Peter Pettino
CSE 3666
16 March, 2024

Homework 4

1)

a)

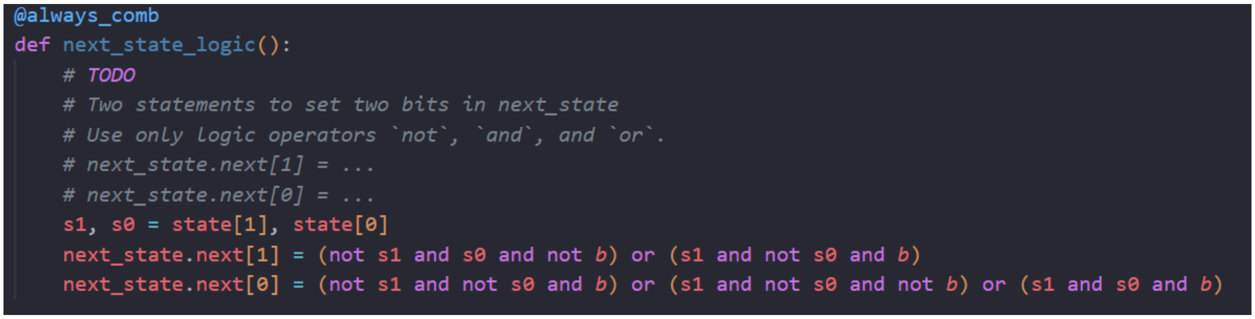
state[1]	state[0]	b	next_state[1]	next_state[0]	z
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	0	1	0

next_state[1] = s1’s0b’ + s1s0’b

next_state[0] s1’s0’b + s1s0’b’ + s1s0b

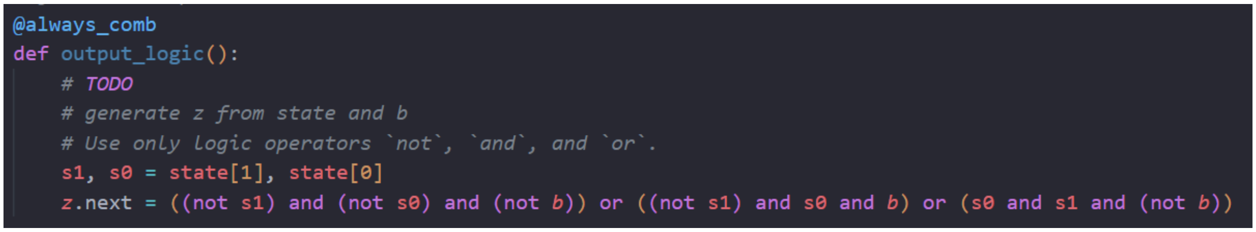
z = s1’s0’b’ + s1’s0b + s0s1b’

b)



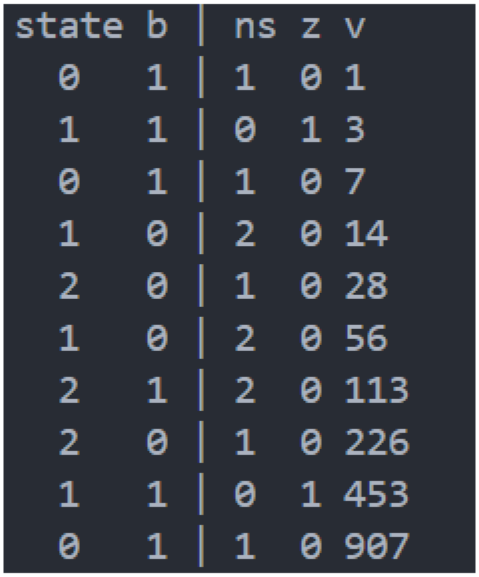
```
@always_comb
def next_state_logic():
    # TODO
    # Two statements to set two bits in next_state
    # Use only logic operators `not`, `and`, and `or`.
    # next_state.next[1] = ...
    # next_state.next[0] = ...
    s1, s0 = state[1], state[0]
    next_state.next[1] = (not s1 and s0 and not b) or (s1 and not s0 and b)
    next_state.next[0] = (not s1 and not s0 and b) or (s1 and not s0 and not b) or (s1 and s0 and b)
```

c)



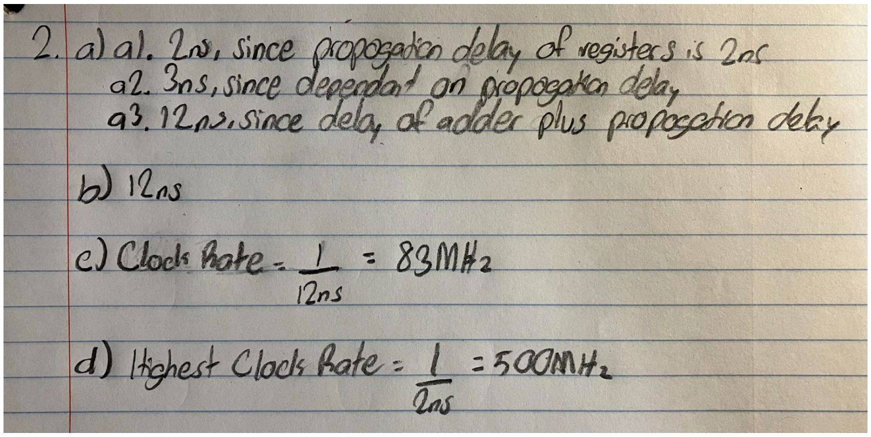
```
@always_comb
def output_logic():
    # TODO
    # generate z from state and b
    # Use only logic operators `not`, `and`, and `or`.
    s1, s0 = state[1], state[0]
    z.next = ((not s1) and (not s0) and (not b)) or ((not s1) and s0 and b) or (s0 and s1 and (not b))
```

d)



```
state b | ns z v
  0   1 | 1  0 1
  1   1 | 0  1 3
  0   1 | 1  0 7
  1   0 | 2  0 14
  2   0 | 1  0 28
  1   0 | 2  0 56
  2   1 | 2  0 113
  2   0 | 1  0 226
  1   1 | 0  1 453
  0   1 | 1  0 907
```

2)



2. a) a1. 2ns, since propogation delay of registers is 2ns
a2. 3ns, since dependant on propogation delay
a3. 12ns, since delay of adder plus propogation delay

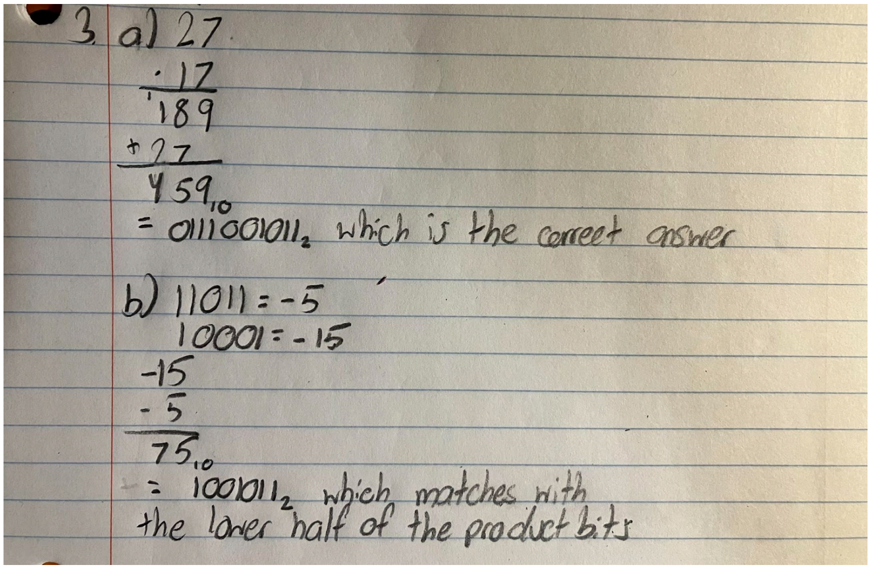
b) 12ns

c) Clock Rate = $\frac{1}{12ns}$ = 83MHz

d) Highest Clock Rate = $\frac{1}{2ns}$ = 500MHz

3)

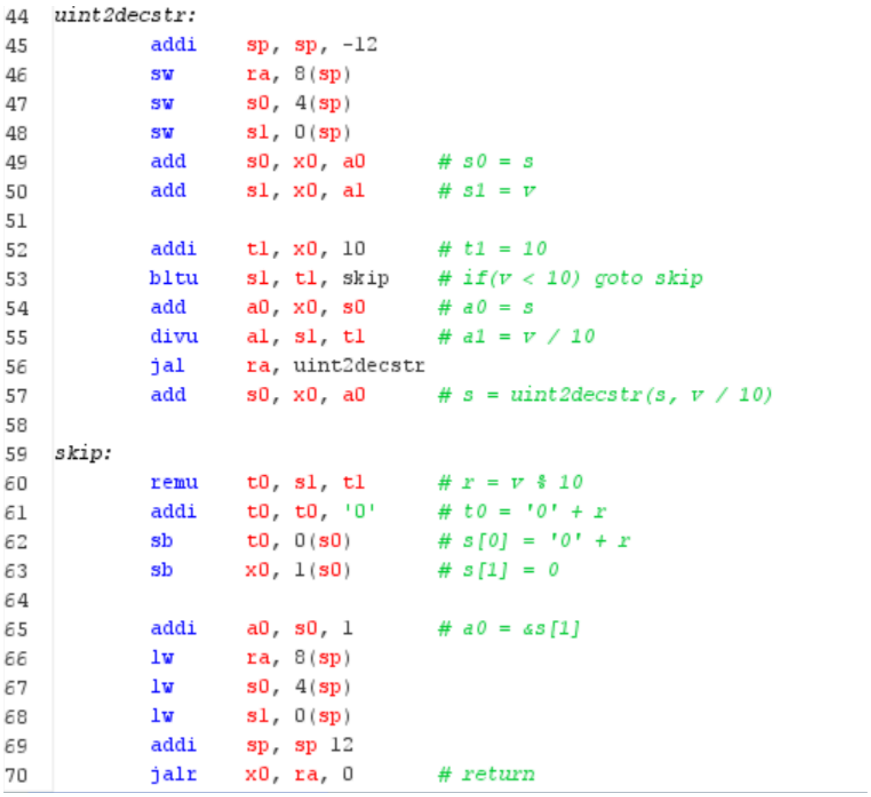
Steps	Multiplicand	Multiplier	Product
init	0000011011	10001	0000000000
1	0000110110	01000	0000011011
2	0001101100	00100	0000011011
3	0011011000	00010	0000011011
4	0110110000	00001	0000011011
5	1101100000	00000	0111001011



3. a) 27
· 17
189
+ 27
459$_{10}$
= 0111001011$_2$ which is the correct answer

b) 11011 = -5
10001 = -15
-15
- 5
75$_{10}$
= 1001011$_2$ which matches with the lower half of the product bits

4)



```
uint2decstr:
        addi    sp, sp, -12
        sw      ra, 8(sp)
        sw      s0, 4(sp)
        sw      s1, 0(sp)
        add     s0, x0, a0      # s0 = s
        add     s1, x0, a1      # s1 = v

        addi    t1, x0, 10      # t1 = 10
        bltu    s1, t1, skip    # if(v < 10) goto skip
        add     a0, x0, s0      # a0 = s
        divu    a1, s1, t1      # a1 = v / 10
        jal     ra, uint2decstr
        add     s0, x0, a0      # s = uint2decstr(s, v / 10)

skip:
        remu    t0, s1, t1      # r = v % 10
        addi    t0, t0, '0'     # t0 = '0' + r
        sb      t0, 0(s0)       # s[0] = '0' + r
        sb      x0, 1(s0)       # s[1] = 0

        addi    a0, s0, 1       # a0 = &s[1]
        lw      ra, 8(sp)
        lw      s0, 4(sp)
        lw      s1, 0(sp)
        addi    sp, sp 12
        jalr    x0, ra, 0       # return
```