**PRE-LAB QUESTIONS OR EXERCISES**

N/A

**PROBLEMS ENCOUNTERED**

N/A

**REQUIREMENTS NOT MET**

N/A

**FUTURE WORK/APPLICATIONS**

This lab had us create a traffic light controller, which can already be applied to future applications because it is something that is already used daily. This lab also introduced us to more in-depth state machines and how we can create diagrams to help aid us in the creation of these state machines. The use of VHDL code instead of creating large amounts of combinatorial logic is also something that may help us in the future.

**PRE-LAB REQUIREMENTS (Design, Schematic, ASM Chart, VHDL, etc.)**

**ASM Chart:**

A close up of a street

Description automatically generated

**Next State Truth Table:**

A close up of a piece of paper

Description automatically generated

**Design/Simulation:**

A screenshot of a cell phone

Description automatically generated

**A screenshot of a cell phone

Description automatically generated**

The first section has no emergency vehicle and no car waiting so the light always remains green.

An emergency vehicle approaches so the light immediately turns yellow for at least one clock, but no more than two, then turns red for four clocks. Because there is no more emergency vehicle it goes back to green.

Car waiting turns true, so when the end of the current cycle occurs the light turns yellow for two clocks and red for four and finally goes back to green.

**VHDL Code:**

library ieee; use ieee.std\_logic\_1164.all;

entity Lab5\_DFF\_Traf\_Cont is port (

Q3, Q2, Q1, Q0: in std\_logic;

EV, CW: in std\_logic;

D3, D2, D1, D0: out std\_logic;

G, Y, R: out std\_logic

);

end Lab5\_DFF\_Traf\_Cont;

architecture logic OF Lab5\_DFF\_Traf\_Cont IS

begin

--D3 = (/Q3\*Q2\*Q1\*Q0)+(Q3\*/Q2\*/Q1\*/Q0)+(Q3\*/Q2\*/Q1\*Q0)+(Q3\*/Q2\*Q1\*/Q0)+(Q3\*/Q2\*Q1\*Q0\*EV)

--D2 = (/Q3\*/Q2\*/Q1\*/Q0\*EV)+(/Q3\*/Q2\*/Q1\*Q0\*EV)+(/Q3\*/Q2\*Q1\*/Q0\*EV)+(/Q3\*/Q2\*Q1\*Q0\*/EV)+(/Q3\*/Q2\*Q1\*Q0\*EV)+(/Q3\*Q2\*/Q1\*/Q0\*/EV)+(/Q3\*Q2\*/Q1\*/Q0\*EV)+(/Q3\*Q2\*/Q1\*Q0\*/EV\*CW)+(/Q3\*Q2\*/Q1\*Q0\*EV)+(/Q3\*Q2\*Q1\*/Q0)

--D1 = (Q3+Q2+Q1+Q0+EV)(Q3+Q2+/Q1+/Q0+EV)(Q3+/Q2+Q1+Q0+EV)(Q3+/Q2+Q1/Q0+EV+CW)(Q3+/Q2+/Q1+/Q0)(/Q3+Q2+Q1+Q0)(/Q3+Q2+/Q1+/Q0+EV)

--D0 = (Q3+Q2+Q1+/Q0+EV)(Q3+Q2+/Q1+/Q0+EV)(Q3+/Q2+Q1+/Q0+EV+CW)(Q3+/Q2+Q1+/Q0+EV+/CW)(Q3+/Q2+/Q1+/Q0)(/Q3+Q2+Q1+/Q0)(/Q3+Q2+/Q1+/Q0+EV)

--G = (/Q3\*/Q2\*/Q1\*/Q0\*/EV)+(/Q2\*/Q2\*/Q1\*Q0\*/EV)+(/Q3\*/Q2\*Q1\*/Q0\*/EV)+(/Q3\*/Q2\*Q1\*Q0\*/EV)+(/Q3\*Q2\*/Q1\*/Q0\*/EV)+(/Q3\*Q2\*/Q1\*Q0\*/EV\*/CW) (/Q3\*Q2\*/Q1\*Q0\*/EV\*CW)+(Q3\*/Q2\*Q1\*Q0\*/EV)

--Y = (/Q3\*/Q2\*/Q1\*/Q0\*EV)+(/Q2\*/Q2\*/Q1\*Q0\*EV)+(/Q3\*/Q2\*Q1\*/Q0\*EV)+(/Q3\*/Q2\*Q1\*Q0\*EV)+(/Q3\*Q2\*/Q1\*/Q0\*EV)+(/Q3\*Q2\*/Q1\*Q0\*EV)+(/Q3\*Q2\*Q1\*/Q0)(/Q3\*Q2\*Q1\*Q0)

--R = (Q3\*/Q2\*/Q1\*/Q0)+(Q3\*/Q2\*/Q1\*Q0)+(Q3\*/Q2\*Q1\*/Q0)+(Q3\*/Q2\*Q1\*Q0\*/EV)+(Q3\*/Q2\*Q1\*Q0\*EV)

D3<=(((NOT Q3)AND(Q2)AND(Q1)AND(Q0))

OR((Q3)AND(NOT Q2)AND(NOT Q1)AND(NOT Q0))

OR((Q3)AND(NOT Q2)AND(NOT Q1)AND(Q0))

OR((Q3)AND(NOT Q2)AND(Q1)AND(NOT Q0))

OR((Q3)AND(NOT Q2)AND(Q1)AND(Q0)AND(EV)));

D2<=(((NOT Q3)AND(NOT Q2)AND(NOT Q1)AND(NOT Q0)AND(EV))

OR((NOT Q3)AND(NOT Q2)AND(NOT Q1)AND(Q0)AND(EV))

OR((NOT Q3)AND(NOT Q2)AND(Q1)AND(NOT Q0)AND(EV))

OR((NOT Q3)AND(NOT Q2)AND(Q1)AND(Q0)AND(NOT EV))

OR((NOT Q3)AND(NOT Q2)AND(Q1)AND(Q0)AND(EV))

OR((NOT Q3)AND(Q2)AND(NOT Q1)AND(NOT Q0)AND(NOT EV))

OR((NOT Q3)AND(Q2)AND(NOT Q1)AND(NOT Q0)AND(EV))

OR((NOT Q3)AND(Q2)AND(NOT Q1)AND(Q0)AND(NOT EV)AND(CW))

OR((NOT Q3)AND(Q2)AND(NOT Q1)AND(Q0)AND(EV))

OR((NOT Q3)AND(Q2)AND(Q1)AND(NOT Q0)));

D1<=(((Q3)OR(Q2)OR(Q1)OR(Q0)OR(EV))

AND((Q3)OR(Q2)OR(NOT Q1)OR(NOT Q0)OR(EV))

AND((Q3)OR(NOT Q2)OR(Q1)OR(Q0)OR(EV))

AND((Q3)OR(NOT Q2)OR(Q1)OR(NOT Q0)OR(EV)OR(CW))

AND((Q3)OR(NOT Q2)OR(NOT Q1)OR(NOT Q0))

AND((NOT Q3)OR(Q2)OR(Q1)OR(Q0))

AND((NOT Q3)OR(Q2)OR(NOT Q1)OR(NOT Q0)OR(EV)));

D0<=(((Q3)OR(Q2)OR(Q1)OR(NOT Q0)OR(EV))

AND((Q3)OR(Q2)OR(NOT Q1)OR(NOT Q0)OR(EV))

AND((Q3)OR(NOT Q2)OR(Q1)OR(NOT Q0)OR(EV)OR(CW))

AND((Q3)OR(NOT Q2)OR(Q1)OR(NOT Q0)OR(EV)OR(NOT CW))

AND((Q3)OR(NOT Q2)OR(NOT Q1)OR(NOT Q0))

AND((NOT Q3)OR(Q2)OR(Q1)OR(NOT Q0))

AND((NOT Q3)OR(Q2)OR(NOT Q1)OR(NOT Q0)OR(EV)));

G<=(((NOT Q3)AND(NOT Q2)AND(NOT Q1)AND(NOT Q0)AND(NOT EV))

OR((NOT Q3)AND(NOT Q2)AND(NOT Q1)AND(Q0)AND(NOT EV))

OR((NOT Q3)AND(NOT Q2)AND(Q1)AND(NOT Q0)AND(NOT EV))

OR((NOT Q3)AND(NOT Q2)AND(Q1)AND(Q0)AND(NOT EV))

OR((NOT Q3)AND(Q2)AND(NOT Q1)AND(NOT Q0)AND(NOT EV))

OR((NOT Q3)AND(Q2)AND(NOT Q1)AND(Q0)AND(NOT EV)AND(NOT CW))

OR((NOT Q3)AND(Q2)AND(NOT Q1)AND(Q0)AND(NOT EV)AND(CW)));

Y<=(((NOT Q3)AND(NOT Q2)AND(NOT Q1)AND(NOT Q0)AND(EV))

OR((NOT Q3)AND(NOT Q2)AND(NOT Q1)AND(Q0)AND(EV))

OR((NOT Q3)AND(NOT Q2)AND(Q1)AND(NOT Q0)AND(EV))

OR((NOT Q3)AND(NOT Q2)AND(Q1)AND(Q0)AND(EV))

OR((NOT Q3)AND(Q2)AND(NOT Q1)AND(NOT Q0)AND(EV))

OR((NOT Q3)AND(Q2)AND(NOT Q1)AND(Q0)AND(EV))

OR((NOT Q3)AND(Q2)AND(Q1)AND(NOT Q0))

OR((NOT Q3)AND(Q2)AND(Q1)AND(Q0)));

R<=(((Q3)AND(NOT Q2)AND(NOT Q1)AND(NOT Q0))

OR((Q3)AND(NOT Q2)AND(NOT Q1)AND(Q0))

OR((Q3)AND(NOT Q2)AND(Q1)AND(NOT Q0))

OR((Q3)AND(NOT Q2)AND(Q1)AND(Q0)AND(NOT EV))

OR((Q3)AND(NOT Q2)AND(Q1)AND(Q0)AND(EV)));

end logic;