

KOCAELİ ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ



ALU

MÜHENDİSLİK TASARIMI -2

Final Raporu

Şüheda ARAS

170207050

Bölümü: Elektronik ve Haberleşme Mühendisliği

Danışman: Prof.Dr. Ali TANGEL

KOCAELİ, 2020

İÇİNDEKİLER

1.GİRİŞ	5
1.1 FPGA Nedir?	5
1.2 Neden FPGA?	5
2.ALU (Arithmetic Logic Unit - Aritmetik Mantık Birimi).....	6
3.RTL ŞEMATİĞİ	8
4.KOD	8
4.1 Aritmetik Ünitenin Kodu:	9
4.2 Lojik Ünitenin Kodu:	9
4.3 Mux:	10
4.4 ALU (Ana Kod):	10
5.SİMÜLASYON SONUÇLARI	12
5.1 Aritmetik	12
5.2 Lojik	13
6.COMPONENTSİZ ALU TASARIMI.....	14
6.1 Kod:	14
6.2 RTL Şematığı:	15
7.SONUÇ.....	15
KAYNAKÇA	17

ŞEKİLLER DİZİNİ

ŞEKİL 2. 1: ÜÇ ADET COMPONENT BLOĞU İLE OLUŞTURULAN ALU BLOK DİYAGRAMI	6
ŞEKİL 2. 2: ALU İŞLEV TABLOSU	7
ŞEKİL 3. 1: RTL ŞEMATİĞİ.....	8
ŞEKİL 4. 1: ARİTMETİK ÜNİTE KODU.....	9
ŞEKİL 4. 2: LOJİK ÜNİTE KODU	9
ŞEKİL 4. 3: MUX KODU	10
ŞEKİL 4. 4: ANA KOD	11
ŞEKİL 5. 1: SEL= “7”DURUMUNDA SİMÜLASYON SONUCU	12
ŞEKİL 5. 2: SEL = “1”DURUMUNDA SİMÜLASYON SONUCU	12
ŞEKİL 5. 3: SEL= “10”DURUMUNDA SİMÜLASYON SONUCU	13
ŞEKİL 5. 4: SEL= “11”DURUMUNDA SİMÜLASYON SONUCU	13
ŞEKİL 6. 1: ALU KODU	14
ŞEKİL 6. 2: RTL ŞEMATİĞİ.....	15

ÖZET

Yaptığımız tasarımın aritmetik ünite de toplama ve çıkarma ,lojik ünite de ise OR, AND, XOR gibi mantıksal işlemlerini yapabilmesi hedeflenmektedir. Gerçekleştirilmesi istenilen özellikler tabloda verilmiştir. Tabloya uygun kodlama, Vivado programında VHDL dili kullanılarak yapılmıştır. RTL şematiği başarılı bir şekilde oluşturulmuş olup simülasyonda gözlemlenmiştir.

1.GİRİŞ

1.1 FPGA Nedir?

FPGA (Field Programmable Gate Array), basitçe ifade etmek gerekirse içerisinde programlanabilir pek çok mantık devresi barındıran bir tümleşik devredir. FPGA'lar, sayısal (digital) olarak yapılabilen tüm tasarımların tek bir entegre üzerinde gerçekleştirilebilmesine olanak sağlarlar. Aynı FPGA ile çok basit bir lojik devre tasarımından, karmaşık CPU-MCU tasarımına kadar çok geniş bir yelpazede çalışma imkânı sunulmaktadır.

FPGA ile donanım tasarımı yapabilmek amacıyla geliştirilmiş birçok dil mevcuttur. Bu dillerden yaygın olarak kullanılan diller VHDL ve Verilog'dur. Biz bu projede VHDL dilini kullanacağız.

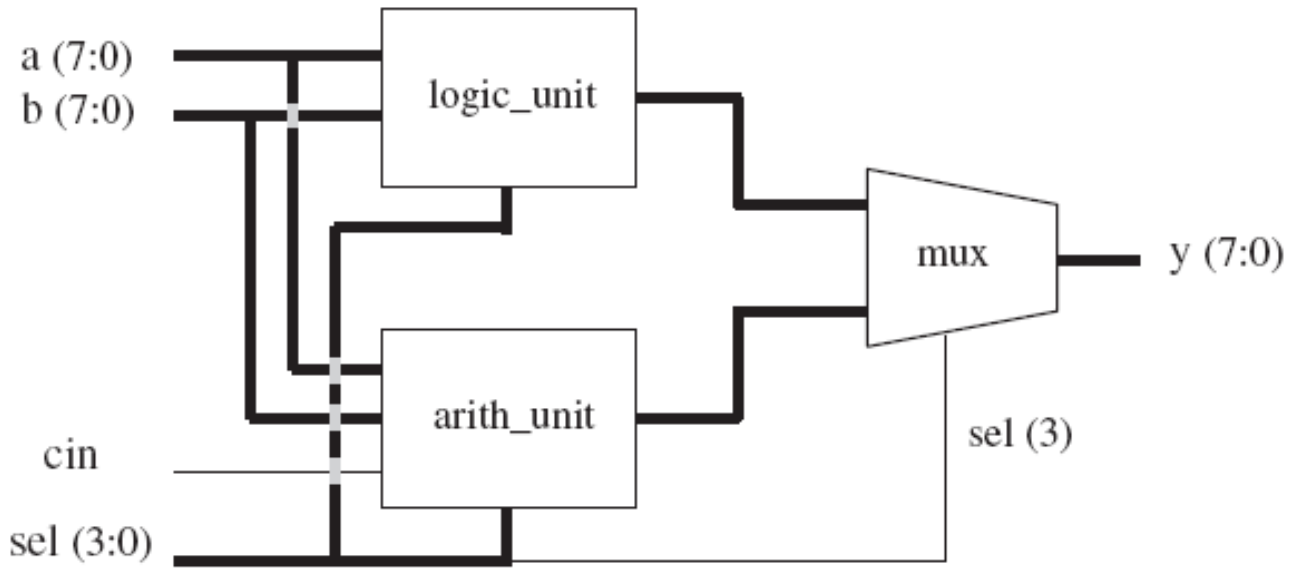
1.2 Neden FPGA?

FPGA'lar yüksek oranda paralel veri işleme olanağı sağlamaktadır. Geleneksel bilgisayar mimarisinden farklı olarak, birbiri ile neden-sonuç ilişkisi bulunmayan birimler eş-zamanlı çalışabilmektedir. Bu çalışma şekli, tüm sistemin geleneksel bilgisayar mimarisine göre daha düşük frekanslarda çalışabilmesine müsaade etmektedir. Özellikle görüntü işleme, sinyal işleme gibi yoğun bilgi girişi olan ve elde edilen verinin kısa sürede işlenmesi gibi durumlar için neredeyse rakipsiz bir seçenektir. FPGA'ların bir diğer avantajı ise düşük güç tüketmesidir. Özellikle robotik, uzay ve havacılık gibi enerji kaynaklarının kısıtlı olduğu, neredeyse gerçek zamanlı işlenmesi gereken pek çok bilginin olduğu ortamlarda eşsiz bir seçenek olmaktadır. FPGA'lar ayrıca pek çok digital entegrenin yapacağı görevleri tek başına yapabilecekleri şekilde görevlendirilebildikleri için, devre kartlarının daha küçük bir yapıda tasarlanabilmelerine olanak sağlamaktadır. Bu, özellikle mobil robotik gibi fiziksel alan kısıtlaması olan tasarımlarda, çok önemli bir fayda sağlamaktadır. FPGA; sayısal işaret işleme, radar haberleşme sistemi, uzay, savunma sistemleri, ASIC, medikal resimleme, robotik, ses tanıma, şifreleme ve kod çözme gibi bir çok alanda kullanılmaktadır. [1]

2.ALU (Arithmetic Logic Unit - Aritmetik Mantık Birimi)

Şekil 5.10'da bir ALU blok diyagramı verilmiştir. ALU, işlemcinin mantıksal ve matematiksel işlemleri gerçekleştirebilmesini sağlayan bölümdür. Çıkış, sel ucunun en değerli biti (3. bit) tarafından seçilirken (aritmetik ya da lojik), özel işlemler ise sel'in diğer 3 biti tarafından belirlenmektedir. İlgili kod parçası aşağıdaki gibidir. Tamamı eş zamanlı yapılardan oluşan bir kod olmasının yanı sıra, aritmetik ve lojik işlemlerin her ikisini de gerçekleyebilmesi için aynı veri türü kullanımını içermektedir. Bunun için de ieee kütüphanesinin içinde bulunan std_logic_unsigned paketi tanımlanmıştır.

Aşağıdaki kodda ana kodun (alu.vhd) yanı sıra yukarıda bahsettiğimiz üç COMPONENT bloğu da bulunmaktadır (logic_unit, arith_unit ve mux). Görülebileceği gibi COMPONENT blokları ana kod içerisinde bildirilmişlerdir.



Şekil 2. 1: Üç adet COMPONENT bloğu ile oluşturulan ALU blok diyagramı

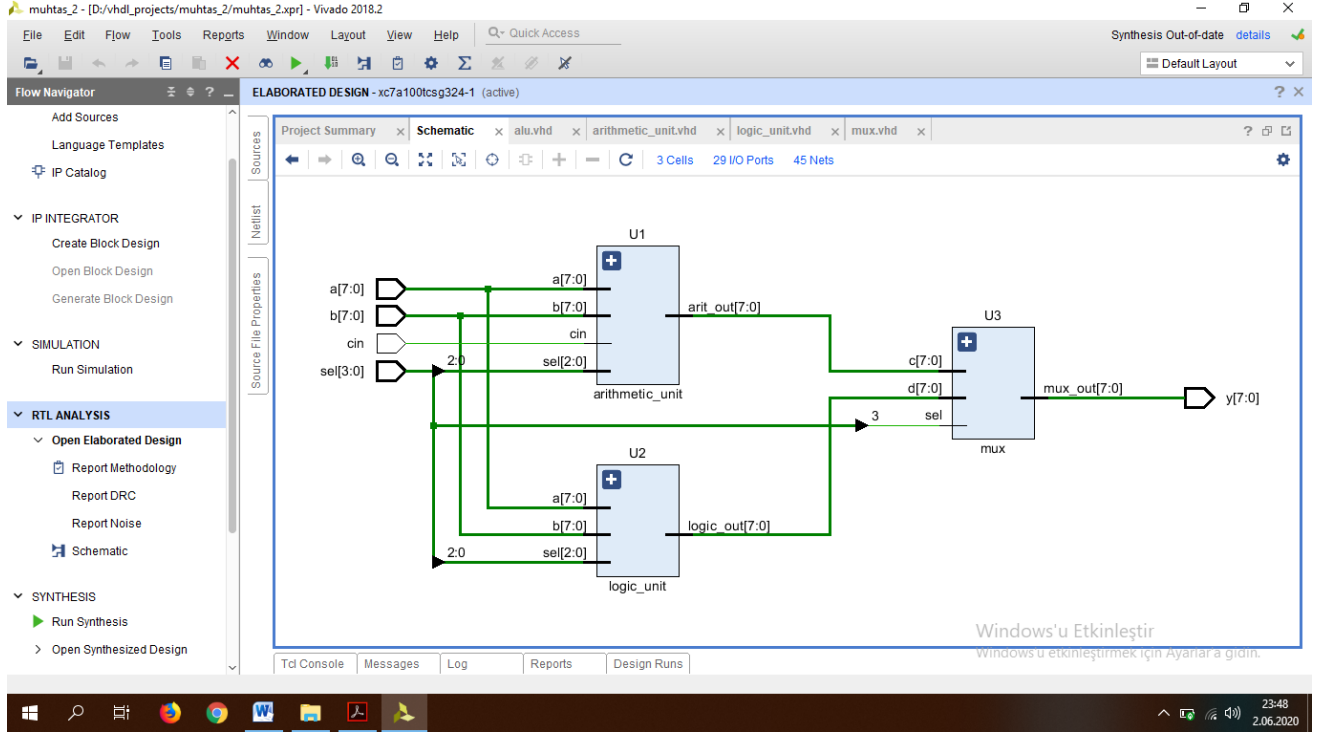
Verilen ALU'nun yaptığı işlemler tablo olarak şekil 2.2'de verilmiştir.

sel	işlem	fonksiyon	ünite
0000	$y \leq a$	a'yı ata	aritmetik
0001	$y \leq a+1$	a'yı arttır ve ata	
0010	$y \leq a-1$	a'yı azalt ve ata	
0011	$y \leq b$	b'yi ata	
0100	$y \leq b+1$	b'yi arttır ve ata	
0101	$y \leq b-1$	b'yi azalt ve ata	
0110	$y \leq a+b$	a ve b'nin toplamını ata	
0111	$y \leq a+b+cin$	a ve b' nin Cin ile toplamını ata	
1000	$y \leq \text{NOT } a$	a'nın tersini al ve ata	lojik
1001	$y \leq \text{NOT } b$	b'nin tersini al ve ata	
1010	$y \leq a \text{ AND } b$	AND	
1011	$y \leq a \text{ OR } b$	OR	
1100	$y \leq a \text{ NAND } b$	NAND	
1101	$y \leq a \text{ NOR } b$	NOR	
1110	$y \leq a \text{ XOR } b$	XOR	
1111	$y \leq a \text{ XNOR } b$	XNOR	

Şekil 2. 2: ALU işlev tablosu

3.RTL ŞEMATİĞİ

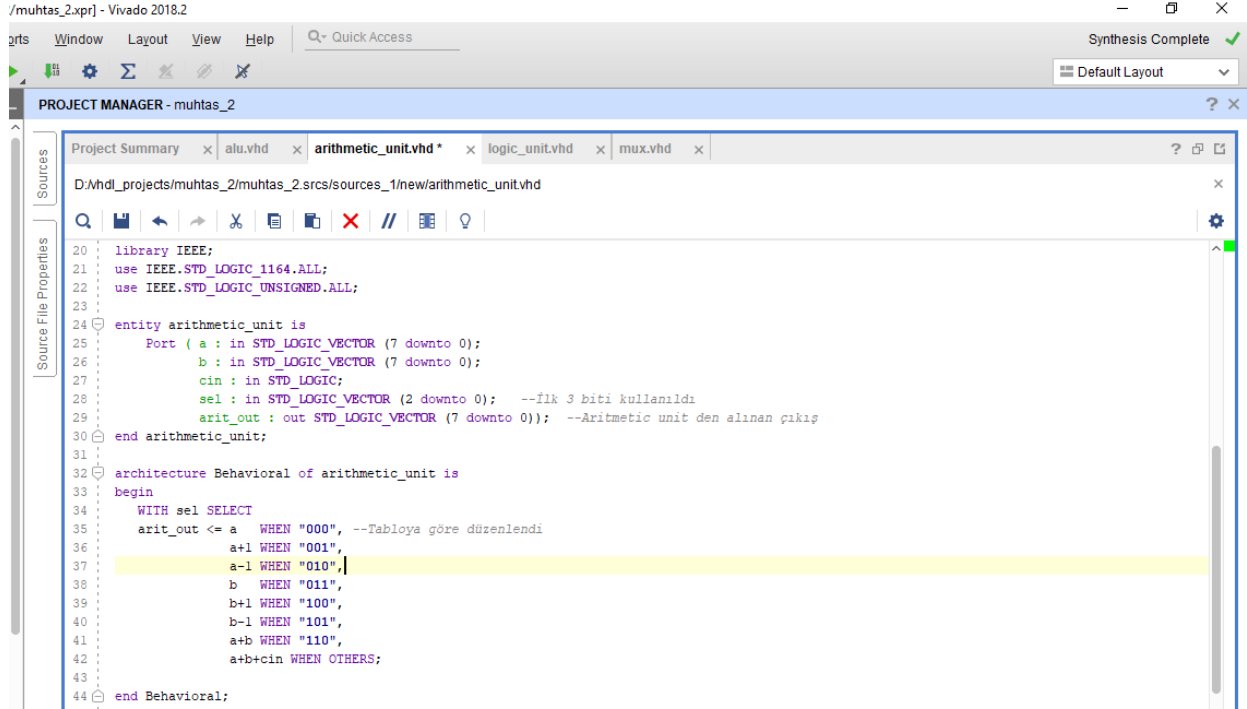
Tasarlanan ALU'nun RTL şematiği Şekil 3.1 de gösterilmiştir.



Şekil 3. 1: RTL Şematiği

4.KOD

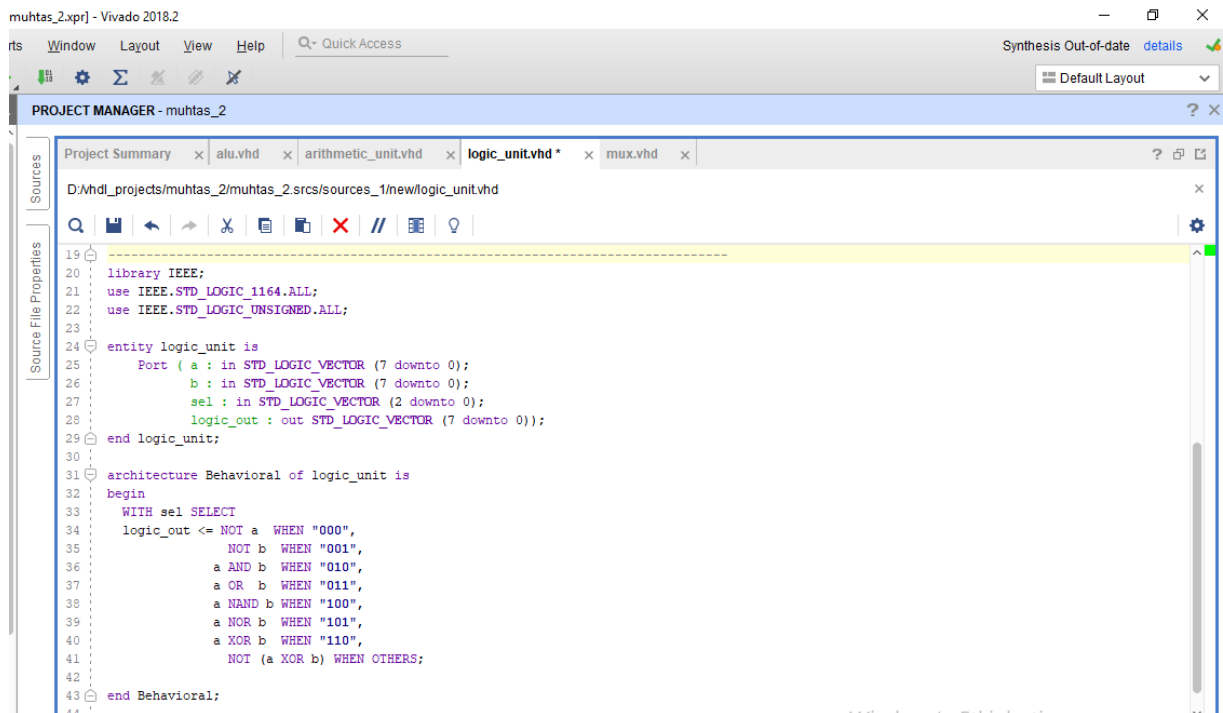
4.1 Aritmetik Ünitenin Kodu:



```
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_UNSIGNED.ALL;
23
24 entity arithmetic_unit is
25     Port ( a : in STD_LOGIC_VECTOR (7 downto 0);
26           b : in STD_LOGIC_VECTOR (7 downto 0);
27           cin : in STD_LOGIC;
28           sel : in STD_LOGIC_VECTOR (2 downto 0); --İlk 3 biti kullanıldı
29           arit_out : out STD_LOGIC_VECTOR (7 downto 0)); --Aritmetik unit den alınan çıkış
30 end arithmetic_unit;
31
32 architecture Behavioral of arithmetic_unit is
33 begin
34     WITH sel SELECT
35     arit_out <= a WHEN "000", --Tabloya göre düzenlendi
36                a+1 WHEN "001",
37                a-1 WHEN "010",
38                b WHEN "011",
39                b+1 WHEN "100",
40                b-1 WHEN "101",
41                a+b WHEN "110",
42                a+b+cin WHEN OTHERS;
43
44 end Behavioral;
```

Şekil 4. 1: Aritmetik Ünite Kodu

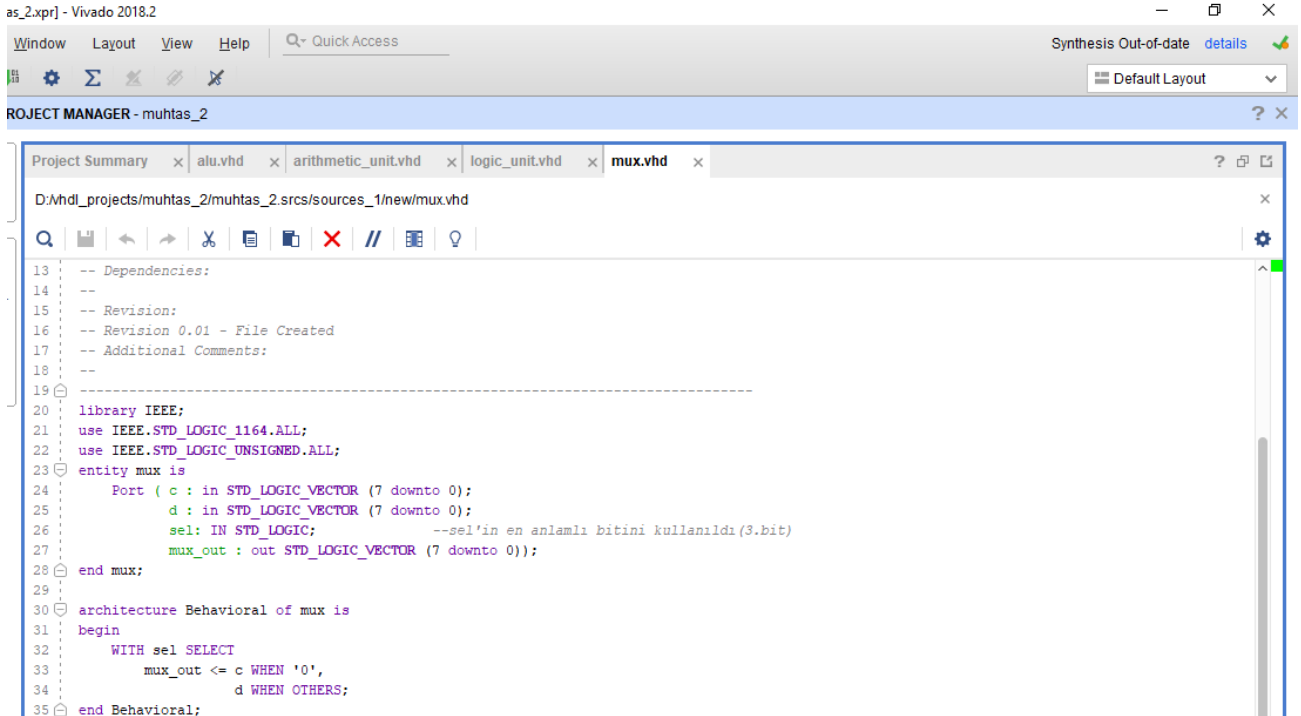
4.2 Lojik Ünitenin Kodu:



```
19
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_UNSIGNED.ALL;
23
24 entity logic_unit is
25     Port ( a : in STD_LOGIC_VECTOR (7 downto 0);
26           b : in STD_LOGIC_VECTOR (7 downto 0);
27           sel : in STD_LOGIC_VECTOR (2 downto 0);
28           logic_out : out STD_LOGIC_VECTOR (7 downto 0));
29 end logic_unit;
30
31 architecture Behavioral of logic_unit is
32 begin
33     WITH sel SELECT
34     logic_out <= NOT a WHEN "000",
35                 NOT b WHEN "001",
36                 a AND b WHEN "010",
37                 a OR b WHEN "011",
38                 a NAND b WHEN "100",
39                 a NOR b WHEN "101",
40                 a XOR b WHEN "110",
41                 NOT (a XOR b) WHEN OTHERS;
42
43 end Behavioral;
```

Şekil 4. 2: Lojik Ünite Kodu

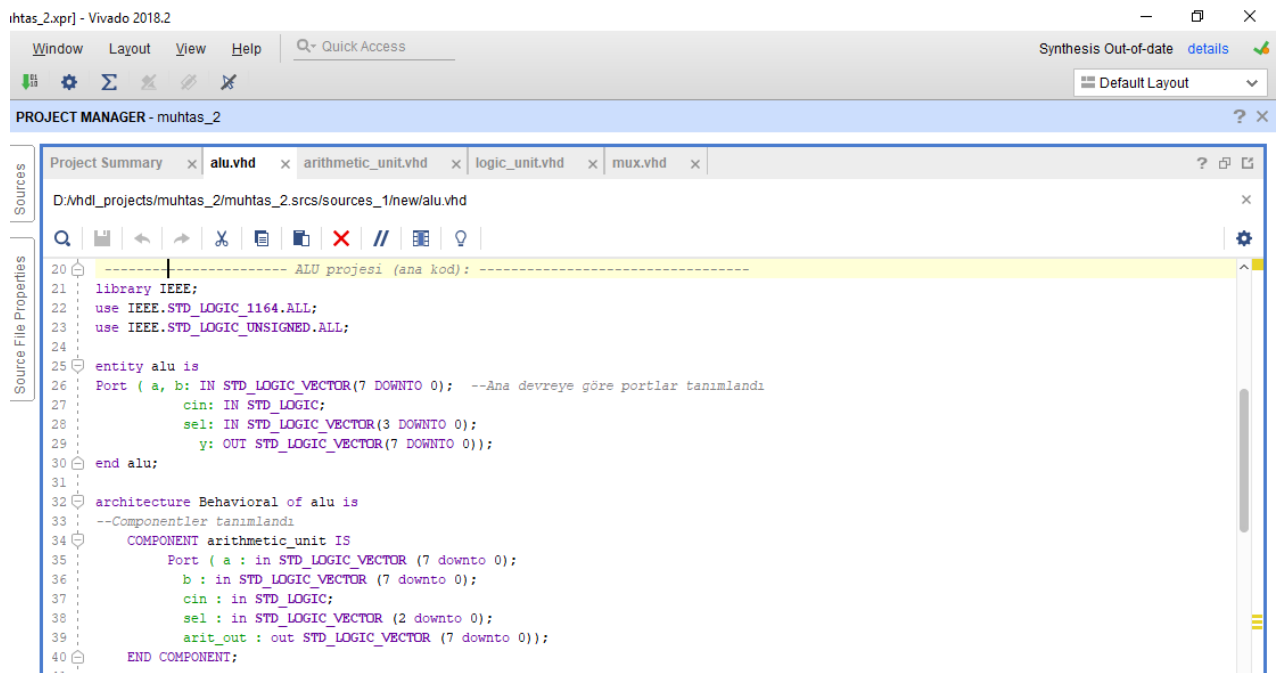
4.3 Mux:



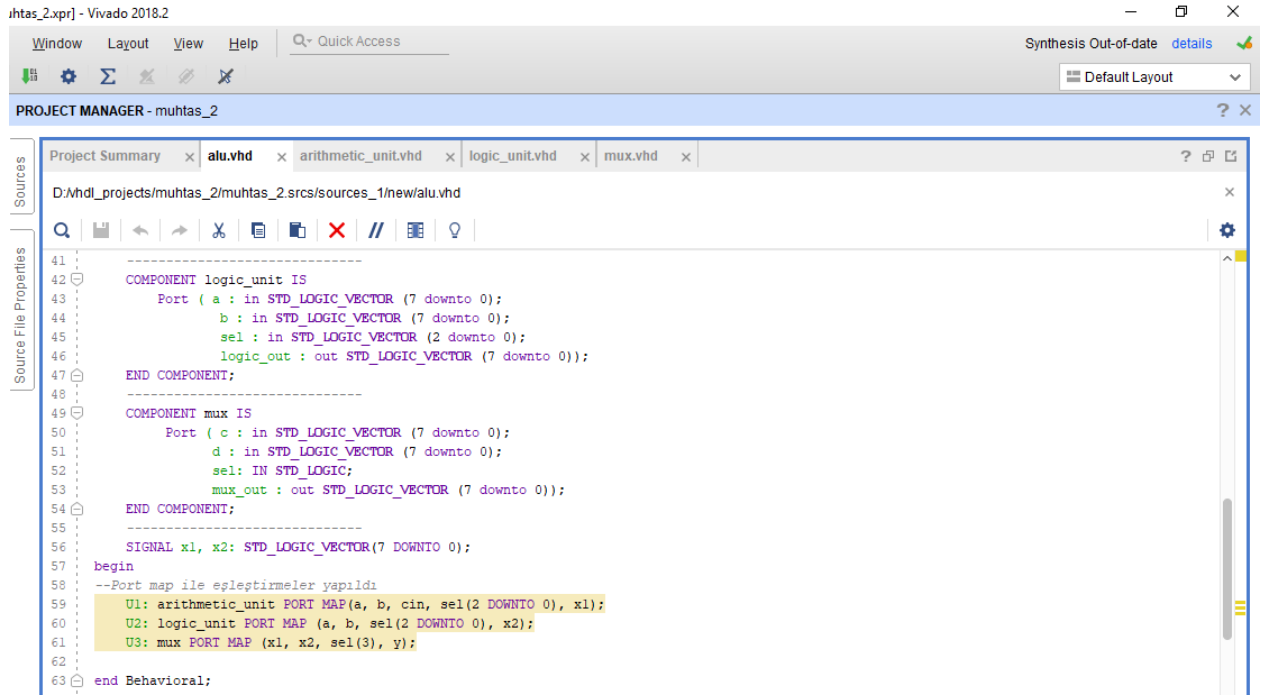
```
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_UNSIGNED.ALL;
23 entity mux is
24     Port ( c : in STD_LOGIC_VECTOR (7 downto 0);
25           d : in STD_LOGIC_VECTOR (7 downto 0);
26           sel: IN STD_LOGIC;           --sel'in en anlamlı bitini kullanıldı (3.bit)
27           mux_out : out STD_LOGIC_VECTOR (7 downto 0));
28 end mux;
29
30 architecture Behavioral of mux is
31 begin
32     WITH sel SELECT
33         mux_out <= c WHEN '0',
34                 d WHEN OTHERS;
35 end Behavioral;
```

Şekil 4. 3: Mux Kodu

4.4 ALU (Ana Kod):



```
20 ----- ALU projesi (ana kod): -----
21 library IEEE;
22 use IEEE.STD_LOGIC_1164.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 entity alu is
26     Port ( a, b: IN STD_LOGIC_VECTOR(7 DOWNTO 0); --Ana devreye göre portlar tanımlandı
27           cin: IN STD_LOGIC;
28           sel: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
29           y: OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
30 end alu;
31
32 architecture Behavioral of alu is
33     --Componentler tanımlandı
34     COMPONENT arithmetic_unit IS
35         Port ( a : in STD_LOGIC_VECTOR (7 downto 0);
36               b : in STD_LOGIC_VECTOR (7 downto 0);
37               cin : in STD_LOGIC;
38               sel : in STD_LOGIC_VECTOR (2 downto 0);
39               arit_out : out STD_LOGIC_VECTOR (7 downto 0));
40     END COMPONENT;
```



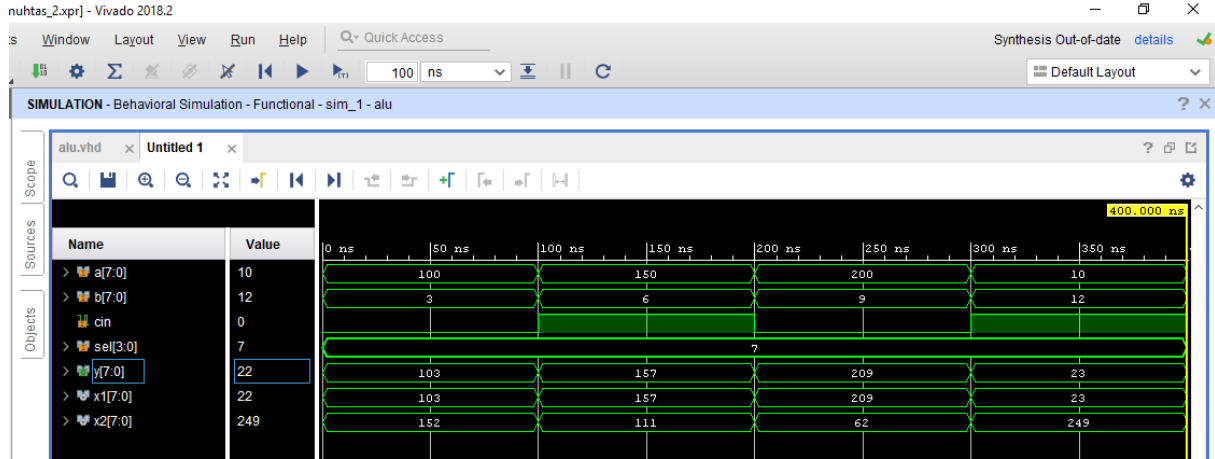
Şekil 4. 4: Ana Kod

Şekilde görüldüğü gibi ana kodda üç uyarı verildi fakat sentez başarılı bir şekilde tamamlanmış olup simülasyonda herhangi bir problem gözlenmemiştir.

5.SİMÜLASYON SONUÇLARI

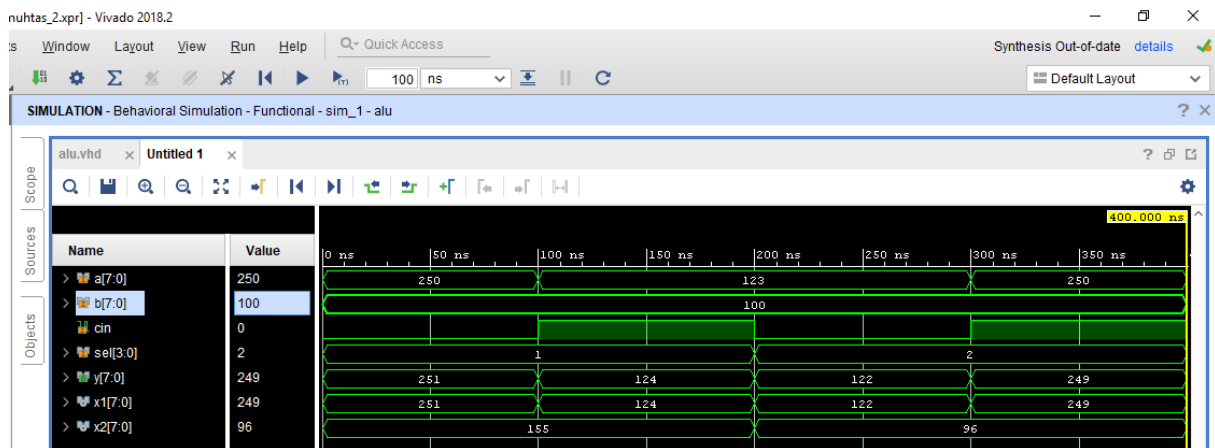
5.1 Aritmetik

sel= “7”durumunda yani sel= “0111” olduğunda $y = a + b + cin$ ’e eşittir. Şekil 5.1’de bu durumu simülasyonda gerçekleştirdik.



Şekil 5. 1: sel= “7”durumunda Simülasyon Sonucu

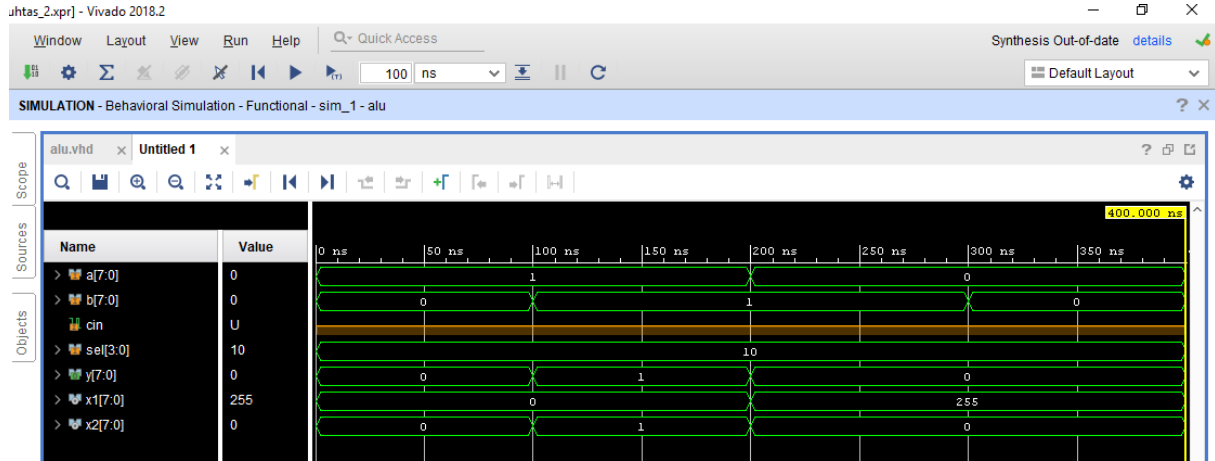
sel= “1”durumunda yani sel= “0001” olduğunda $y = a + 1$ ’e eşittir. İlk 200ns’de bu durum gözlemlendi. sel= “2”durumunda yani sel= “0010” olduğunda $y = a - 1$ ’e eşittir. 200ns-400ns arasında ise bu durum Şekil 5.2’de gözlemlendi.



Şekil 5. 2: sel= “1”durumunda Simülasyon Sonucu

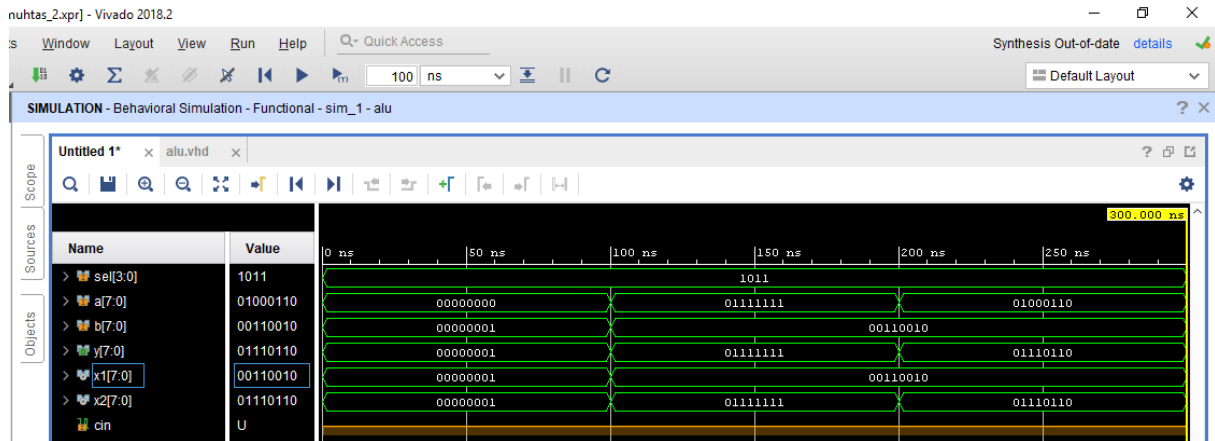
5.2 Lojik

sel= “10” durumunda yani sel= “1010” olduğunda $y = a \text{ AND } b$ ’ye eşittir. Şekil 5.3’de bu durumu simülasyonda gerçekleştirdik.



Şekil 5. 3: sel= “10”durumunda Simülasyon Sonucu

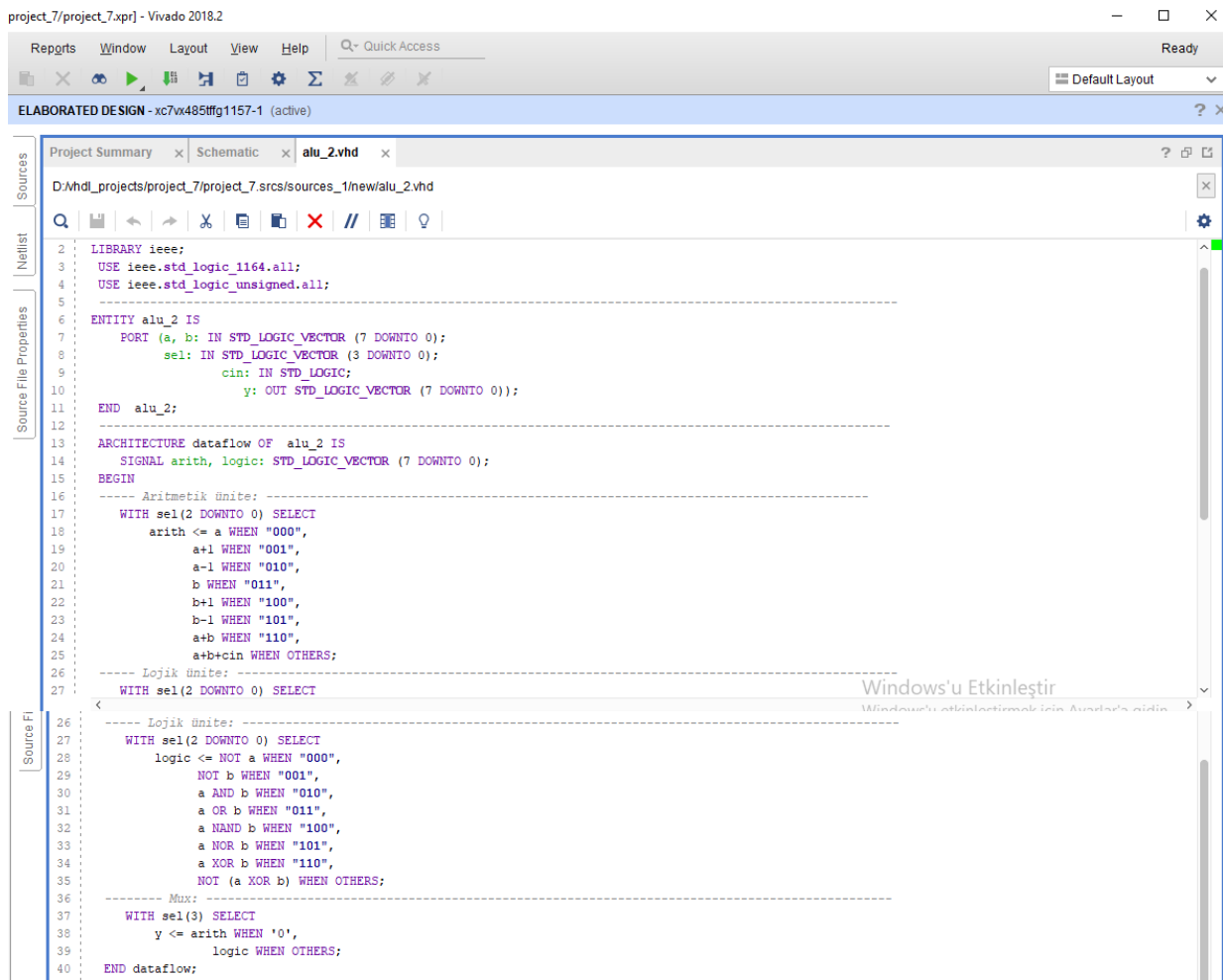
sel= “11” durumunda yani sel= “1011” olduğunda $y = a \text{ OR } b$ ’ye eşittir. Şekil 5.4 de bu durumu simülasyonda gerçekleştirdik.



Şekil 5. 4: sel= “11”durumunda Simülasyon Sonucu

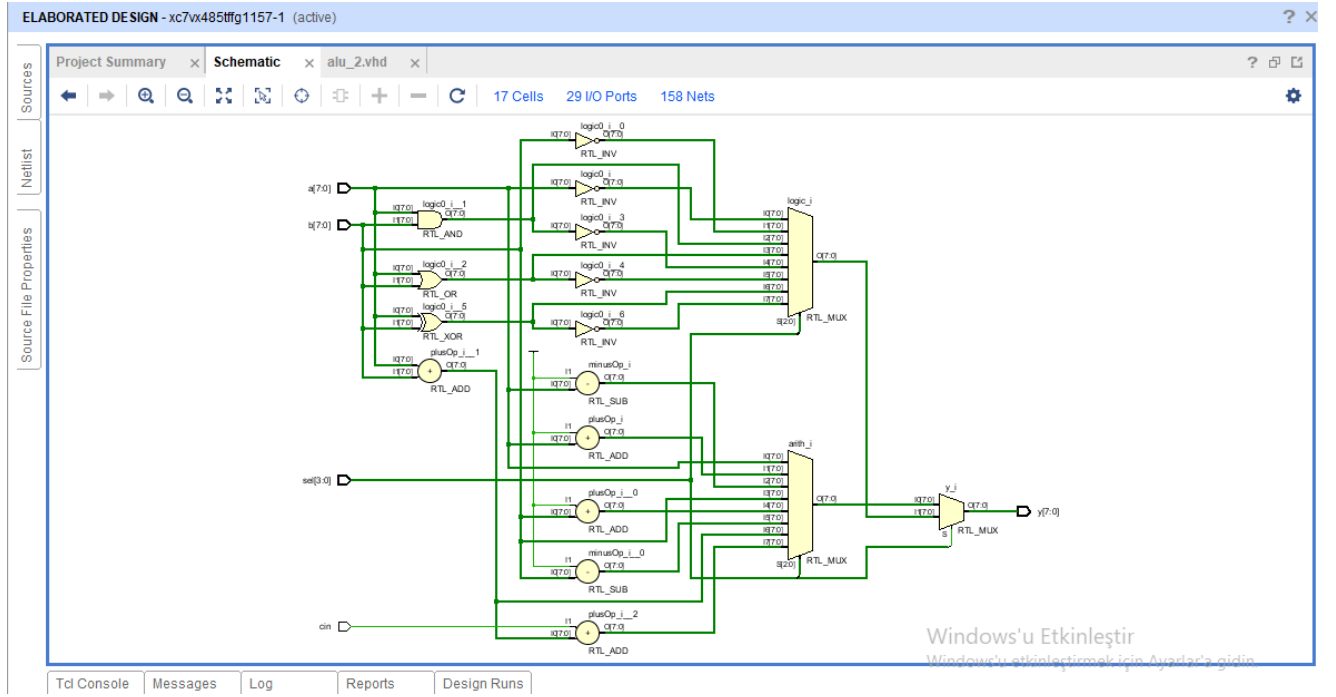
6.COMPONENTSİZ ALU TASARIMI

6.1 Kod:



Şekil 6. 1: ALU Kodu

6.2 RTL Şematiği:



Şekil 6. 2: RTL Şematiği

7.SONUÇ

Sonuç olarak yaptığımız tasarımın amacı aritmetik ünite de toplama ve çıkarma, lojik ünite de ise OR, AND, XOR gibi mantıksal işlemleri yapabilmesidir. Gerçekleştirilmesi istenilen özellikler tabloda verilmiştir. Bunun için Vivado programında VHDL dili kullanılarak iki farklı kodlama yapılmıştır. Biri component kullanılarak gerçekleştirilmiştir ve buna bağlı olarak RTL şematikleri de farklıdır. Simülasyonda aynı sonuçlar gözlemlenmiştir.

KAYNAKÇA

[1] Çavuşlu M.A. (2015). *VHDL ile sayısal tasarım ve FPGA uygulamaları* (1.Baskı).
İstanbul: İnkılap Kitabevi

