



Name:	Muhammad Attiq
Reg. No:	FA23-BCE-060
Section:	BCE-5A
Subject:	Computer Organization & Architecture
Instructor:	Dr. Irfanullah
Lab Report:	5

Lab # 05 Introduction to Verilog HDL

Combinational Circuits 1

Objectives

- Gate Level Modeling

In-Lab

Tasks (Use Xilinx)

(Define input and output as vectors ; and not multiple variables)

- Write the Verilog gate-level description and testbench for a circuit with 3-bit input(X) and 1-bit output (Y). The output is 1 if the input number is a palindrome (reads the same backward and forward eg 101 or 010). The minterms of the output will be

```
module palindrome(input [2:0]x,output y);
```

$$Y(X_2,X_1,X_0) = \sum(0,2,5,7) \text{ i.e. } 000,010,101 \text{ and } 111$$

```
1  .data
2  prompt:    .ascii "Enter the 3 bit number: \n"
3  result:    .ascii "The number is palindrome\n"
4  not_result: .ascii "The number is not palindrome\n"
5  .text
6  main:
7      li $v0, 4
8      la $a0, prompt
9      syscall
10     li $v0, 5
11     syscall
12     move $t0, $v0
13     srl $t1, $t0, 2
14     andi $t1, $t1, 1
15     andi $t2, $t0, 1
16     beq $t1, $t2, yes
17     j no
18 yes:
19     li $v0, 4
20     la $a0, result
21     syscall
22     j exit
23 no:
```

```

li $v0, 4
la $a0, not_result
syscall
exit:
li $v0, 10
syscall

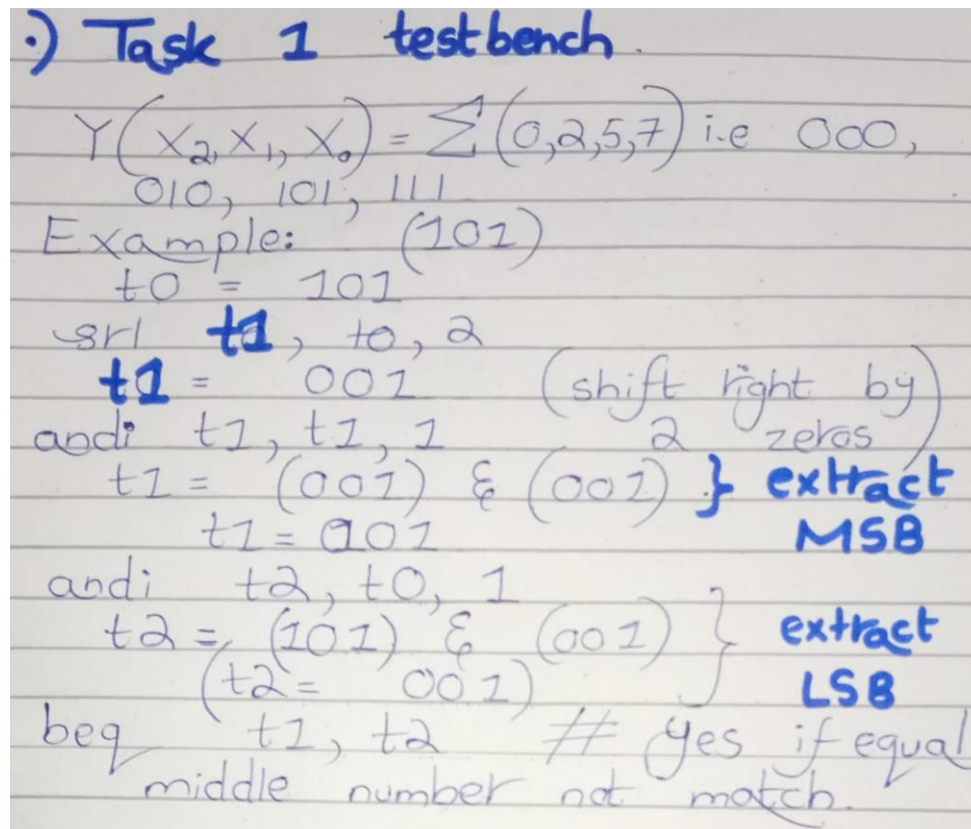
```

Enter the 3 bit number:

101

The number is palindrome

Output



Task 1 Testbench

- Write the Verilog gate-level description and testbench for a circuit with 6-bit input(X) and 1-bit output (Y). It compares the first three bits of the inputs with the latter three and sets output equal to 1 when both are equal.

$$Y = X_0 * X_3 + X_1 * X_4 + X_2 * X_5$$

```

1  .data
2  prompt:      .ascii "Enter the 6-bit number: "
3  yes_result:  .ascii "The number is equal\n"
4  no_result:   .ascii "The number is not equal\n"
5  .text
6  main:
7      li $v0, 4
8      la $a0, prompt
9      syscall
10     li $v0, 5
11     syscall
12     move $t0, $v0
13     andi $t0, $t0, 0x3F
14     andi $t1, $t0, 0x7
15     srl  $t2, $t0, 3
16     andi $t2, $t2, 0x7
17     bne $t1, $t2, no
18     j yes
19 yes:
20     li $v0, 4
21     la $a0, yes_result
22     syscall
23     j exit
24 no:
25     li $v0, 4
26     la $a0, no_result
27     syscall
28 exit:
29     li $v0, 10
30     syscall

```

Enter the 6-bit number: 101101

The number is equal

Output

Task 2 testbench

$$Y = X_0 * X_3 + X_1 * X_4 + X_2 * X_5$$

Example: (101101)

$t_0 = 101101$

$\text{andi } t_1, t_0, 0x7$

$t_1 = 101101 \& 000111$

$t_1 = 101$

$\text{srl } t_2, t_0, 3$ (shift right by 3)

$t_2 = 000101$

$\text{andi } t_2, t_2, 0x7 \neq t_1 = 101$

if ($t_1 = t_2$) then bits are equal

Task 2 Testbench

- Write the Verilog gate-level description and testbench for a circuit with 6-bit input(X) and 1-bit output (Y). It compares the first three bits of the inputs with the latter three and sets output equal to 1 when both are mirror image. For example 100 001 or 110 011

$$Y = X_0 * X_5 + X_1 * X_4 + X_2 * X_3$$

```

1  .data
2      prompt: .ascii "Enter a 6-bit number (0-63): "
3      result_true: .ascii "Output Y = 1 (Mirror image)\n"
4      result_false: .ascii "Output Y = 0 (Not mirror image)\n"
5  .text
6  .globl main
7  main:
8      li $v0, 4
9      la $a0, prompt
10     syscall
11     li $v0, 5
12     syscall
13     move $t0, $v0
14     andi $t1, $t0, 1
15     andi $t2, $t0, 2
16     andi $t3, $t0, 4
17     andi $t4, $t0, 8
18     andi $t5, $t0, 16
19     andi $t6, $t0, 32
20     srl $t2, $t2, 1
21     srl $t3, $t3, 2
22     srl $t4, $t4, 3
23     srl $t5, $t5, 4
24     srl $t6, $t6, 5
25     and $s0, $t1, $t6
26     and $s1, $t2, $t5
27     and $s2, $t3, $t4
28     or $s3, $s0, $s1
29     or $s4, $s3, $s2
30     beq $s4, $zero, print_false
31 print_true:
32     li $v0, 4
33     la $a0, result_true
34     syscall
35     j exit
36 print_false:
37     li $v0, 4
38     la $a0, result_false
39     syscall
40 exit:
41     li $v0, 10
42     syscall

```

Enter a 6-bit number (0-63): 110011
Output Y = 1 (Mirror image)

Output

(Task 3 testbench)

```

t0 = 110011
t1 = 000001 # andi t0, 1
t2 = 000010 # andi t0, 2
t3 = 000000 # andi t0, 4
t4 = 000000 # andi t0, 8
t5 = 010000 # andi t0, 16
t6 = 100000 # andi t0, 32

t2 = 000001 # t2 >> 1
t3 = 000000 # t3 >> 2
t4 = 000000 # t4 >> 3
t5 = 000001 # t5 >> 4
t6 = 000001 # t6 >> 5

s0 = (t2 & t6) # 1
s1 = (t2 & t5) # 1
s2 = (t3 & t4) # 0

s3 = (s0 | s1) # s3 = 1
s4 = (s3 | s2) # s4 = 1

```

;) So the number is mirror image of upper 3 and last 3 bits

Task 3 Testbench

Critical Analysis / Conclusion

Doing these logic tasks in MIPS assembly is a great learning exercise. It forces you to understand how a processor works with simple ones and zeros. You learn to break down a problem into tiny, step-by-step instructions, which shows you how software commands are connected to the hardware's physical logic. This is a

fundamental skill for writing efficient, low-level code for systems that need to be fast or run on simple devices.

Lab Assessment		
Pre Lab	/5	/25
Performance	/5	
Results	/5	
Viva	/5	
Critical Analysis	/5	
Instructor Signature and Comments		