Homework 4

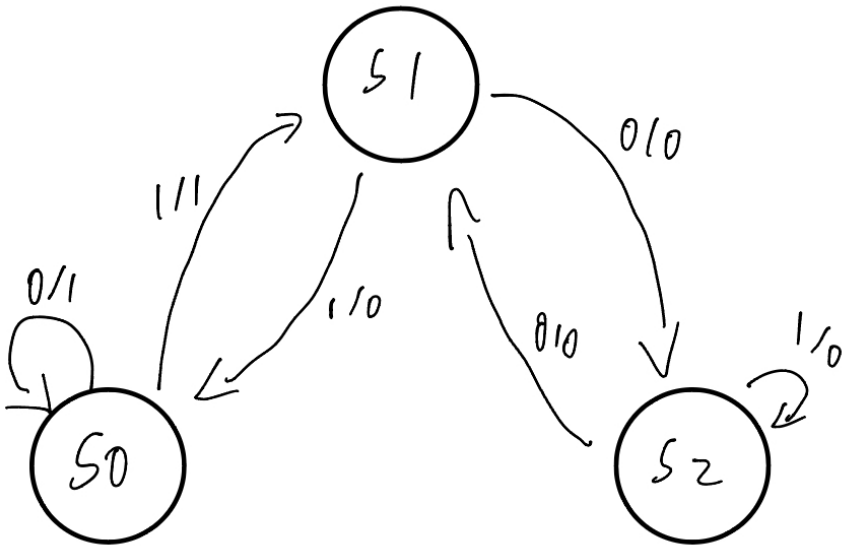
Benny Chen

October 23, 2022

Problem 1

State table:

State[1]	State[0]	b	NextState[1]	NextState[0]	z
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	0	0	0
1	0	0	0	1	0
1	0	1	1	0	0



next_stage_logic:

```
def next_state_logic():
    next_state.next[0] =
    (b & ~state[0] & ~state[1])|(~b & ~state[0] & state[1])
    next_state.next[1] =
    (~b & state[0] & ~state[1])|(b & ~state[0] & state[1])
```

output_logic:

```
def output_logic():
    z.next = 1 if state[1] == 0 and state[0] == 0 else 0
```

Out for 111000101

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

Problem 2

Consider the multiplier we have studied. Inside the control module, there is also a register that counts the steps and a combinational circuit. Assume the following.

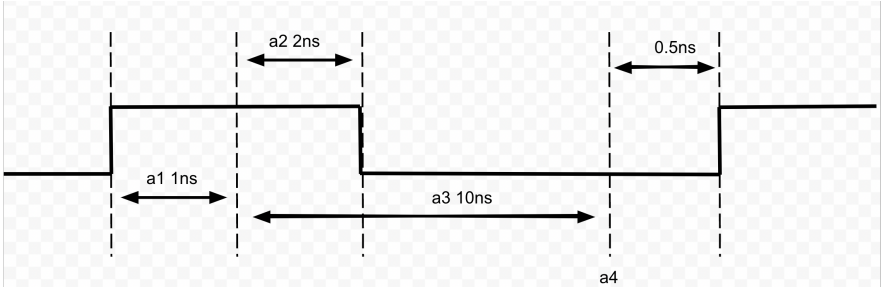
- The delay of the adder is 10 ns,
- The delay of the combinational circuit in the control module is 2 ns.
- The setup time, the hold time, and the propagation delay of the registers is 0.5 ns, 0.2 ns, and 1 ns, respectively.
- Do not consider the delays on wire.

Answer the following questions. Round answers to the nearest tenth if necessary. For example, enter 3 for 3, 0.3 for 1/3, and 0.7 for 2/3.

a.

In a figure like below, show the timing of the following events.

1. a1. Reg-Ready. The output of registers is available.
2. a2. Control-Ready. The output of control module is available
3. a3. Adder-Ready. The output of adder is available.
4. a4. Latest. The input to registers must be ready. This is the example in the figure.



b.

What is the minimum cycle time for this multiplier to work properly?

The minimum cycle time is 10 ns + 1 ns + 0.5 ns = 11.5 ns.

c.

What is the highest clock rate in MHz that this multiplier can run at?

The highest clock rate is 1/11.5 ns * 1000 = 86.96 MHz or 87 MHz.

d.

If we build sequential circuit with the same kind of registers, what is the highest clock rate in MHz we can achieve?

The highest clock rate is 1 ns + 0.5 ns = 1.5 ns. The highest clock rate is 1/1.5 ns * 1000 = 666.67 MHz or 667 MHz.

Problem 3

Multiplier:

01110 01011 = 459 27 * 17 = 459 Same value if consider unsigned

Steps	Multiplicand	Multiplier	Product
init	00000 11011	10001	00000 00000
1	00001 10110	01000	00000 11011
2	00011 01100	00100	00000 11011
3	00110 11000	00010	00000 11011
4	01101 10000	00001	00000 11011
5	11011 00000	00000	01110 01011

Problem 4

Translate the following C function to RISC-V assembly code.

```
char * uint2decstr(char s[], unsigned int v)
{
    unsigned int r;

    if (v >= 10) {
        s = uint2decstr(s, v / 10);
    }
    r = v % 10; // remainder
    s[0] = '0' + r;
    s[1] = 0;
    return &s[1]; // return the address of s[1]
}
```

Solution:

```
#       CSE 3666 uint2decstr

.globl  main

.data

str:    .space 128

    .text
main:
# set all bytes in the buffer to 'A'
la      s1, str
    addi    a1, x0, 0
addi    a2, x1, 128
li      t2, 'A'
clear:
```

```
    add     t0, s1, a1
sb      t1, 0(t0)
    addi    a1, a1, 1
bne     a1, a2, clear

    addi    a0, s1, 0
    # change -1 to other numbers to test
    li  a1, 2022
jal     ra, uint2decstr

    addi    a0, s1, 0
    addi    a7, x0, 4
    ecall

exit:   addi    a7, x0, 10
    ecall

# char * uint2decstr(char *s, unsigned int v)
# the function converts unsigned value to decimal string
# Here are some examples:
# 0:    "0"
# 2022: "2022"
# -1:   "4294967295"

uint2decstr:
addi sp, sp, -8 # reserve space for return address and v
sw ra, 4(sp) # save return address
sw a1, 0(sp) # save v
addi t0, x0, 10 # t0 = 10
bltu a1, t0, cont # if v < 10 skips to cont
divu a1, a1, t0 # v = v / 10
jal ra, uint2decstr # call uint2decstr(s, v / 10)
cont:
lw ra, 4(sp) # restore return address
lw a1, 0(sp) # restore v
addi t0, x0, 10 # t0 = 10
remu t1, a1, t0 # t0 = v % 10
addi t0, t1, '0' # Set t0 to charcter of t1
sb t0, 0(a0) # s[0] = '0' + t1
sb t0, 1(a0) # s[1] = 0
addi a0, a0, 1 # a0 = &s[1]
addi sp, sp, 8 # free space
jr ra # return
```