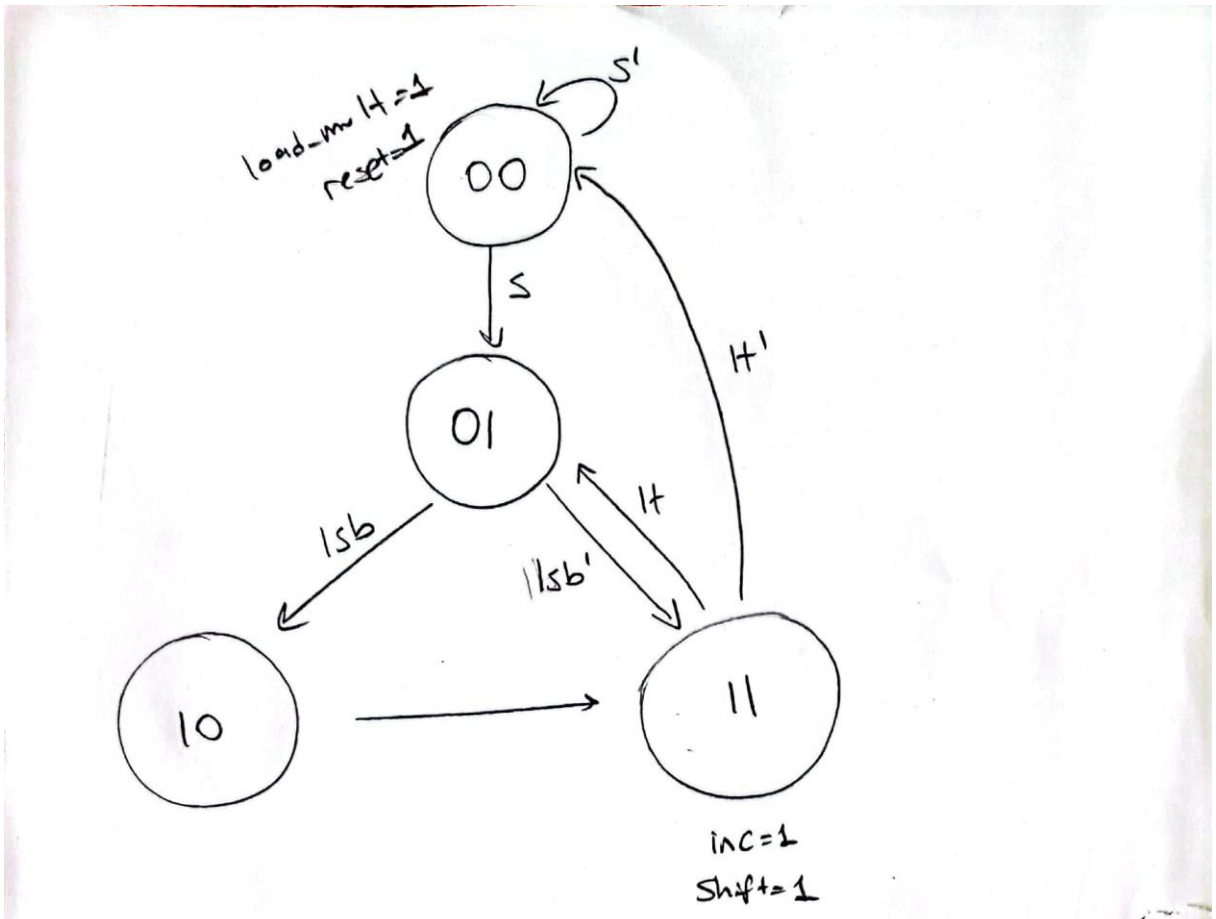
CSE 331 FALL 2020
HOMEWORK 3
-

BARIŞ AYYILDIZ
1901042252

QUESTION-1

STATE DIAGRAM



Inputs: s (start)
lsb (least significant bit)
lt (counter less than 32)

Outputs: reset (reset)
inc (increment counter by 1)
shift (shift right)
load_mult (load multiplicant)

STATE TABLE

Inputs					Outputs					
s1	s0	s	lsb	lt	n1	n0	reset	inc	shift	load_mult
0	0	0	x	x	0	0	0	0	0	0
0	0	1	x	x	0	1	1	0	0	1
0	1	x	0	x	1	1	0	0	0	0
0	1	x	1	x	1	0	0	0	0	0
1	0	x	x	x	1	1	0	0	0	0
1	1	x	x	0	0	0	0	1	1	0
1	1	x	x	1	0	1	0	1	1	0

BOOLEAN EXPRESSION

n1 = s1's0lsb + s1's0lsb' + s1s0'

= s1's0 + s1s0'

= s1 XOR s0

n0 = s1's0's + s1's0lsb' + s1s0' + s1s0lt

reset = s1's0's

inc = s1s0lt' + s1s0lt

= s1s0

shift = s1s0

load_mult = s1's0's

I could not implement multiplexer in verilog...

QUESTION-2

NOT_32.V

Takes a 32 bit register and reverses it's bits

OR_32.V

Or's two 32 bit registers

AND_32.V

And's two 32 bit registers

XOR_32.V

Xor's two 32 bit registers

NOR_32.V

Nor's two 32 bit registers

FULL_ADDER.V
Full adder module works with single bits

ADDER_32.V
Adds two 32 bit registers with using 32 of full_adder modules

TWOS_COMPLEMENT.V
Takes the 2's complement of a 32 bit register with using not_32 and adder_32 modules

SUB_32.V
Subtracts two 32 bit registers with using not_32 and adder_32 modules

SLT_32.V
Takes two 32 bit registers, if the first register is smaller than the second one return 32 of 1 bits else returns 32 of 0 bits

MUX8X3
8x3 multiplexer

ALU_32.V
32 bit arithmetic logic unit with 3 bit opcode

TESTBENCHES
Adder testbench

```
# 96 + 36 = 132
# 22 + 12 = 34
```

And testbench

```
# 00000000000000000000000000000111 AND 00000000000000000000000000001111 = 00000000000000000000000000000111
# 11111111111111111111111111100000 AND 00000000000000000000000111111111 = 00000000000000000000000111100000
```

Nor testbench

```
# 00000000000000000000000000000111 NOR 00000000000000000000000000001111 = 11111111111111111111111111110000
# 00000000000000111111111111100000 NOR 00000000000000000000000111111111 = 11111111111111000000000000000000
```

Or testbench

```
# 00000000000000000000000000000111 OR 00000000000000000000000000001111 = 00000000000000000000000000001111
# 00000000000000111111111111100000 OR 00000000000000000000000111111111 = 00000000000000111111111111111111
```

Slt testbench

```
# 10 SLT 15 = 11111111111111111111111111111111
# 30 SLT 25 = 00000000000000000000000000000000
```

Sub testbench

```
# 96 - 36 = 60
# 22 - 12 = 10
```

Xor testbench

```
# 00000000000000000000000000000111 XOR 00000000000000000000000111111111 = 00000000000000000000000111111000
# 11111111111111111111111111110000 XOR 00000000000000001111111111111111 = 11111111111111110000000000001111
```

ALU testbench

```
# 12 + 24 = 36
#  7 + 38 = 45
# 00000000000000000000000000000111 XOR 00000000000000000000000000000010 = 00000000000000000000000000000101
# 11111111111111111111111111111111 XOR 00000000000000000000000000000111 = 11111111111111111111111111111000
# 36 - 24 = 12
# 90 - 80 = 10
# 10 < 20 = 11111111111111111111111111111111
# 15 < 10 = 00000000000000000000000000000000
# 00000000000000000000000000000111 NOR 00000000000000000000000000000010 = 11111111111111111111111111111000
# 11111111111111111111111111111001 NOR 11111111111111111111111111001001 = 00000000000000000000000000000110
# 00000000000000000000000000000111 AND 00000000000000000000000000000010 = 00000000000000000000000000000010
# 11111111111111111111111111111111 AND 11100000000111000000000000000111 = 11100000000111000000000000000111
# 00000000000000000000000000111111 OR 00000000000000001110000000000110 = 00000000000000001110000000111111
# 11111111111111111111111111111111 OR 11100000000111000000000000000111 = 11111111111111111111111111111111
```