A picture containing chart

Description automatically generated

# Counter.vhd

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity counter is

port(

reset, clock: in std\_logic;

done: out std\_logic;

count\_out: out std\_logic\_vector(31 downto 0)

);

end counter;

architecture counter\_logic of counter is

signal counter\_next: std\_logic\_vector(31 downto 0) := "00000000000000000000000000000000";

signal counter\_1s: std\_logic\_vector(31 downto 0) := "00000000000000000000000000000001";

signal counter\_3s: std\_logic\_vector(31 downto 0) := "00000000000000000000000000000011";

signal counter\_6s: std\_logic\_vector(31 downto 0) := "00000000000000000000000000000110";

signal done\_next: std\_logic;

begin

count\_out <= counter\_next;

done <= done\_next;

process(clock) begin

if(rising\_edge(clock)) then

if(reset='1') then

counter\_next <= "00000000000000000000000000000000";

done\_next <= '0';

else

counter\_next <= counter\_next + 1;

if(counter\_next = counter\_1s or counter\_next=counter\_3s or counter\_next=counter\_6s) then

done\_next <= '1';

else

done\_next <= '0';

end if;

end if;

end if;

end process;

end architecture;

# Lab2.vhd

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity Lab2 is

port(

reset, start, clock: in std\_logic;

red, yellow, green: out std\_logic;

count\_out: out std\_logic\_vector(31 downto 0)

);

end Lab2;

architecture Lab2\_logic of Lab2 is

type state\_available is (Idle, Ready, Set, Go);

signal present\_state, next\_state : state\_available;

signal count\_reset: std\_logic;

signal done: std\_logic;

component counter

port(

reset, clock: in std\_logic;

done: out std\_logic;

count\_out: out std\_logic\_vector(31 downto 0)

);

end component;

begin

count\_reset <= reset or start;

c0: counter port map (count\_reset, clock, done, count\_out);

process(clock, reset) begin

if(reset='1') then

present\_state <= Idle;

elsif(rising\_edge(clock)) then

present\_state <= next\_state;

end if;

end process;

process(present\_state, start, done) begin

case present\_state is

when Idle =>

red <= '1';

yellow <= '0';

green <= '0';

if(start='0') then

next\_state <= Idle;

else

next\_state <= Ready;

end if;

when Ready =>

red <= '1';

yellow <= '0';

green <= '0';

if(done = '1') then

next\_state <= Set;

else

next\_state <= Ready;

end if;

when Set =>

red <= '0';

yellow <= '1';

green <= '0';

if(done = '1') then

next\_state <= Go;

else

next\_state <= Set;

end if;

when Go =>

red <= '0';

yellow <= '0';

green <= '1';

if(done = '1') then

next\_state <= Idle;

else

next\_state <= Go;

end if;

end case;

end process;

end Lab2\_logic;