**Exercise 1**

Diagram

Description automatically generated

The LUT equation is: O = I0 & I2 + !I0 & I1 + !I1 & !I2

Equation Breakdown:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Partial Results** | | | **Final Result** |
| Partial Equ. | I0 & I2 | !I0 & I1 | !I1 & !I2 | I0 & I2 + !I0 & I1 + !I1 & !I2 |
| 000 | 0 | 0 | 1 | 1 |
| 001 | 0 | 0 | 0 | 0 |
| 010 | 0 | 1 | 0 | 1 |
| 011 | 0 | 1 | 0 | 1 |
| 100 | 0 | 0 | 1 | 1 |
| 101 | 1 | 0 | 0 | 1 |
| 110 | 0 | 0 | 0 | 0 |
| 111 | 1 | 0 | 0 | 1 |

This LUT equation uses three separate AND statements (all ORed together) to build a result. Each partial result has 2 cases where they are set and between each partial result, no ‘1’s overlap.

**Exercise 2**

Diagram

Description automatically generated

The LUT equation is: O = I0 & !I1 & !I2 + !I0 & I1 + I1 & I2

Equation Breakdown:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Partial Results** | | | **Final Result** |
| Partial Equ. | I0 & !I1 & !I2 | !I2 + !I0 | I1 & I2 | O = I0 & !I1 & !I2 + !I0 & I1 + I1 & I2 |
| 000 | 0 | 0 | 0 | 0 |
| 001 | 0 | 0 | 0 | 0 |
| 010 | 0 | 1 | 0 | 1 |
| 011 | 0 | 0 | 0 | 0 |
| 100 | 1 | 0 | 0 | 1 |
| 101 | 0 | 0 | 1 | 1 |
| 110 | 0 | 0 | 0 | 0 |
| 111 | 0 | 0 | 1 | 1 |

This LUT equation uses three separate AND statements (all ORed together) to build a result. One partial equation is more complex than the other two.

Graphical user interface, timeline

Description automatically generated**Exercise 3**

library ieee;

use ieee.std\_logic\_1164.all;

use work.Hw5Q3pkgSoln.all;

entity Hw5Q3memTB is

end Hw5Q3memTB;

architecture tb of Hw5Q3memTB is

signal addr : std\_logic\_vector (a\_width-1 downto 0) := (others => '0');

signal din : std\_logic\_vector (d\_width-1 downto 0) := (others => '0');

signal clk, we : std\_logic := '0';

signal dout : std\_logic\_vector (d\_width-1 downto 0) := (others => '0');

begin

dut : entity work.Hw5Q3mem

port map (addr => addr, din => din, clk => clk, we => we, dout => dout);

clk <= not clk after clk\_per/2;

stimuli : process

begin

we <= '0';

addr <= "000";

wait for 10 ns;

assert (x"FE" = dout) report "error in result. dout is " &

to\_hstring (dout) & " and should be = FE";

wait for 10 ns;

-- code to read and check location 1

addr <= "001";

wait for 10 ns;

assert (x"FE" = dout) report "error in result. dout is " &

to\_hstring (dout) & " and should be = FE";

wait for 10 ns;

-- code to read and check location 6

addr <= "110";

wait for 10 ns;

assert (x"FE" = dout) report "error in result. dout is " &

to\_hstring (dout) & " and should be = FE";

wait for 10 ns;

wait until clk = '0';

we <= '1';

addr <= "000";

din <= x"FF";

wait until clk = '0';

addr <= "100";

wait until clk = '0';

addr <= "110";

-- code to read and check location 0

wait until clk = '0';

addr <= "000";

wait for 10 ns;

assert (x"FF" = dout) report "error in result. dout is " &

to\_hstring (dout) & " and should be = FF";

wait for 10 ns;

-- read and check location 4

addr <= "100";

wait for 10 ns;

assert (x"FF" = dout) report "error in result. dout is " &

to\_hstring (dout) & " and should be = FF";

wait for 10 ns;

-- read and check location 6

addr <= "110";

wait for 10 ns;

assert (x"FF" = dout) report "error in result. dout is " &

to\_hstring (dout) & " and should be = FF";

wait for 10 ns;

wait;

end process;

end tb;

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

use work.Hw5Q3pkgSoln.all;

entity Hw5Q3mem is

port (

addr: in std\_logic\_vector (a\_width -1 downto 0);

din: in std\_logic\_vector (d\_width -1 downto 0);

clk, we: in std\_logic;

dout: out std\_logic\_vector (d\_width -1 downto 0)

);

end entity Hw5Q3mem;

architecture obehavior of Hw5Q3mem is

signal mem : mem\_type(0 to (a\_width\*\*2)-1)(d\_width-1 downto 0) := (x"FE", x"0F",others => x"01");

begin

process (clk, we) is begin

if (rising\_edge(clk) and we = '1') then

mem(to\_integer(unsigned(addr))) <= din;

end if;

end process;

dout <= mem(to\_integer(unsigned(addr)));

end obehavior;

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

package Hw5Q3pkgSoln is

constant a\_width: positive := 3;

constant d\_width: positive := 8;

-- add a constant d\_width and set it to 8

type mem\_type is array (natural range <>) of std\_logic\_vector;

constant clk\_per: time := 20 ns;

end package Hw5Q3pkgSoln;