# 16-BIT MICROPROCESSOR USING VERILOG

EC-448 Major Project

By Soham Patel (16EC085)

# CONTENTS

# OBJECTIVE

- To design a 16-bit microprocessor (RISC)
- Develop using Xilinx ISE Design Suite and verilog HDL

# SPECIFICATIONS

- 16-bit instruction set
- Interface 64k memory
- ALU supports signed binary
- Harvard Architecture
- 8 general purpose registers
- 18 executable instructions

# INSTRUCTION TABLE

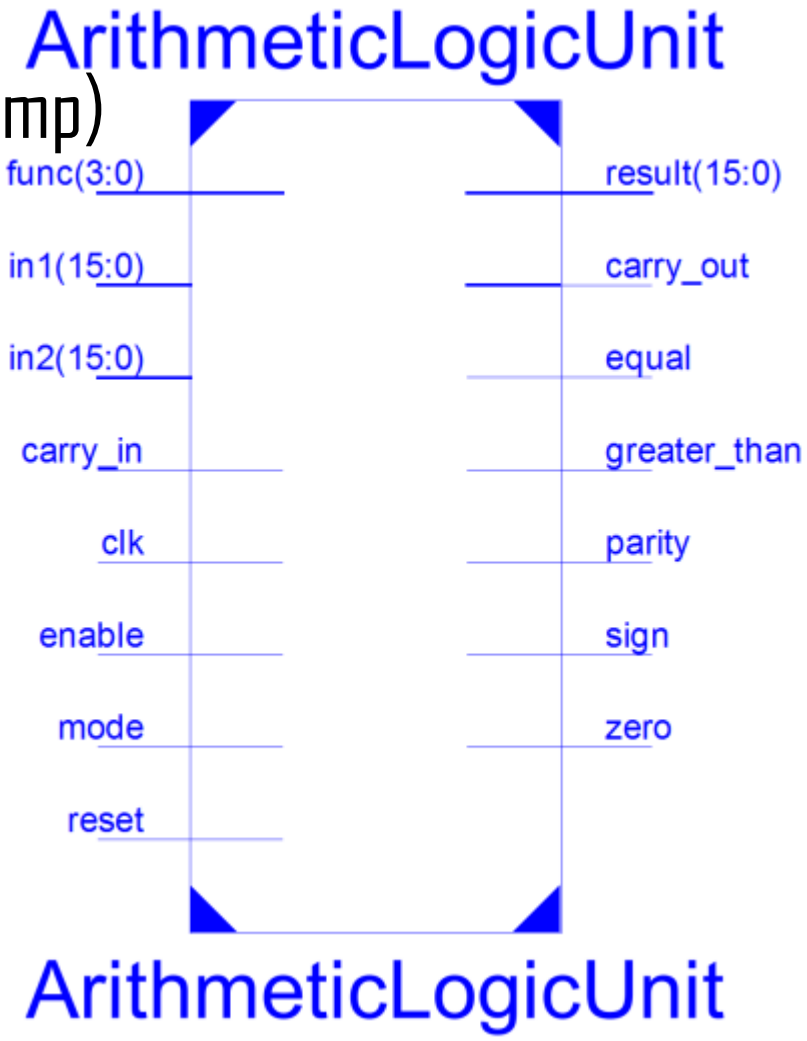
| Sr. No | Instruction Type | Instruction | 1st Cycle | | 2nd Cycle | | Information |
|---|---|---|---|---|---|---|---|
| | | | 1st Byte (reg) | 2nd Byte (opcode) | 2 Bytes (data/addr) | | |
| 1 | Load / Store operations | LDM | 00 to 07 | 01 | Address | | Loads data from memory address to register |
| 2 | | LDR | 00 to 07 (source) | 02 | 00 | 00 to 07 (other) | Loads data from source register to other register |
| 3 | | LDV | 00 to 07 | 03 | Data | | Loads data to register |
| 4 | | STR | 00 to 07 | 04 | Address | | Stores data from register to memory address |
| 5 | Arithmetic operations | ADR | 00 | 10 | 0000 | | Adds data from reg 0 and reg 1 |
| 6 | | ADD | 00 | 11 | Data | | Adds data to accumulator (reg 0) |
| 7 | | SBR | 00 | 12 | 0000 | | Substracts data in reg 0 from reg 1 (r0 - r1) |
| 8 | | SUB | 00 | 13 | Data | | Substracts data in accumulator (r0) from given data |
| 9 | | INC | 00 | 14 | 0000 | | Increments the value of accumulator (r0) by 1 |
| 10 | | DEC | 00 | 15 | 0000 | | Decrements the value of accumulator (r0) by 1 |
| 11 | Logical operations | XOR | 00 | 1f | Data | | Performs xor operation on accumulator and given data |
| 12 | | AND | 00 | 1e | Data | | Performs and operation on accumulator and given data |
| 13 | | OR | 00 | 1d | Data | | Performs or operation on accumulator and given data |
| 14 | | XNOR | 00 | 1c | Data | | Performs xnor operation on accumulator and given data |
| 15 | | NAND | 00 | 1b | Data | | Performs nand operation on accumulator and given data |
| 16 | | NOR | 00 | 1a | Data | | Performs nor operation on accumulator and given data |
| 17 | | INV | 00 | 19 | 0000 | | Performs inverter operation on accumulator |
| 18 | Other operation | NOP | 00 | 00 | 0000 | | Performs no operation |
| 19 | | REG | 00 to 07 | ff | 0000 | | Loads data of source register on data bus |

# BLOCKS DEVELOPED

- ALU
- Instruction Register
- Program Counter
- Memory
- Flag Register
- Decoder
- General Purpose Register
- Top module

# ALU

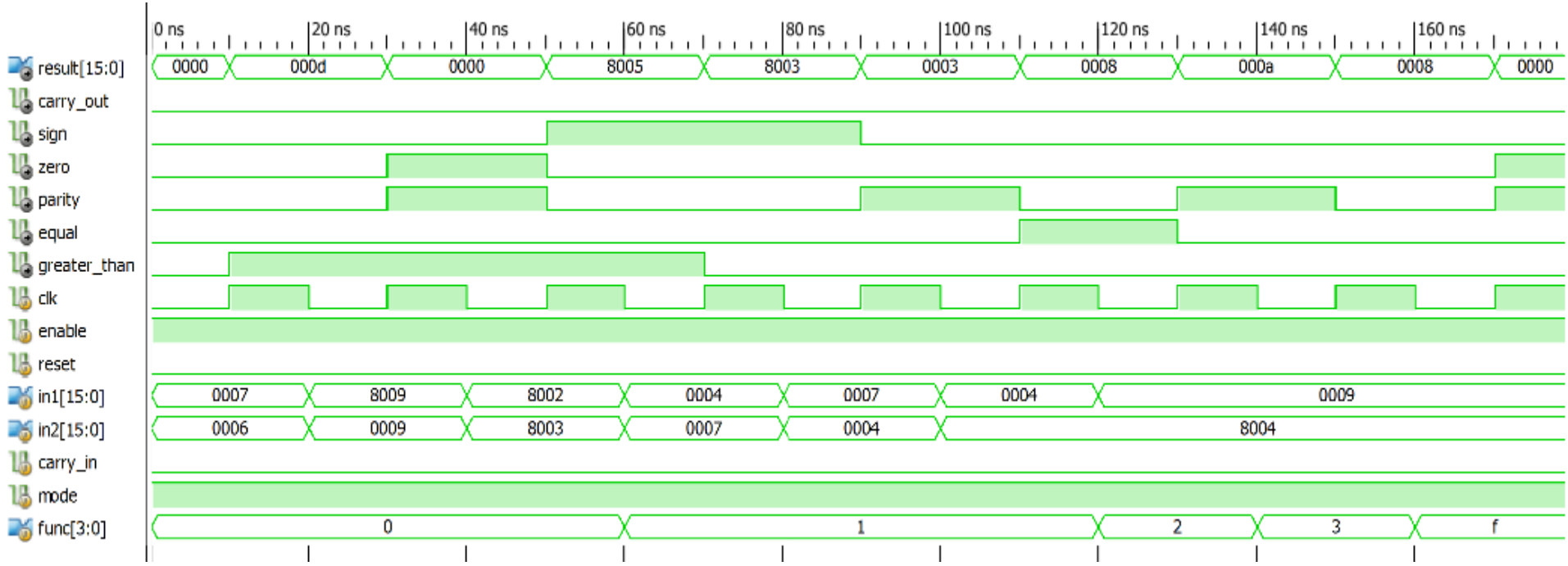
- Verilog functions (add, inv, 2's comp)
- I/O 16-bit SM format
- 2's comp
- 2 modes (A and L)
- Around 15 functions
- Carry look-ahead adder



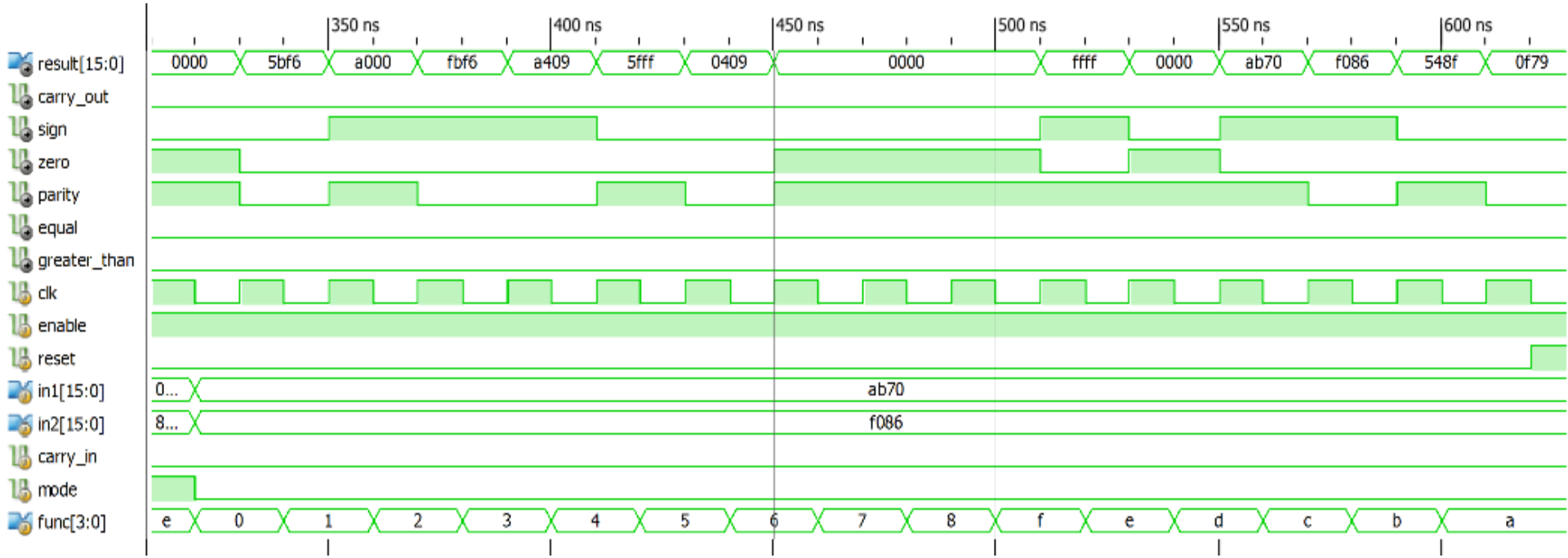
# ALU FUNCTIONS

| Arithmetic functions (Mode 1) | Logic Functions (Mode 0) |
|---|---|
| 0 Addition | 0 XOR |
| 1 Substraction | 1 AND |
| 2 Increment | 2 OR |
| 3 Decrement | 3 XNOR |
| | 4 NAND |
| | 5 NOR |
| | A INV IN1 |
| | B INV IN2 |
| | C IN2 |
| | D IN1 |
| | E LOW |
| | F HIGH |

# AU TEST


0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
160 ns
result[15:0]
0000
000d
0000
8005
8003
0003
0008
000a
0008
0000
carry_out
sign
zero
parity
equal
greater_than
clk
enable
reset
in1[15:0]
0007
8009
8002
0004
0007
0004
0009
in2[15:0]
0006
0009
8003
0007
0004
8004
carry_in
mode
func[3:0]
0
1
2
3
f

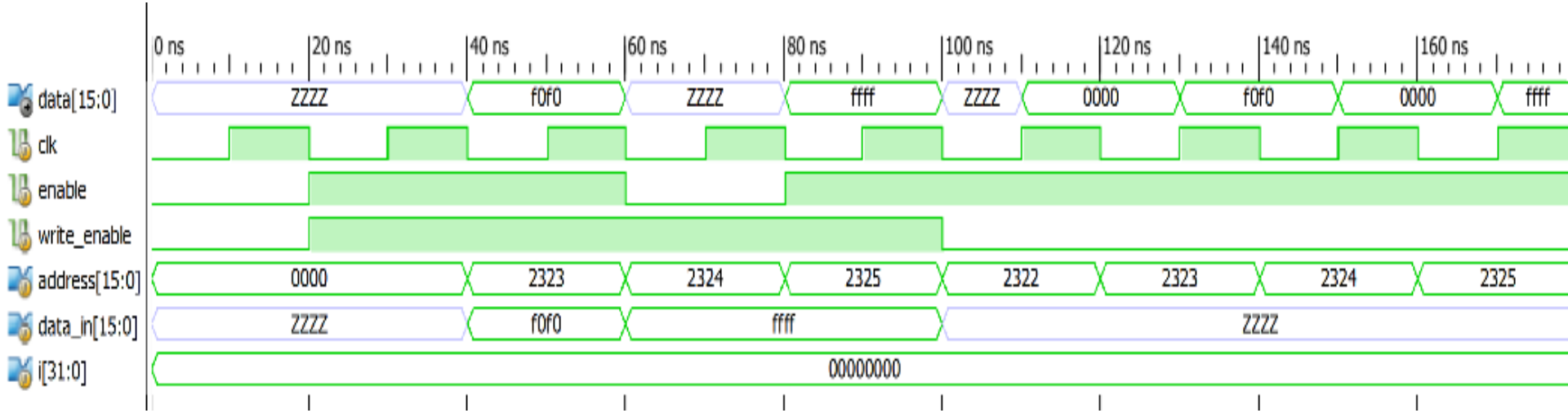
# LU TEST


350 ns
400 ns
450 ns
500 ns
550 ns
600 ns
result[15:0]
0000
5bf6
a000
fbf6
a409
5fff
0409
0000
ffff
0000
ab70
f086
548f
0f79
carry_out
sign
zero
parity
equal
greater_than
clk
enable
reset
in1[15:0]
0...
ab70
in2[15:0]
8...
f086
carry_in
mode
func[3:0]
e
0
1
2
3
4
5
6
7
8
f
e
d
c
b
a

# MEMORY

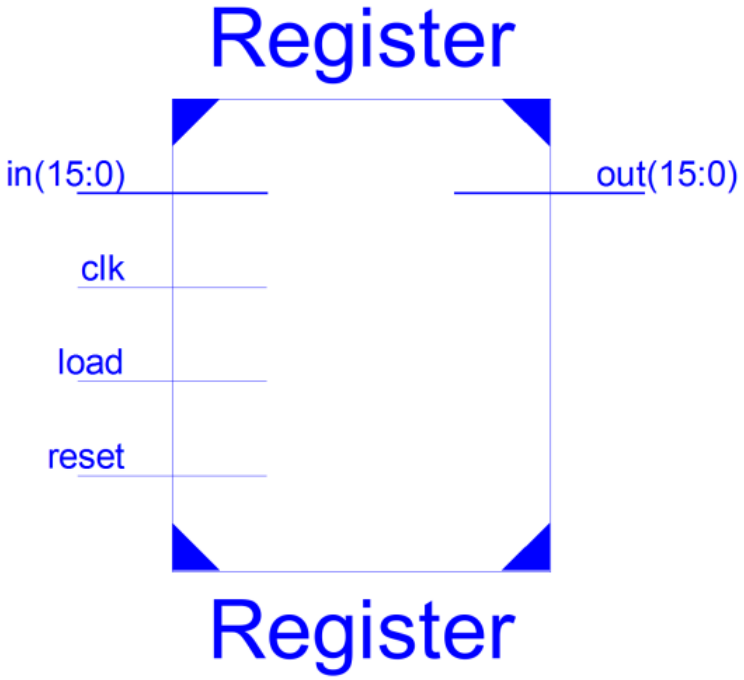
- Two-dimension array of reg variable
- Read, Write and Idle modes
- "$readmemb" file io system task
- Control output using Enable pin
- Data I/O using same bus
- Memory containing instruction or machine code

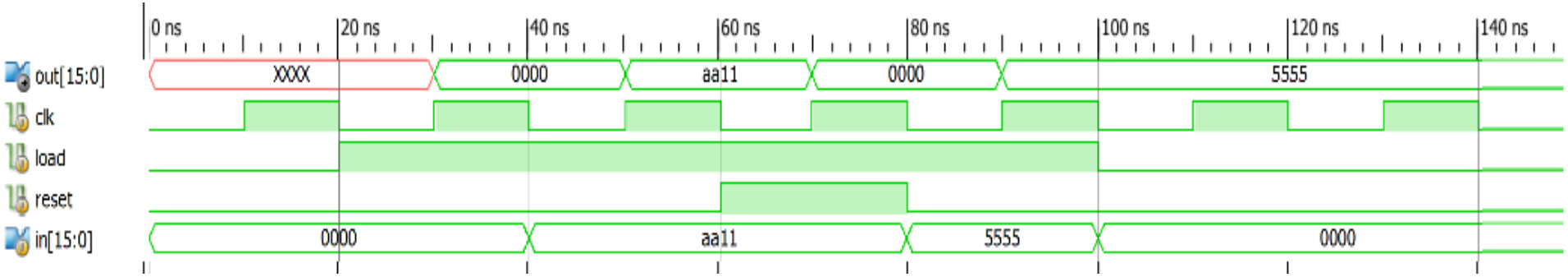
# TEST MEMORY


0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
160 ns
data[15:0]
ZZZZ
f0f0
ZZZZ
ffff
ZZZZ
0000
f0f0
0000
ffff
clk
enable
write_enable
address[15:0]
0000
2323
2324
2325
2322
2323
2324
2325
data_in[15:0]
ZZZZ
f0f0
ffff
ZZZZ
i[31:0]
00000000

# REGISTER

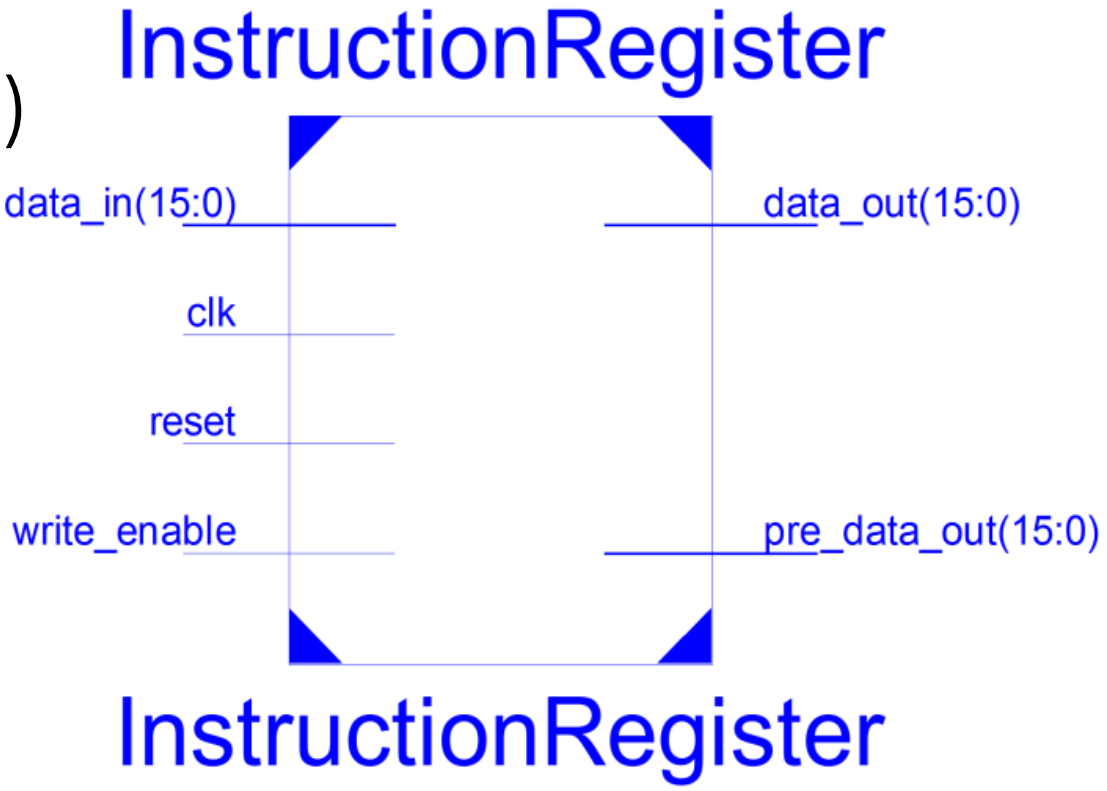
- Positive edge triggered
- Control using Load



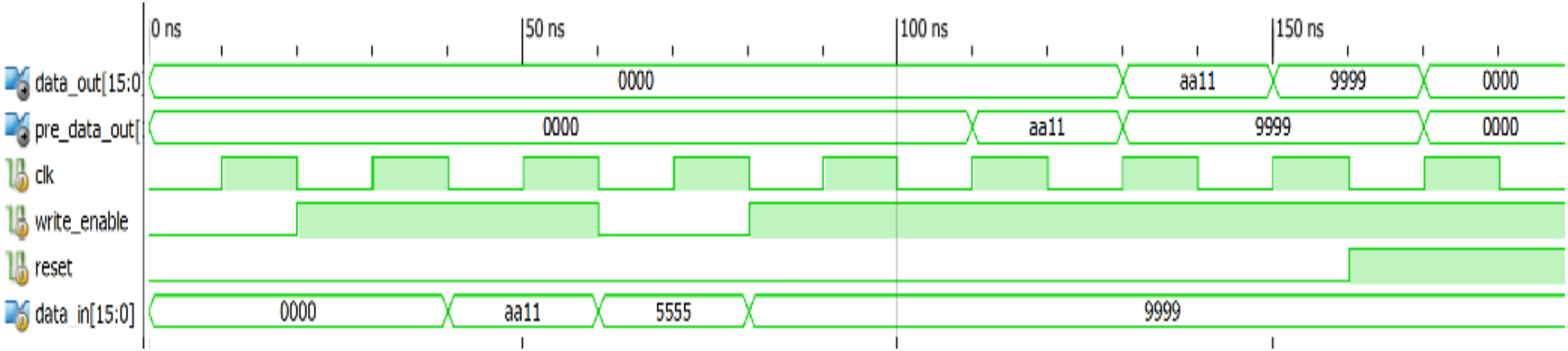



# INSTRUCTION REGISTER

- 16-bit
- 4 stage PFU (queue)
- Control using Write Enable

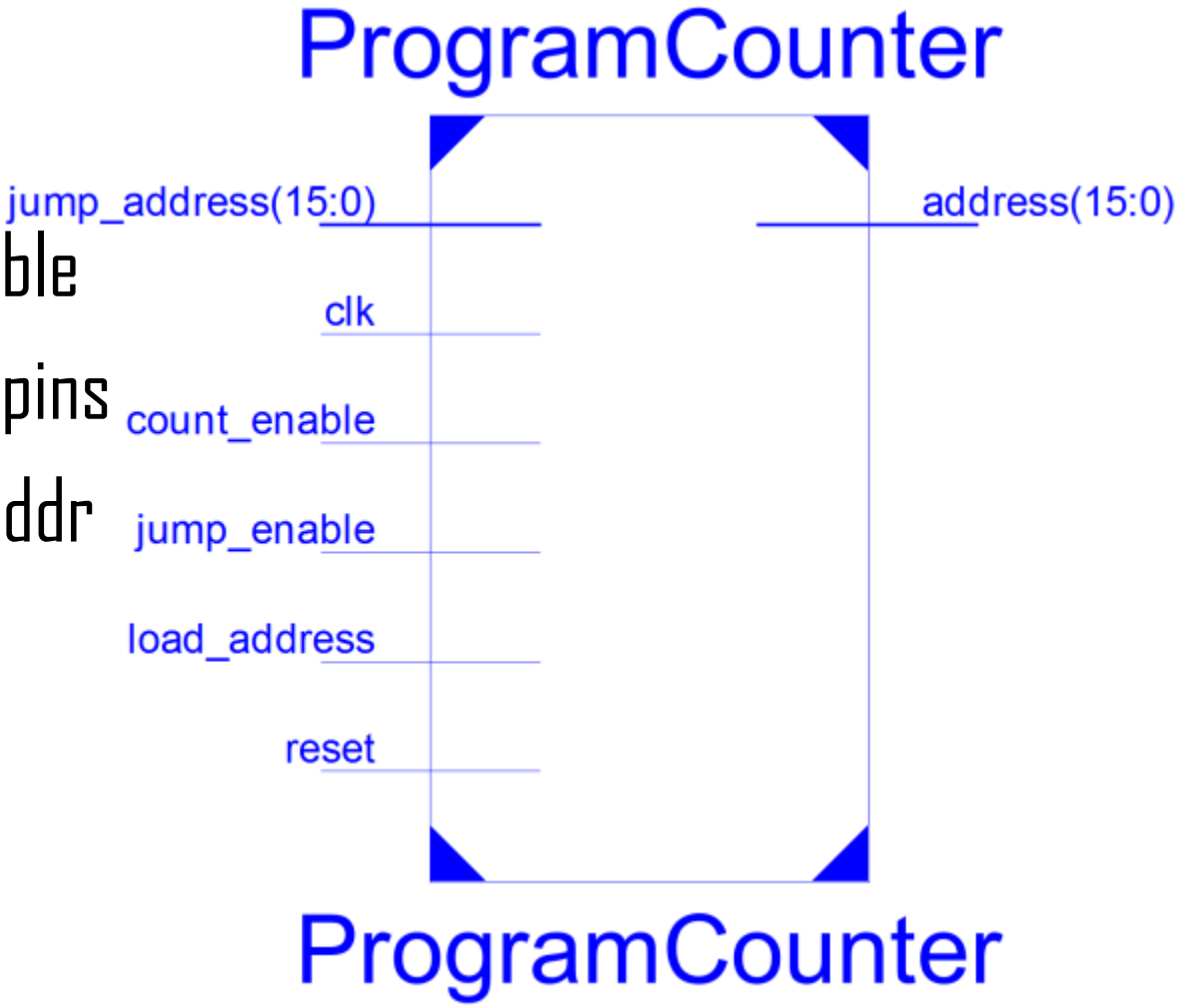


# IR TEST

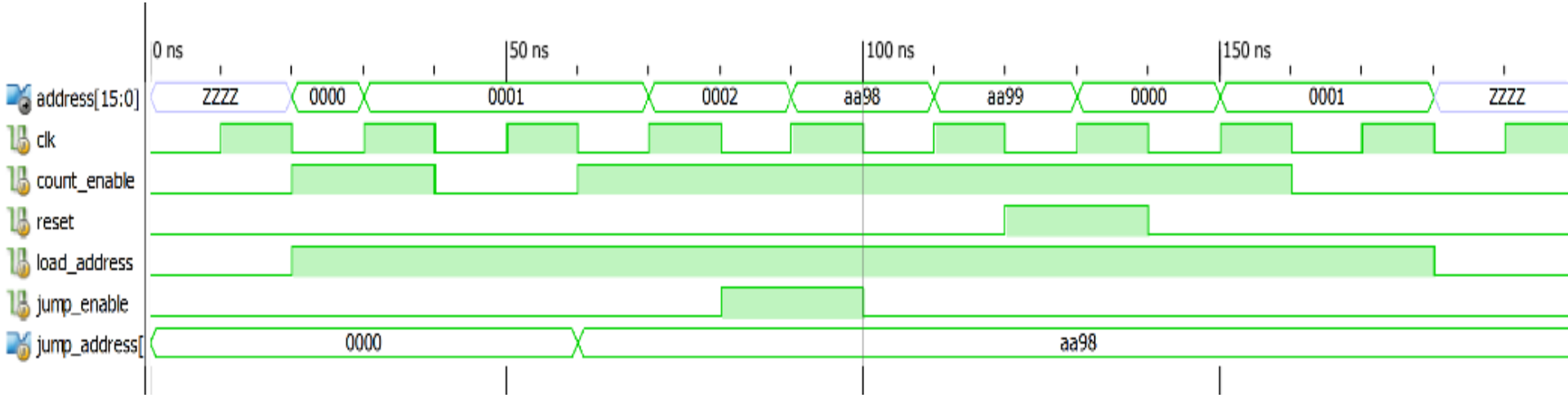

0 ns
50 ns
100 ns
150 ns
data_out[15:0]
0000
aa11
9999
0000
pre_data_out[
0000
aa11
9999
0000
clk
write_enable
reset
data_in[15:0]
0000
aa11
5555
9999

# PROGRAM COUNTER

- 16-bit counter
- Jump and jump_enable
- Clear, count enable pins
- Control using load addr

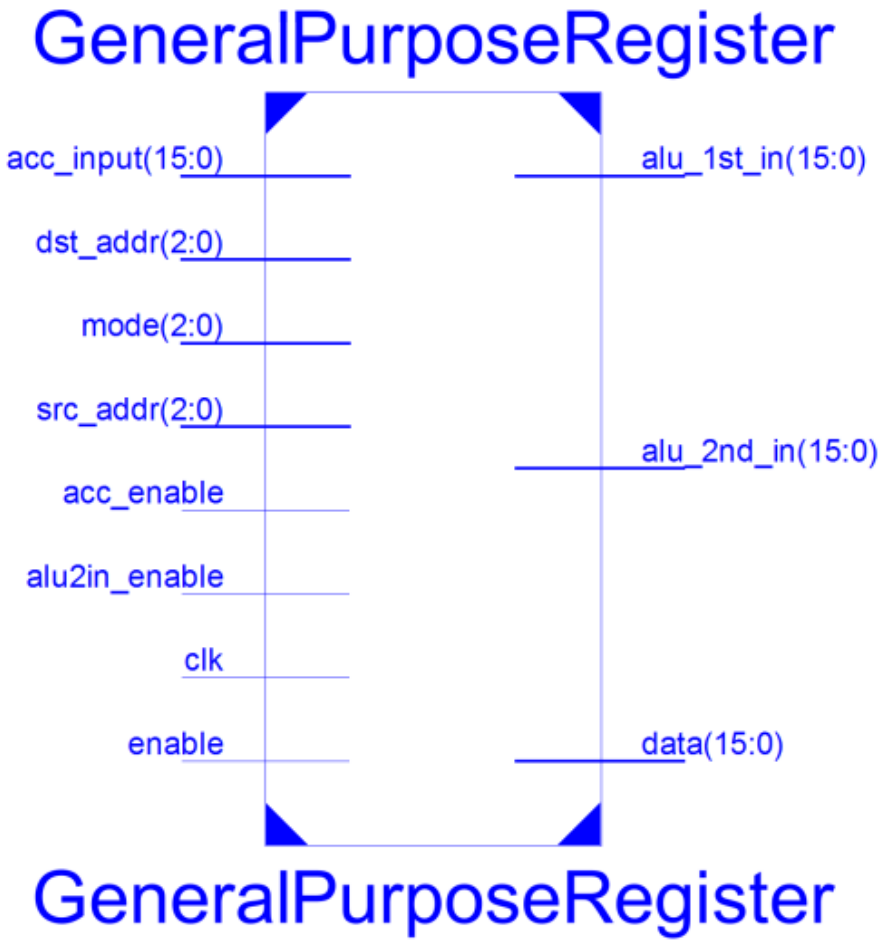


# PC TEST

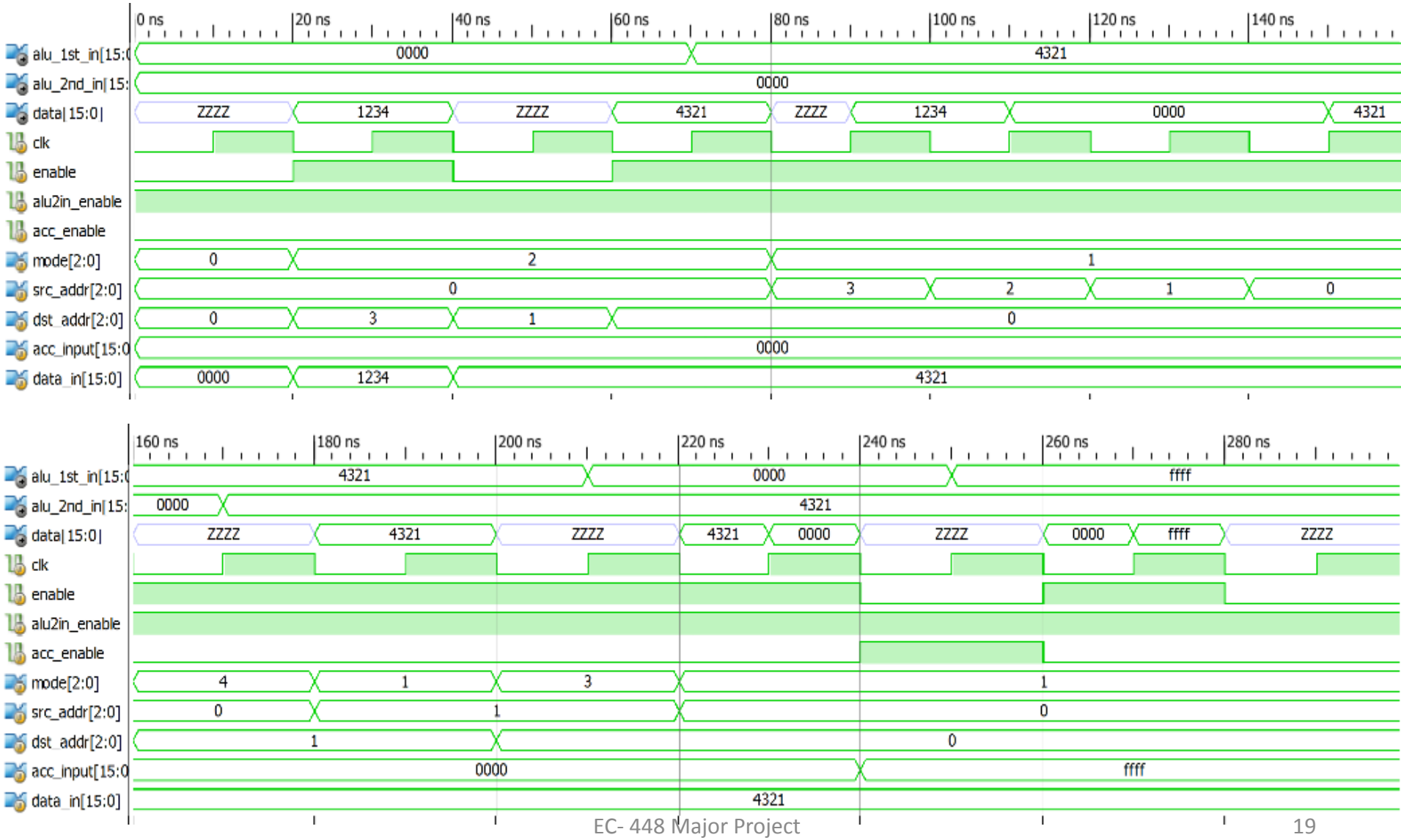

0 ns
50 ns
100 ns
150 ns
address[15:0]
ZZZZ
0000
0001
0002
aa98
aa99
0000
0001
ZZZZ
clk
count_enable
reset
load_address
jump_enable
jump_address[
0000
aa98

# GENERAL PURPOSE REGISTER

- Read, Write, Reg Transfer, Idle modes
- 16-bit regiters
- I/O using same data bus
- Src and Dst Addr bus

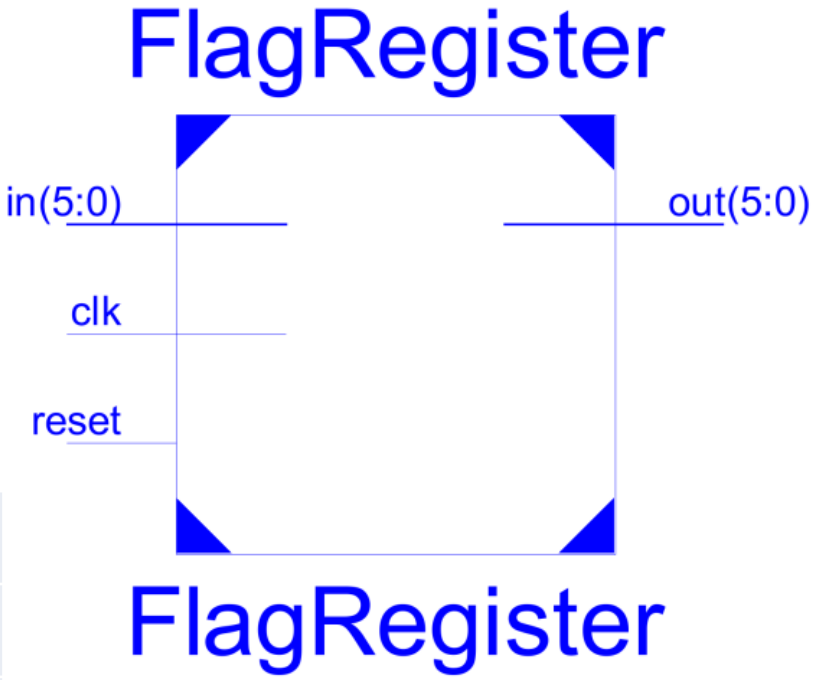


# GPR TEST


0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
alu_1st_in[15:0]
0000
4321
alu_2nd_in[15:
0000
data[15:0]
ZZZZ
1234
ZZZZ
4321
ZZZZ
1234
0000
4321
clk
enable
alu2in_enable
acc_enable
mode[2:0]
0
2
1
src_addr[2:0]
0
3
2
1
0
dst_addr[2:0]
0
3
1
0
acc_input[15:0]
0000
data_in[15:0]
0000
1234
4321
160 ns
180 ns
200 ns
220 ns
240 ns
260 ns
280 ns
alu_1st_in[15:0]
4321
0000
ffff
alu_2nd_in[15:
0000
4321
data[15:0]
ZZZZ
4321
ZZZZ
4321
0000
ZZZZ
0000
ffff
ZZZZ
clk
enable
alu2in_enable
acc_enable
mode[2:0]
4
1
3
1
src_addr[2:0]
0
1
0
dst_addr[2:0]
1
0
acc_input[15:0]
0000
ffff
data_in[15:0]
4321

# FLAG REGISTER

- Controlled by ALU
- Useable by Decoder
- 6 – bit

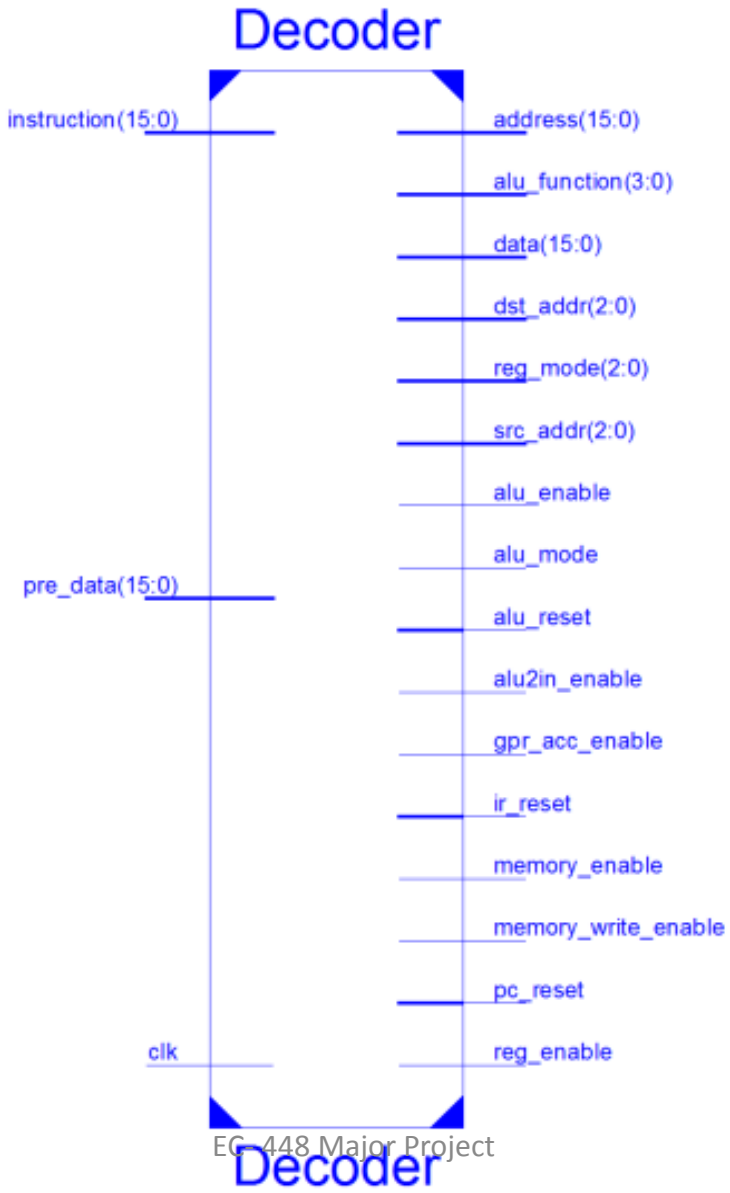



| FLAG | Purpose |
|---|---|
| 0 | Carry |
| 1 | Zero |
| 2 | Sign |
| 3 | Greater than |
| 4 | Equal |
| 5 | Parity |

# DECODER/ CONTROL UNIT

- Decodes instruction set
- Generates appropriate control signals
- Commands all components
- Significant component

# DECODER BLOCK



# TOP MODULE

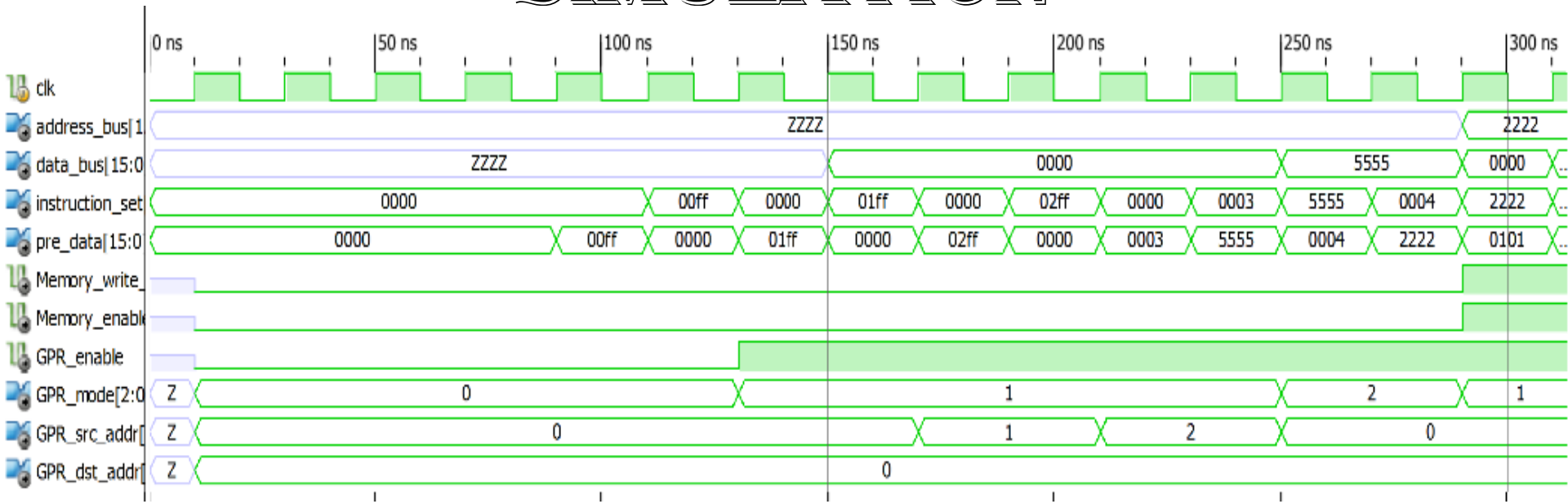
- All components are assembled
- Only clock is provided
- Interconnection of buses and wires

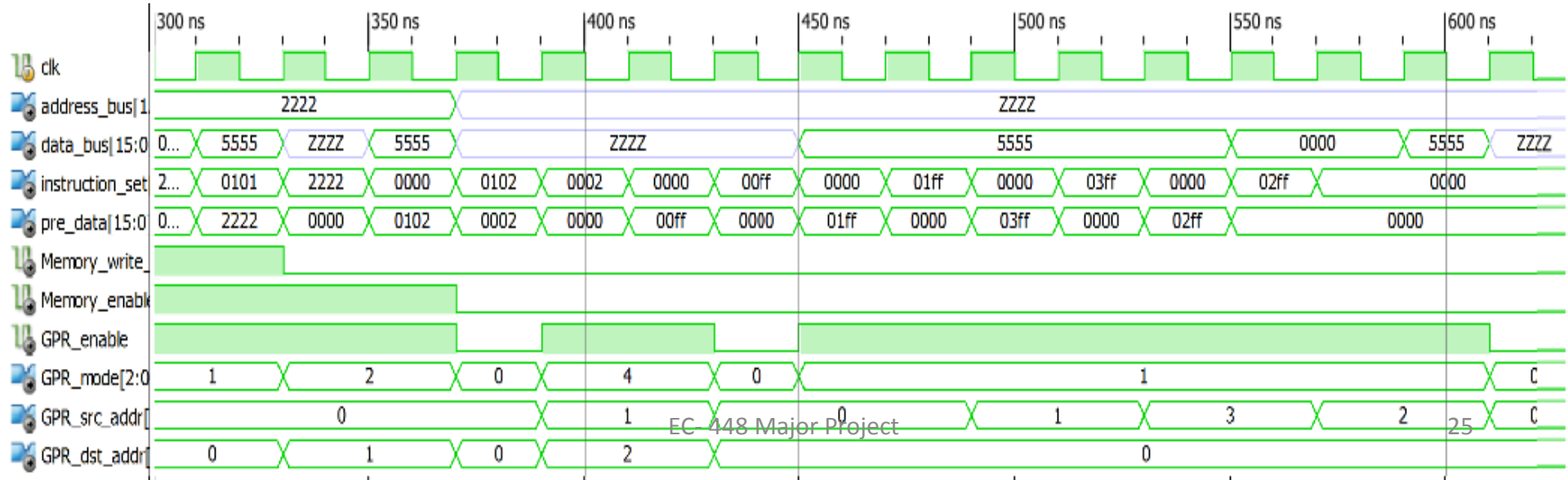
# TEST 1

**Program :-**

```
NOP                                                                    0000
REG R0 // loads data in register 0 to data bus                         00ff
REG R1 // loads data in register 1 to data bus                         0000
REG R2 // loads data in register 2 to data bus                         01ff
LDV R0, 5555H // loads 5555 in register 0                              0000
STR R0, [2222H] // stores data in register 0 to memory location 2222   02ff
LDM R1, [2222H] // loads data from memory location 2222 to register 1  0000
NOP                                                                    0003
LDR R2, R1 // loads data from register 1 to register 2                 5555
NOP                                                                    0004
REG R0 // loads data in register 0 to data bus                         2222
REG R1 // loads data in register 1 to data bus                         0101
REG R3 // loads data in register 1 to data bus                         2222
REG R2 // loads data in register 2 to data bus                         0000
NOP                                                                    0102
                                                                       0002
                                                                       0000
                                                                       00ff
                                                                       0000
                                                                       01ff
                                                                       0000
                                                                       03ff
                                                                       0000
                                                                       02ff
                                                                       0000
                                                                       0000
```

# SIMULATION


0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
clk
address_bus[1
data_bus[15:0
instruction_set
pre_data[15:0
Memory_write_
Memory_enabl
GPR_enable
GPR_mode[2:0
GPR_src_addr[
GPR_dst_addr[
ZZZZ
2222
0000
5555
00ff
01ff
02ff
0003
0004
0101


300 ns
350 ns
400 ns
450 ns
500 ns
550 ns
600 ns
clk
address_bus[1
data_bus[15:0
instruction_set
pre_data[15:0
Memory_write_
Memory_enabl
GPR_enable
GPR_mode[2:0
GPR_src_addr[
GPR_dst_addr[
2222
ZZZZ
5555
0000
0101
0102
0002
00ff
01ff
03ff
02ff

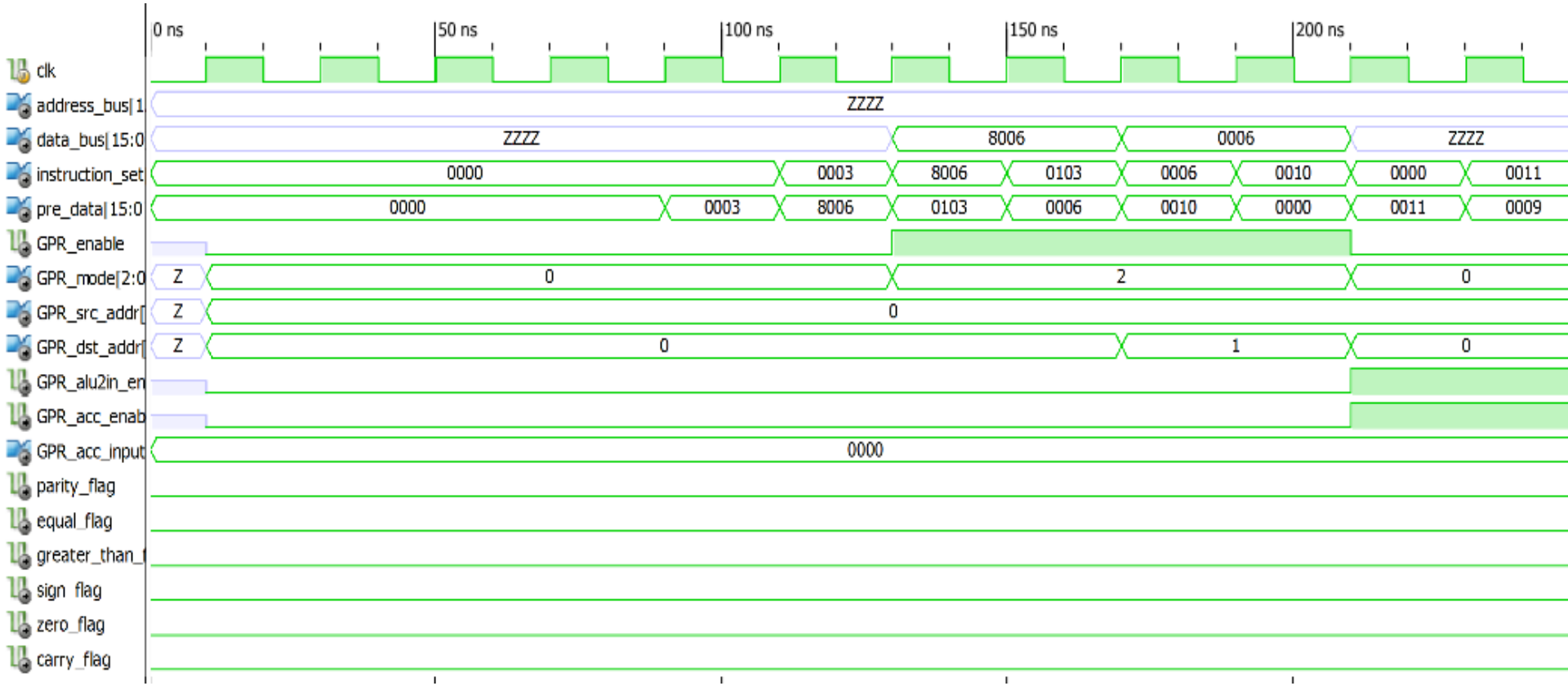
# TEST 2

**Program :-**

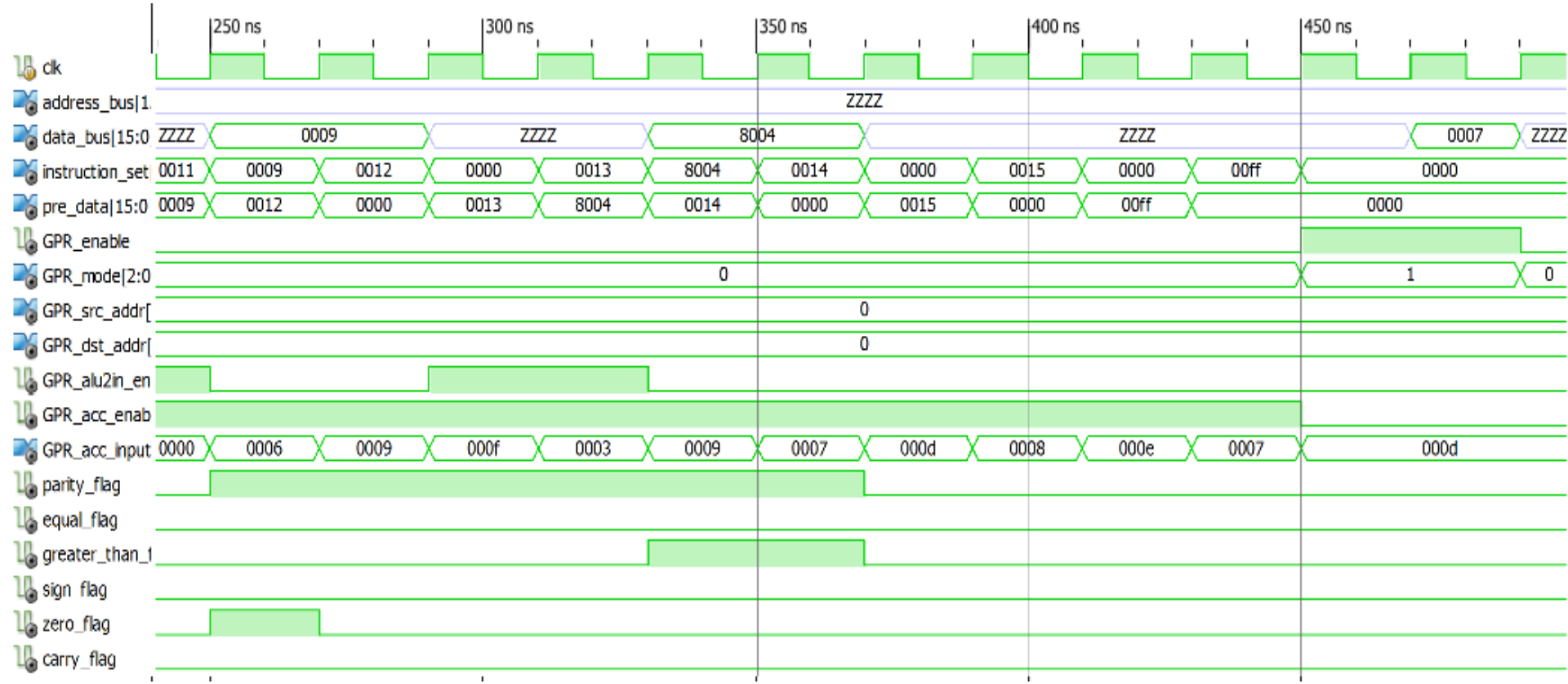
```
NOP
LDV R0, 8006H // loads -6 in register 0 (accumulator)
LDV R1, 0006H // loads 6 in register 1
ADR // adds reg 1 and reg 0 and stores result in accumulator
ADD 0009H // adds 9 to reg 0 and stores in accumulator
SBR // subtracts reg 1 from reg 0
SUB 8004 // subtracts -4 from reg 0
INC // increments value in accumulator
DEC // decrements value in accumulator
REG R0 // loads data in reg 0 (accumulator) to data bus
NOP
```

```
0000
0003
8006
0103
0006
0010
0000
0011
0009
0012
0000
0013
8004
0014
0000
0015
0000
00ff
0000
0000
```

# SIMULATION


0 ns
50 ns
100 ns
150 ns
200 ns
clk
address_bus[1
data_bus[15:0
instruction_set
pre_data[15:0
GPR_enable
GPR_mode[2:0
GPR_src_addr[
GPR_dst_addr[
GPR_alu2in_en
GPR_acc_enab
GPR_acc_input
parity_flag
equal_flag
greater_than_f
sign_flag
zero_flag
carry_flag
ZZZZ
ZZZZ
8006
0006
ZZZZ
0000
0003
8006
0103
0006
0010
0000
0011
0000
0003
8006
0103
0006
0010
0000
0011
0009
Z
0
2
0
Z
0
Z
0
1
0
0000

# SIMULATION

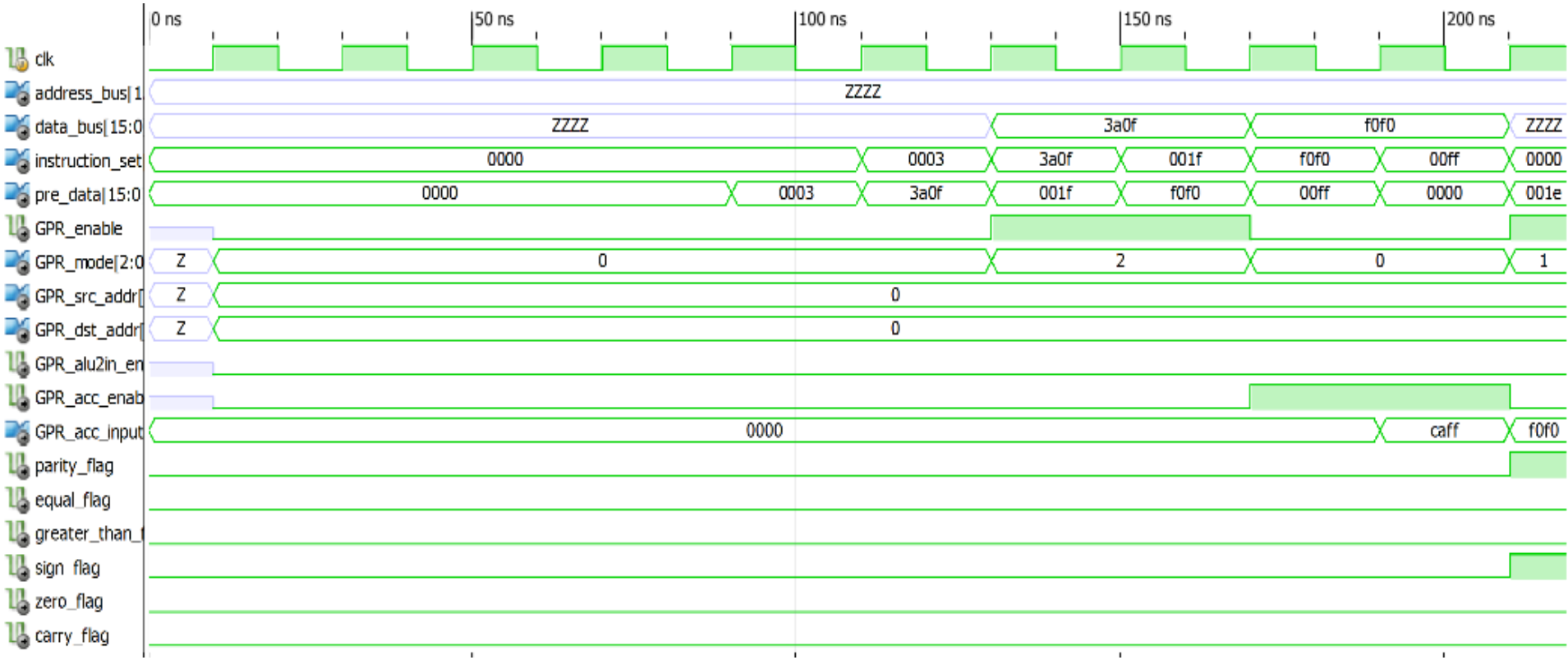

250 ns
300 ns
350 ns
400 ns
450 ns
clk
address_bus[1
ZZZZ
data_bus[15:0
ZZZZ
0009
ZZZZ
8004
ZZZZ
0007
ZZZZ
instruction_set
0011
0009
0012
0000
0013
8004
0014
0000
0015
0000
00ff
0000
pre_data[15:0
0009
0012
0000
0013
8004
0014
0000
0015
0000
00ff
0000
GPR_enable
GPR_mode[2:0
0
1
0
GPR_src_addr[
0
GPR_dst_addr[
0
GPR_alu2in_en
GPR_acc_enab
GPR_acc_input
0000
0006
0009
000f
0003
0009
0007
000d
0008
000e
0007
000d
parity_flag
equal_flag
greater_than_f
sign_flag
zero_flag
carry_flag

# TEST 3

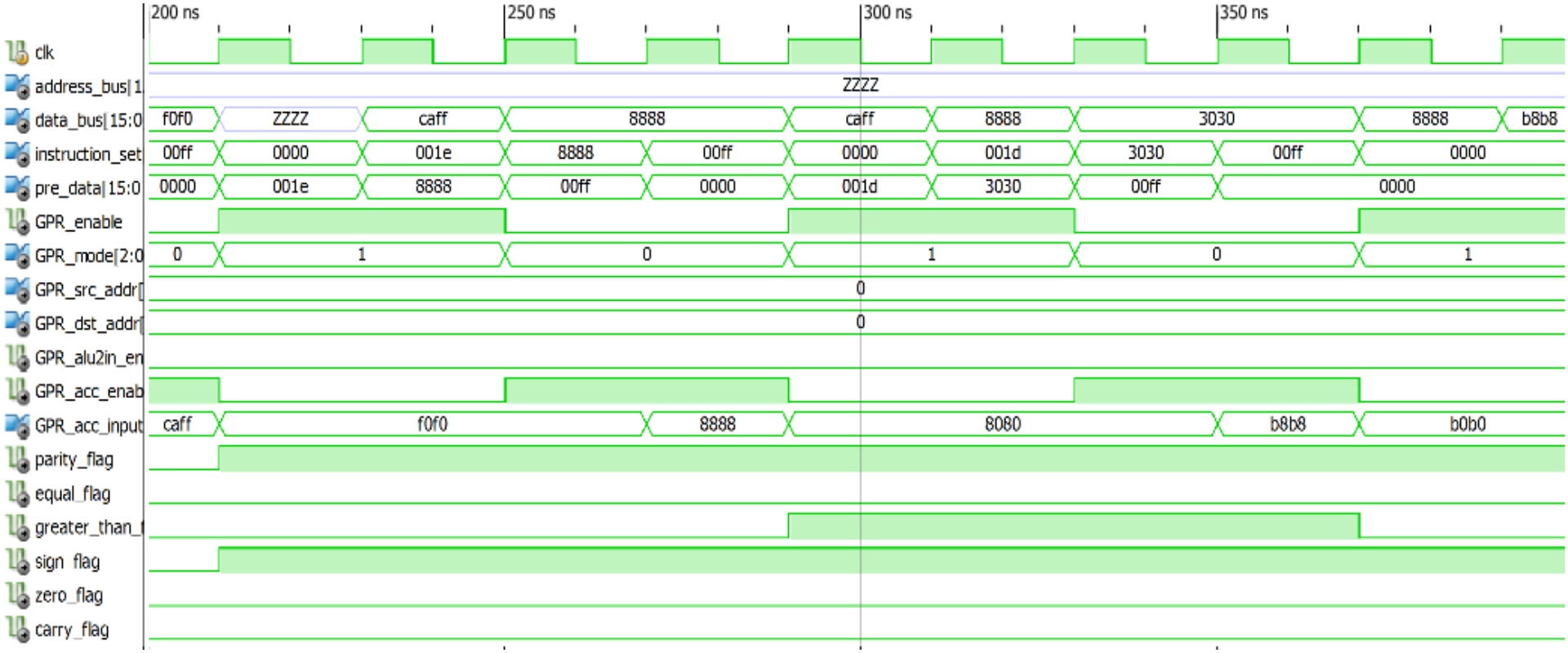
**Program :-**

| | |
|---|---|
| | 0000 |
| NOP | 0003 |
| LDV R0, 3A0FH // loads 3a0f in reg 0 (accumulator) | 3A0F |
| XOR F0F0H // performs xor operation of reg 0 and f0f0 | 001F |
| REG R0 // loads data in reg 0 (accumulator) to data bus | F0F0 |
| AND 8888H // performs and operation of reg 0 and 8888 | 00FF |
| REG R0 // loads data in reg 0 (accumulator) to data bus | 0000 |
| OR 3030H // performs or operation of reg 0 and 3030 | 001E |
| REG R0 // loads data in reg 0 (accumulator) to data bus | 8888 |
| NOP | 00FF |
| | 0000 |
| | 001D |
| | 3030 |
| | 00FF |
| | 0000 |
| | 0000 |

# SIMULATION


0 ns
50 ns
100 ns
150 ns
200 ns
clk
address_bus[1
ZZZZ
data_bus[15:0
ZZZZ
3a0f
f0f0
ZZZZ
instruction_set
0000
0003
3a0f
001f
f0f0
00ff
0000
pre_data[15:0
0000
0003
3a0f
001f
f0f0
00ff
0000
001e
GPR_enable
GPR_mode[2:0
Z
0
2
0
1
GPR_src_addr[
Z
0
GPR_dst_addr[
Z
0
GPR_alu2in_en
GPR_acc_enab
GPR_acc_input
0000
caff
f0f0
parity_flag
equal_flag
greater_than_
sign_flag
zero_flag
carry_flag

# SIMULATION


200 ns
250 ns
300 ns
350 ns
clk
address_bus[1
data_bus[15:0
instruction_set
pre_data[15:0
GPR_enable
GPR_mode[2:0
GPR_src_addr[
GPR_dst_addr[
GPR_alu2in_en
GPR_acc_enab
GPR_acc_input
parity_flag
equal_flag
greater_than_
sign_flag
zero_flag
carry_flag
ZZZZ
f0f0 ZZZZ caff 8888 caff 8888 3030 8888 b8b8
00ff 0000 001e 8888 00ff 0000 001d 3030 00ff 0000
0000 001e 8888 00ff 0000 001d 3030 00ff 0000
0 1 0 1 0 1
0
0
caff f0f0 8888 8080 b8b8 b0b0

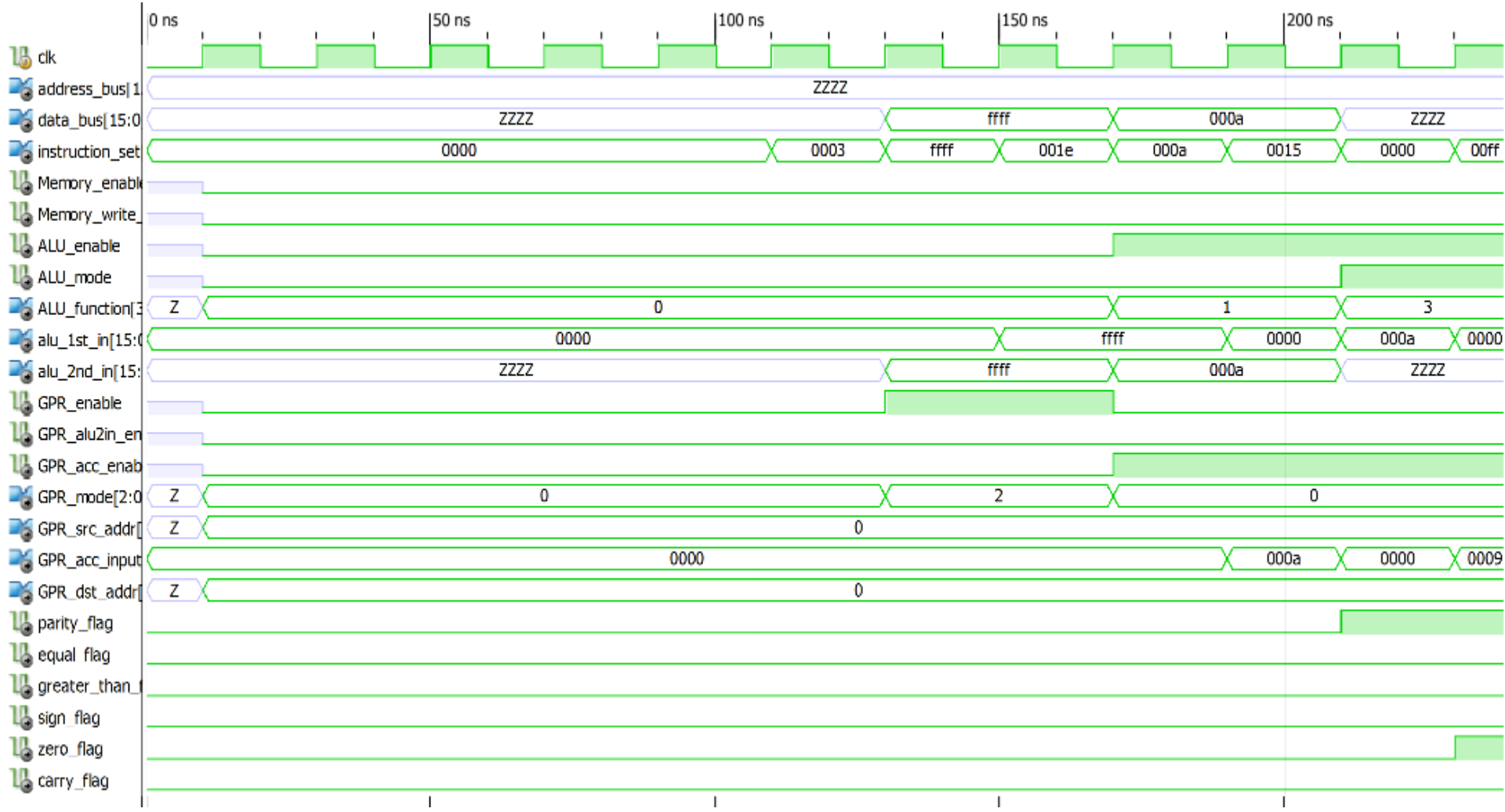
# TEST 4

**Program :-**

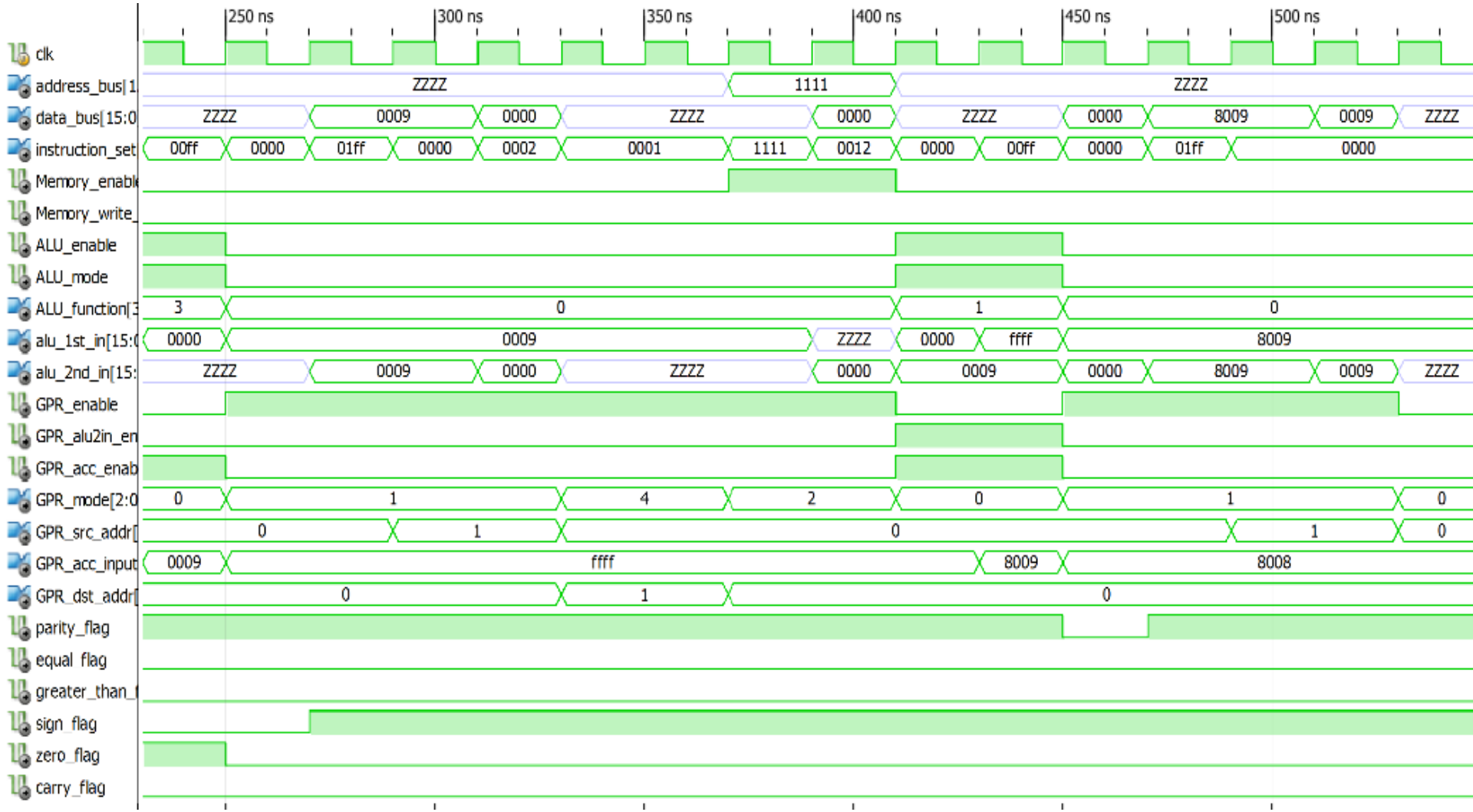
```
NOP
LDV R0, FFFFH // loads ffff to reg 0
AND 000AH // performs and operation with reg 0 and 00a
DEC // decrease the value of reg 0 by 1
REG R0 // loads data in reg 0 (accumulator) to data bus
REG R1 // loads data in reg 1 to data bus
LDR R1, R0 // loads data from reg 0 to reg 1
LDM R0, [1111H] // loads data from memory location 1111 to reg 0
SUB // subtracts reg 1 from reg 0
REG R0 // loads data in reg 0 (accumulator) to data bus
REG R1 // loads data in reg 1 to data bus
NOP
```

```
0000
0003
ffff
001e
000a
0015
0000
00ff
0000
01ff
0000
0002
0001
0001
1111
0012
0000
00ff
0000
01ff
0000
0000
```

# SIMULATION


0 ns
50 ns
100 ns
150 ns
200 ns
clk
address_bus[1
data_bus[15:0
instruction_set
Memory_enabl
Memory_write_
ALU_enable
ALU_mode
ALU_function[3
alu_1st_in[15:0
alu_2nd_in[15:
GPR_enable
GPR_alu2in_en
GPR_acc_enab
GPR_mode[2:0
GPR_src_addr[
GPR_acc_input
GPR_dst_addr[
parity_flag
equal_flag
greater_than_
sign_flag
zero_flag
carry_flag
ZZZZ
ffff
000a
0000
0003
001e
0015
00ff
Z
0
1
3
2
0009

# SIMULATION


250 ns
300 ns
350 ns
400 ns
450 ns
500 ns
clk
address_bus[1
data_bus[15:0
instruction_set
Memory_enabl
Memory_write_
ALU_enable
ALU_mode
ALU_function[3
alu_1st_in[15:0
alu_2nd_in[15:
GPR_enable
GPR_alu2in_en
GPR_acc_enab
GPR_mode[2:0
GPR_src_addr[
GPR_acc_input
GPR_dst_addr[
parity_flag
equal_flag
greater_than_
sign_flag
zero_flag
carry_flag
ZZZZ
1111
0009
0000
0002
0001
0012
00ff
01ff
8009
ffff
8008

# LIMITATIONS

- ALU
- Interrupt handling
- I/O Protocols
- Stack memory and register
- Limited memory interface
- Branch Instructions
- Other

# REFERENCES


- NPTEL lecture on microprocessor by Prof. Shantanu
- ARM System Developer's Guide book
- Microprocessors and Interfacing book by Douglass Hall
- GeekforGeeks.org
- Ben eater Youtube Channel
- ChipVerify.com

# Thank You