***XILINE VIVADO Simulation:***

**Boolean Expression Design:**

**(A, B, C, D)=**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 1 | 1 |
| 1 | 1 |  | 1 |
| 1 | 1 | 1 |  |
| 1 |  |  | 1 |

**X=**(C/B+AC/D/+ABD+A/CD/+A/B/C+B/CD/)

**Explanation:**

In this task we implement the K\_map actually k\_map is used to convert the truth table to Boolean expression as shown above, this work on 1,2 4, 8 pairs , Here we deal with SOP mean sum of product to make it simple we take the 1 in K\_map if it was POS (product of sum) we take the 0 in K\_Map.

**Truth Table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **Y** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

**Explanation:**

The above table called truth table , it play a leading role in design of logic circuit, without this we are not able to create logic circuit in this task we have given some parameters, where it was locate we take here 1 and other will be zero.

**VHDL Code:**

**Design.vhd:**

-- Simple design

library IEEE;

use IEEE.std\_logic\_1164.all;

entity combine is

port(

a: in std\_logic;

b: in std\_logic;

c: in std\_logic;

d: in std\_logic;

q: out std\_logic);

end or\_gate;

architecture Behavioral of combine is

begin

process(a, b,c,d)

begin

if(a='0' and b='0' and c='0' and d='0') then

q<='0';

else if(a='0' and b='0' and c='0' and d='1') then

q<='0';

else if(a='0' and b='0' and c='1' and d='0') then

q<='1';

else if(a='0' and b='0' and c='1' and d='1') then

q<='1';

else if(a='0' and b='1' and c='0' and d='0') then

q<='1';

else if(a='0' and b='1' and c='0' and d='1') then

q<='1';

else if(a='0' and b='1' and c='1' and d='0') then

q<='1';

else if(a='0' and b='1' and c='1' and d='1') then

q<='0';

else if(a='1' and b='0' and c='0' and d='0') then

q<='1';

else if(a='1' and b='0' and c='0' and d='1') then

q<='0';

else if(a='1' and b='0' and c='1' and d='0') then

q<='1';

else if(a='1' and b='0' and c='1' and d='1') then

q<='0';

else if(a='1' and b='1' and c='0' and d='0') then

q<='1';

else if(a='1' and b='1' and c='0' and d='1') then

q<='1';

else if(a='1' and b='1' and c='1' and d='0') then

q<='0';

else if(a='1' and b='1' and c='1' and d='1') then

q<='1';

end if;

end if;

end if;

end if;

end if;

end if;

end if;

end if;

end if;

end if;

end if;

end if;

end if;

end if;

end if;

end if;

end process;

end Behavioral;

**TestBench.vhd**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity testbench is

-- empty

end testbench;

architecture tb of testbench is

-- DUT component

component combine is

port(

a: in std\_logic;

b: in std\_logic;

c: in std\_logic;

d: in std\_logic;

q: out std\_logic);

end component;

signal a\_in, b\_in,c\_in,d\_in, q\_out: std\_logic;

begin

-- Connect DUT

DUT: combine port map(a\_in, b\_in,c\_in,d\_in, q\_out);

process

begin

a\_in <= '0';

b\_in <= '0';

c\_in <= '0';

d\_in <= '0';

wait for 100 ns;

a\_in <= '0';

b\_in <= '0';

c\_in <= '0';

d\_in <= '1';

wait for 100 ns;

a\_in <= '0';

b\_in <= '0';

c\_in <= '1';

d\_in <= '0';

wait for 100 ns;

a\_in <= '0';

b\_in <= '0';

c\_in <= '1';

d\_in <= '1';

wait for 100 ns;

a\_in <= '0';

b\_in <= '1';

c\_in <= '0';

d\_in <= '0';

wait for 100 ns;

a\_in <= '0';

b\_in <= '1';

c\_in <= '0';

d\_in <= '1';

wait for 100 ns;

a\_in <= '0';

b\_in <= '1';

c\_in <= '1';

d\_in <= '0';

wait for 100 ns;

a\_in <= '0';

b\_in <= '1';

c\_in <= '1';

d\_in <= '1';

wait for 100 ns;

a\_in <= '1';

b\_in <= '0';

c\_in <= '0';

d\_in <= '0';

wait for 100 ns;

a\_in <= '1';

b\_in <= '0';

c\_in <= '0';

d\_in <= '1';

wait for 100 ns;

a\_in <= '1';

b\_in <= '0';

c\_in <= '1';

d\_in <= '0';

wait for 100 ns;

a\_in <= '1';

b\_in <= '0';

c\_in <= '1';

d\_in <= '1';

wait for 100 ns;

a\_in <= '1';

b\_in <= '1';

c\_in <= '0';

d\_in <= '0';

wait for 100 ns;

a\_in <= '1';

b\_in <= '1';

c\_in <= '0';

d\_in <= '1';

wait for 100 ns;

a\_in <= '1';

b\_in <= '1';

c\_in <= '1';

d\_in <= '0';

wait for 100 ns;

a\_in <= '1';

b\_in <= '1';

c\_in <= '1';

d\_in <= '1';

wait for 100 ns;

-- Clear inputs

a\_in <= '0';

b\_in <= '0';

c\_in <= '0';

d\_in <= '0';

wait;

end process;

end tb;

**Waveform:**



