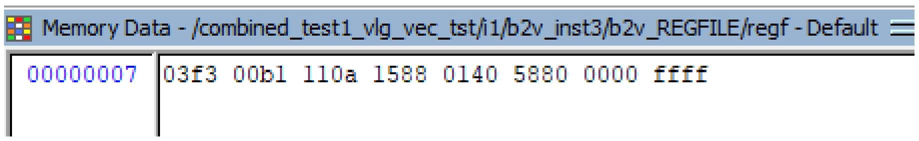


Figure 29: Simulation Result of LSR R1, R6, #8 Operation.

14) The simulation results for LDR operation is given in Figures 30 and 31. R0 is changed to 0xffff which is the data stored in memory address $A.


Memory Data - /combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_REGFILE/regf - Default

```
00000007  03f3 00b1 110a 1588 0140 5880 0000 ffff
```

Figure 30: Register File After The 14th Instruction.

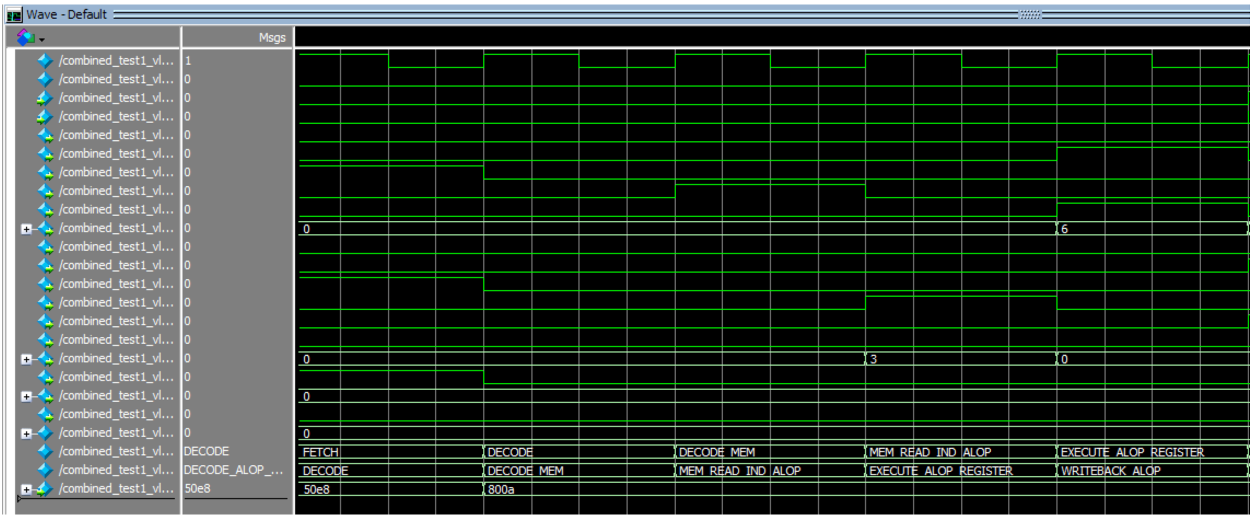
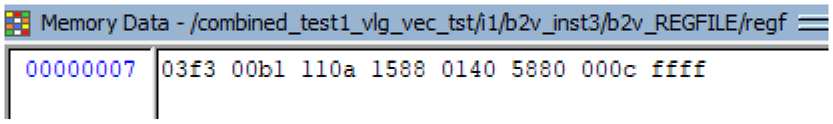


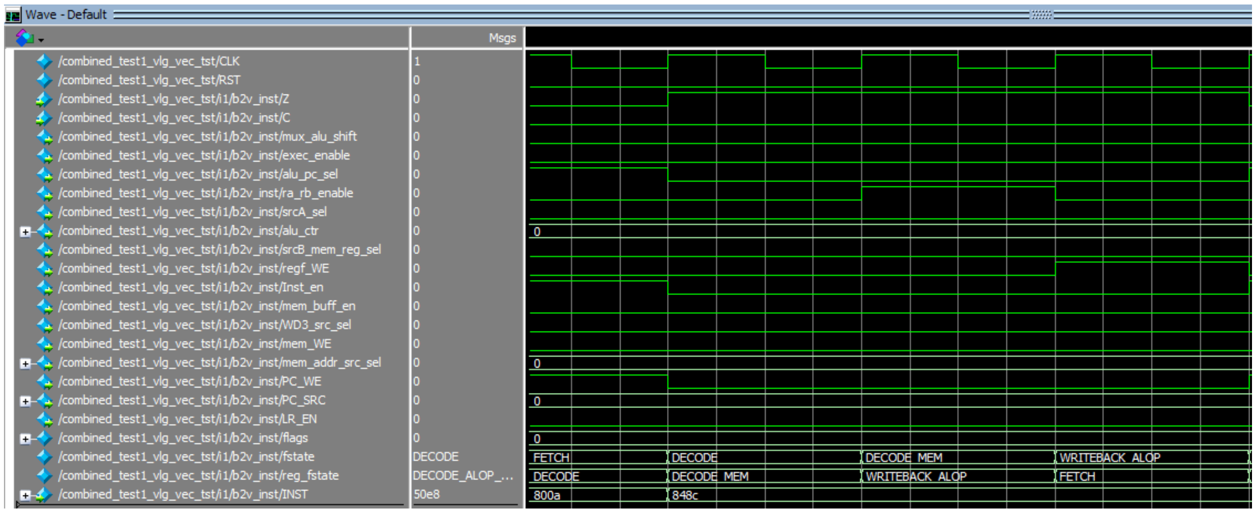
Figure 31: Simulation Result of LDR R0, $A Operation.

15) The simulation results for MDR operation is given in Figures 32 and 33. As can be seen from the figure below, R1 value is changed to 0xC.



```
Memory Data - /combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_REGFILE/regf
00000007   03f3 00b1 110a 1588 0140 5880 000c ffff
```

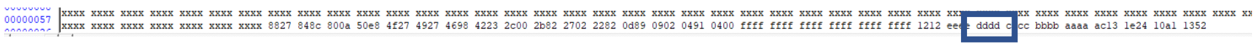
Figure 32: Register File After The 15th Instruction.



```
Wave - Default
                                                      Msgs
/combined_test1_vlg_vec_tst/CLK                       1
/combined_test1_vlg_vec_tst/RST                       0
/combined_test1_vlg_vec_tst/i1/b2v_inst/Z             0
/combined_test1_vlg_vec_tst/i1/b2v_inst/C             0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mux_alu_shift 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/exec_enable   0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_pc_sel    0
/combined_test1_vlg_vec_tst/i1/b2v_inst/ra_rb_enable  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcA_sel      0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_ctr       0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcB_mem_reg_sel 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/regf_WE       0
/combined_test1_vlg_vec_tst/i1/b2v_inst/Inst_en       0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_buff_en   0
/combined_test1_vlg_vec_tst/i1/b2v_inst/WD3_src_sel   0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_WE        0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_addr_src_sel 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_WE         0
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_SRC        0
/combined_test1_vlg_vec_tst/i1/b2v_inst/LR_EN         0
/combined_test1_vlg_vec_tst/i1/b2v_inst/flags         0
/combined_test1_vlg_vec_tst/i1/b2v_inst/fstate        DECODE
/combined_test1_vlg_vec_tst/i1/b2v_inst/reg_fstate    DECODE_ALOP_...
/combined_test1_vlg_vec_tst/i1/b2v_inst/INST          50e8
```

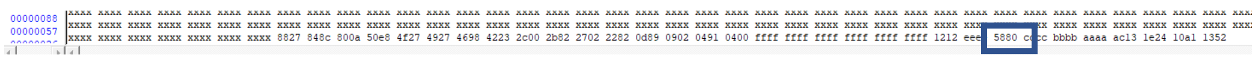
Figure 33: Simulation Result of MDR R1, #C Operation.

16) The simulation results for SDM operation is given in Figures 34, 35 and 36. The data in R2 is loaded to memory address $7. The difference can be seen below.



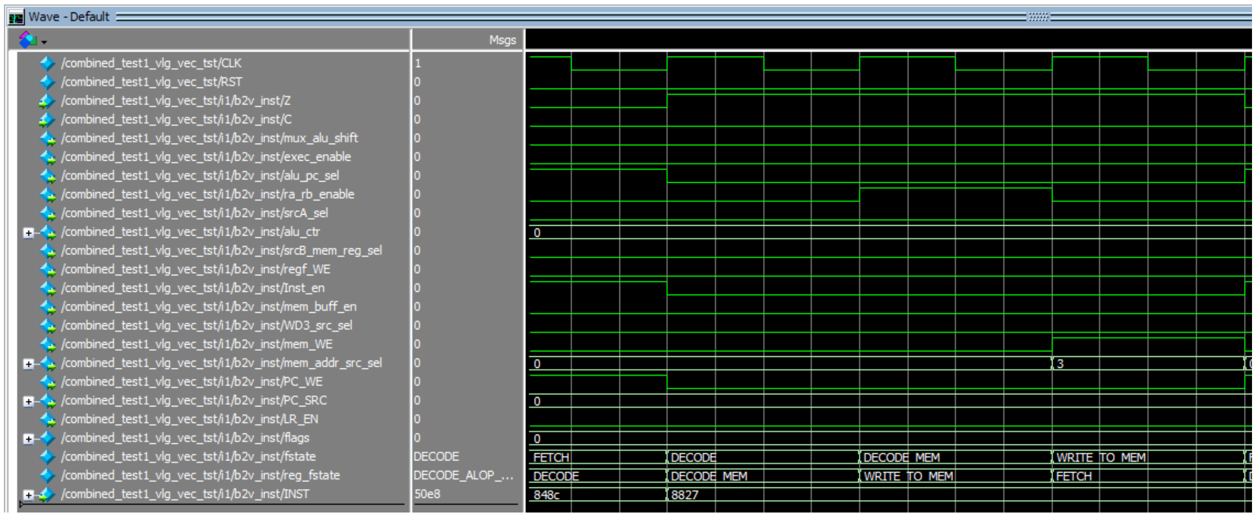
```
00000057 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx
         xxxx xxxx xxxx xxxx xxxx xxxx xxxx 8827 848c 800a 50e8 4f27 4927 4698 4223 2c00 2b82 2702 2282 0d89 0902 0491 0400 ffff ffff ffff ffff ffff ffff 1212 eeee dddd cccc bbbb aaaa ac13 1e24 10a1 1352
```

Figure 34: Memory After The 16th Instruction.



```
00000088 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
00000057 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
         xxxx xxxx xxxx xxxx xxxx xxxx xxxx 8827 848c 800a 50e8 4f27 4927 4698 4223 2c00 2b82 2702 2282 0d89 0902 0491 0400 ffff ffff ffff ffff ffff ffff 1212 eeee 5880 cccc bbbb aaaa ac13 1e24 10a1 1352
```

Figure 35: Memory After The 16th Instruction.

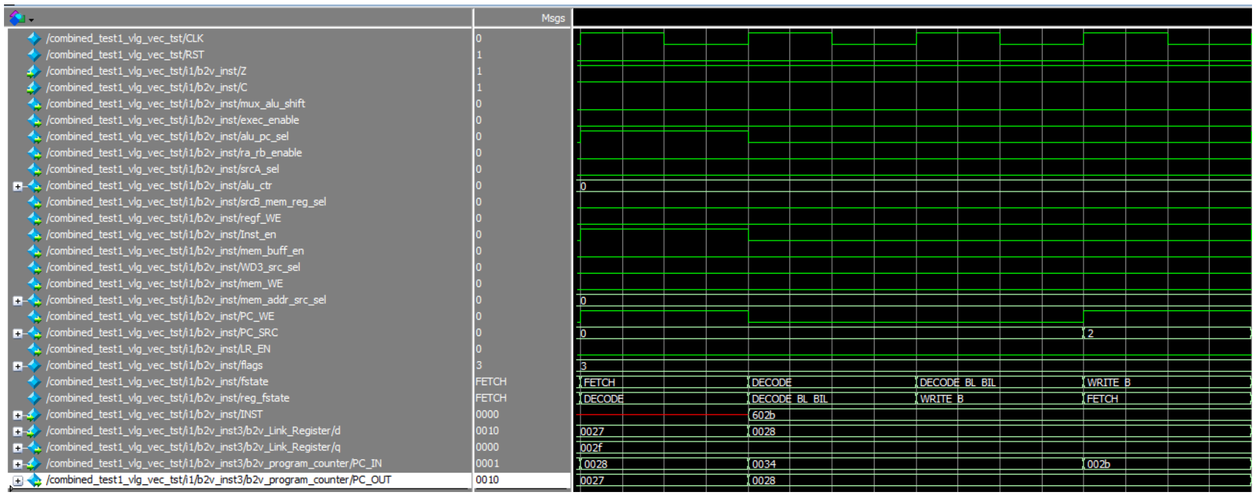


```
Wave - Default
                                                  Msgs
/combined_test1_vlg_vec_tst/CLK                   1
/combined_test1_vlg_vec_tst/RST                   0
/combined_test1_vlg_vec_tst/i1/b2v_inst/Z         0
/combined_test1_vlg_vec_tst/i1/b2v_inst/C         0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mux_alu_shift 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/exec_enable 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_pc_sel 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/ra_rb_enable 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcA_sel  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_ctr   0     0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcB_mem_reg_sel 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/regf_WE   0
/combined_test1_vlg_vec_tst/i1/b2v_inst/Inst_en   0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_buff_en 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/WD3_src_sel 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_WE    0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_addr_src_sel 0     0                                3
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_WE     0
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_SRC    0     0
/combined_test1_vlg_vec_tst/i1/b2v_inst/LR_EN     0
/combined_test1_vlg_vec_tst/i1/b2v_inst/flags     0     0
/combined_test1_vlg_vec_tst/i1/b2v_inst/fstate    DECODE      FETCH   DECODE      DECODE_MEM  WRITE_TO_MEM
/combined_test1_vlg_vec_tst/i1/b2v_inst/reg_fstate DECODE_ALOP_... DECODE DECODE_MEM WRITE_TO_MEM FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/INST      50e8        848c    8827
```

Figure 36: Simulation Result of SDM R2, $7 Operation.

Branch instructions are tested consecutively. In other words, when one branch is taken the next instruction will be another branch instruction. In general, branches are set to be taken. As can be seen from the figures below, the PC jumps.

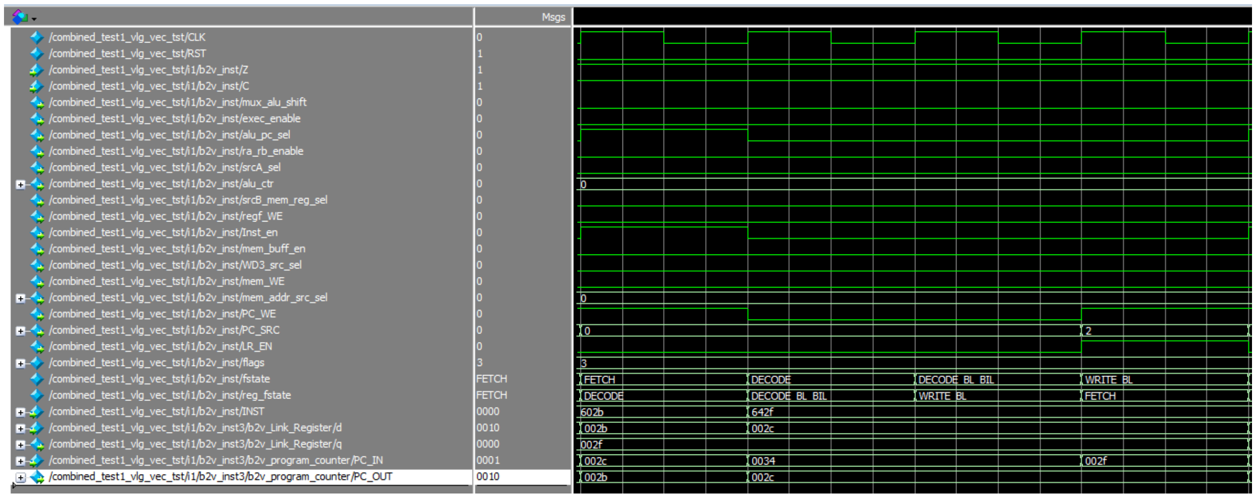
17) The simulation results for B operation is given in Figure 37. The change in PC can be seen below as 2b (especially in Figure 38).



```
                                                                     Msgs
/combined_test1_vlg_vec_tst/CLK                                      0
/combined_test1_vlg_vec_tst/RST                                      1
/combined_test1_vlg_vec_tst/i1/b2v_inst/Z                            1
/combined_test1_vlg_vec_tst/i1/b2v_inst/C                            1
/combined_test1_vlg_vec_tst/i1/b2v_inst/mux_alu_shift                0
/combined_test1_vlg_vec_tst/i1/b2v_inst/exec_enable                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_pc_sel                   0
/combined_test1_vlg_vec_tst/i1/b2v_inst/ra_rb_enable                 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcA_sel                     0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_ctr                      0      0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcB_mem_reg_sel             0
/combined_test1_vlg_vec_tst/i1/b2v_inst/regf_WE                      0
/combined_test1_vlg_vec_tst/i1/b2v_inst/Inst_en                      0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_buff_en                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/WD3_src_sel                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_WE                       0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_addr_src_sel             0      0
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_WE                        0
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_SRC                       0      0                                      2
/combined_test1_vlg_vec_tst/i1/b2v_inst/LR_EN                        0
/combined_test1_vlg_vec_tst/i1/b2v_inst/flags                        3      3
/combined_test1_vlg_vec_tst/i1/b2v_inst/fstate                       FETCH  FETCH   DECODE         DECODE_BL_BIL  WRITE_B
/combined_test1_vlg_vec_tst/i1/b2v_inst/reg_fstate                   FETCH  DECODE  DECODE_BL_BIL  WRITE_B        FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/INST                         0000           602b
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_Link_Register/d         0010   0027    0028
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_Link_Register/q         0000   002f
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_program_counter/PC_IN   0001   0028    0034                          002b
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_program_counter/PC_OUT  0010   0027    0028
```

Figure 37: Simulation Result of B 2B Operation.

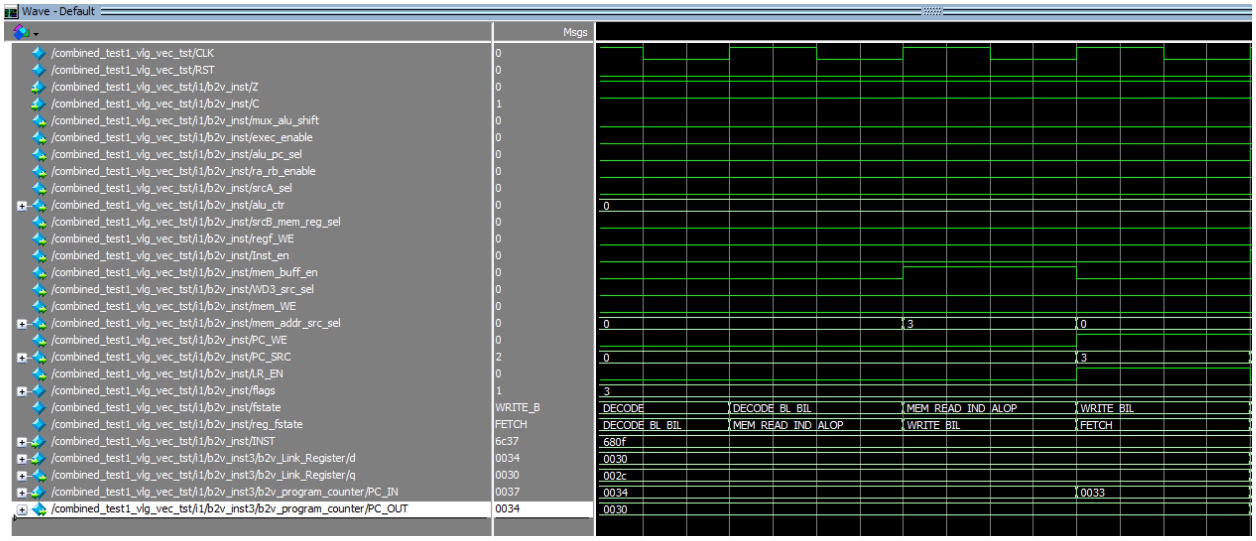
18) The simulation results for BL operation is given in Figure 38. The change in PC and LR can be seen below as 2f and 2c respectively.



```
                                                                     Msgs
/combined_test1_vlg_vec_tst/CLK                                      0
/combined_test1_vlg_vec_tst/RST                                      1
/combined_test1_vlg_vec_tst/i1/b2v_inst/Z                            1
/combined_test1_vlg_vec_tst/i1/b2v_inst/C                            1
/combined_test1_vlg_vec_tst/i1/b2v_inst/mux_alu_shift                0
/combined_test1_vlg_vec_tst/i1/b2v_inst/exec_enable                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_pc_sel                   0
/combined_test1_vlg_vec_tst/i1/b2v_inst/ra_rb_enable                 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcA_sel                     0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_ctr                      0      0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcB_mem_reg_sel             0
/combined_test1_vlg_vec_tst/i1/b2v_inst/regf_WE                      0
/combined_test1_vlg_vec_tst/i1/b2v_inst/Inst_en                      0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_buff_en                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/WD3_src_sel                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_WE                       0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_addr_src_sel             0      0
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_WE                        0
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_SRC                       0      0                                      2
/combined_test1_vlg_vec_tst/i1/b2v_inst/LR_EN                        0
/combined_test1_vlg_vec_tst/i1/b2v_inst/flags                        3      3
/combined_test1_vlg_vec_tst/i1/b2v_inst/fstate                       FETCH  FETCH   DECODE         DECODE_BL_BIL  WRITE_BL
/combined_test1_vlg_vec_tst/i1/b2v_inst/reg_fstate                   FETCH  DECODE  DECODE_BL_BIL  WRITE_BL       FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/INST                         0000   602b    642f
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_Link_Register/d         0010   002b    002c
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_Link_Register/q         0000   002f
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_program_counter/PC_IN   0001   002c    0034                          002f
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_program_counter/PC_OUT  0010   002b    002c
```

Figure 38: Simulation Result of BL 2F Operation.

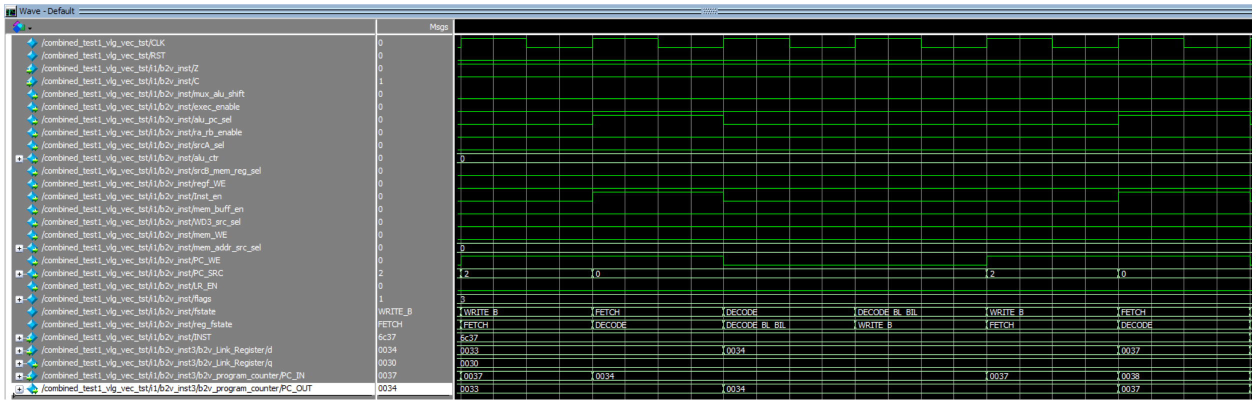
19) The simulation results for BIL operation is given in Figure 39. The change in PC and LR can be seen below as 33 and 30 respectively. Memory address $7 stores 0x33.



```
Wave - Default
                                                                  Msgs
/combined_test1_vlg_vec_tst/CLK                                   0
/combined_test1_vlg_vec_tst/RST                                   0
/combined_test1_vlg_vec_tst/i1/b2v_inst/Z                         0
/combined_test1_vlg_vec_tst/i1/b2v_inst/C                         1
/combined_test1_vlg_vec_tst/i1/b2v_inst/mux_alu_shift             0
/combined_test1_vlg_vec_tst/i1/b2v_inst/exec_enable               0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_pc_sel                0
/combined_test1_vlg_vec_tst/i1/b2v_inst/ra_rb_enable              0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcA_sel                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_ctr                   0        0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcB_mem_reg_sel          0
/combined_test1_vlg_vec_tst/i1/b2v_inst/regf_WE                   0
/combined_test1_vlg_vec_tst/i1/b2v_inst/Inst_en                   0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_buff_en               0
/combined_test1_vlg_vec_tst/i1/b2v_inst/WD3_src_sel               0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_WE                    0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_addr_src_sel          0        0 | 3 | 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_WE                     0
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_SRC                    2        0 | 3
/combined_test1_vlg_vec_tst/i1/b2v_inst/LR_EN                     0
/combined_test1_vlg_vec_tst/i1/b2v_inst/flags                     1        3
/combined_test1_vlg_vec_tst/i1/b2v_inst/fstate                    WRITE_B  DECODE | DECODE_BL_BIL | MEM_READ_IND_ALOP | WRITE_BIL
/combined_test1_vlg_vec_tst/i1/b2v_inst/reg_fstate                FETCH    DECODE_BL_BIL | MEM_READ_IND_ALOP | WRITE_BIL | FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/INST                      6c37     680f
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_Link_Register/d      0034     0030
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_Link_Register/q      0030     002c
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_program_counter/PC_IN  0037   0034 | 0033
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_program_counter/PC_OUT 0034   0030
```

Figure 39: Simulation Result of BIL $7 Operation.

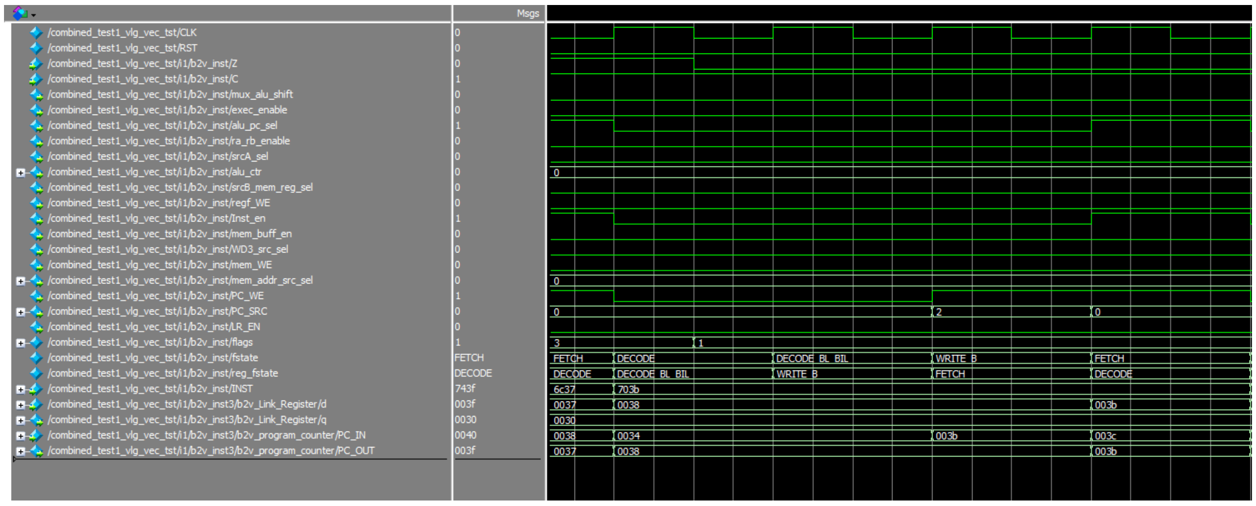
20) The simulation results for BIZ operation is given in Figure 40. The change in PC can be seen below as 37.



```
Wave - Default
                                                                  Msgs
/combined_test1_vlg_vec_tst/CLK                                   0
/combined_test1_vlg_vec_tst/RST                                   0
/combined_test1_vlg_vec_tst/i1/b2v_inst/Z                         0
/combined_test1_vlg_vec_tst/i1/b2v_inst/C                         1
/combined_test1_vlg_vec_tst/i1/b2v_inst/mux_alu_shift             0
/combined_test1_vlg_vec_tst/i1/b2v_inst/exec_enable               0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_pc_sel                0
/combined_test1_vlg_vec_tst/i1/b2v_inst/ra_rb_enable              0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcA_sel                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_ctr                   0        0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcB_mem_reg_sel          0
/combined_test1_vlg_vec_tst/i1/b2v_inst/regf_WE                   0
/combined_test1_vlg_vec_tst/i1/b2v_inst/Inst_en                   0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_buff_en               0
/combined_test1_vlg_vec_tst/i1/b2v_inst/WD3_src_sel               0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_WE                    0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_addr_src_sel          0        0
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_WE                     0
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_SRC                    2        2 | 0 | 2 | 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/LR_EN                     0
/combined_test1_vlg_vec_tst/i1/b2v_inst/flags                     1        3
/combined_test1_vlg_vec_tst/i1/b2v_inst/fstate                    WRITE_B  WRITE_B | FETCH | DECODE | DECODE_BL_BIL | WRITE_B | FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/reg_fstate                FETCH    FETCH | DECODE | DECODE_BL_BIL | WRITE_B | FETCH | DECODE
/combined_test1_vlg_vec_tst/i1/b2v_inst/INST                      6c37     6c37
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_Link_Register/d      0034     0033 | 0034 | 0037
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_Link_Register/q      0030     0030
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_program_counter/PC_IN  0037   0037 | 0034 | 0037 | 0038
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_program_counter/PC_OUT 0034   0033 | 0034 | 0037
```

Figure 40: Simulation Result of BIZ 0x37 Operation.

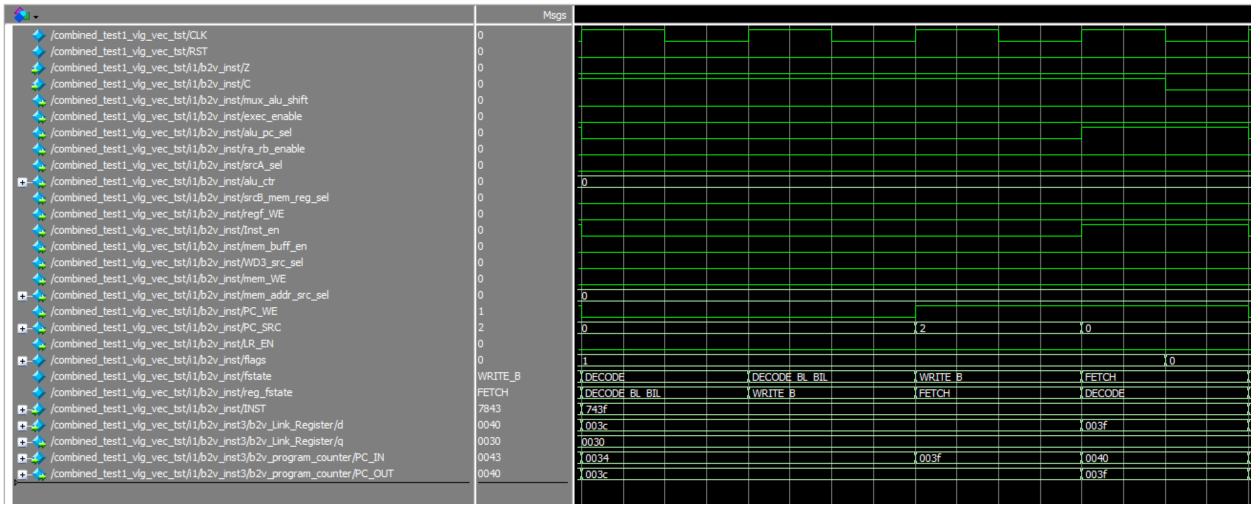
21) The simulation results for BNZ operation is given in Figure 41. The change in PC can be seen below as 3b.



```
/combined_test1_vlg_vec_tst/CLK                                  0
/combined_test1_vlg_vec_tst/RST                                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/Z                        0
/combined_test1_vlg_vec_tst/i1/b2v_inst/C                        1
/combined_test1_vlg_vec_tst/i1/b2v_inst/mux_alu_shift            0
/combined_test1_vlg_vec_tst/i1/b2v_inst/exec_enable              0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_pc_sel               1
/combined_test1_vlg_vec_tst/i1/b2v_inst/ra_rb_enable             0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcA_sel                 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_ctr                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcB_mem_reg_sel         0
/combined_test1_vlg_vec_tst/i1/b2v_inst/regf_WE                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/Inst_en                  1
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_buff_en              0
/combined_test1_vlg_vec_tst/i1/b2v_inst/WD3_src_sel              0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_WE                   0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_addr_src_sel         0
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_WE                    1
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_SRC                   0
/combined_test1_vlg_vec_tst/i1/b2v_inst/LR_EN                    0
/combined_test1_vlg_vec_tst/i1/b2v_inst/flags                    1
/combined_test1_vlg_vec_tst/i1/b2v_inst/fstate                   FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/reg_fstate               DECODE
/combined_test1_vlg_vec_tst/i1/b2v_inst/INST                     743f
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_Link_Register/d     003f
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_Link_Register/q     0030
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_program_counter/PC_IN   0040
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_program_counter/PC_OUT  003f
```

Figure 41: Simulation Result of BNZ 3B Operation.

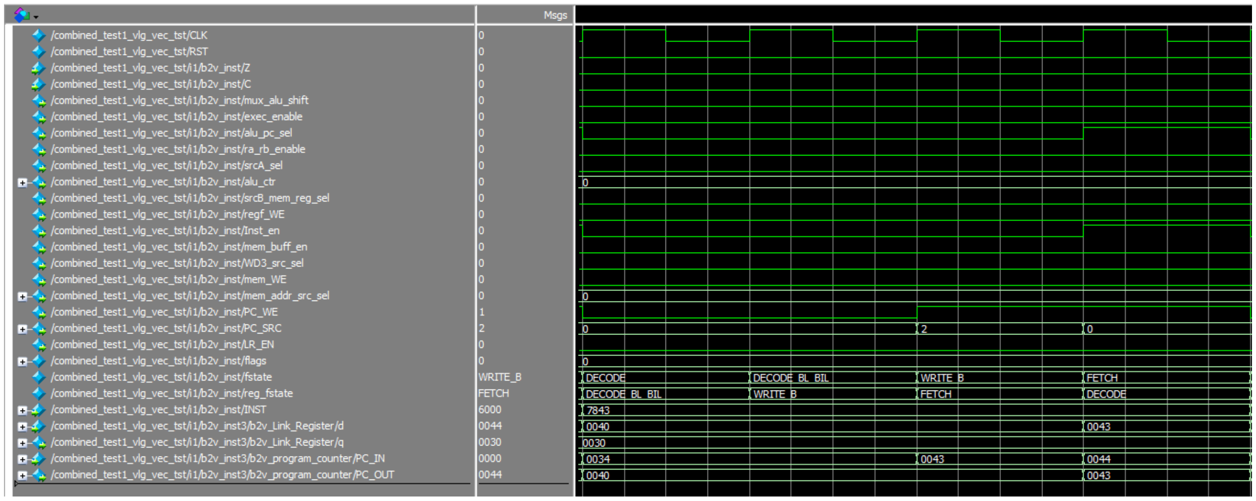
22) The simulation results for BIC operation is given in Figure 42. The change in PC can be seen below as 3f.



```
/combined_test1_vlg_vec_tst/CLK                                  0
/combined_test1_vlg_vec_tst/RST                                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/Z                        0
/combined_test1_vlg_vec_tst/i1/b2v_inst/C                        0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mux_alu_shift            0
/combined_test1_vlg_vec_tst/i1/b2v_inst/exec_enable              0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_pc_sel               0
/combined_test1_vlg_vec_tst/i1/b2v_inst/ra_rb_enable             0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcA_sel                 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_ctr                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcB_mem_reg_sel         0
/combined_test1_vlg_vec_tst/i1/b2v_inst/regf_WE                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/Inst_en                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_buff_en              0
/combined_test1_vlg_vec_tst/i1/b2v_inst/WD3_src_sel              0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_WE                   0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_addr_src_sel         0
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_WE                    1
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_SRC                   2
/combined_test1_vlg_vec_tst/i1/b2v_inst/LR_EN                    0
/combined_test1_vlg_vec_tst/i1/b2v_inst/flags                    0
/combined_test1_vlg_vec_tst/i1/b2v_inst/fstate                   WRITE_B
/combined_test1_vlg_vec_tst/i1/b2v_inst/reg_fstate               FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/INST                     7843
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_Link_Register/d     0040
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_Link_Register/q     0030
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_program_counter/PC_IN   0043
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_program_counter/PC_OUT  0040
```

Figure 42: Simulation Result of BIC 3f Operation.

23) The simulation results for BNC operation is given in Figure 43. The change in PC can be seen below as 43.



```
                                                                    Msgs
/combined_test1_vlg_vec_tst/CLK                                     0
/combined_test1_vlg_vec_tst/RST                                     0
/combined_test1_vlg_vec_tst/i1/b2v_inst/Z                           0
/combined_test1_vlg_vec_tst/i1/b2v_inst/C                           0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mux_alu_shift               0
/combined_test1_vlg_vec_tst/i1/b2v_inst/exec_enable                 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_pc_sel                  0
/combined_test1_vlg_vec_tst/i1/b2v_inst/ra_rb_enable                0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcA_sel                    0
/combined_test1_vlg_vec_tst/i1/b2v_inst/alu_ctr                     0
/combined_test1_vlg_vec_tst/i1/b2v_inst/srcB_mem_reg_sel            0
/combined_test1_vlg_vec_tst/i1/b2v_inst/regf_WE                     0
/combined_test1_vlg_vec_tst/i1/b2v_inst/Inst_en                     0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_buff_en                 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/WD3_src_sel                 0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_WE                      0
/combined_test1_vlg_vec_tst/i1/b2v_inst/mem_addr_src_sel            0
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_WE                       1
/combined_test1_vlg_vec_tst/i1/b2v_inst/PC_SRC                      2
/combined_test1_vlg_vec_tst/i1/b2v_inst/LR_EN                       0
/combined_test1_vlg_vec_tst/i1/b2v_inst/flags                       0
/combined_test1_vlg_vec_tst/i1/b2v_inst/fstate                      WRITE_B
/combined_test1_vlg_vec_tst/i1/b2v_inst/reg_fstate                  FETCH
/combined_test1_vlg_vec_tst/i1/b2v_inst/INST                        6000
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_Link_Register/d        0044
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_Link_Register/q        0030
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_program_counter/PC_IN  0000
/combined_test1_vlg_vec_tst/i1/b2v_inst3/b2v_program_counter/PC_OUT 0044
```

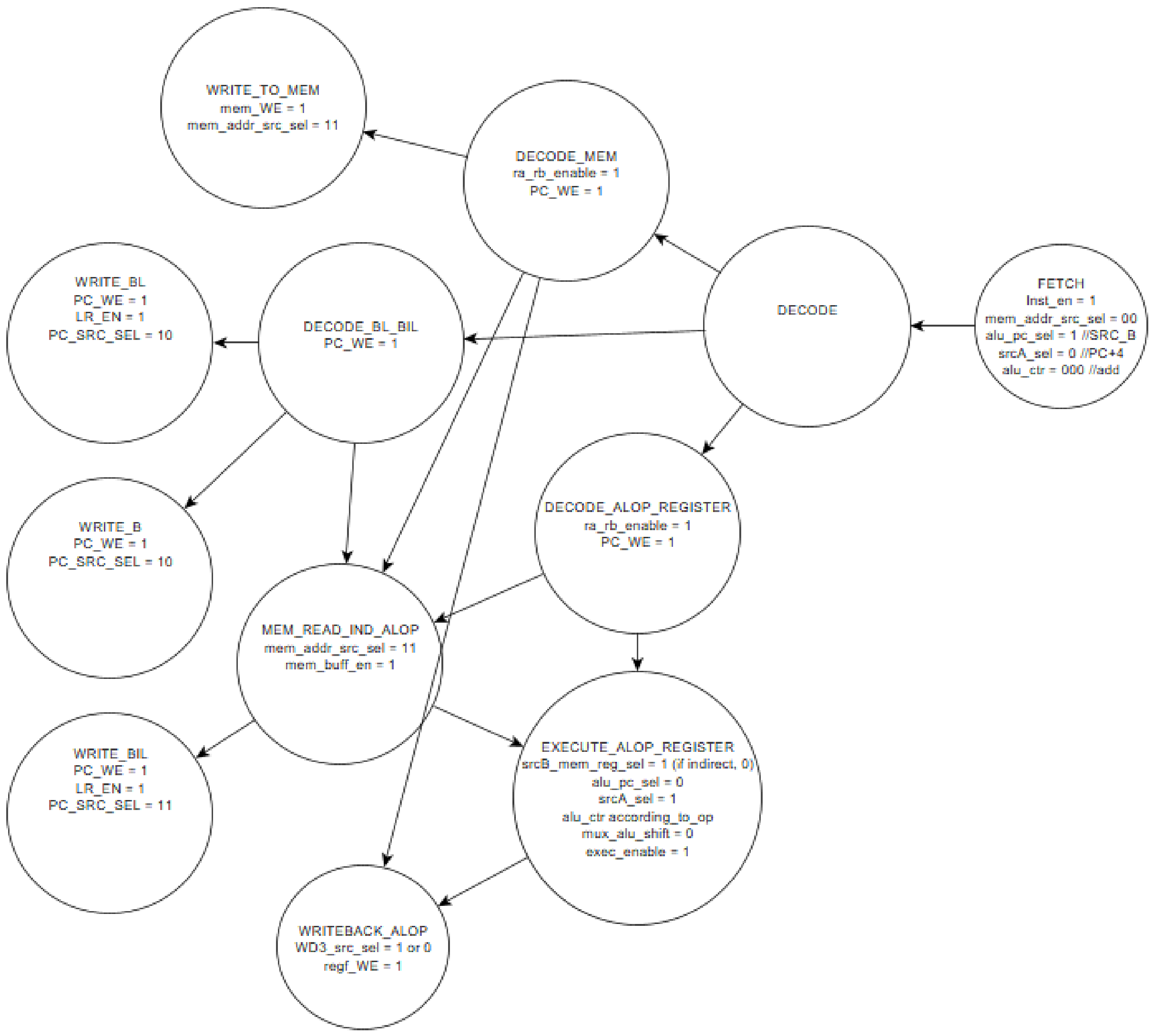
0
0
0 2 0
0
DECODE DECODE_BL_BIL WRITE_B FETCH
DECODE_BL_BIL WRITE_B FETCH DECODE
7843
0040 0043
0030
0034 0043 0044
0040 0043

Figure 43: Simulation Result of SDM R2, $7 Operation.

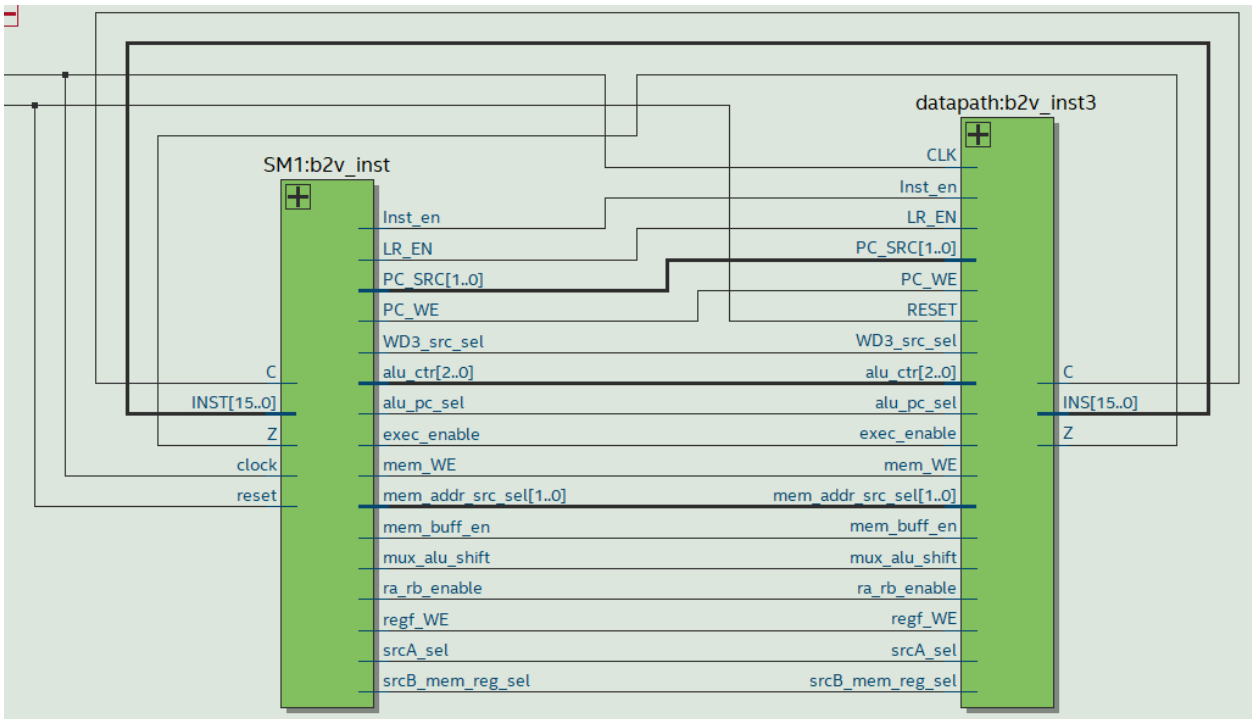
Conclusion

To summarize, in this report an ISA, and a Datapath design is provided. The design is tested by using a controller. However, the details of the controller will not be provided in this report. As can be seen from the tests, datapath works properly for all instruction. Note that, in this multicycle datapath design, only the used parts are needed to be controlled. Thus, other parts are don't care (except write enable signals). When the results are verified by the user, ASM chart can provide help to understand the test results. However, the control signals are also given within the simulation results.

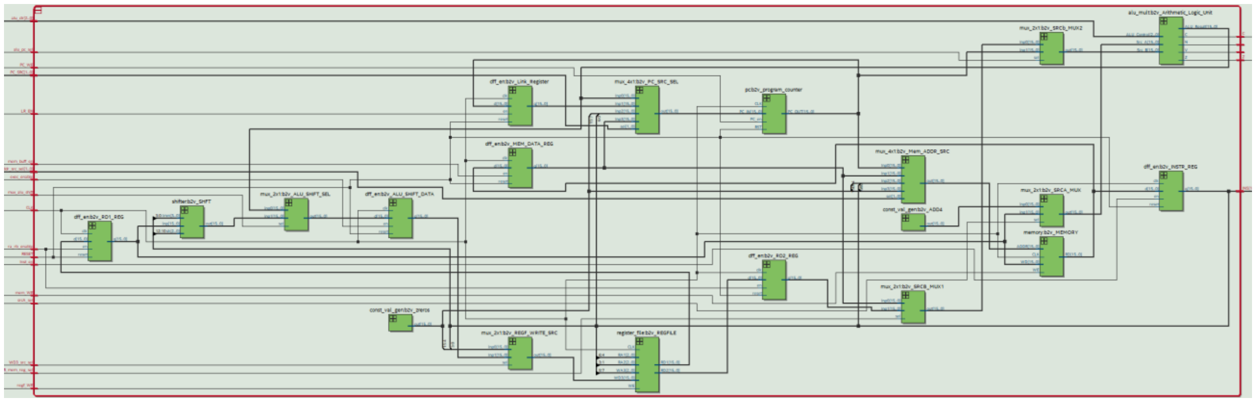
Appendix



A1: ASM Chart of the Multicycle CPU.



A2: Complete Multicycle CPU RTL View.



A3: RTL View of Datapath.