Deney 3) Full Adder

- Bu deneyde, tam toplayıcı (full adder) VHDL dili ile yazılarak **ALTERA-DE0** kartı üzerinde çalıştırılacaktır.
- VHDL temelleri ve component oluşturma öğrenilecektir.

	Family:	**Name:**
ALTERA-DE0	Cyclone III	EP3C16F484C6

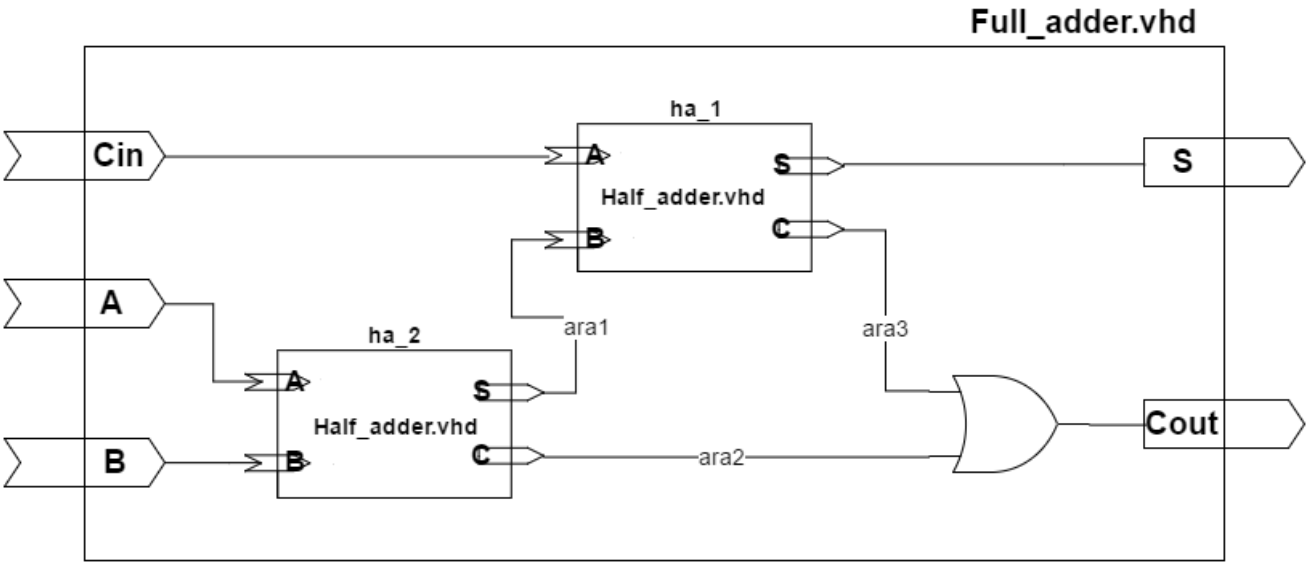
2a) Full adder

Tam toplayıcı verilen iki sayının yanı sıra eldeyi (carry in) de giriş olarak almakta ve sonuç ile elde (carry out) çıkışlarını üretmektedir. Doğruluk tablosu aşağıda verilmiştir.

A,B: 1 bitlik girişler
Cin: 1 bitlik elde girişi
S: 1 bitlik toplam sonucu
Cout: 1 bitlik elde sonucu

A	B	Cin	S	Cout
0	0	0	**0**	**0**
0	0	1	**1**	**0**
0	1	0	**1**	**0**
0	1	1	**0**	**1**
1	0	0	**1**	**0**
1	0	1	**0**	**1**
1	1	0	**0**	**1**
1	1	1	**1**	**1**

Tam toplayıcı tasarlamanın basit bir yolu da iki tane yarım toplayıcıyı birleştirmektir. İki yarım toplayıcıya dayalı olarak tasarlanan tam toplayıcı şematiği aşağıdadır.



Dip not:

*Aynı proje içinde "*Full_adder.vhd*" ve "*Half_adder.vhd*" isimli **iki ayrı kod sayfası** olduğuna dikkat ediniz!!!

*Hierarchy olarak "*Full_adder.vhd*" dosyasının **top level** olmasına dikkat ediniz!!!

*Half_adder.vhd dosyası Full_adder.vhd dosyası içinde **component olarak kullanılacağı için öncelikle architecture ile begin arasında tanımlanmalıdır.**

```
component Half_adder
port(
        A:in std_logic;
        B:in std_logic;
        S:out std_logic;
        C:out std_logic);
end component;
```

*tanımlı component aşağıdaki örnekteki gibi kullanılabilir. **Componentin içindeki unsurlar her zaman solda bağlanacak olan dış unsurlar her zaman sağdadır. Bağlama işareti "=>" 'dir.**

```
ha_1: Half_adder port map(
        A=>Cin,
        B=>ara1,
        S=>S,
        C=>ara3);
```

full_adder.vhd

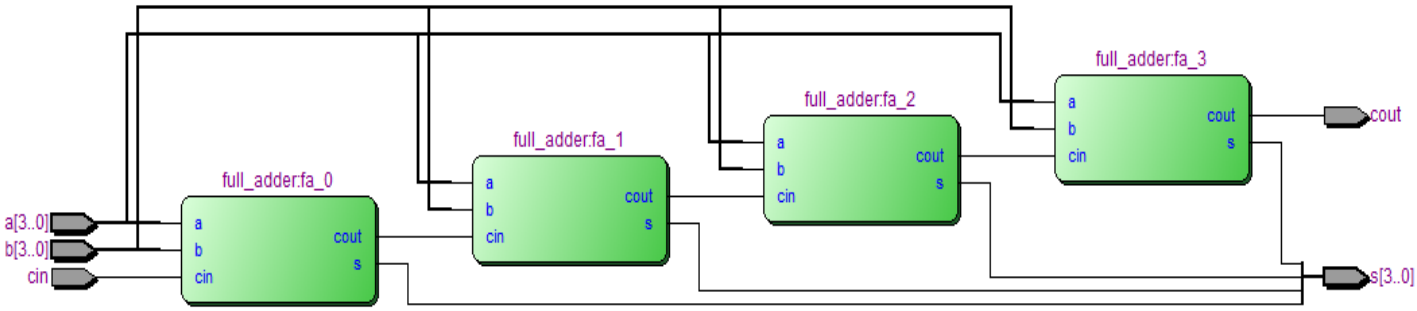
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity full_adder is
        port (a, b, cin : in std_logic;
                s, cout :out std_logic);
end full_adder;

architecture behavior of full_adder is
        component half_adder
        port (a, b : in std_logic;
                s, c :out std_logic);
        end component;
        signal ara1,ara2,ara3:std_logic;
begin
        ha1:half_adder port map(
        a=>cin,
        b=>ara1,
        c=>ara3,
        s=>s);

        ha2:half_adder port map(
        a=>a,
        b=>b,
        c=>ara2,
        s=>ara1);

        cout<=ara2 or ara3;

end behavior;
```

2b) 4bit full adder

Yapılan devre toplayıcı devresidir. Binary olarak girilen 4'er bitlik iki sayının ve elde girişinin toplamı çıkışta binary olarak göstermektedir.

a,b: 4'er bitlik girişler
cin: 1 bitlik elde girişi
s: 4 bitlik toplam sonucu
cout: 1 bitlik elde sonucu

4 bitlik tam toplayıcı tasarlamak için, dört tane 1 bitlik tam toplayıcıyı birleştirilmektedir. Tasarlanacak 4 bitlik tam toplayıcı şematiği aşağıdadır.



Dip not:

*Aynı proje içinde "full_adder_4bit.vhd", "*Full_adder.vhd*" ve "*Half_adder.vhd*" isimli **üç ayrı kod sayfası** olduğuna dikkat ediniz!!!

*Hierarchy olarak "full_adder_4bit.vhd" dosyasının **top level** olmasına dikkat ediniz!!!

*Bir bitten daha fazla sayıda olan girişler için vektör tanımlayınız.
(örn; a:in std_logic_vector(3 downto 0);)

*Bir vektörün elemanlarını ayrı ayrı seçebilirsiniz. (örn: a(0),a(1),a(1:0),a(2:1))

*"Full_adder.vhd" dosyası "full_adder_4bit.vhd" dosyası içinde **component olarak kullanılacağı için öncelikle architecture ile begin arasında tanımlanmalıdır.**

*Componentler arası ara bağlantılar için signal tanımlayınız.

*Tasarladığınız üst seviye kodun alt componentlerinin bağlantısını "RTL viewer" ile göre bilirsiniz. RTL viewer' Analysis&synthesis'in altındaki Netlistviewers'ın altındadır. Yanda gösterilmiştir.