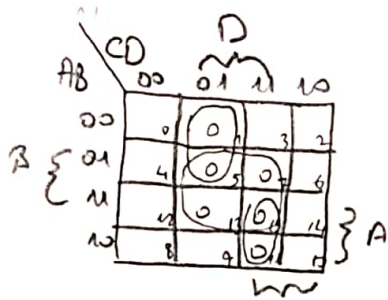


ÖDEV 3

Yigit Bektaş Güşay
090180063
Yazgı

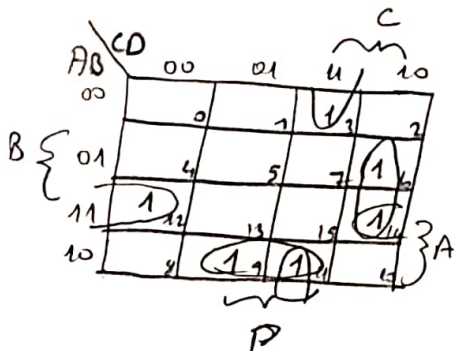
1-

a-) $F(A,B,C,D) = \Pi(1,5,7,11,13,15)$



$$F = (B'D')(A'+C+D')(A+C+D')$$

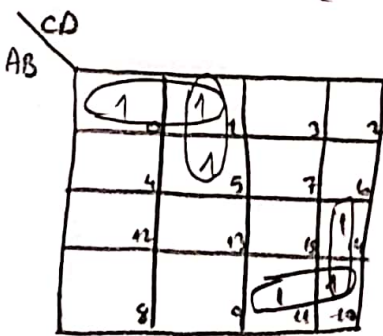
b-) $F(A,B,C,D) = \Sigma(3,6,9,11,12,16)$



$$F = ABD' + B'CD + AB'D + BCD'$$

2-) $F = A'BC'D + AB'CD + A'B'C' + ACD' \Rightarrow F = \Sigma(0,1,5,10,11,14)$

*) Karnaugh Gruplarının toplamı (minterm)



$$F = A'CD + A'BC' + AB'C + ACD'$$

$$F = A'C(B'+D) + AC(B'+D) \rightarrow 8 \text{ literal}$$

*) Quine McCluskey

	Sutun 1	Sutun 2
✓0	0 0 0 0	(0,1) 0 0 0 -
✓1	0 0 0 1	(1,5) 0 - 0 1
✓5	0 1 0 1	(10,11) 1 0 1 -
✓10	1 0 1 0	(10,14) 1 - 1 0
✓11	1 0 1 1	
✓14	1 1 1 0	

$$F = A'BC' + A'CD' + AB'C + ACD'$$

$$F = A'C(B'+D) + AC(B'+D) \rightarrow 8 \text{ literal}$$

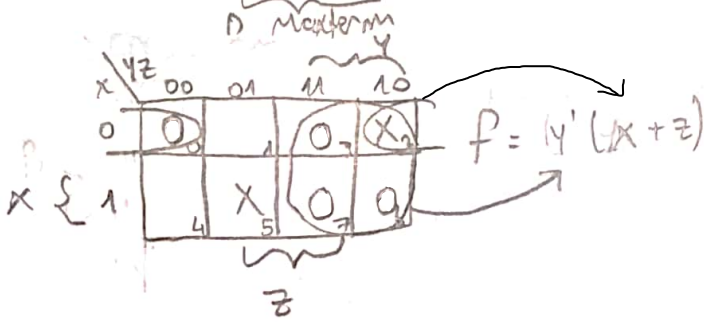
*) Boolean sadeleştirme

$$F = A'C'(BD+B') + AC(B'D+D') = A'C'(D+B') + AC(B'+D') \rightarrow 8 \text{ literal}$$

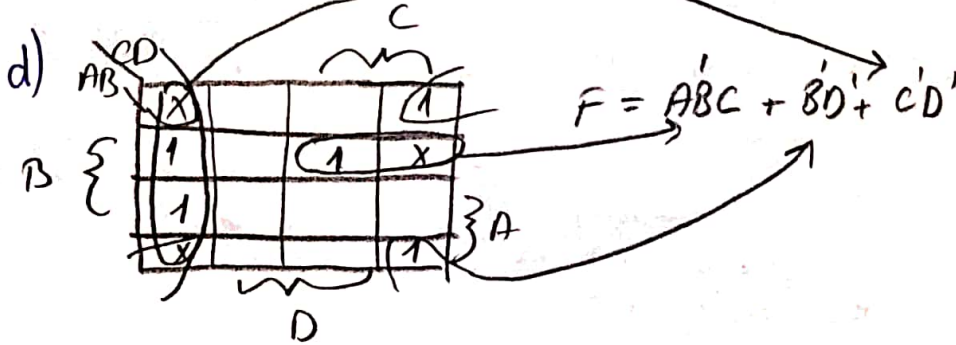
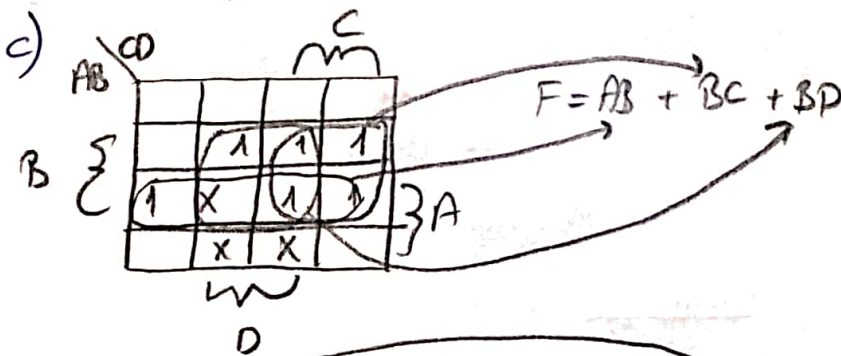
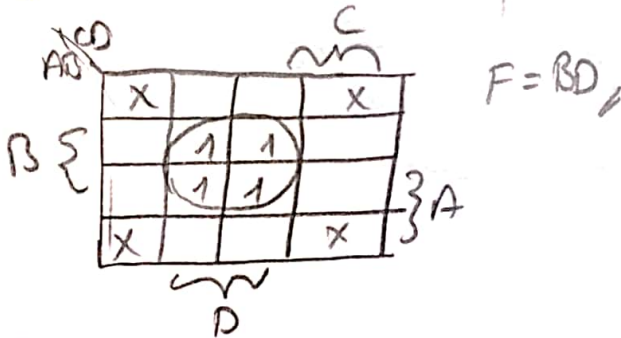
1)

3-)

a) $F(x,y,z) = \Pi(0,3,6,7) + d\Sigma(2,5)$



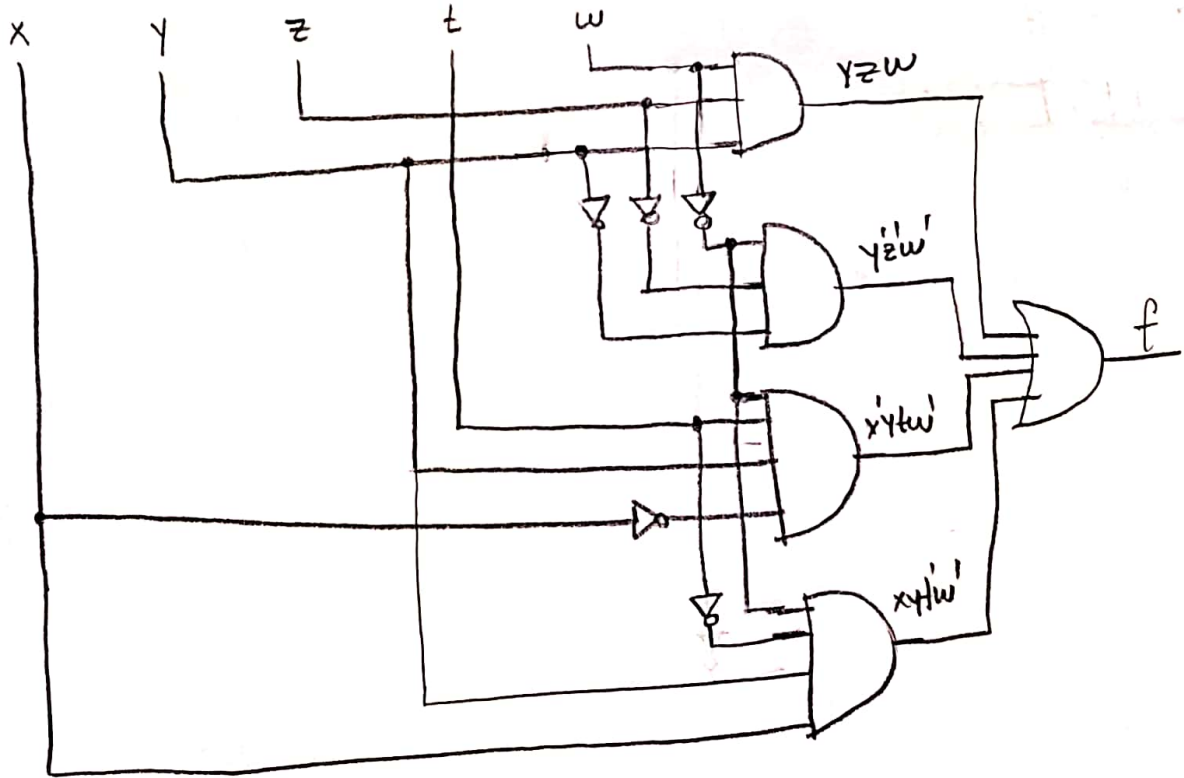
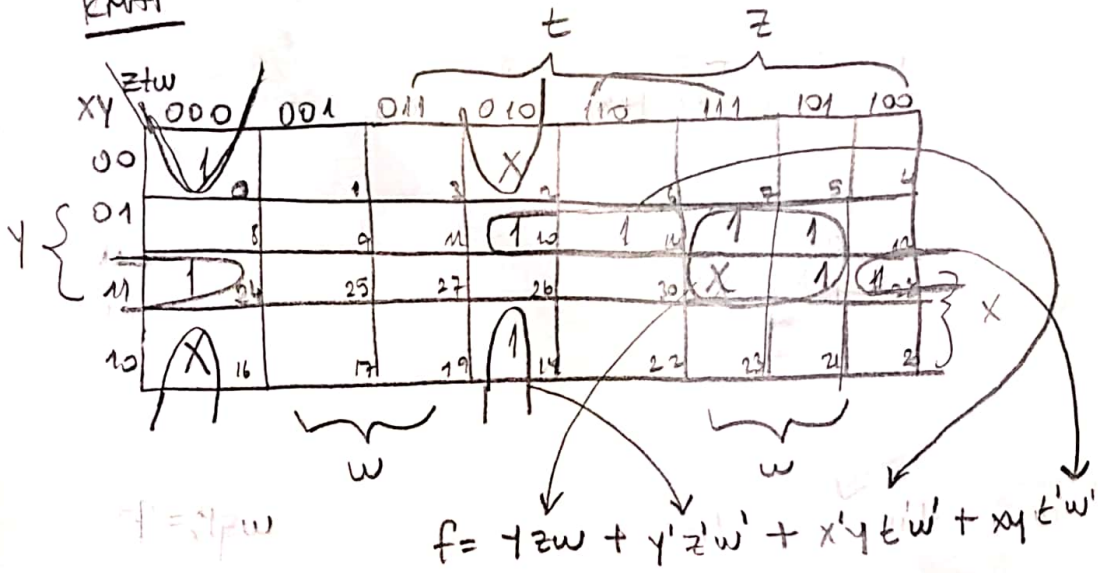
b) $F(A,B,C,D) = \Sigma(5,7,13,15) + d\Sigma(0,2,8,10)$



Part 2 - $f(x,y,z,t,w) = \sum 0,10,13,14,15,18,24,28,29 \rightarrow d \sum 2,16,31$

Yigit Bektaş Gürsey
060180060
yigitm

KMAP



1. VE KAPISI

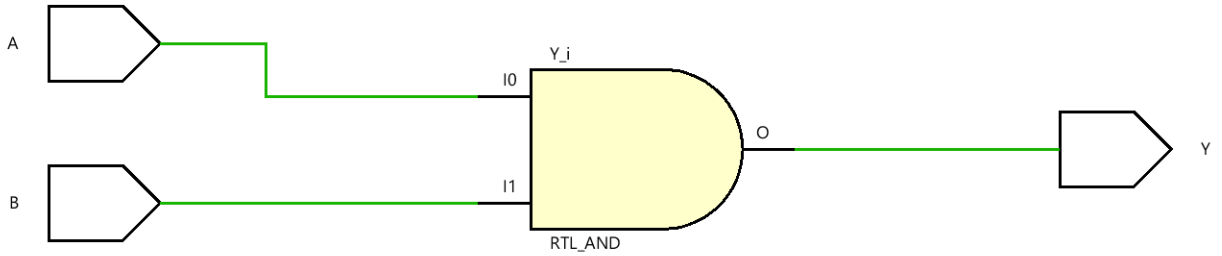
- Ödevde açıklandığı gibi proje oluşturuldu.
- Ve kapısı için bir modül oluşturdum ve modülüne AND_gate ismini verdim.
- Ardından AND_gate için A ve B adında iki input Y adında bir output tanımlayarak behavioral kısmında fonksiyonu gerçekledim.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity AND_gate is
    Port ( A : in STD_LOGIC;
          B : in STD_LOGIC;
          Y : out STD_LOGIC);
end AND_gate;

architecture Behavioral of AND_gate
is
begin
    Y <= A AND B;
end Behavioral;
```

- RTL şematik aşağıda belirtilmiştir.



- Kodumuzun doğruluğunu kontrol etmek için testbench dosyası oluşturuyoruz. Yazılan kod ve simülasyon sonucu aşağıdadır.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity AND_gate_tb is
end AND_gate_tb;

architecture Behavioral of AND_gate_tb is
    COMPONENT AND_gate
    PORT ( A : in STD_LOGIC;
          B : in STD_LOGIC;
          Y : out STD_LOGIC
        );
    END COMPONENT;

    signal A : STD_LOGIC;
    signal B : STD_LOGIC;

    signal Y : STD_LOGIC;

BEGIN
    DUT:AND_gate PORT MAP(A => A,
                          B => B,
                          Y => Y);

    PROCESS
    BEGIN
        -- 00
        A <= '0';
        B <= '0';
        wait for 10ns;
        -- 01
        A <= '0';
        B <= '1';
        wait for 10ns;
        -- 11
        A <= '1';
        B <= '0';
        wait for 10ns;
        -- 10
        A <= '1';
        B <= '1';
        wait;

    end process;
END Behavioral;
```

- Ve kapısı inputların ikisinin de 1 olduğu durumda 1 sonucunu verir , diğer durumlarda 0 verir. Simülasyon sonucumuz bunu doğruluyor.

Name	Value	0 ns	5 ns	10 ns	15 ns	20 ns	25 ns	30 ns	35 ns
A	1								
B	1								
Y	1								

2. DEĞİL KAPISI

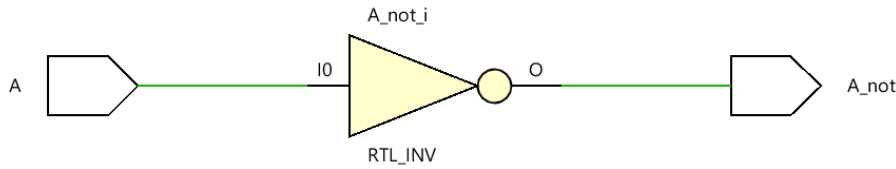
- NOT kapısı için bir modül oluşturdum ve modülüne NOT_gate ismini verdim.
- Ardından NOT_gate için A adında input A_not adında bir output tanımlayarak behavioral kısmında fonksiyonu gerçekledim.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

ENTITY NOT_gate IS
    PORT ( A : in STD_LOGIC;
           A_not : out STD_LOGIC);
end NOT_gate;

ARCHITECTURE Behavioral OF NOT_gate IS
BEGIN
    A_not <= NOT(A);
END Behavioral;
```

- NOT kapısını RTL şematığı aşağıda verilmiştir.



- Kodumuzun doğruluğunu kontrol etmek için testbench dosyası oluşturuyoruz. Yazılan kod ve simülasyon sonucu aşağıdadır.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity NOT_gate_tb is
end NOT_gate_tb;

architecture Behavioral of NOT_gate_tb is

    COMPONENT NOT_gate
    PORT ( A: in STD_LOGIC;
           A_not: out STD_LOGIC);
    END COMPONENT;

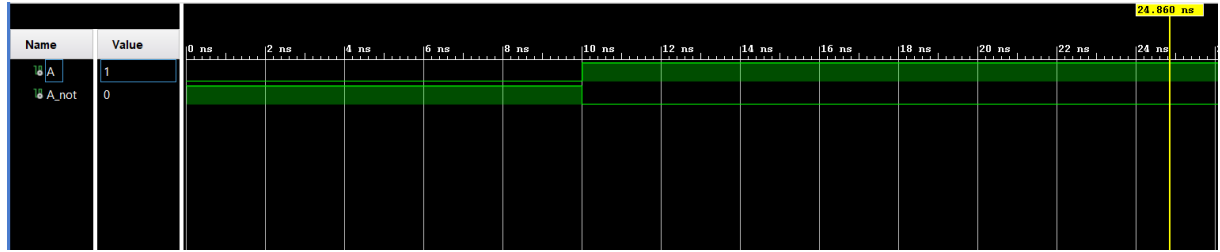
    signal A : STD_LOGIC;
    signal A_not: STD_LOGIC;

BEGIN
    DUT:NOT_gate PORT MAP( A => A,
                           A_not =>
A_not);
    PROCESS
    BEGIN
        --0
        A <= '0';
        wait for 10ns;

        --1
        A <= '1';
        wait;

    end PROCESS;
end Behavioral;
```

- NOT kapısının işlevi verilen inputun değilini almasıdır. 1 verdiğimizde çıkışta 0, 0 verdiğimizde çıkışta 1 görmemiz gerekir. Aşağıdaki simülasyon sonucu bunu doğruluyor.



3. BOOLEAN FONKSİYONUNUN GERÇEKLENMESİ

- İlgili olan fonksiyonu ilk bölümde el ile sadeleştirdik ardından kod üzerinde gerçekledik.
- OR_gate, AND_gate ve NOT_gate modülleri alt modül kullanarak aradaki bağlantıları bu modüllerle bağladık.
- Modülleri kullandığımız kod aşağıdaki gibidir.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

ENTITY NOT_gate IS
    PORT ( A : in STD_LOGIC;
          A_not : out STD_LOGIC);
end NOT_gate;
ARCHITECTURE Behavioral OF NOT_gate IS
BEGIN
    A_not <= NOT(A);
END Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Simplify is
    Port ( X : in STD_LOGIC;
          Y : in STD_LOGIC;
          Z : in STD_LOGIC;
          T : in STD_LOGIC;
          W : in STD_LOGIC;
          OUTA : out STD_LOGIC);
end Simplify;
architecture Behavioral of Simplify is
    component AND_gate is
        port (A, B : in std_logic;
              Y: out std_logic
        );
    end component;
    component NOT_gate is
        port (A : in STD_LOGIC;
              A_not : out STD_LOGIC);
    end component;
    component OR_gate is
        port ( A : in STD_LOGIC;
              B : in STD_LOGIC;
              Y : out STD_LOGIC);
    end component;

    signal x_not,y_not,z_not,t_not,w_not : std_logic;
    signal a, al, b, b1, c, c1, c2, d, d1, d2, o1, o2 :
    std_logic;
```

```
begin

NOTX: NOT_gate port map(X, x_not);
NOTY: NOT_gate port map(Y, y_not);
NOTZ: NOT_gate port map(Z, z_not);
NOTT: NOT_gate port map(T, t_not);
NOTW: NOT_gate port map(W, w_not);--not's of input
variables

anda1: AND_gate port map (y_not, z_not, al);--y'z'
anda: AND_gate port map (al, w_not,a);--y'z'w'

andb1: AND_gate port map (Y , Z, b1);--yz
andb: AND_gate port map (b1, W, b);--yzw

andc1: AND_gate port map (x_not , Y, c1);--x'y
andc2: AND_gate port map (c1, T,c2);--x'yt
andc: AND_gate port map (c2, w_not, c);--x'ytw'

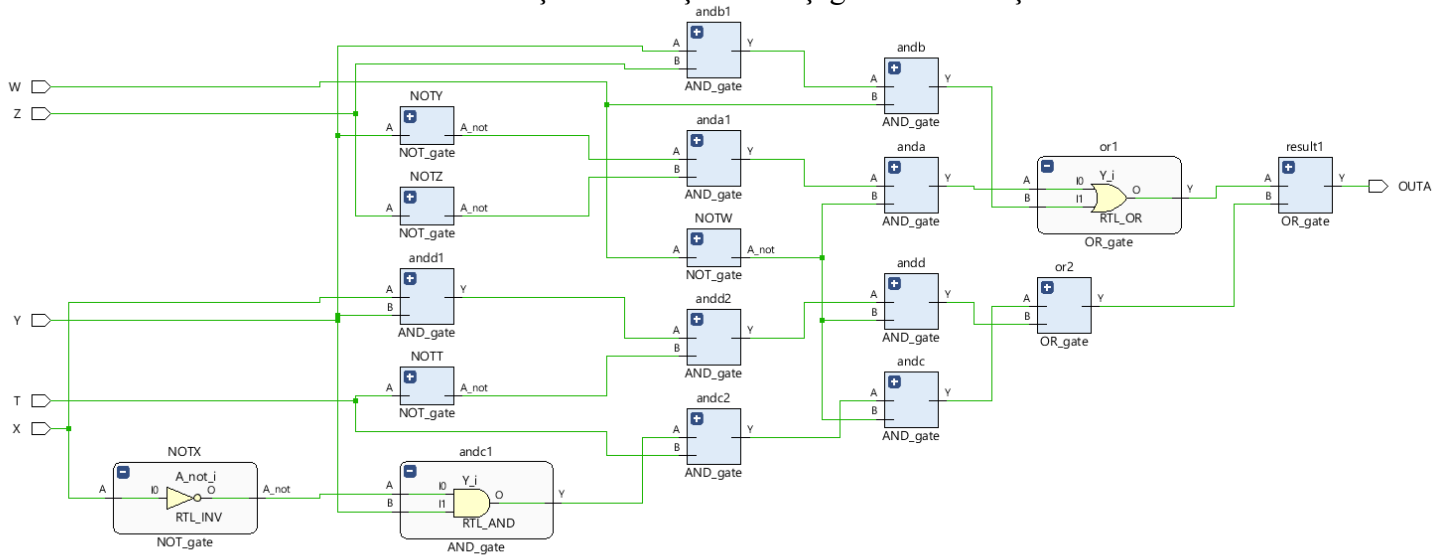
add1: AND_gate port map (X , Y, d1);--xy
add2: AND_gate port map (d1, t_not, d2);--xyt'
add: AND_gate port map (d2, w_not, d);--xyt'w'

or1: OR_gate port map (a, b, o1);--y'z'w' + yzw
or2: OR_gate port map (c, d, o2);--x'ytw' + xyt'w'

result1: OR_gate port map (o1, o2, OUTA);--y'z'w' + yzw
+ x'ytw' + xyt'w'

end Behavioral;
```


- Yukarıdaki kod sonucunda çıkan RTL şematik aşağıda belirtilmiştir.



- Kodumuzun doğruluğunu kontrol etmek için testbench dosyası oluşturuyoruz. Yazılan kod ve simülasyon sonucu aşağıdadır.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY Simplify_tb IS
END Simplify_tb;

ARCHITECTURE Behavioral OF Simplify_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT Simplify
    PORT(
        X : IN  std_logic;
        Y : IN  std_logic;
        Z : IN  std_logic;
        T : IN  std_logic;
        W : IN  std_logic;
        OUTA : OUT std_logic
    );
    END COMPONENT;

    --Inputs
    signal X : std_logic := '0';
    signal Y : std_logic := '0';
    signal Z : std_logic := '0';
    signal T : std_logic := '0';
    signal W : std_logic := '0';

    --Outputs
    signal OUTA : std_logic;

    -- No clocks detected in port list. Replace <clock> below
    with
    -- appropriate port name

BEGIN
    -- Instantiate the Unit Under Test (UUT)
    DUT:Simplify PORT MAP (
        X => X,
        Y => Y,
        Z => Z,
        T => T,
        W => W,
        OUTA => OUTA
    );
END;
```



```
-- Stimulus process
stim_proc: process
begin
    --00000--0
    X <= '0';
    Y <= '0';
    Z <= '0';
    T <= '0';
    W <= '0';
    wait for 10ns;
    --00001--1
    X <= '0';
    Y <= '0';
    Z <= '0';
    T <= '0';
    W <= '1';
    wait for 10ns;
    --00010--2
    X <= '0';
    Y <= '0';
    Z <= '0';
    T <= '1';
    W <= '0';
    wait for 10ns;
    --00011--3
    X <= '0';
    Y <= '0';
    Z <= '0';
    T <= '1';
    W <= '1';
    wait for 10ns;
    --00100--4
    X <= '0';
    Y <= '0';
    Z <= '1';
    T <= '0';
    W <= '0';
    wait for 10ns;
    --00101--5
    X <= '0';
    Y <= '0';
    Z <= '1';
    T <= '0';
    W <= '1';
    wait for 10ns;
    --00110--6
    X <= '0';
    Y <= '0';
    Z <= '1';
    T <= '1';
    W <= '0';
    wait for 10ns;
    --00111--7
    X <= '0';
    Y <= '0';
    Z <= '1';
    T <= '1';
    W <= '1';
    wait for 10ns;
    --01000--8
    X <= '0';
    Y <= '1';
    Z <= '0';
    T <= '0';
    W <= '0';
    wait for 10ns;
    --01001--9
    X <= '0';
    Y <= '1';
    Z <= '0';
    T <= '0';
    W <= '1';
    wait for 10ns;
    --01010--10
    X <= '0';
    Y <= '1';
    Z <= '0';
    T <= '1';
    W <= '0';
    wait for 10ns;
    --01011--11
    X <= '0';
    Y <= '1';
    Z <= '0';
    T <= '1';
    W <= '1';
    wait for 10ns;
    --01100--12
    X <= '0';
    Y <= '1';
    Z <= '1';
    T <= '0';
    W <= '0';
    wait for 10ns;
    --01101--13
    X <= '0';
    Y <= '1';
    Z <= '1';
    T <= '0';
    W <= '1';
    wait for 10ns;
    --01110--14
    X <= '0';
    Y <= '1';
    Z <= '1';
    T <= '1';
    W <= '0';
    wait for 10ns;
    --01111--15
    X <= '0';
    Y <= '1';
    Z <= '1';
    T <= '1';
    W <= '1';
    wait for 10ns;
    --10000--16
    X <= '1';
    Y <= '0';
    Z <= '0';
    T <= '0';
    W <= '0';
```

```
wait for 10ns;
--10001--17
X <= '1';
Y <= '0';
Z <= '0';
T <= '0';
W <= '1';
wait for 10ns;
--10010--18
X <= '1';
Y <= '0';
Z <= '0';
T <= '1';
W <= '0';
wait for 10ns;
--10011--19
X <= '1';
Y <= '0';
Z <= '0';
T <= '1';
W <= '1';
wait for 10ns;
--10100--20
X <= '1';
Y <= '0';
Z <= '1';
T <= '0';
W <= '0';
wait for 10ns;
--10101--21
X <= '1';
Y <= '0';
Z <= '1';
T <= '0';
W <= '1';
wait for 10ns;
--10110--22
X <= '1';
Y <= '0';
Z <= '1';
T <= '1';
W <= '0';
wait for 10ns;
--10111--23
X <= '1';
Y <= '0';
Z <= '1';
T <= '1';
W <= '1';
wait for 10ns;
--11000--24
X <= '1';
Y <= '1';
Z <= '0';
T <= '0';
W <= '0';
wait for 10ns;
--11001--25
X <= '1';
Y <= '1';
Z <= '0';
T <= '0';
W <= '1';
wait for 10ns;
--11010--26
X <= '1';
Y <= '1';
Z <= '0';
T <= '1';
W <= '0';
wait for 10ns;
--11011--27
X <= '1';
Y <= '1';
Z <= '0';
T <= '1';
W <= '1';
wait for 10ns;
--11100--28
X <= '1';
Y <= '1';
Z <= '1';
T <= '0';
W <= '0';
wait for 10ns;
--11101--29
X <= '1';
Y <= '1';
Z <= '1';
T <= '0';
W <= '1';
wait for 10ns;
--11110--30
X <= '1';
Y <= '1';
Z <= '1';
T <= '1';
W <= '0';
wait for 10ns;
--11111--31
X <= '1';
Y <= '1';
Z <= '1';
T <= '1';
W <= '1';
wait for 10ns;

    wait;
end process;
end;
```

- 5 adet input , x y z t w , olduğu için 32 farklı durum (2^5) oluşmuştur. Test bench kodunda görüldüğü üzere tüm durumları input olarak verip simülasyonu kontrol ettim.

$$f(x, y, z, t, w) = y'z'w' + yzw + x'ytw' + xyt'w'$$

- Yukarıda bize soruda verilen fonksiyonun en sade hali belirtilmiştir. Simülasyonun doğruluğunu kontrol etmek için belli aralıklara bakabiliriz. $yzw = 1$, $yzw=1$, $x'ytw'=1$ ve $xyt'w'=1$ olduğu durumların hepsinde OUTA değeri 1 olmalıdır. Aşağıdaki simülasyona baktığımızda bunun gerçekleştiğini görebiliriz.

