Ian Parker

Homework 5

D flip-flop

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity DFlipFlop is

PORT( D,CLOCK: in std\_logic;

Q,QNOT: out std\_logic);

end DFlipFlop;

architecture behavioral of DFlipFlop is

signal Qtemp : std\_logic;

begin

process(CLOCK)

begin

if((CLOCK='1') and (CLOCK'EVENT)) then

Qtemp <= D;

end if;

end process;

Q <= Qtemp;

QNOT <= NOT Qtemp;

end behavioral;

Test Bench

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

use ieee.numeric\_std.all;

entity TestBench\_DFlipFlop is

end TestBench\_DFlipFlop;

architecture behavioral of TestBench\_DFlipFlop is

signal D : std\_logic;

signal CLOCK : std\_logic;

signal Q : std\_logic;

signal QNOT : std\_logic;

component DFlipFlop is port(D,CLOCK : in std\_logic;

Q,QNOT: out std\_logic);

end component;

begin

DFF : DFlipFlop port map(D,CLOCK,Q,QNOT);

process

begin

CLOCK <= '0';

D <= '0';

wait for 10 ns;

CLOCK <= '1';

D <= '0';

wait for 10 ns;

CLOCK <= '0';

D <= '1';

wait for 10 ns;

CLOCK <= '1';

D <= '1';

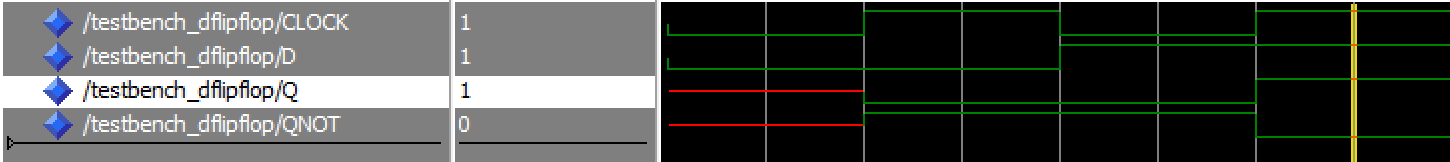
wait for 10 ns;

wait;

end process;

end behavioral;

Simulation



D flip-flop       with enable and reset

library ieee;

use ieee. std\_logic\_1164.all;

use ieee. std\_logic\_arith.all;

use ieee. std\_logic\_unsigned.all;

entity DFlipFlop\_Enable\_Reset is

PORT( D,CLOCK, RESET, ENABLE: in std\_logic;

Q, QNOT: out std\_logic);

end DFlipFlop\_Enable\_Reset;

architecture behavioral of DFlipFlop\_Enable\_Reset is

signal Qtemp : std\_logic;

begin

process(CLOCK, ENABLE, RESET)

begin

if(RESET='1' and RESET'EVENT) then

QTemp <= '0';

elsif(CLOCK='1' and CLOCK'EVENT) then

if(ENABLE='1' and ENABLE'EVENT) then

Qtemp <= D;

end if;

end if;

end process;

Q <= QTemp;

QNOT <= NOT QTemp;

end behavioral;

Test Bench

library ieee;

use ieee. std\_logic\_1164.all;

use ieee. std\_logic\_arith.all;

use ieee. std\_logic\_unsigned.all;

entity TestBench\_DFlipFlop\_Enable\_Reset is

end TestBench\_DFlipFlop\_Enable\_Reset;

architecture behavioral of TestBench\_DFlipFlop\_Enable\_Reset is

signal D : std\_logic;

signal CLOCK : std\_logic;

signal RESET : std\_logic;

signal ENABLE : std\_logic;

signal Q : std\_logic;

signal QNOT : std\_logic;

component DFlipFlop\_Enable\_Reset is port(D,CLOCK,RESET,ENABLE : in std\_logic;

Q,QNOT : out std\_logic);

end component;

begin

DFF : DFlipFlop\_Enable\_Reset port map(D,CLOCK,RESET,ENABLE,Q,QNOT);

process

begin

CLOCK <= '0';

ENABLE <= '0';

RESET <= '0';

D <= '0';

wait for 10 ns;

CLOCK <= '1';

ENABLE <= '0';

RESET <= '0';

D <= '0';

wait for 10 ns;

CLOCK <= '0';

ENABLE <= '1';

RESET <= '0';

D <= '0';

wait for 10 ns;

CLOCK <= '1';

ENABLE <= '1';

RESET <= '0';

D <= '0';

wait for 10 ns;

CLOCK <= '0';

ENABLE <= '0';

RESET <= '1';

D <= '0';

wait for 10 ns;

CLOCK <= '1';

ENABLE <= '0';

RESET <= '1';

D <= '0';

wait for 10 ns;

CLOCK <= '0';

ENABLE <= '1';

RESET <= '1';

D <= '0';

wait for 10 ns;

CLOCK <= '1';

ENABLE <= '1';

RESET <= '1';

D <= '0';

wait for 10 ns;

CLOCK <= '0';

ENABLE <= '0';

RESET <= '0';

D <= '1';

wait for 10 ns;

CLOCK <= '1';

ENABLE <= '0';

RESET <= '0';

D <= '1';

wait for 10 ns;

CLOCK <= '0';

ENABLE <= '1';

RESET <= '0';

D <= '1';

wait for 10 ns;

CLOCK <= '1';

ENABLE <= '1';

RESET <= '0';

D <= '1';

wait for 10 ns;

CLOCK <= '0';

ENABLE <= '0';

RESET <= '1';

D <= '1';

wait for 10 ns;

CLOCK <= '1';

ENABLE <= '0';

RESET <= '1';

D <= '1';

wait for 10 ns;

CLOCK <= '0';

ENABLE <= '1';

RESET <= '1';

D <= '1';

wait for 10 ns;

CLOCK <= '1';

ENABLE <= '1';

RESET <= '1';

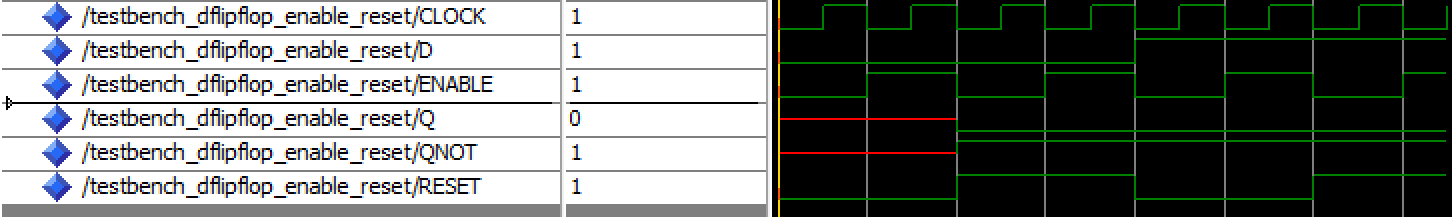
D <= '1';

wait;

end process;

end behavioral;

Simulation



1. J-K flip flop with asynchronous set

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity JKFlipFlop is

port(J,K,SET,CLOCK:in std\_logic;

Q,QNOT:out std\_logic);

end JKFlipFlop;

architecture Behavioral of JKFlipFlop is

signal Qtemp : std\_logic;

begin

process(CLOCK,J,K,SET)

begin

if (CLOCK'EVENT AND CLOCK='1') then

Qtemp <= J;

end if;

if(SET = '1') then

Qtemp <= '1';

elsif(SET = '0') then

if (J='0' and K='0')then

Qtemp<=Qtemp;

elsif(j='0' and k='1') then

Qtemp<='0';

elsif (J='1' and K='0') then

Qtemp<='1';

elsif(J='1' and K='1')then

Qtemp<= NOT Qtemp;

end if;

end if;

Q<= Qtemp;

QNOT<= NOT Qtemp;

end process;

end Behavioral;

Test Bench

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

use ieee.std\_logic\_unsigned.all;

entity TestBench\_JKFlipFlop is

end TestBench\_JKFlipFlop;

architecture behavioral of TestBench\_JKFlipFlop is

signal J : std\_logic;

signal K : std\_logic;

signal SET : std\_logic;

signal CLOCK : std\_logic;

signal Q : std\_logic;

signal QNOT : std\_logic;

component JKFlipFlop is port(J,K,SET,CLOCK : in std\_logic;

Q,QNOT: out std\_logic);

end component;

begin

JKFF : JKFlipFlop port map(J,K,SET,CLOCK,Q,QNOT);

process

begin

CLOCK <= '0';

J <= '0';

K <= '0';

SET <= '0';

wait for 10 ns;

CLOCK <= '1';

J <= '0';

K <= '0';

SET <= '0';

wait for 10 ns;

CLOCK <= '0';

J <= '1';

K <= '0';

SET <= '0';

wait for 10 ns;

CLOCK <= '1';

J <= '1';

K <= '0';

SET <= '0';

wait for 10 ns;

CLOCK <= '0';

J <= '0';

K <= '1';

SET <= '0';

wait for 10 ns;

CLOCK <= '1';

J <= '0';

K <= '1';

SET <= '0';

wait for 10 ns;

CLOCK <= '0';

J <= '1';

K <= '1';

SET <= '0';

wait for 10 ns;

CLOCK <= '1';

J <= '1';

K <= '1';

SET <= '0';

wait for 10 ns;

CLOCK <= '0';

J <= '0';

K <= '0';

SET <= '1';

wait for 10 ns;

CLOCK <= '1';

J <= '0';

K <= '0';

SET <= '1';

wait for 10 ns;

CLOCK <= '0';

J <= '1';

K <= '0';

SET <= '1';

wait for 10 ns;

CLOCK <= '1';

J <= '1';

K <= '0';

SET <= '1';

wait for 10 ns;

CLOCK <= '0';

J <= '0';

K <= '1';

SET <= '1';

wait for 10 ns;

CLOCK <= '1';

J <= '0';

K <= '1';

SET <= '1';

wait for 10 ns;

CLOCK <= '0';

J <= '1';

K <= '1';

SET <= '1';

wait for 10 ns;

CLOCK <= '1';

J <= '1';

K <= '1';

SET <= '1';

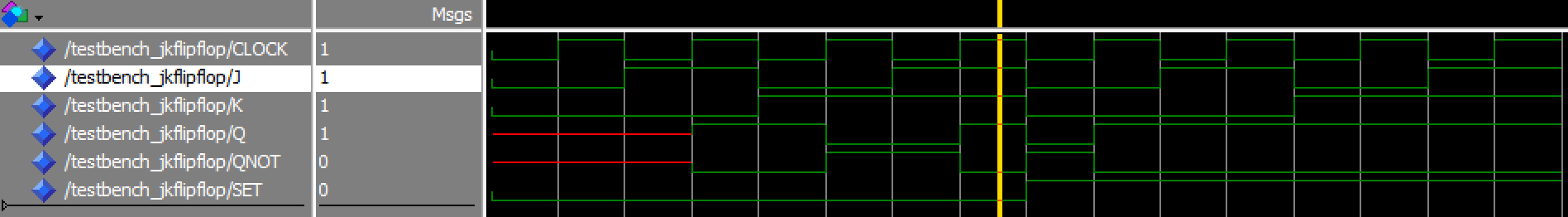
wait for 10 ns;

wait;

end process;

end behavioral;

Simulation



T Flip flop with asynchronous clear

library ieee;

use ieee.std\_logic\_1164.all;

entity TFlipFlop is

port (T,CLOCK,CLEAR : in std\_logic;

Q,QNOT : out std\_logic);

end TFlipFlop;

architecture Behavioral of TFlipFlop is

signal Qtemp : std\_logic;

begin

process (T,CLOCK,CLEAR)

begin

if (CLOCK'EVENT AND CLOCK='1') then

Qtemp <= T;

end if;

if (CLEAR = '1') then

Qtemp <= T;

elsif(CLEAR = '0') then

if (T = '0') then

Qtemp <= Qtemp;

elsif(T = '1') then

Qtemp <= NOT Qtemp;

end if;

end if;

end process;

Q <= Qtemp;

QNOT <= NOT Qtemp;

end Behavioral;

Test Bench

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

use ieee.std\_logic\_unsigned.all;

entity TestBench\_TFlipFlop is

end TestBench\_TFlipFlop;

architecture behavioral of TestBench\_TFlipFlop is

signal T : std\_logic;

signal CLOCK : std\_logic;

signal CLEAR : std\_logic;

signal Q : std\_logic;

signal QNOT : std\_logic;

component TFlipFlop is port(T,CLOCK,CLEAR : in std\_logic;

Q,QNOT: out std\_logic);

end component;

begin

TFF : TFlipFlop port map(T,CLOCK,CLEAR,Q,QNOT);

process

begin

CLOCK <= '0';

T <= '0';

CLEAR <= '0';

wait for 10 ns;

CLOCK <= '1';

T <= '0';

CLEAR <= '0';

wait for 10 ns;

CLOCK <= '0';

T <= '1';

CLEAR <= '0';

wait for 10 ns;

CLOCK <= '1';

T <= '1';

CLEAR <= '0';

wait for 10 ns;

CLOCK <= '0';

T <= '0';

CLEAR <= '1';

wait for 10 ns;

CLOCK <= '1';

T <= '0';

CLEAR <= '1';

wait for 10 ns;

CLOCK <= '0';

T <= '1';

CLEAR <= '1';

wait for 10 ns;

CLOCK <= '1';

T <= '1';

CLEAR <= '1';

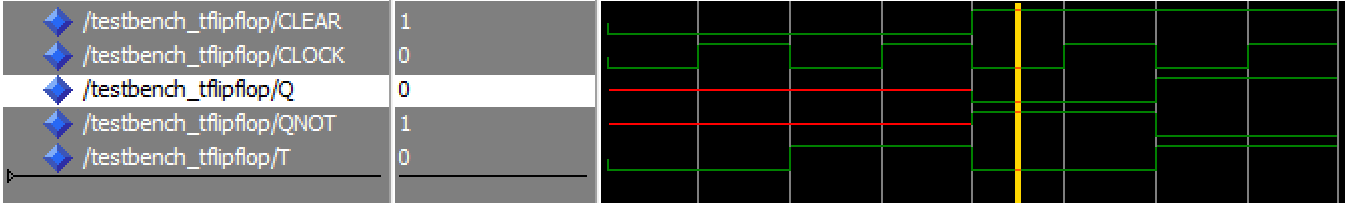
wait for 10 ns;

wait;

end process;

end behavioral;

Simulation



YOUTUBE VIDEO

https://youtu.be/XTUBRQY-n0g