Graphical user interface

Description automatically generated

# RegN.vhd

-- ENABLE ACTIVE LOW N-bit REGISTER

library ieee;

use ieee.std\_logic\_1164.all;

entity regN is

generic(N: integer:= 6);

port(

clock: in std\_logic;

en: in std\_logic;

D: in std\_logic\_vector((N-1) downto 0);

Q: out std\_logic\_vector((N-1) downto 0)

);

end regN;

architecture regN\_arch of regN is begin

process(clock) begin

if(rising\_edge(clock)) then

if(en='0') then

Q <= D;

end if;

end if;

end process;

end regN\_arch;

# tristateN.vhd

-- ENABLE ACTIVE LOW N-bit TRISTATE BUFFER

library ieee;

use ieee.std\_logic\_1164.all;

entity tristateN is

generic(N: integer:= 6);

port(

enable: in std\_logic;

input: in std\_logic\_vector((N-1) downto 0);

output: out std\_logic\_vector((N-1) downto 0)

);

end tristateN;

architecture tristate\_arch of tristateN is begin

output <= input when(enable = '0') else (OTHERS => 'Z');

end tristate\_arch;

# Lab3.vhd

library ieee;

use ieee.std\_logic\_1164.all;

entity Lab3 is

port(

clock: in std\_logic;

Ld: in std\_logic;

ReadAddr: in std\_logic\_vector(1 downto 0);

WriteAddr: in std\_logic\_vector(1 downto 0);

Memin: in std\_logic\_vector(5 downto 0);

Memout: out std\_logic\_vector(5 downto 0)

);

end Lab3;

architecture Lab3\_arch of Lab3 is

signal rout0, rout1, rout2, rout3: std\_logic\_vector(5 downto 0);

signal ren0, ren1, ren2, ren3: std\_logic;

signal tri0, tri1, tri2, tri3: std\_logic;

component regN is

generic(N: integer:= 6);

port(

clock: in std\_logic;

en: in std\_logic;

D: in std\_logic\_vector(N-1 downto 0);

Q: out std\_logic\_vector(N-1 downto 0)

);

end component;

component tristateN is

generic(N: integer:= 6);

port(

enable: in std\_logic;

input: in std\_logic\_vector((N-1) downto 0);

output: out std\_logic\_vector((N-1) downto 0)

);

end component;

begin

reg0: regN generic map(N=>6) port map(clock, ren0, Memin, rout0);

t0: tristateN generic map(N=>6) port map(tri0, rout0, Memout);

reg1: regN generic map(N=>6) port map(clock, ren1, Memin, rout1);

t1: tristateN generic map(N=>6) port map(tri1, rout1, Memout);

reg2: regN generic map(N=>6) port map(clock, ren2, Memin, rout2);

t2: tristateN generic map(N=>6) port map(tri2, rout2, Memout);

reg3: regN generic map(N=>6) port map(clock, ren3, Memin, rout3);

t3: tristateN generic map(N=>6) port map(tri3, rout3, Memout);

process(ReadAddr) begin

end process;

process(clock) begin

if(rising\_edge(clock)) then

case ReadAddr is

when "00" =>

tri0 <= '0';

tri1 <= '1';

tri2 <= '1';

tri3 <= '1';

when "01" =>

tri0 <= '1';

tri1 <= '0';

tri2 <= '1';

tri3 <= '1';

when "10" =>

tri0 <= '1';

tri1 <= '1';

tri2 <= '0';

tri3 <= '1';

when "11" =>

tri0 <= '1';

tri1 <= '1';

tri2 <= '1';

tri3 <= '0';

when others =>

tri0 <= '1';

tri1 <= '1';

tri2 <= '1';

tri3 <= '1';

end case;

if(Ld='1') then

case WriteAddr is

when "00" =>

ren0 <= '0';

ren1 <= '1';

ren2 <= '1';

ren3 <= '1';

when "01" =>

ren0 <= '1';

ren1 <= '0';

ren2 <= '1';

ren3 <= '1';

when "10" =>

ren0 <= '1';

ren1 <= '1';

ren2 <= '0';

ren3 <= '1';

when "11" =>

ren0 <= '1';

ren1 <= '1';

ren2 <= '1';

ren3 <= '0';

when others =>

ren0 <= '1';

ren1 <= '1';

ren2 <= '1';

ren3 <= '1';

end case;

else

ren0 <= '1';

ren1 <= '1';

ren2 <= '1';

ren3 <= '1';

end if;

end if;

end process;

end Lab3\_arch;