Kyle Loyka

Homework 3

ECEN 449-503

Due: 6 April 2016

1.

a) This will also search sub directories, if you only want the current directory add the flag "-maxdepth 1"
find ~/usr –type f –size +10M

b) This will also search sub directories, if you only want the current directory add the flag "-maxdepth 1"
find ~/doc –name '*.tar' –delete

c) **find . -maxdepth 1 -name '*.txt' | xargs grep -i "int"**

d) **fgrep –o "Error" logfile.txt | wc –l**

e) **lpr -# 3 –P HPLaserJet4300 my_driver.c**

2.

a)

Round Robin	
Exec. TIME	**TASK**
0	A
4	B
7	C
10	A
14	D
16	A
20	E
24	A
26	(end)

b)

First Come First Served	
Exec. TIME	**TASK**
0	A
14	B
17	C
20	D
22	E
26	(end)

c) **Round Robin wait:**

$[(26-0)_A+(7-1)_B+(10-2)_C+(16-7)_D+(24-18)_E]/5$ = **11**

First Come First Served wait:

$[(14-0)_A+(17-1)_B+(20-2)_C+(22-7)_D+(26-18)_E]/5$ = **14.2**

3.

a) Static Timing Analysis

NODE	MAX DELAY
P	10
R	5
Q	15
S	25
T	30
OUT	40

b) The delay calculated using STA is not accurate. Gates **G4** and **G5** make the circuit perform in a way such that the value of node **T** only depends on the **d** signal. This means the actual delay of the circuit is **30ns**.

4.

a) out = $\bar{S}bc + Sab$

b) NS = $\bar{S}abc + Sac + bd$

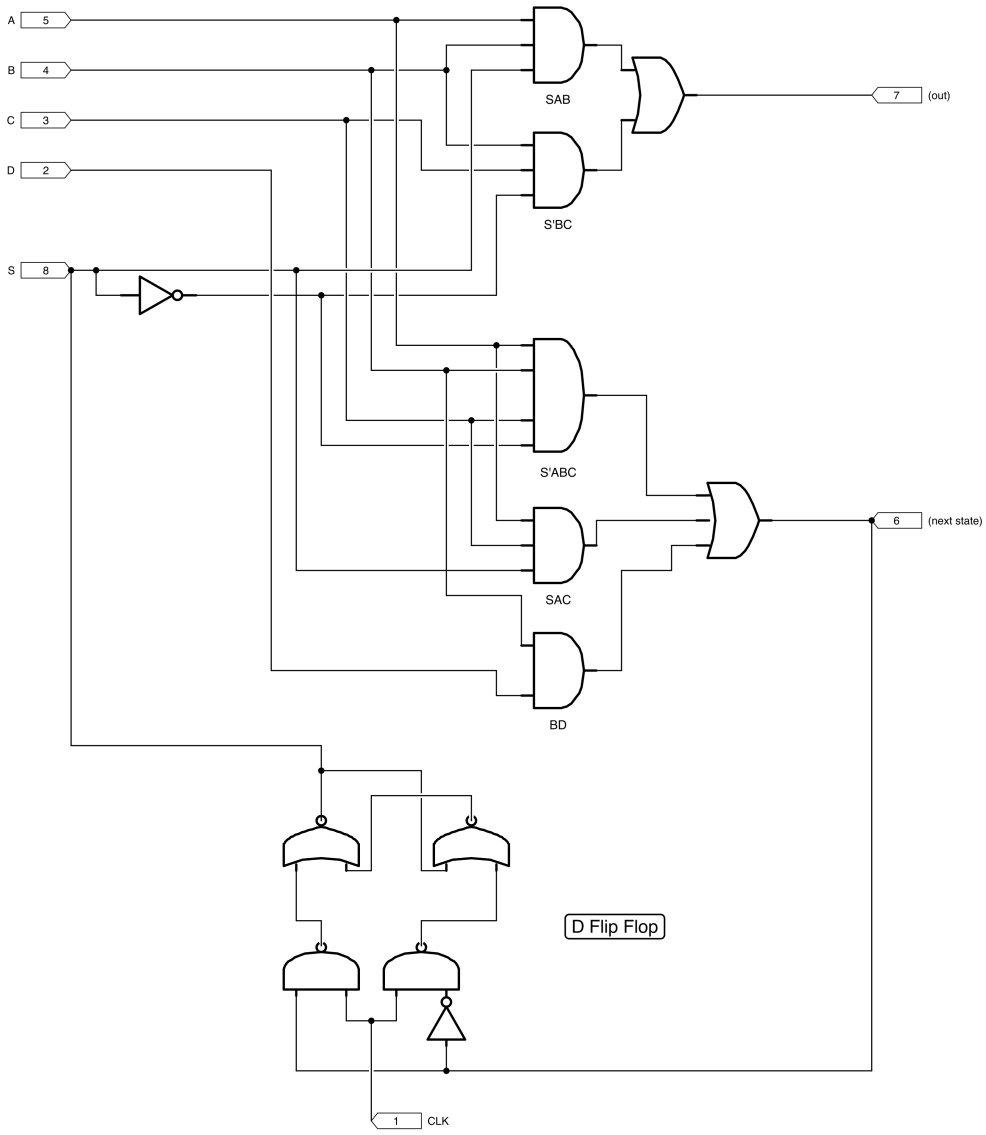
c) (see next page)

d) At least two CLBs are needed to implement F.

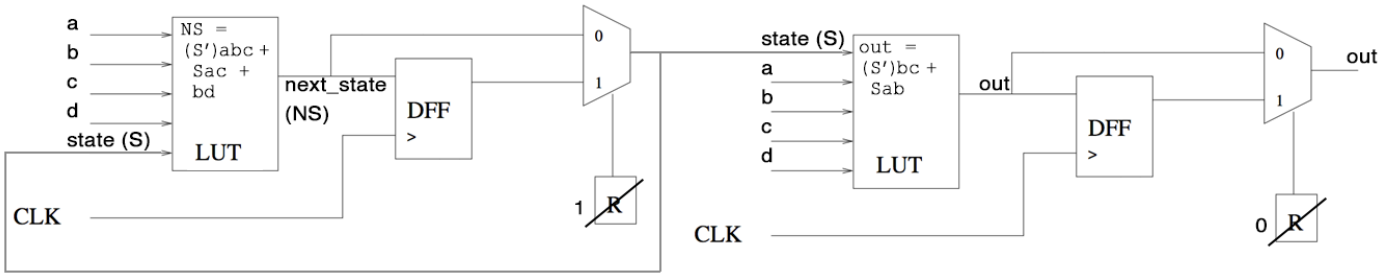
The first CLB's LUT implements the next state logic. The Configuration bit for the first MUX would be permanently set at 1 since the D Flip Flop is needed for sequential logic. The output of this CLB would be plugged in to the input of the second CLB's LUT and into the input of the first LUT. The LUT would also take inputs A,B,C,D.

The second CLB's LUT implements the out signal logic. The inputs for the second LUT are state, A,B,C,D. The configuration bit for the second MUX would be permanently set at 0 since the D Flip Flop is not needed. The output of this CLB is the (out) signal.

c)


A
5
B
4
C
3
D
2
S
8
SAB
S'BC
S'ABC
SAC
BD
7
(out)
6
(next state)
D Flip Flop
1
CLK

d)


a
b
c
d
state (S)
NS =
(S')abc +
Sac +
bd
LUT
next_state
(NS)
DFF
>
0
1
1
R
CLK
state (S)
a
b
c
d
out =
(S')bc +
Sab
LUT
out
DFF
>
0
1
out
0
R
CLK