



TOBB Ekonomi ve Teknoloji Üniversitesi
Elektrik – Elektronik Mühendisliği Bölümü
ELE 422 – CMOS VLSI Tasarımı
Proje Raporu

Adı Soyadı	Seyyid Hikmet Çelik
Numara	181201047
Tarih	01/09/2022

1 Giriş

Proje olarak 8bit x 8bit çarpıcı devresi tasarlanmıştır. Ön raporda da belirtildiği gibi gecikme ve güç performansı açısından iyi olduğu araştırılarak görülen [1][5][6] Wallace Tree ve Hybrid (Booth+Wallace) algoritmaları kullanan çarpıcı devrelerinin şematikleri çizilmiş (önce kapilar, sonra alt modüller, daha sonra çarpıcı devresi), devrenin çıkışları birim eviriciye benzetilerek dc analizle transistör genişlikleri ayarlanmış, transient analizlerle gecikme süreleri karşılaştırılmış ve sonuç olarak çizilmesi daha mantıklı olan Wallace Tree çarpıcı devresi seçilmiştir.

Devreler 8 bitlik işaretli (signed) iki sayıyı çarpabilmektedir (8 bitlik işaretli sayı aralığı -128'den +127'ye) ve 16 bitlik işaretli (signed) (-16256 (-128 x +127) 'dan +16384 (-128 x -128) 'e) çıkış vermektedir. (Normalde 16 bit işaretli sayılar -32768'den +32767'ye kadardır fakat çarpımların alacağı maksimum ve minimum değerler bunu sınırlıyor.)

Şematikler <https://github.com/pareddy113/Design-of-various-multiplier-Array-Booth-Wallace> adresinde bulunan kapı seviyesinde yazılmış verilog kodlarından tek tek elle şematiğe dökülmüştür. (Daha öncesinde şematiğe geçmeden de verilog kodları testbenchler ile doğrulanmıştır.) Daha sonrasında ise layoutu çizilmeye karar verilen Wallace Tree çarpıcı devresinin de layoutu çizilerek DRC ve LVS kontrollerinden de geçmiştir. Layoutlar oldukça sıkıştırılarak gerekirse metal2, metal3 de kullanılarak alandan tasarruf edilmeye çalışılmıştır.

2 Özeti

Kullanılan Çarpıcı Algoritması: Wallace Tree

Kullanılan Teknoloji: 600nm (ami06)

Girişler: 8 bit işaretli x sayısı ve 8 bit işaretli y sayısı

Cıkışlar: 16 bit işaretli p sayısı

Transistör genişlikleri: pmos 2.4um, nmos 1.5um

(nand kapısı hariç, nand kapısında tam tersi pmos 1.5um iken nmos 2.4um)

Transistör sayısı: 1344 pmos, 1344 nmos, toplam 2688

Layout Dikey ve Yatay Uzunlukları: 732.9um dikey, 305.7um yatay

Layout Alanı: 224047.53um²

DRC: GEÇTİ

LVS: GEÇTİ

Verilog Testleri: GEÇTİ

Transient ve DC Analizler: GEÇTİ

DC analizde, parazitik kapasitanslar eklenmiş halde, kritik (en uzun) yola sahip ve birim eviriciye benzetilmiş p<15> çıkışının (diğer çıkışlar da benzer) orta nokta voltajları: 2.45V - 2.5V

Transient analizde, parazitik kapasitanslar eklenmiş halde, yükselme ve düşme süreleri: 0.1408us yükselme, 0.1347us düşme

3 Wallace Tree Çarpıcı Devresi

Wallace Tree çarpıcı devresi 8 bitlik iki işaretli sayıyı çarparak 16 bitlik işaretli sayı çıkışını verir.

Wallace Tree çarpıcı devresi tasarımda 8 And Array, 6 Adder Array, 8 Full Adder modülü ve 8 xor, 2 not kapısı kullanılmıştır.

Bölüm 4'te kapı ve alt modüllerin tasarımını daha detaylı açıklanmıştır.

Başlangıçta Hybrid çarpıcı devresi denenmiş fakat daha sonra vazgeçilmiştir. Hybrid çarpıcı hem yükselseme hem de düşme zamanlarında 0.2ns kadar daha hızlıdır fakat layoutunu elle çizmek çok zor olacağından aradaki çok çok küçük bu zaman farkı göz ardı edilerek Wallace Tree algoritmasından devam edilmiştir. Bölüm 5'te daha detaylı anlatılmaktadır.

3.1 Wallace Tree Çarpıcı Devresi Verilog Kodu

Devre aşağıdaki linkte bulunan verilog kodundan yola çıkılarak, aşağıda kod parçasında görüldüğü gibi daha modülerize hale getirilerek ve transistör ve alandan tasarruf edebilmek adına, kullanılan kapı sayısı azaltılarak (örneğin xor+not yerine direkt xnor kapısı tasarlanıp kullanılması daha fazla efor gerektiriyor ama 14 transistör yerine 12 transistör kullanılmasını sağlıyor), gereksiz işlemler çıkarılarak elle şematiğe dökülmüştür. Şematiğe geçmeden önce verilog kodu Vivado'da test edilmiştir.

[https://github.com/pareddy113/Design-of-various-multiplier-Array-Booth-Wallace
/blob/master/Wallace%20Tree%20Multiplier/Wallace%20Tree%20multiplier.v](https://github.com/pareddy113/Design-of-various-multiplier-Array-Booth-Wallace/blob/master/Wallace%20Tree%20Multiplier/Wallace%20Tree%20multiplier.v)

```
// 8bit x 8bit Wallace Tree Multiplier

`timescale 1ns/1ps
module wallace(x,y,p);
input [7:0] x,y;
output [15:0] p;

wire [6:0] ip0,ip1,ip2,ip3,ip4,ip5,ip6,ip7;
wire [7:0] si,iip;
wire [6:0] s1,s2,s3,s4,s5,s6;
wire [7:0] c1,c2,c3,c4,c5,c6,c7;

// first and array
and_array I0(y[0], x, {ip0[5:0],p[0]}, si[0]);
not I1(ip0[6],si[0]);

// second
and_array I2(y[1], x, ip1, si[1]);

// third
and_array I3(y[2], x, ip2, si[2]);

// fourth
and_array I4(y[3], x, ip3, si[3]);
```

```

// fifth
and_array I5(y[4], x, ip4, si[4]);

// sixth
and_array I6(y[5], x, ip5, si[5]);

// seventh
and_array I7(y[6], x, ip6, si[6]);

// eight
wire niip7;
and_array I8(y[7], x, iip[6:0], niip7);
not I9(iip[7], niip7);

xor I10(ip7[0],y[7],iip[0]);
xor I11(ip7[1],y[7],iip[1]);
xor I12(ip7[2],y[7],iip[2]);
xor I13(ip7[3],y[7],iip[3]);
xor I14(ip7[4],y[7],iip[4]);
xor I15(ip7[5],y[7],iip[5]);
xor I16(ip7[6],y[7],iip[6]);
xnor I17(si[7],y[7],iip[7]);

// first adder array
adder_array I18(si[0], ip0, {si[1],ip1}, ip2, s1, c1, p[1]);

// second
adder_array I19(si[2], s1, c1, ip3, s2, c2, p[2]);

// third
adder_array I20(si[3], s2, c2, ip4, s3, c3, p[3]);

// fourth
adder_array I21(si[4], s3, c3, ip5, s4, c4, p[4]);

// fifth
adder_array I22(si[5], s4, c4, ip6, s5, c5, p[5]);

// sixth
adder_array I23(si[6], s5, c5, ip7, s6, c6, p[6]);

// seventh
FA I24(s6[0],c6[0],y[7],c7[0],p[7]);
FA I25(s6[1],c6[1],c7[0],c7[1],p[8]);
FA I26(s6[2],c6[2],c7[1],c7[2],p[9]);

```

```

FA I27(s6[3],c6[3],c7[2],c7[3],p[10]);
FA I28(s6[4],c6[4],c7[3],c7[4],p[11]);
FA I29(s6[5],c6[5],c7[4],c7[5],p[12]);
FA I30(s6[6],c6[6],c7[5],c7[6],p[13]);
FA I31(si[7],c6[7],c7[6],c7[7],p[14]);
xor I32(p[15],1'b1,c7[7]);

endmodule

module HA(a,b,c,s);
input a,b;

output c,s;

and I0(c,a,b);
xor I1(s,a,b);
endmodule

module FA(a,b,c,cy,sm);
input a,b,c;

output cy,sm;

wire x,y,z;

HA I0(a,b,x,z);
HA I1(z,c,y,sm);
or I2(cy,x,y);
endmodule

module and_array(y, x, ip, si);
input y;
input [7:0] x;

output [6:0] ip;
output si;

and I0(ip[0],y,x[0]);
and I1(ip[1],y,x[1]);
and I2(ip[2],y,x[2]);
and I3(ip[3],y,x[3]);
and I4(ip[4],y,x[4]);
and I5(ip[5],y,x[5]);
and I6(ip[6],y,x[6]);
nand I7(si, y,x[7]);
endmodule

module adder_array(si, s1, c1, ip, s2, c2, p);

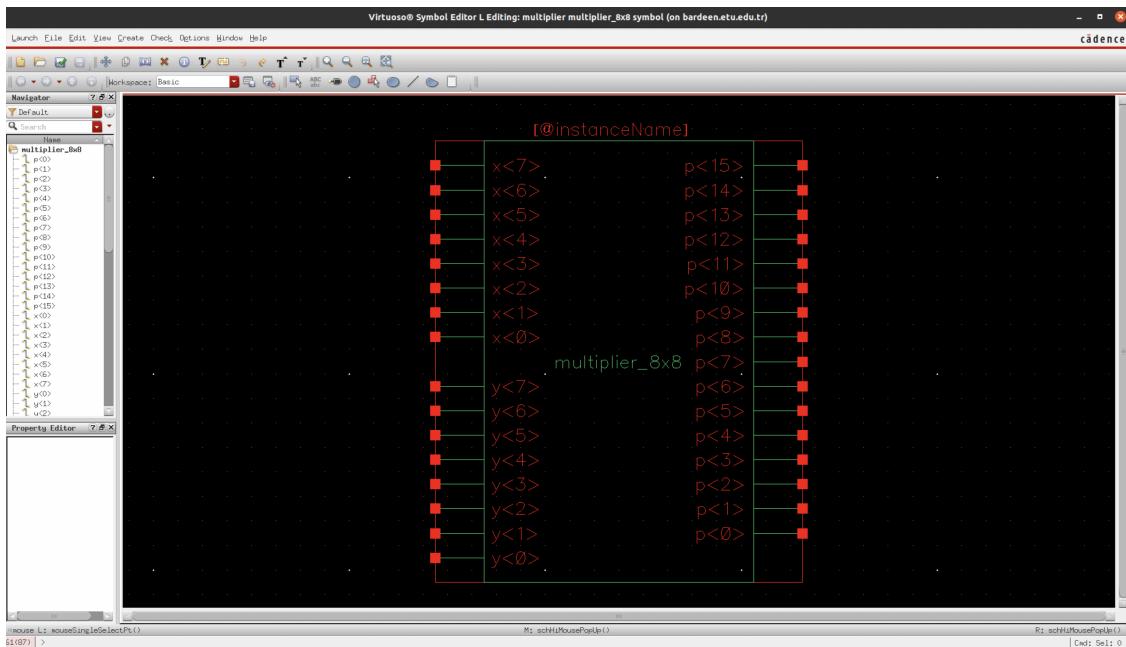
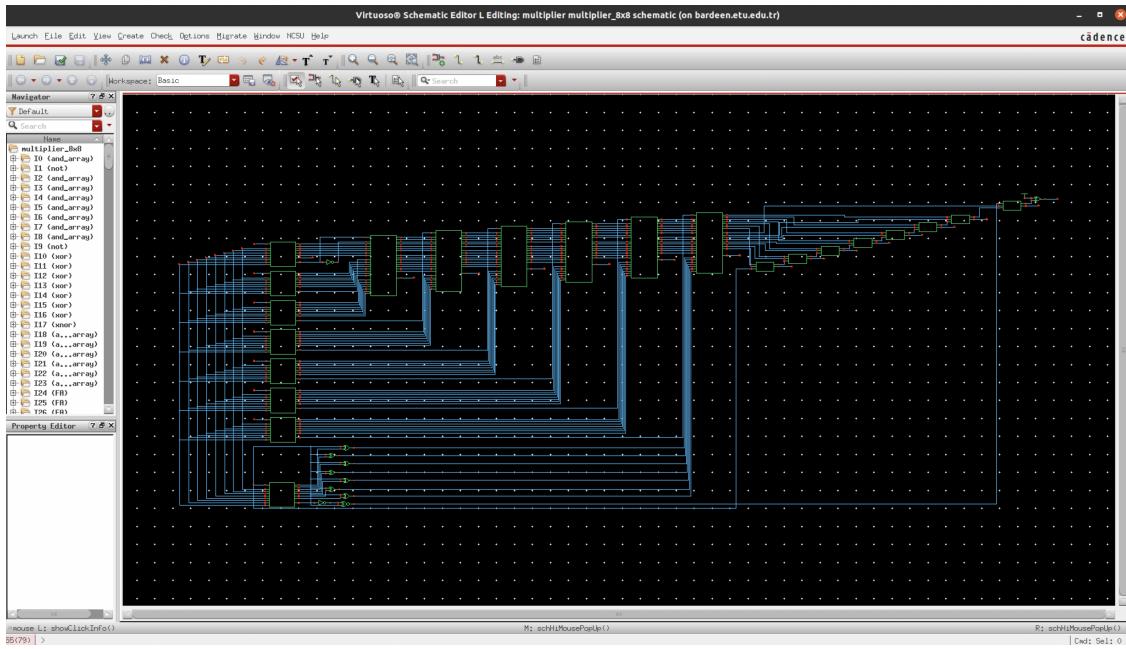
```

```
input si;
input [6:0] s1;
input [7:0] c1;
input [6:0] ip;

output [6:0] s2;
output [7:0] c2;
output p;

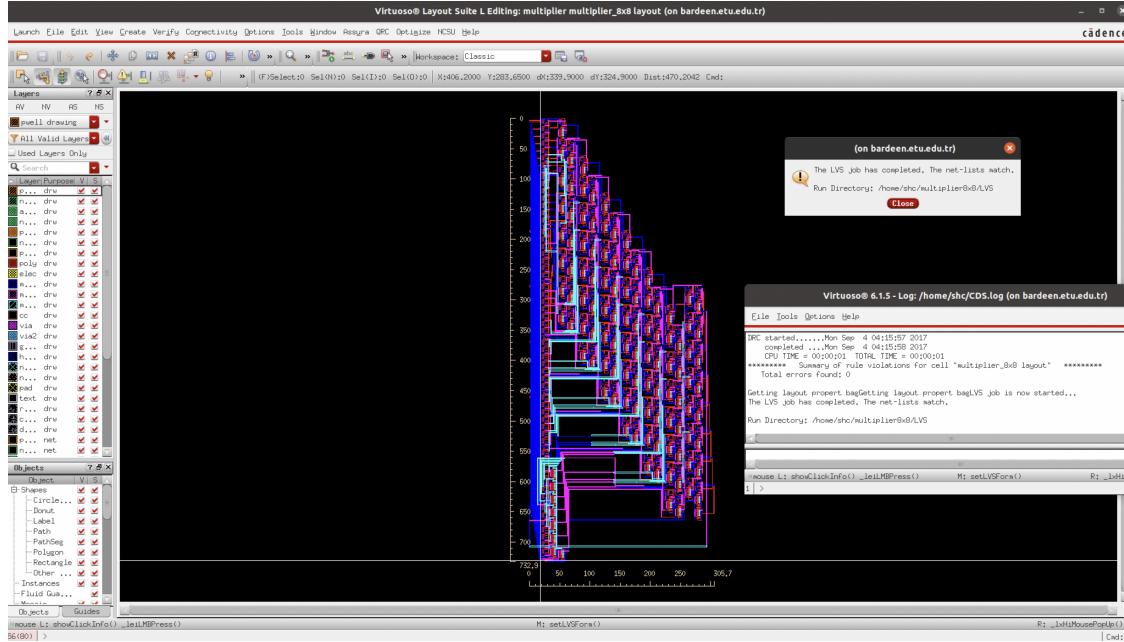
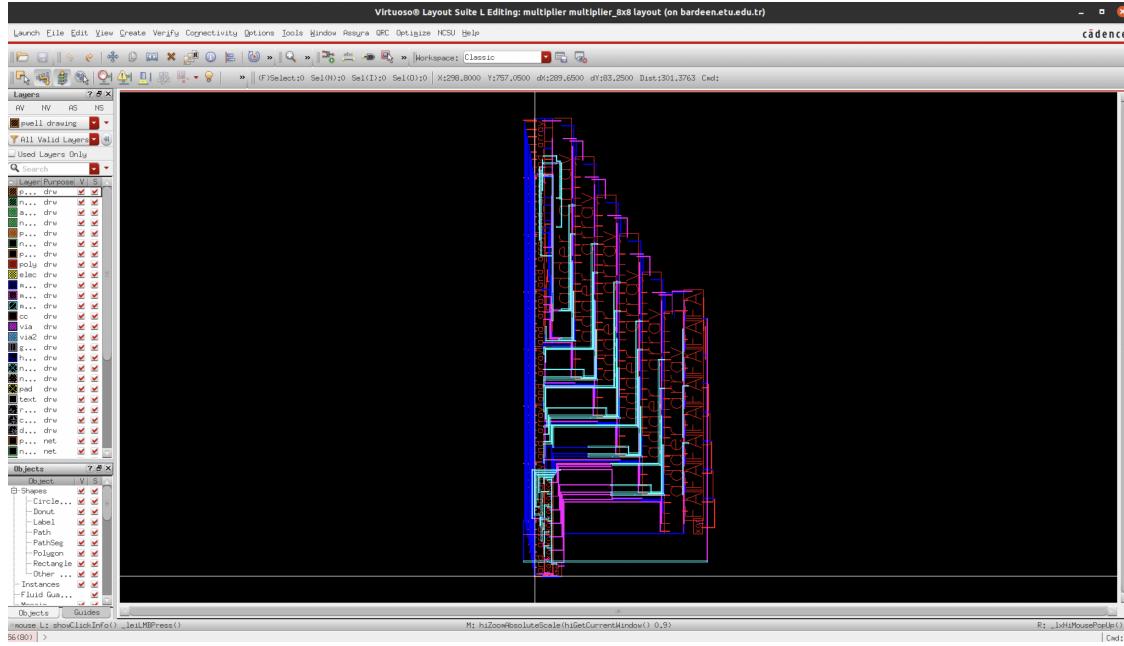
HA I0(s1[0],c1[0],      c2[0],p);
FA I1(s1[1],c1[1],ip[0],c2[1],s2[0]);
FA I2(s1[2],c1[2],ip[1],c2[2],s2[1]);
FA I3(s1[3],c1[3],ip[2],c2[3],s2[2]);
FA I4(s1[4],c1[4],ip[3],c2[4],s2[3]);
FA I5(s1[5],c1[5],ip[4],c2[5],s2[4]);
FA I6(s1[6],c1[6],ip[5],c2[6],s2[5]);
FA I7(si,    c1[7],ip[6],c2[7],s2[6]);
endmodule
```

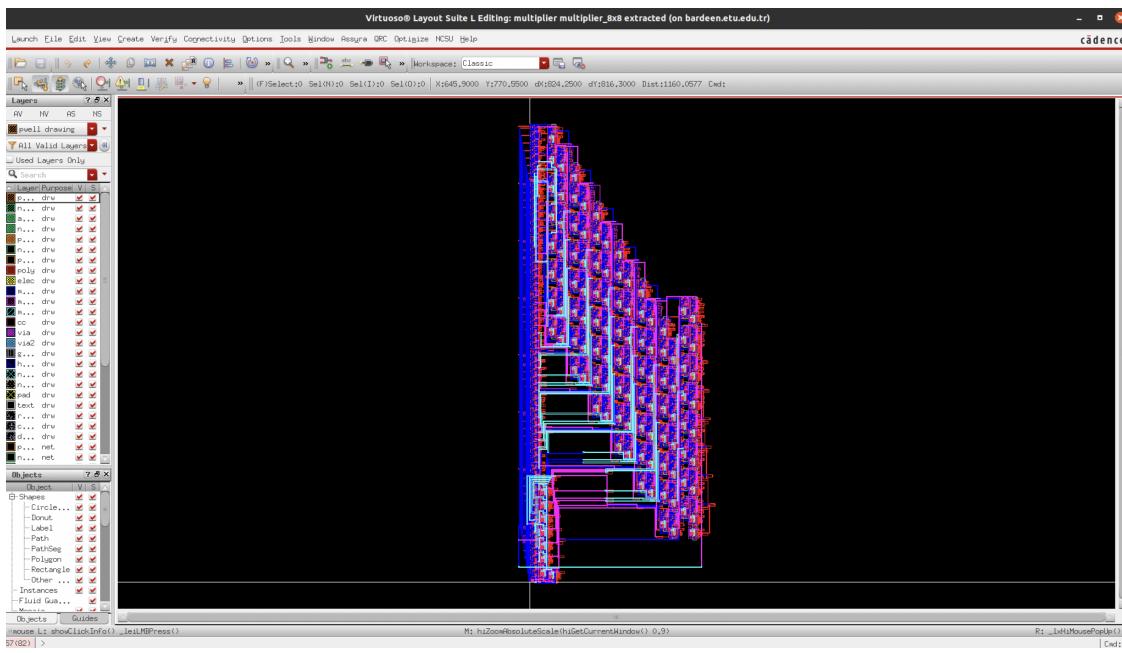
3.2 Wallace Tree Çarpıcı Devresi Sematik ve Sembol



3.3 Wallace Tree Çarpıcı Devresi Layout ve Extracted

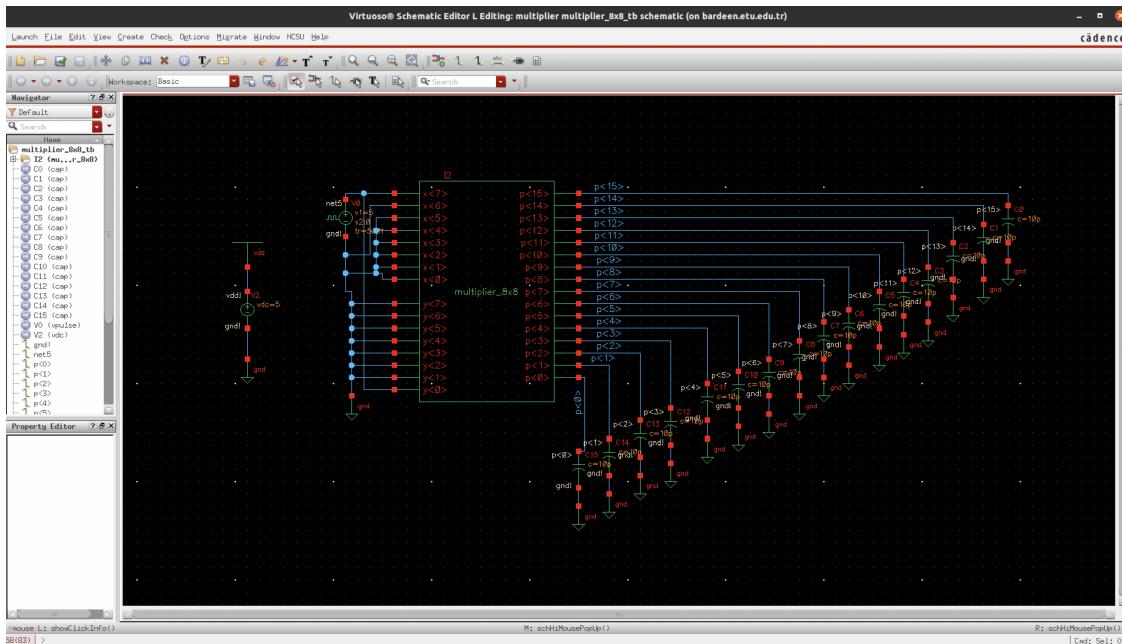
Wallace Tree çarpıcı devresi layoutu dikeyde 732.9um, yatayda 305.7um uzunluğundadır ve 224047.53um^2 alana sahiptir. DRC ve LVS'den geçmiştir.





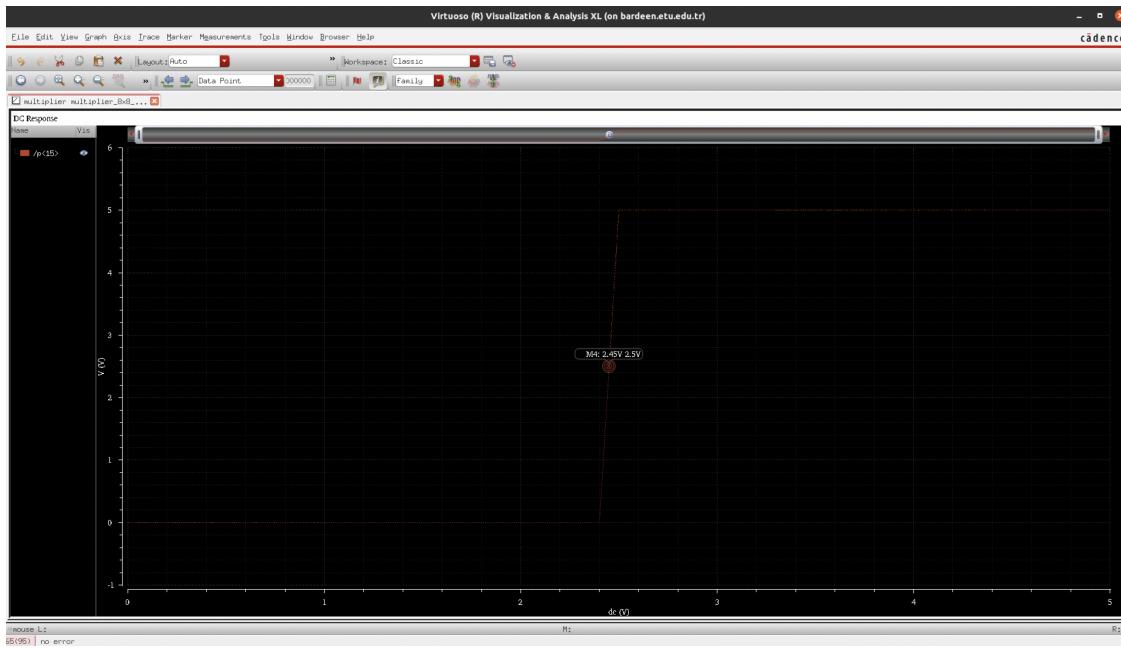
3.4 Wallace Tree Çarpıcı Devresi Testbench

DC ve Transient analiz için testbench devresi aşağıdaki gibi kurulmuştur.



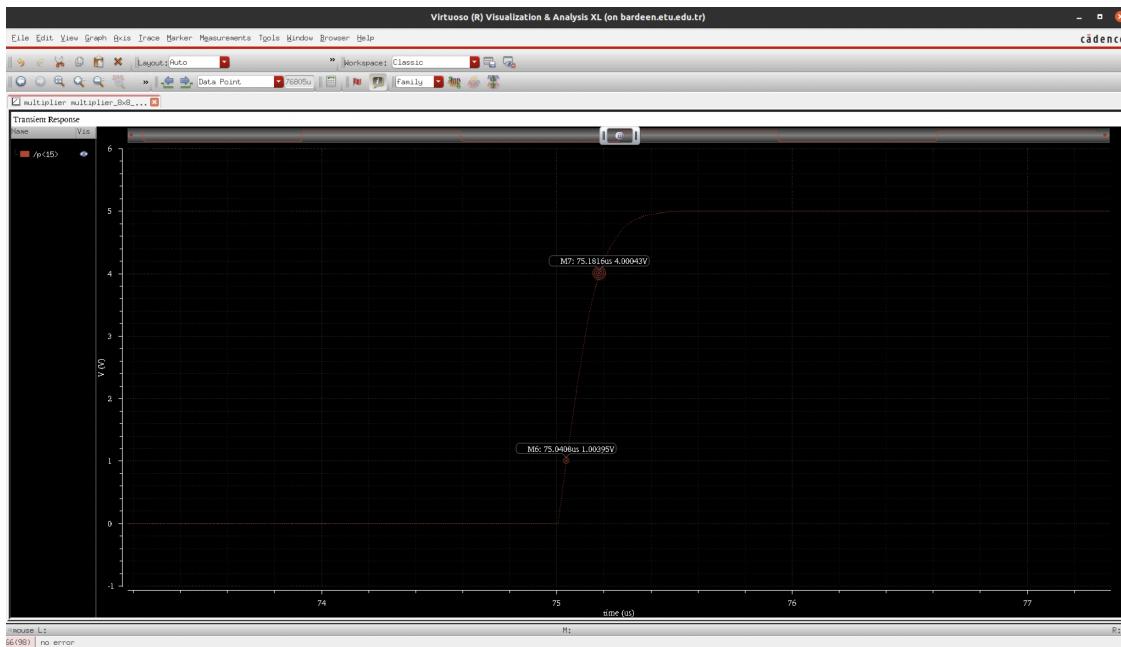
3.4.1 Wallace Tree Çarpıcı Devresi DC Analiz

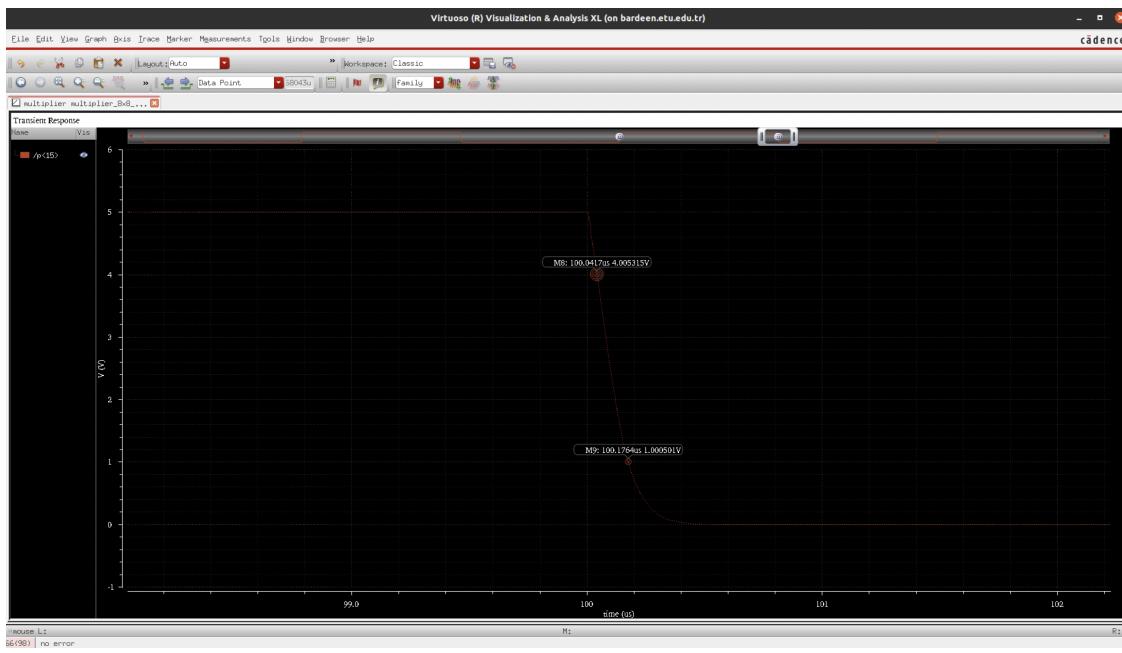
Aşağıdaki grafikte görüleceği üzere, parazitik kapasitanslar ekliyken $p<15>$ (en uzun yoldaki) çıkış birim eviriciye benzetilmiş ve orta noktası $2.45V - 2.5V$ olarak bulunmuştur. Neredeyse simetriktir.



3.4.2 Wallace Tree Çarpıcı Devresi Transient Analiz

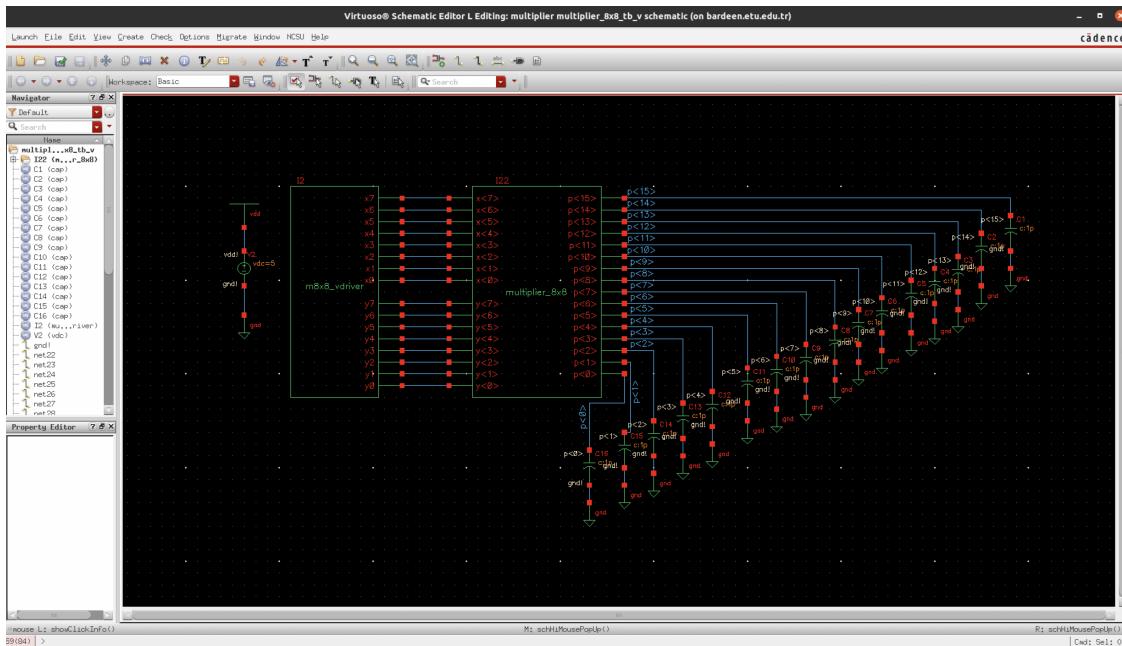
Transient analizde de yine $p<15>$ çıkışına bakılmış, yükselme zamanı $0.1408\mu s$, düşme zamanı $0.1347\mu s$ olarak bulunmuştur. Neredeyse simetriktir.





3.5 Wallace Tree Çarpıcı Devresi Verilog Testbench

Verimixe uygun değerler girildikten sonra aşağıda görülen verilog kodu ve verilog testbench devresi ile spectreVerilog simülasyonu yapılmış ve tüm ihtimaller kontrol edilemese bile en uç ve ara değerlere bakılmış ve devrenin denenen ihtimallerin hepsi (10 durum) için doğru çarpım sonuçları verdiği görülmüştür.




```

x3, x2, x1, x0}), $signed({y7, y6, y5, y4, y3, y2, y1, y0}), $signed(mult), mult[15], mult[14],
mult[13], mult[12], mult[11], mult[10], mult[9], mult[8], mult[7], mult[6], mult[5], mult[4],
mult[3], mult[2], mult[1], mult[0]);

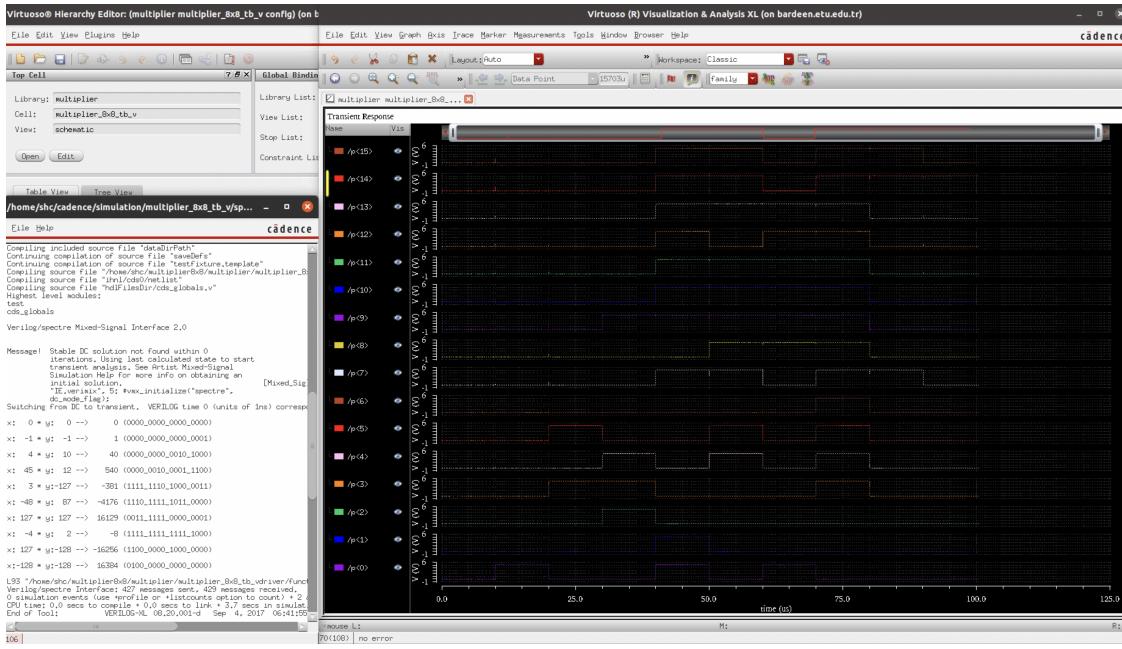
// 127*-128
$display("");
x7=1'b0; x6=1'b1; x5=1'b1; x4=1'b1; x3=1'b1; x2=1'b1; x1=1'b1; x0=1'b1;
y7=1'b1; y6=1'b0; y5=1'b0; y4=1'b0; y3=1'b0; y2=1'b0; y1=1'b0; y0=1'b0;
mult = $signed({x7, x6, x5, x4, x3, x2, x1, x0})*$signed({y7, y6, y5, y4, y3, y2, y1, y0});
#1;
$display("x:%d * y:%d --> %d (%b%b%b%b_%b%b%b%b%b%b%b%b%b%b%b)", $signed({x7, x6, x5, x4,
x3, x2, x1, x0}), $signed({y7, y6, y5, y4, y3, y2, y1, y0}), $signed(mult), mult[15], mult[14],
mult[13], mult[12], mult[11], mult[10], mult[9], mult[8], mult[7], mult[6], mult[5], mult[4],
mult[3], mult[2], mult[1], mult[0]);

// -128*-128
$display("");
x7=1'b1; x6=1'b0; x5=1'b0; x4=1'b0; x3=1'b0; x2=1'b0; x1=1'b0; x0=1'b0;
y7=1'b1; y6=1'b0; y5=1'b0; y4=1'b0; y3=1'b0; y2=1'b0; y1=1'b0; y0=1'b0;
mult = $signed({x7, x6, x5, x4, x3, x2, x1, x0})*$signed({y7, y6, y5, y4, y3, y2, y1, y0});
#1;
$display("x:%d * y:%d --> %d (%b%b%b%b_%b%b%b%b%b%b%b%b%b%b%b)", $signed({x7, x6, x5, x4,
x3, x2, x1, x0}), $signed({y7, y6, y5, y4, y3, y2, y1, y0}), $signed(mult), mult[15], mult[14],
mult[13], mult[12], mult[11], mult[10], mult[9], mult[8], mult[7], mult[6], mult[5], mult[4],
mult[3], mult[2], mult[1], mult[0]);

$display("");
$finish;
end

endmodule

```



4 Tasarlanan Kapı ve Alt Modüller

Devrede iki girişli xor, xnor, or, nand, and ve not kapılarına ihtiyaç duyulmuş ve **6 çeşit mantıksal kapı** tasarlanmıştır. (Aslında daha az çeşit kapı ile de aynı devre tasarlanabilirdi fakat transistör ve alandan kazanabilmek amacıyla, bir mantıksal işlemi birden fazla kapıyla yapmaktansa tek kapı ile yapmak daha mantıklı gelmiştir.)

Ayrıca, devre daha modülerize hale getirilerek alt modüller olan **Half Adder (Yarım Toplayıcı)**, **Full Adder (Tam Toplayıcı)**, **And Array (Ve Kapısı Dizisi)**, **Adder Array (Toplayıcı Dizisi)** gibi ağaç yapısını sağlayan **4 alt modül** tasarlanmıştır.

Transistör genişlikleri, kapilar başlangıçta birim eviriciye benzetilerek dc simülasyonla ayarlanmış fakat kapilar tek başlarına bu genişliklerde en iyi sonucu verirken ana devreye geçildikten sonra en uzun yola sahip olan $p<15>$ çıkışına bakılmış ve kapı genişlikleri dikey layout uzunlukları aynı tutulacak, çarpıcı devresi en iyi dc simülasyon sonucunu verecek ve pmos'lar 2.4um, nmoslar 1.5um genişliğe sahip olacak şekilde yeniden tasarlanmıştır. (nand kapısı hariç, nand kapısında tam tersi pmoslar 1.5um iken nmoslar 2.4um fakat yine de layout dikey uzunluğu aynı) Kapıların layoutları, en büyük kapı olan xor baz alınarak aynı dikey uzunluğa (14.1um) getirilecek şekilde çizilmiştir.

Kapılar için transistör genişliklerinin tablosu Tablo 1'de görülebilir.

Tablo 1: Transistör Genişlikleri Tablosu

Kapılar	Çarpıcı Devresi Tasarlanmadan Önce	Çarpıcı Devresi Tasarlandıktan Sonra
nand	pmos 1.5um, nmos 2.4um	pmos 1.5um, nmos 2.4um
and	pmos 2.25um, nmos 1.5um	pmos 2.4um, nmos 1.5um
xor	pmos 2.4um, nmos 1.5um	pmos 2.4um, nmos 1.5um
xnor	pmos 2.4um, nmos 1.5um	pmos 2.4um, nmos 1.5um
or	pmos 3.45um, nmos 1.5um	pmos 2.4um, nmos 1.5um
not	pmos 2.4um, nmos 1.5um	pmos 2.4um, nmos 1.5um

Devredeki tüm kapı ve alt modüllerin layoutlarındaki dikey, yatay uzunluk, alan ve transistör sayısı tablosu Tablo 2'de görülebilir.

Tablo 2: Kapı ve Alt Modüllerin Uzunluk, Alan ve Transistör Sayısı Tablosu

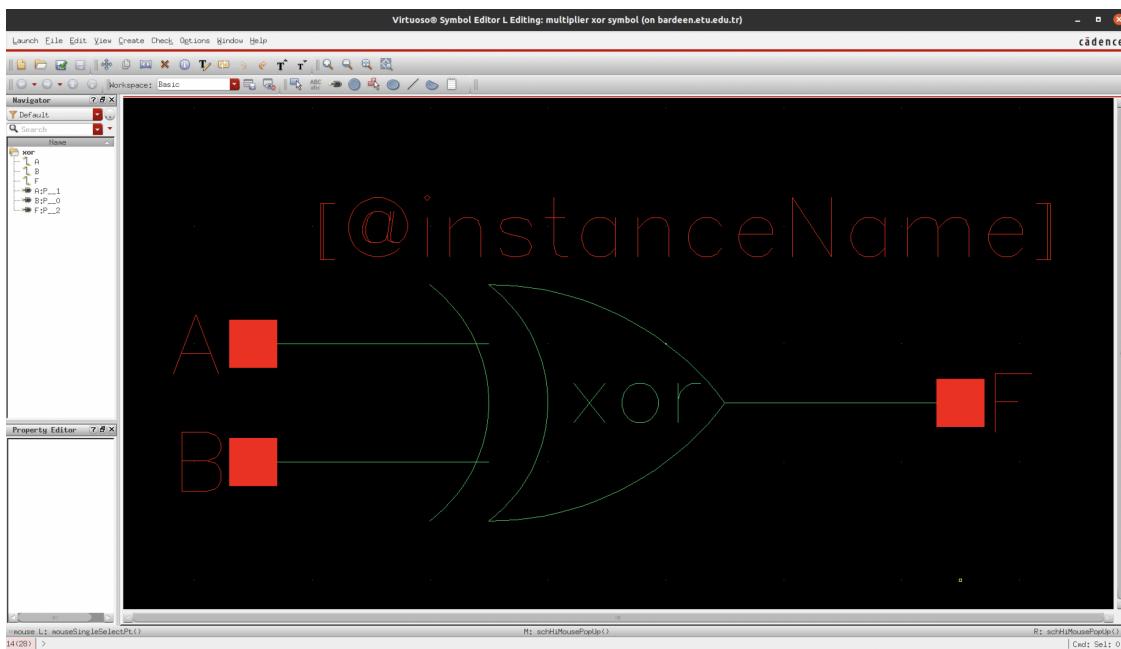
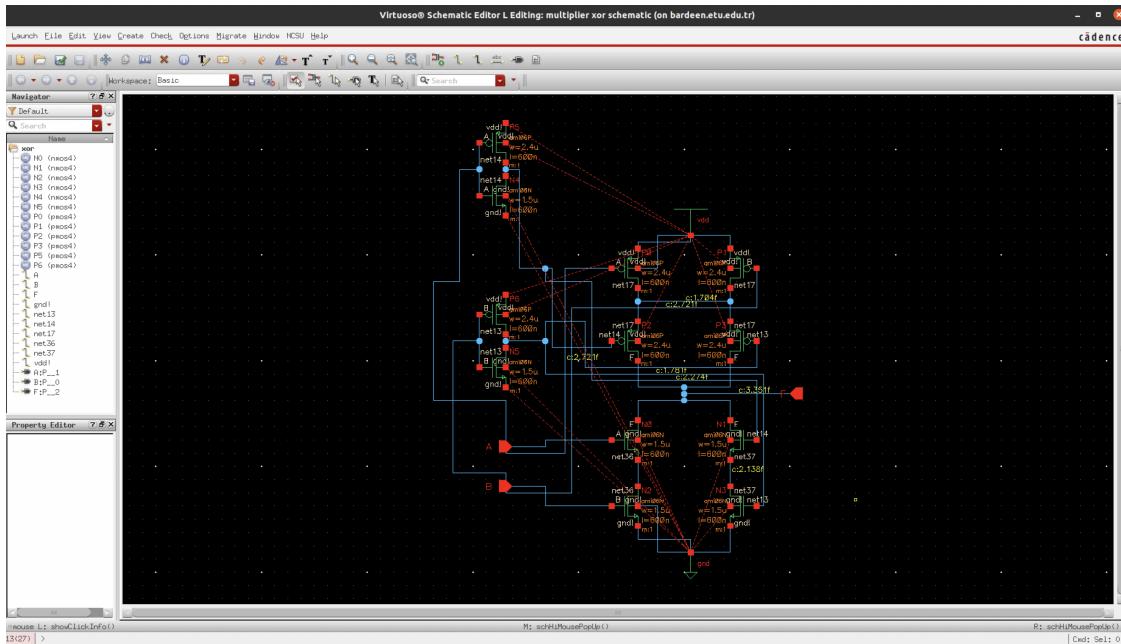
Kapı ve Alt Modüller	Dikey Uzunluk	Yatay Uzunluk	Alan	Transistör Sayısı
nand	14.1um	9.6um	135.36um ²	2 pmos, 2 nmos, toplam 4
and	14.1um	14.4um	203.04um ²	3 pmos, 3 nmos, toplam 6
xor	14.1um	24um	338.4um ²	6 pmos, 6 nmos, toplam 12
xnor	14.1um	24um	338.4um ²	6 pmos, 6 nmos, toplam 12
or	14.1um	14.4um	203.04um ²	3 pmos, 3 nmos, toplam 6
not	14.1um	7.2um	101.52um ²	1 pmos, 1 nmos, toplam 2
half adder	14.7um	36um	529.2um ²	9 pmos, 9 nmos, toplam 18
full adder	29.7um	48um	1425.6um ²	21 pmos, 21 nmos, toplam 42
and array	15.6um	93.6um	1460.16um ²	23 pmos, 23 nmos, toplam 46
adder array	29.7um	355.2um	10549.44um ²	156 pmos, 156 nmos, toplam 312

4.1 xor

xor kapısı $F = A \cdot B' + A' \cdot B$ işlemini yapar.

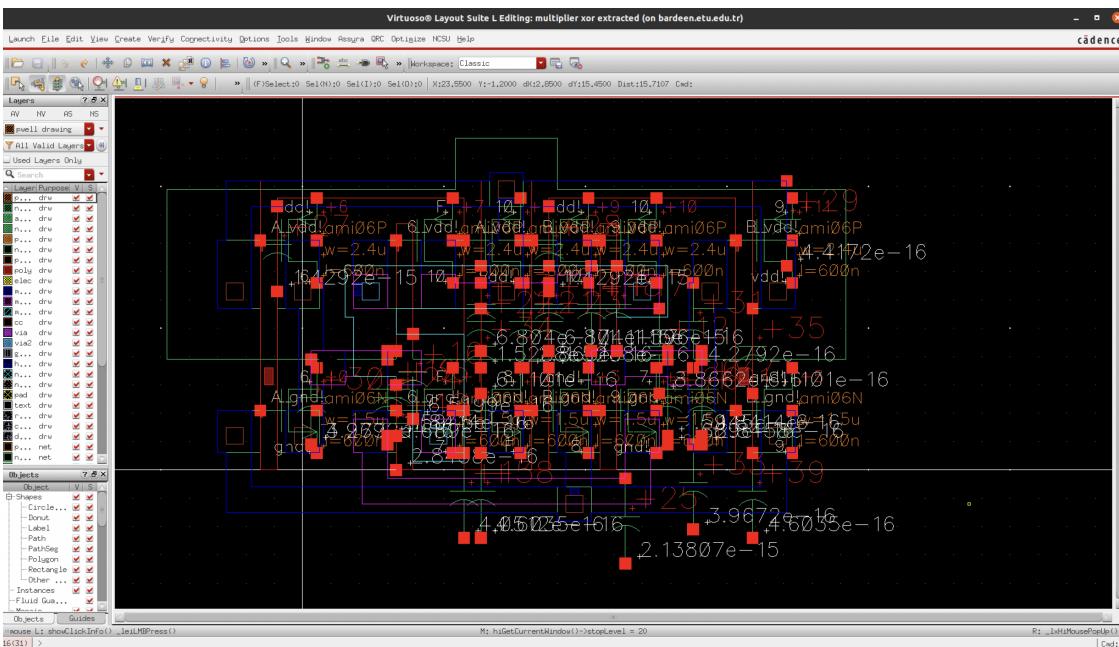
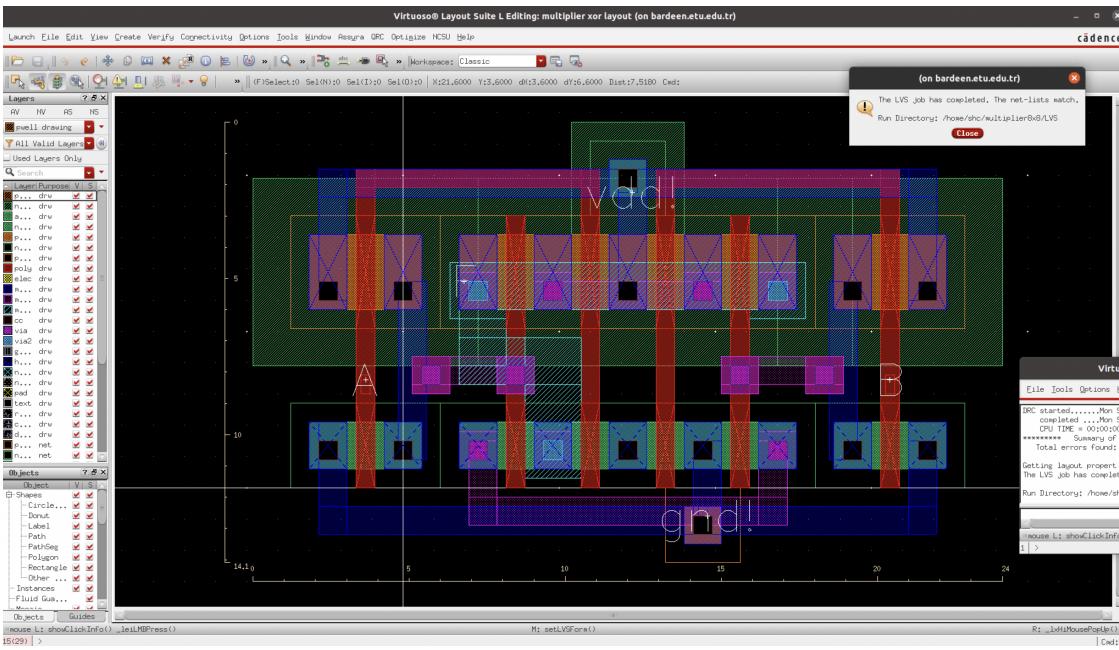
4.1.1 xor Şematik ve Sembol

xor CMOS tasarımda 6 nmos, 6 pmos toplam 12 transistör vardır.



4.1.2 xor Layout ve Extracted

xor layoutu dikeyde 14.1um, yatayda 24um uzunluğundadır ve 338.4um^2 alana sahiptir.

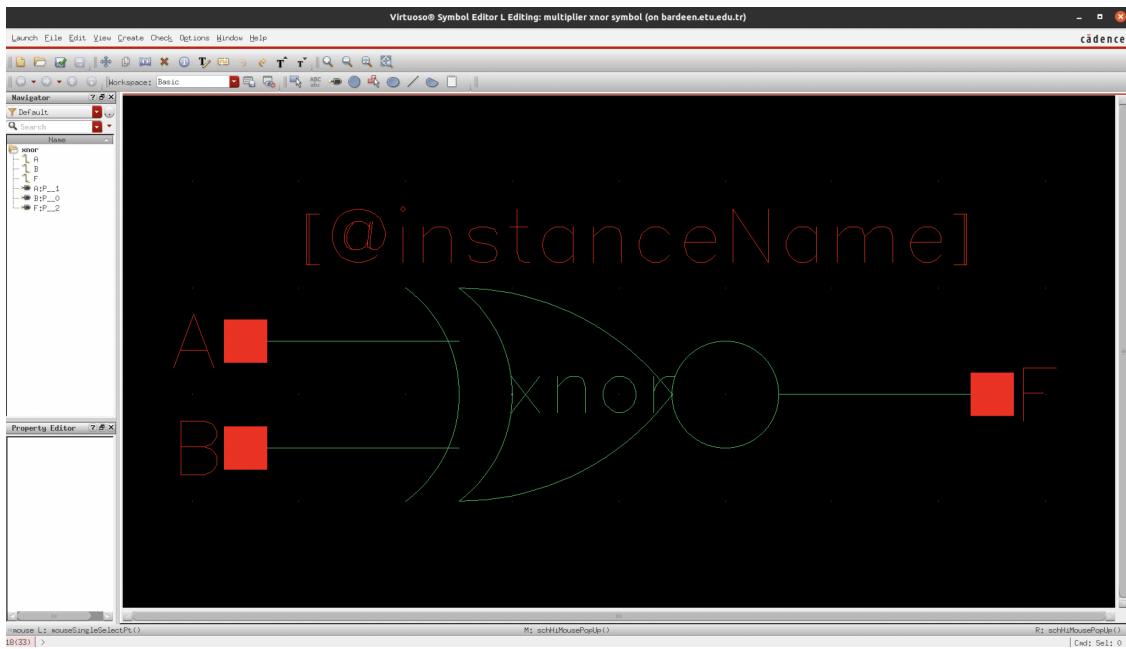
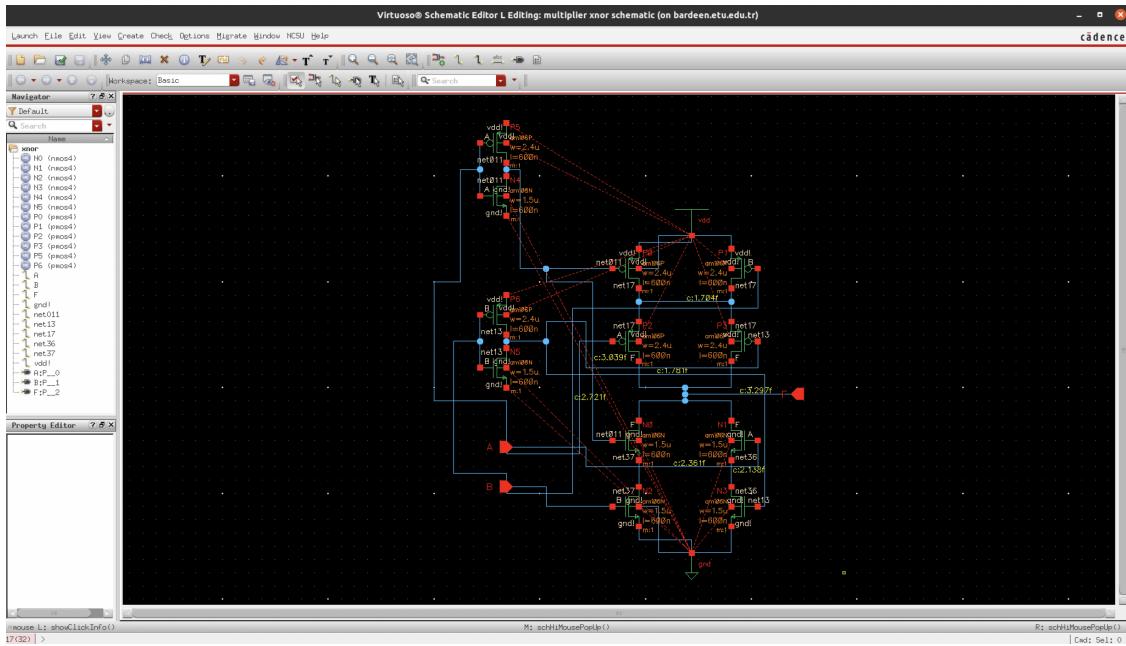


4.2 xnor

xnor kapısı $F = A \cdot B + A' \cdot B'$ işlemini yapar.

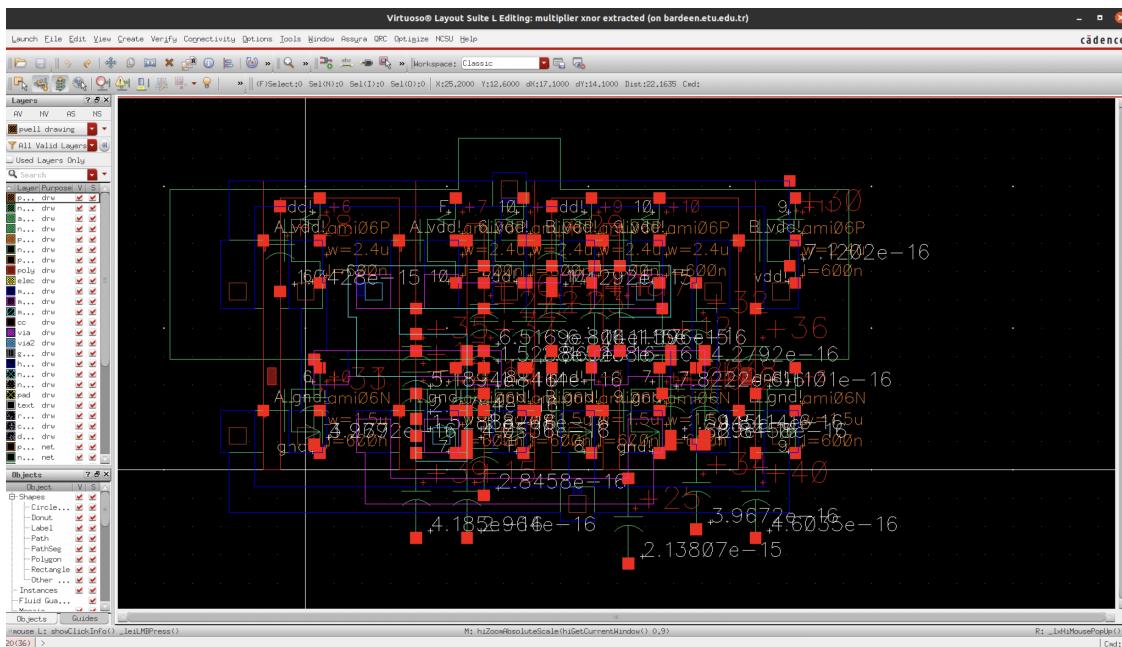
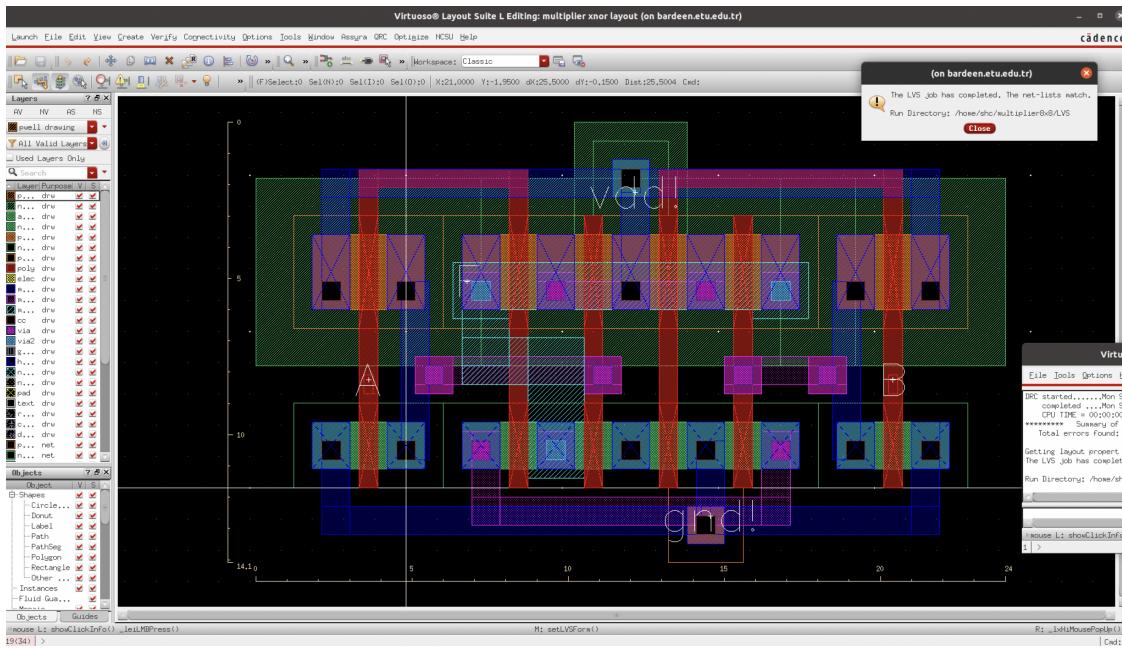
4.2.1 xnor Şematik ve Sembol

xnor CMOS tasarımda 6 nmos, 6 pmos toplam 12 transistör vardır.



4.2.2 xnor Layout ve Extracted

xnor layoutu dikeyde 14.1um, yatayda 24um uzunluğundadır ve 338.4um^2 alana sahiptir.

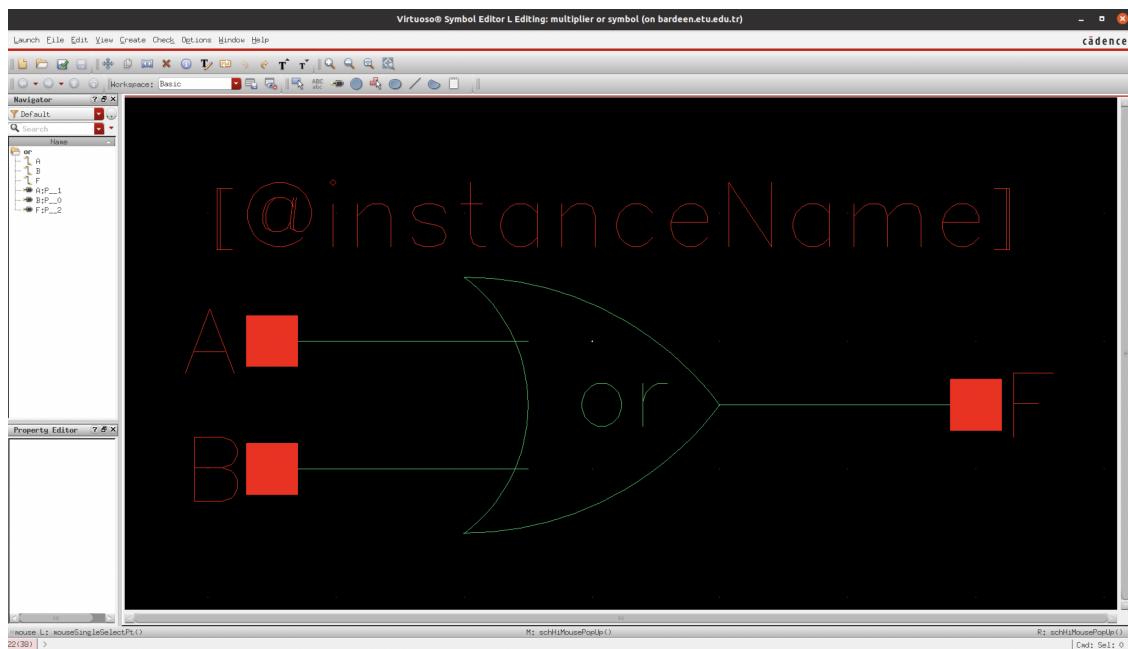
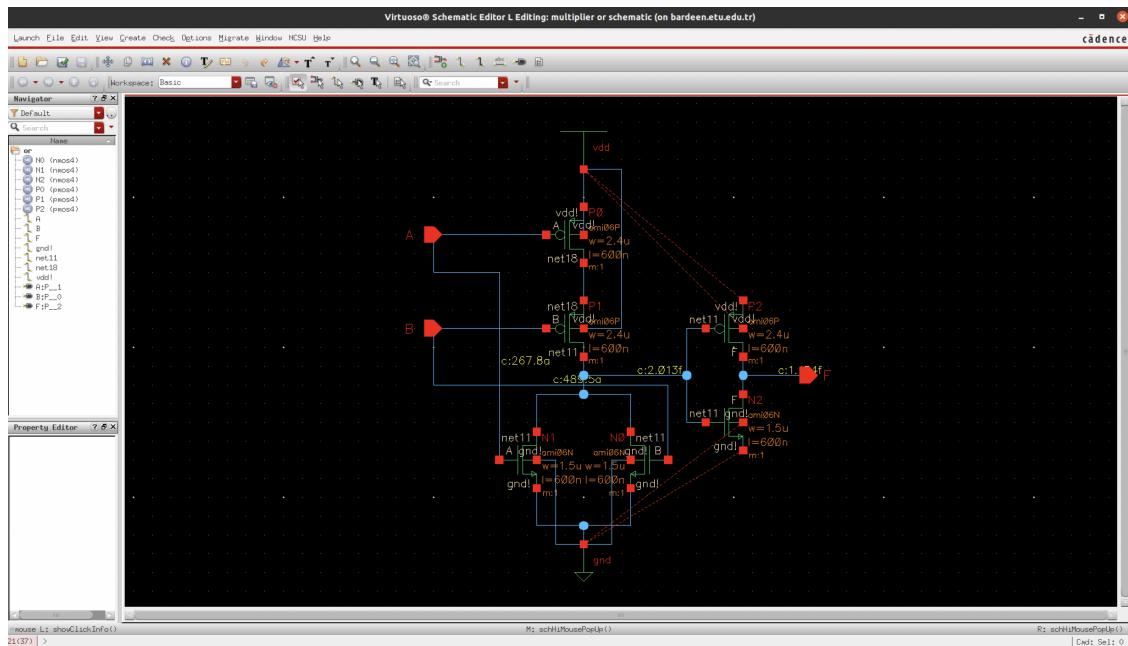


4.3 or

or kapısı $F = A + B$ işlemini yapar.

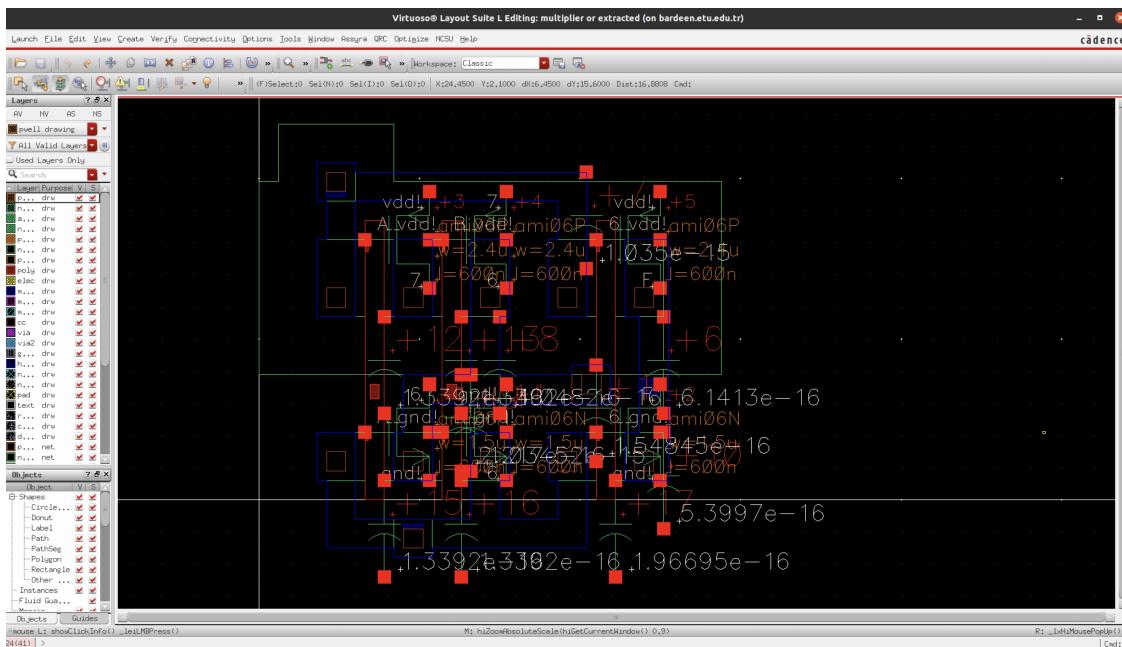
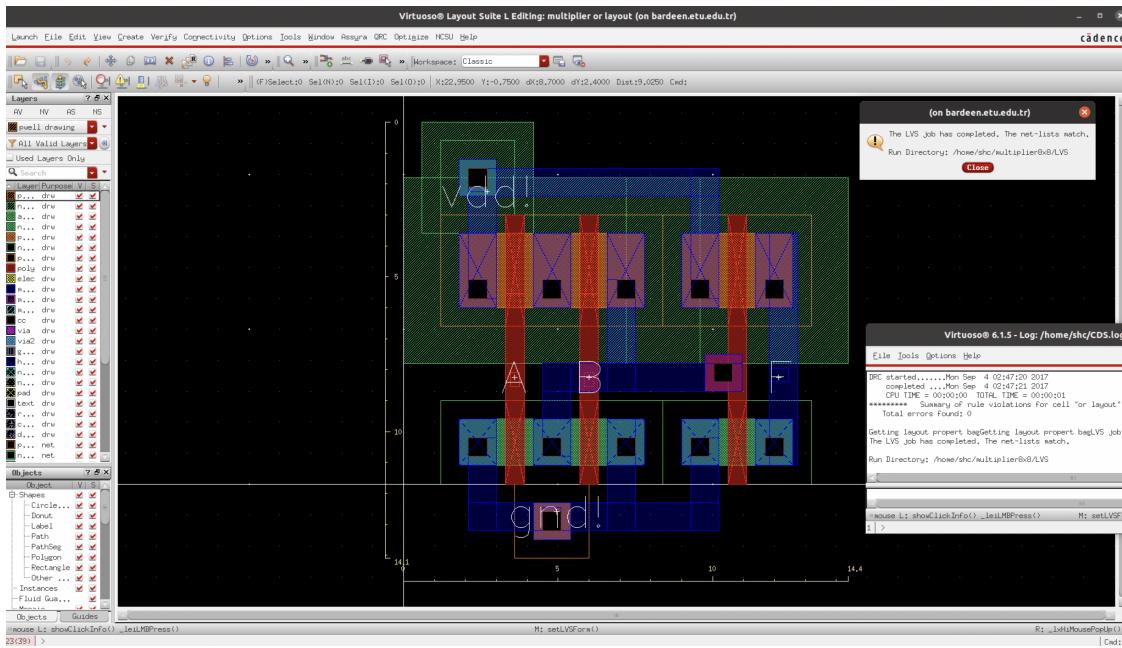
4.3.1 or Şematik ve Sembol

or CMOS tasarımda 3 nmos, 3 pmos toplam 6 transistör vardır.



4.3.2 or Layout ve Extracted

or layoutu dikeyde 14.1um, yatayda 14.4um uzunlugundadir ve 203.04um² alana sahiptir.

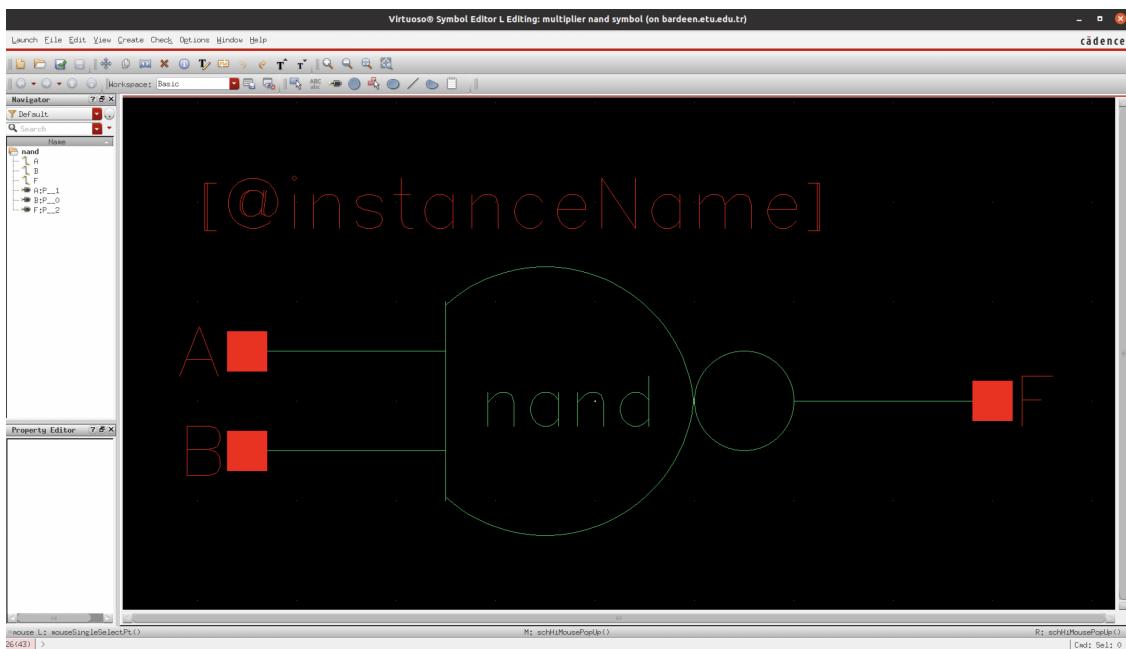
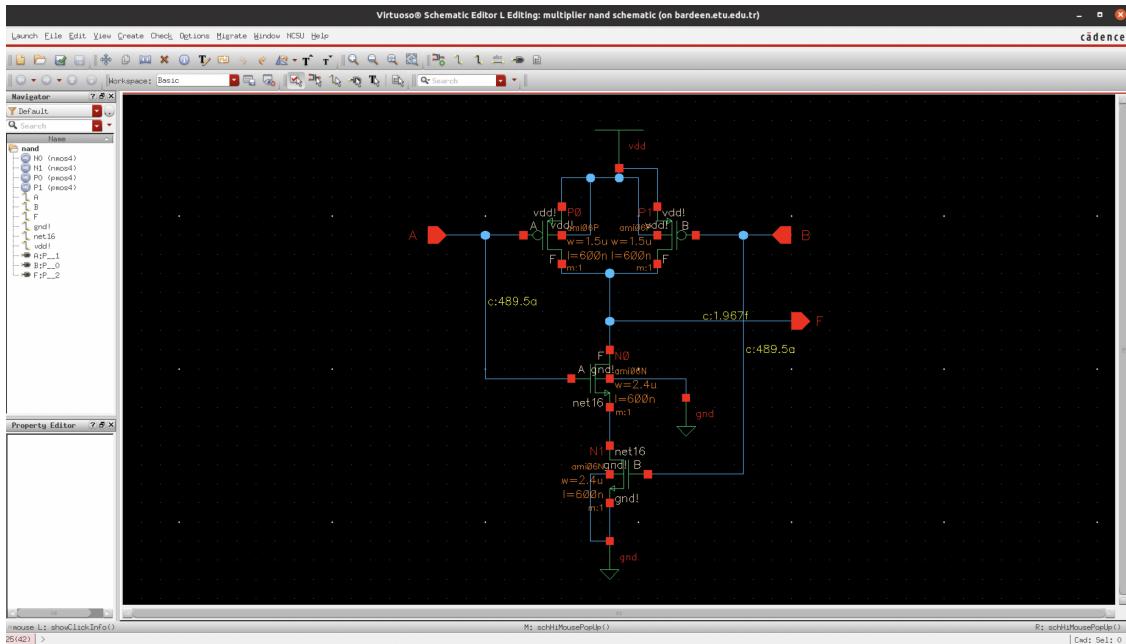


4.4 nand

nand kapısı $F = A' + B'$ işlemini yapar.

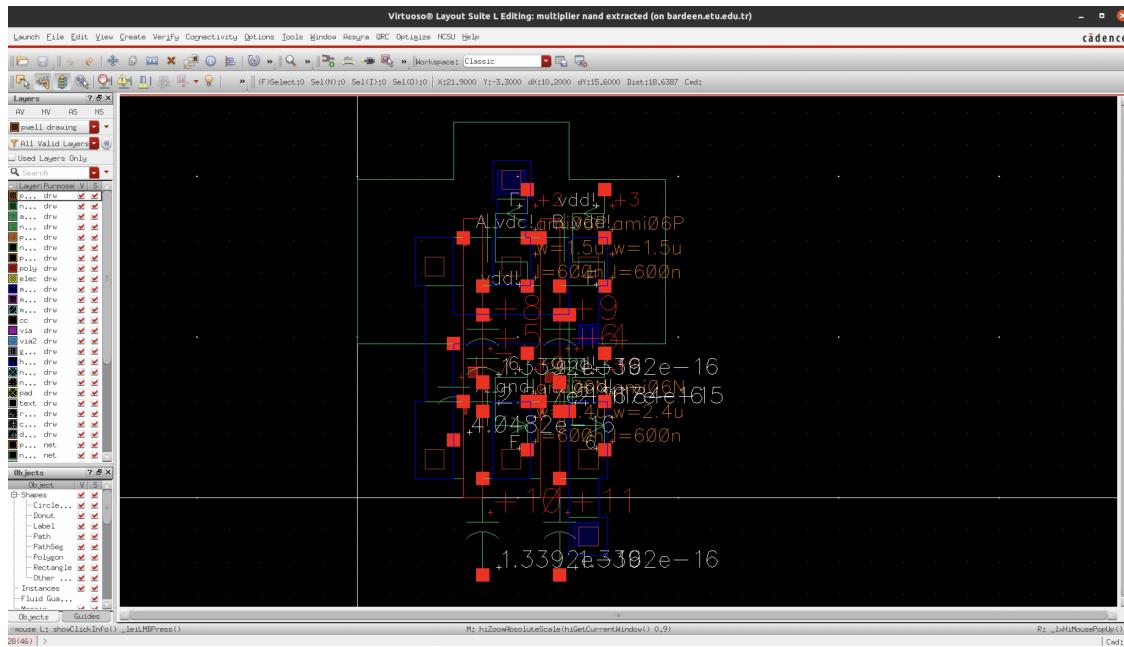
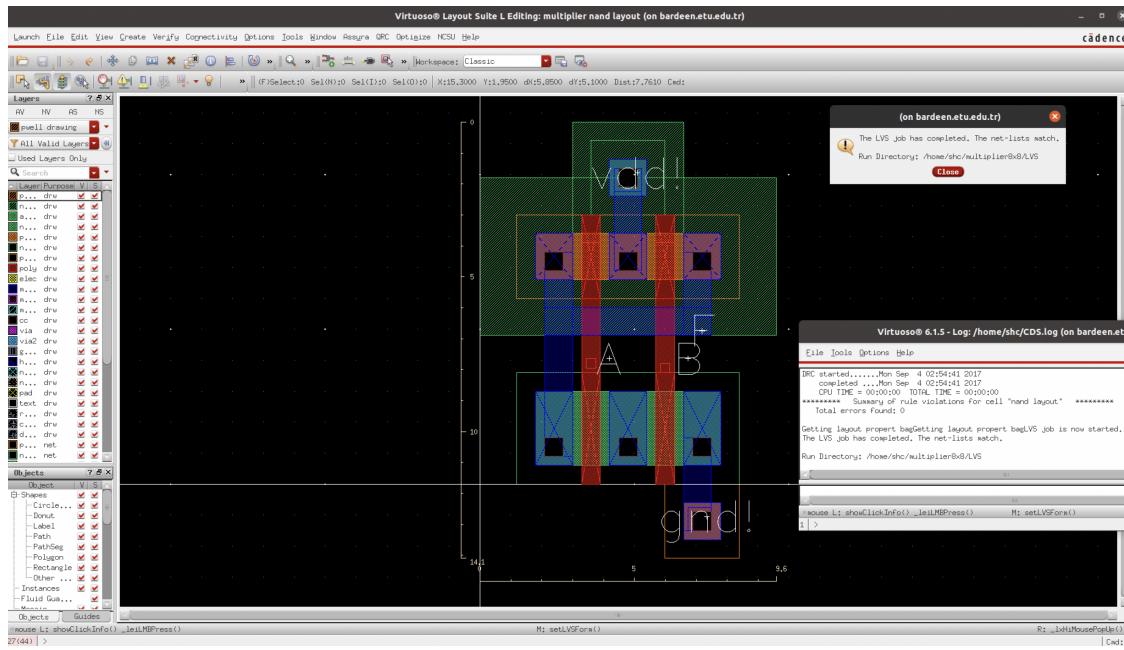
4.4.1 nand Şematik ve Sembol

nand CMOS tasarımda 2 nmos, 2 pmos toplam 4 transistör vardır.



4.4.2 nand Layout ve Extracted

nand layoutu dikeyde 14.1um, yatayda 9.6um uzunluğundadır ve 135.36um^2 alana sahiptir.

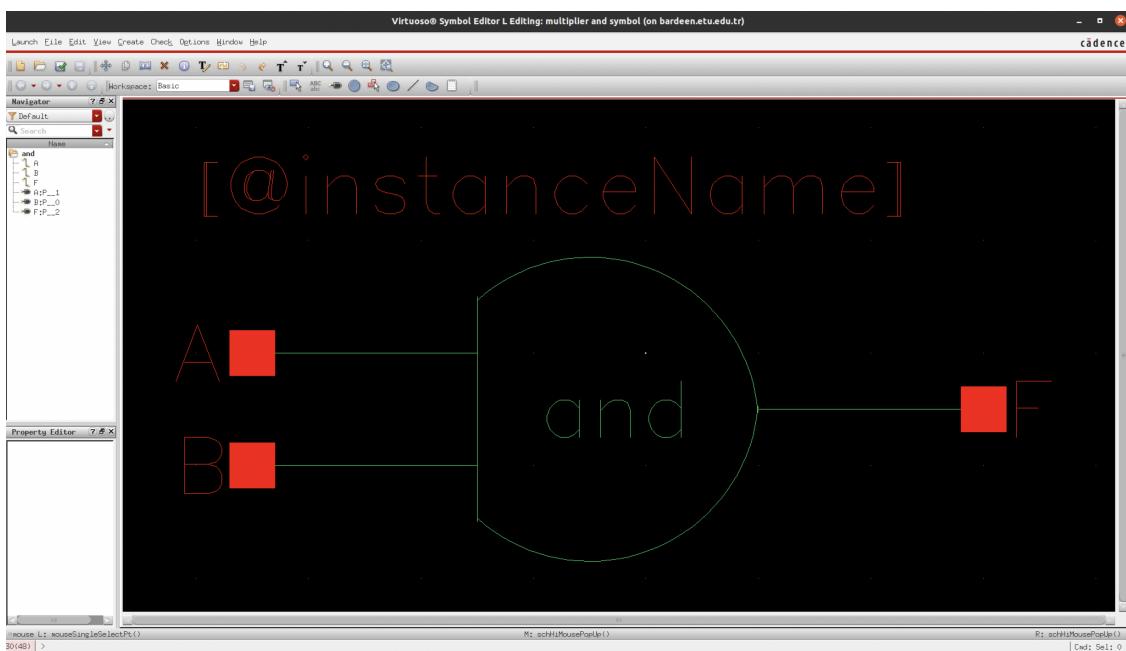
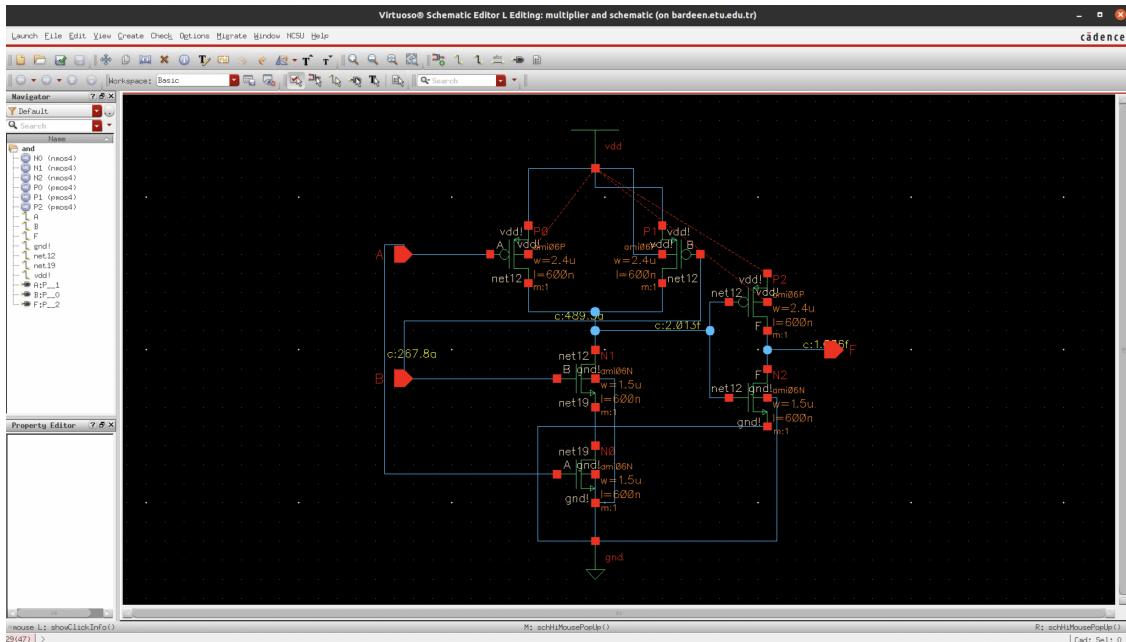


4.5 and

and kapısı $F = A \cdot B$ işlemini yapar.

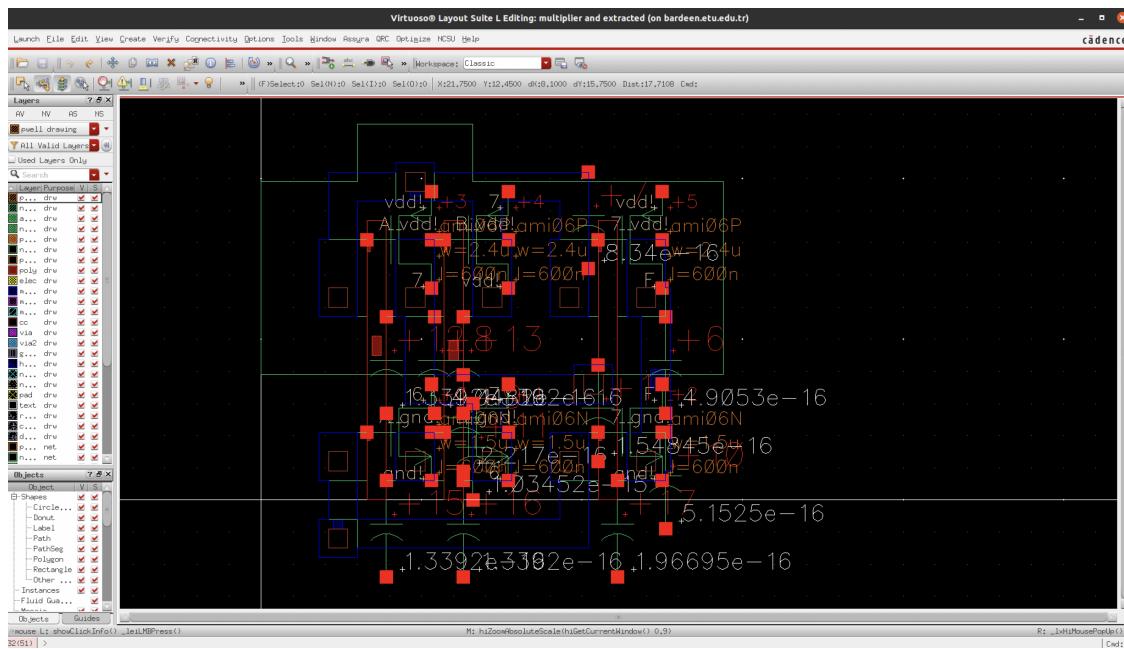
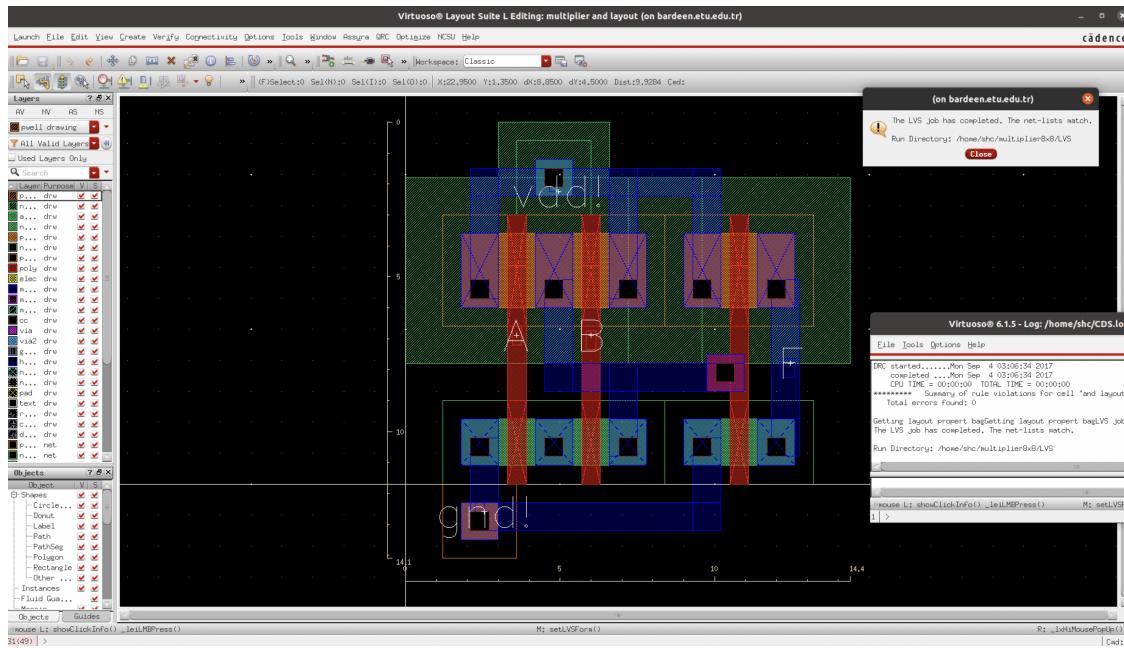
4.5.1 and Şematik ve Sembol

and CMOS tasarımda 3 nmos, 3 pmos toplam 6 transistör vardır.



4.5.2 and Layout ve Extracted

and layoutu dikeyde 14.1um, yatayda 14.4um uzunluğundadır ve 203.04um^2 alana sahiptir.

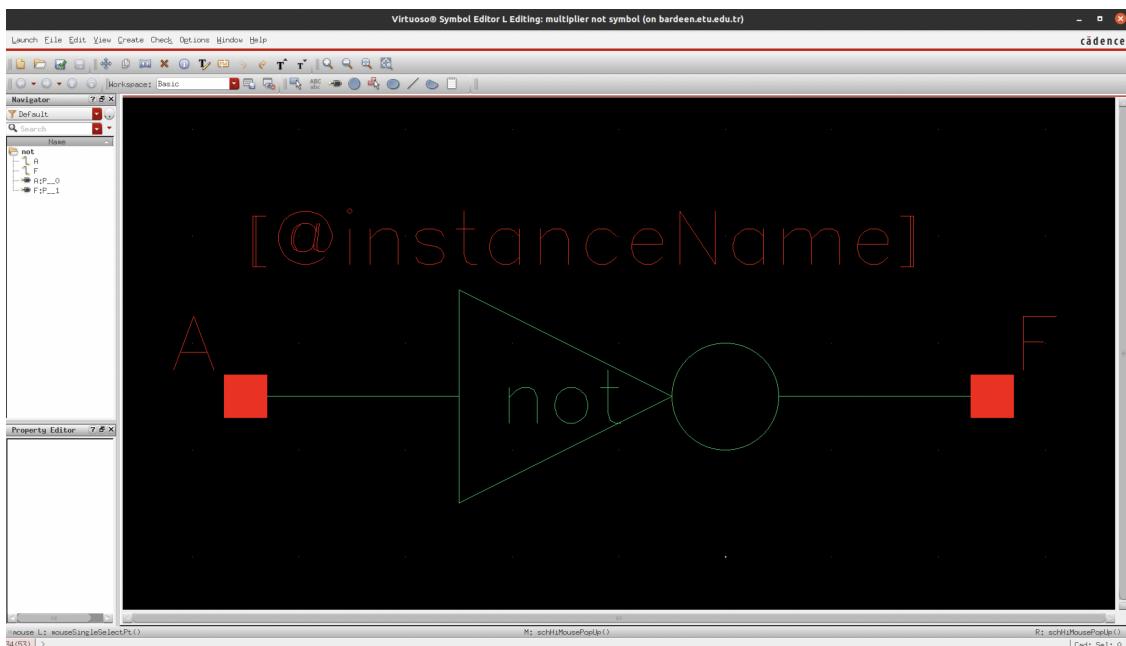
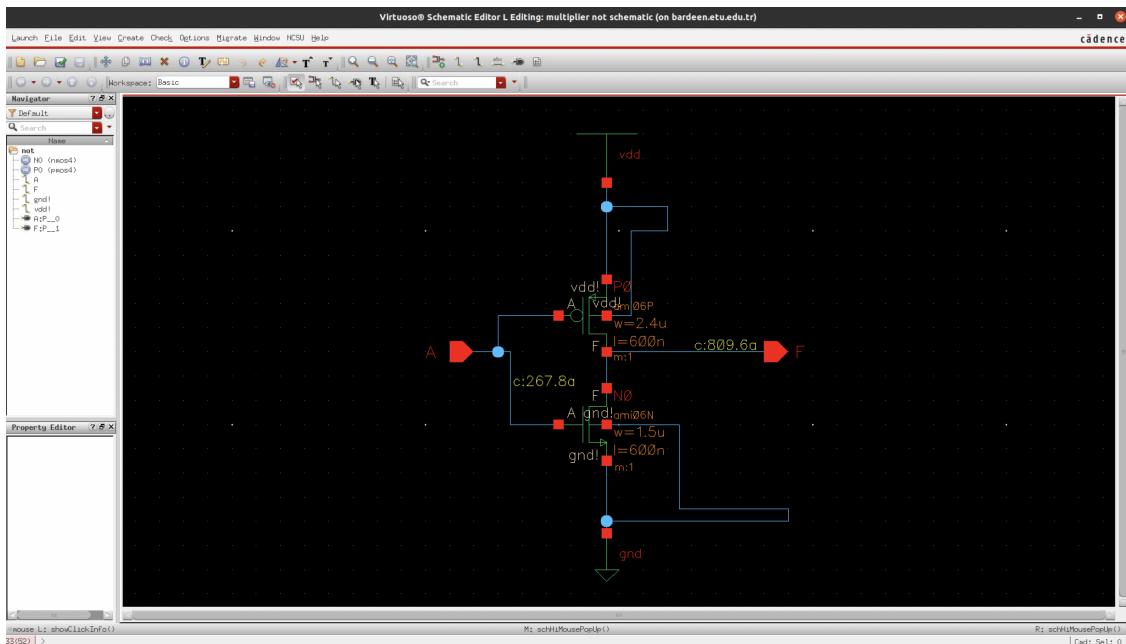


4.6 not

not kapısı $F = A'$ işlemini yapar.

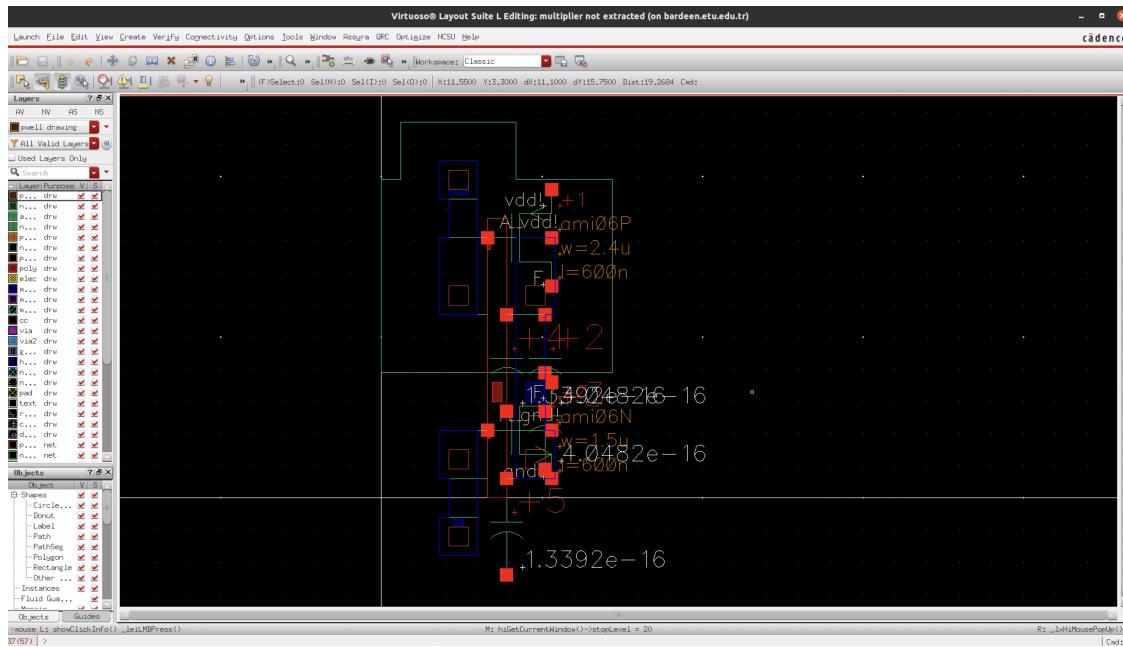
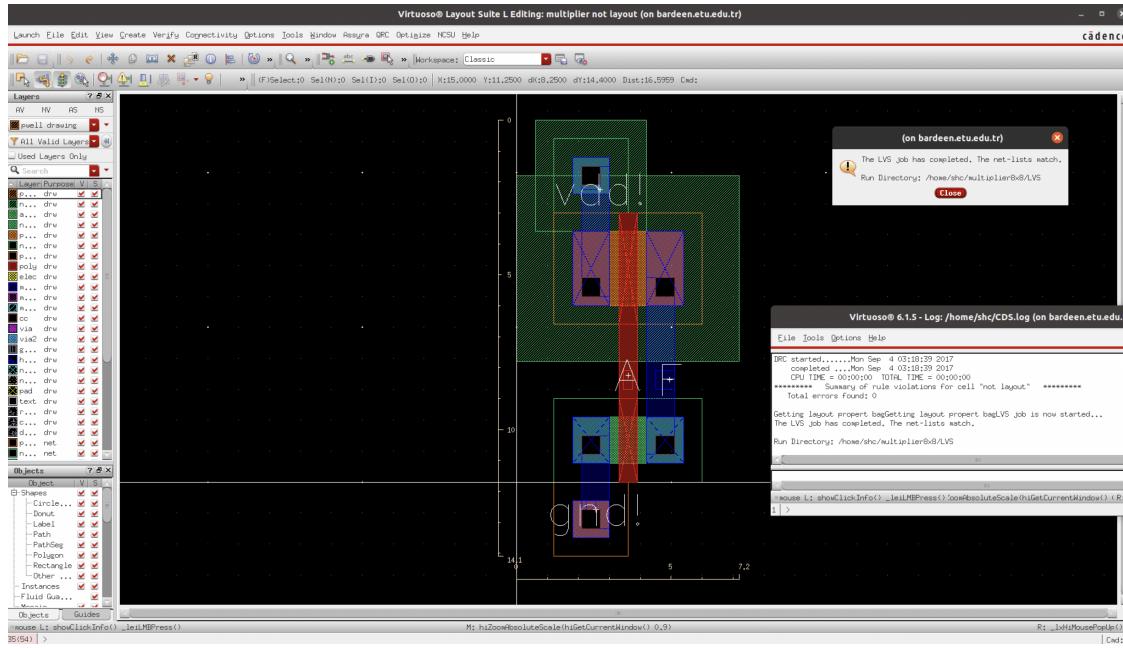
4.6.1 not Şematik ve Sembol

not CMOS tasarımda 1 nmos, 1 pmos toplam 2 transistör vardır.



4.6.2 not Layout ve Extracted

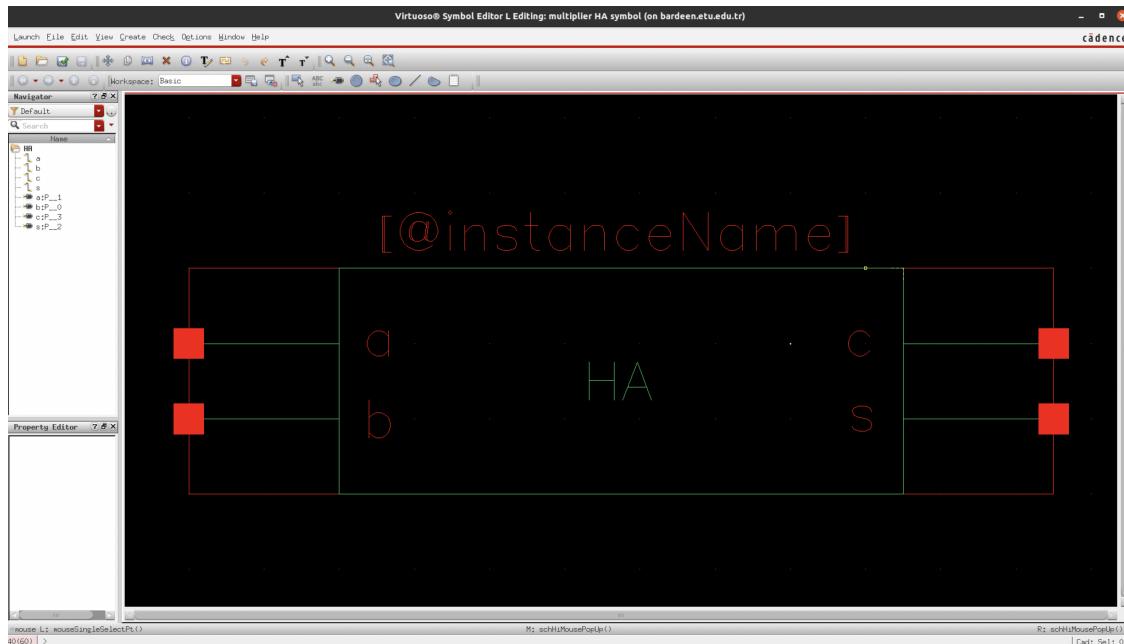
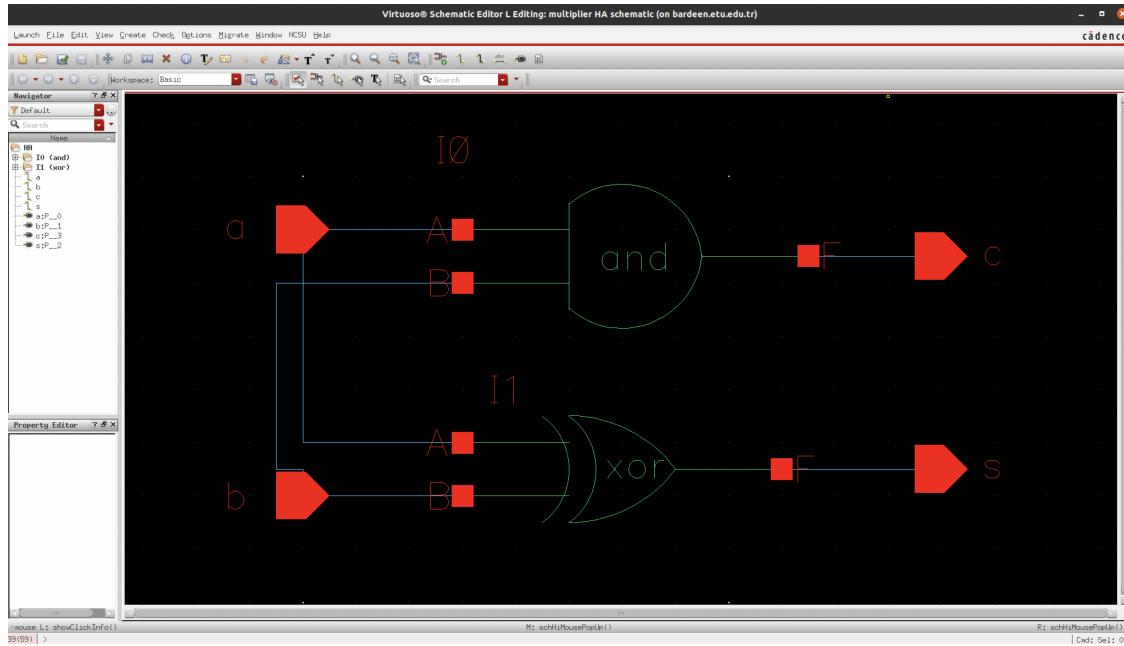
not layoutu dikeyde 14.1um, yatayda 7.2um uzunluğundadır ve $101,52\text{um}^2$ alana sahiptir.



4.7 Half Adder

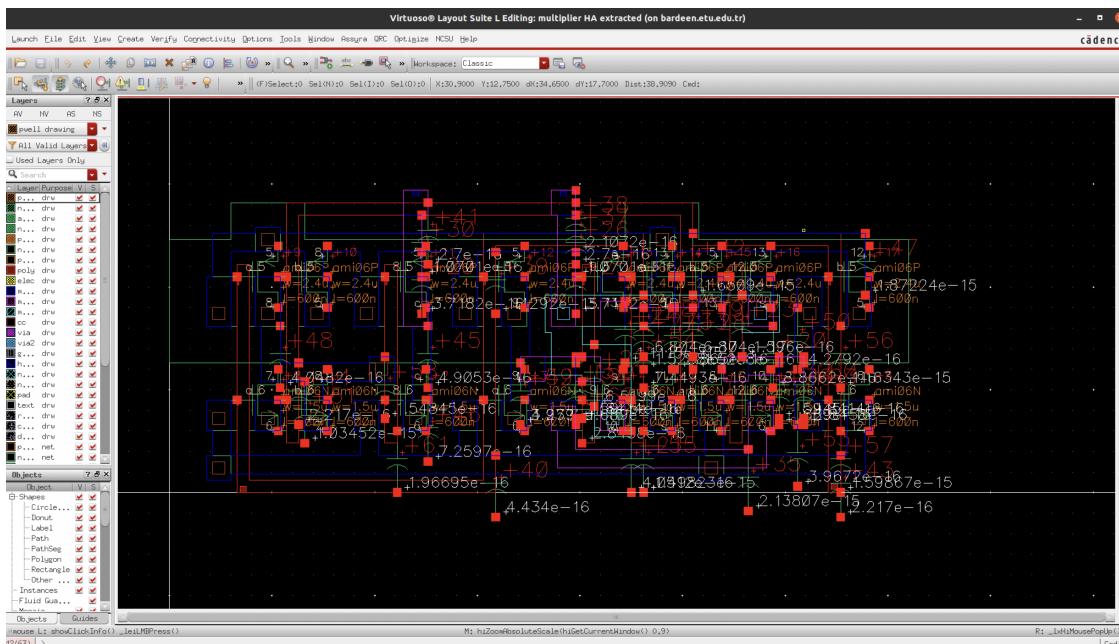
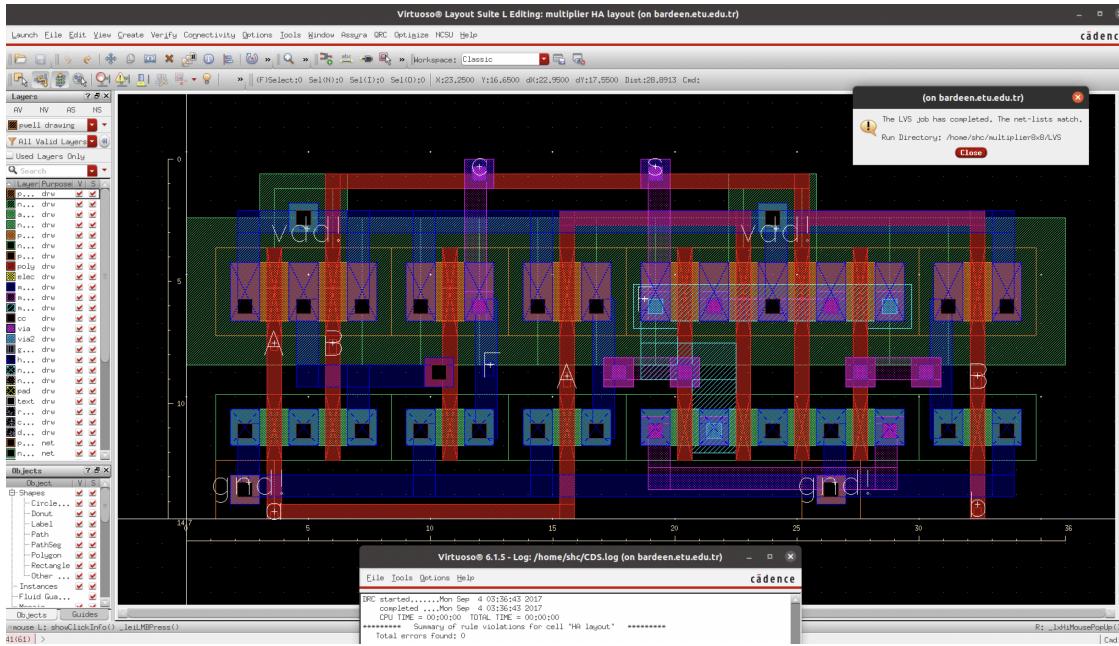
4.7.1 Half Adder Şematik ve Sembol

Half Adder tasarımda 1 and ve 1 xor kapısı vardır.



4.7.2 Half Adder Layout ve Extracted

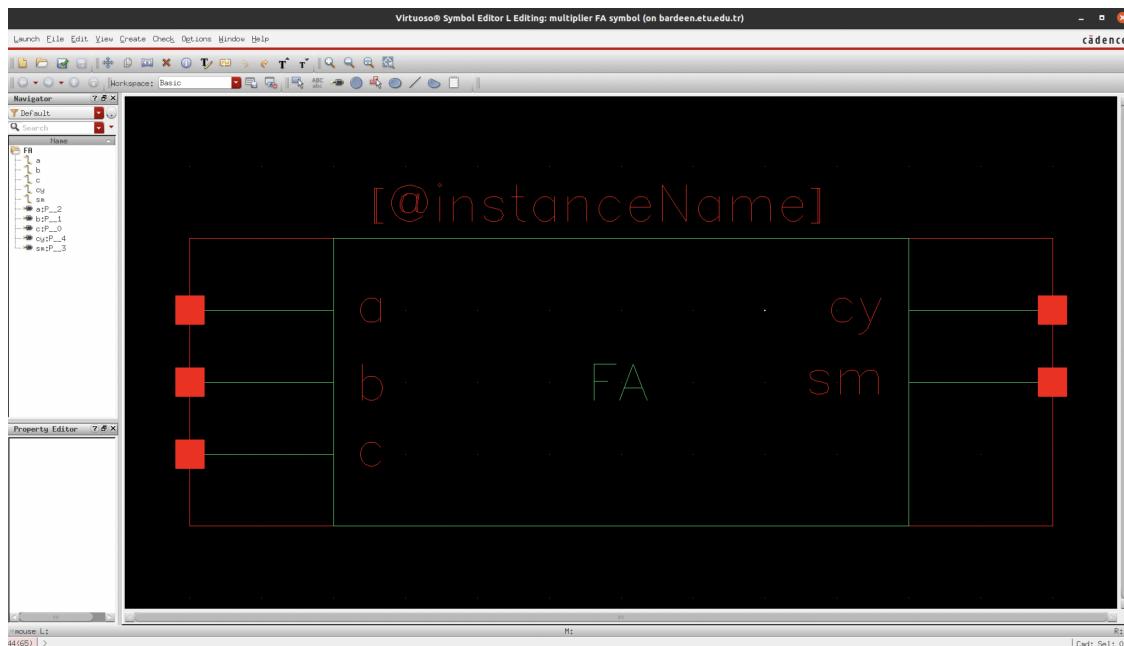
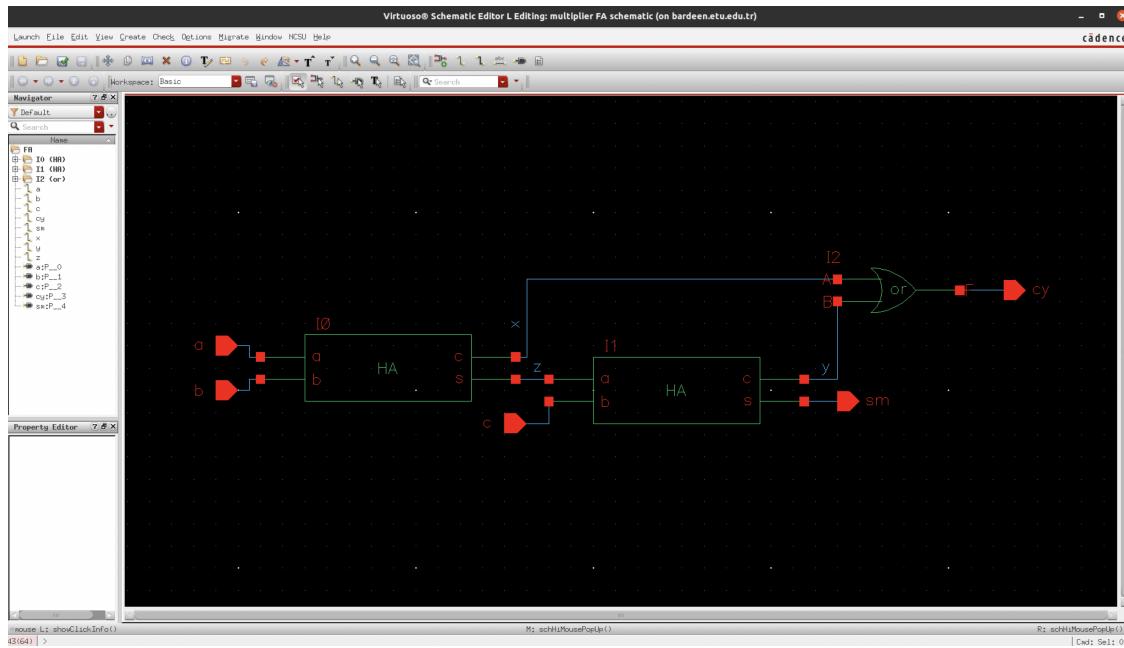
Half Adder layoutu dikeyde 14.7um, yatayda 36um uzunluğundadır ve 529.2um² alana sahiptir.



4.8 Full Adder

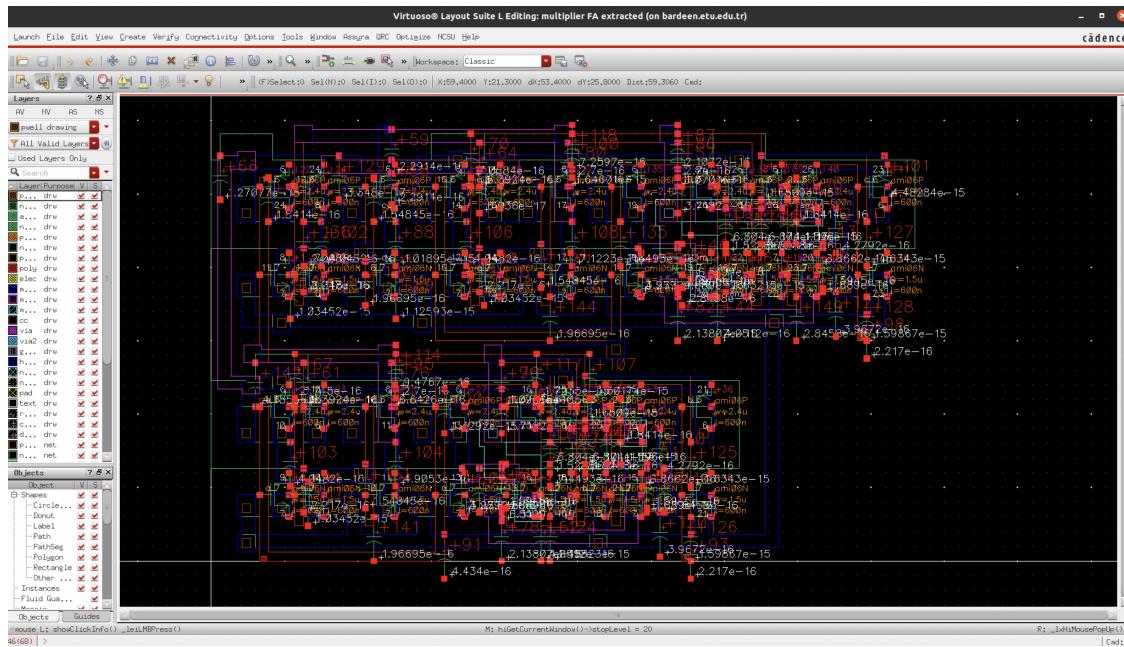
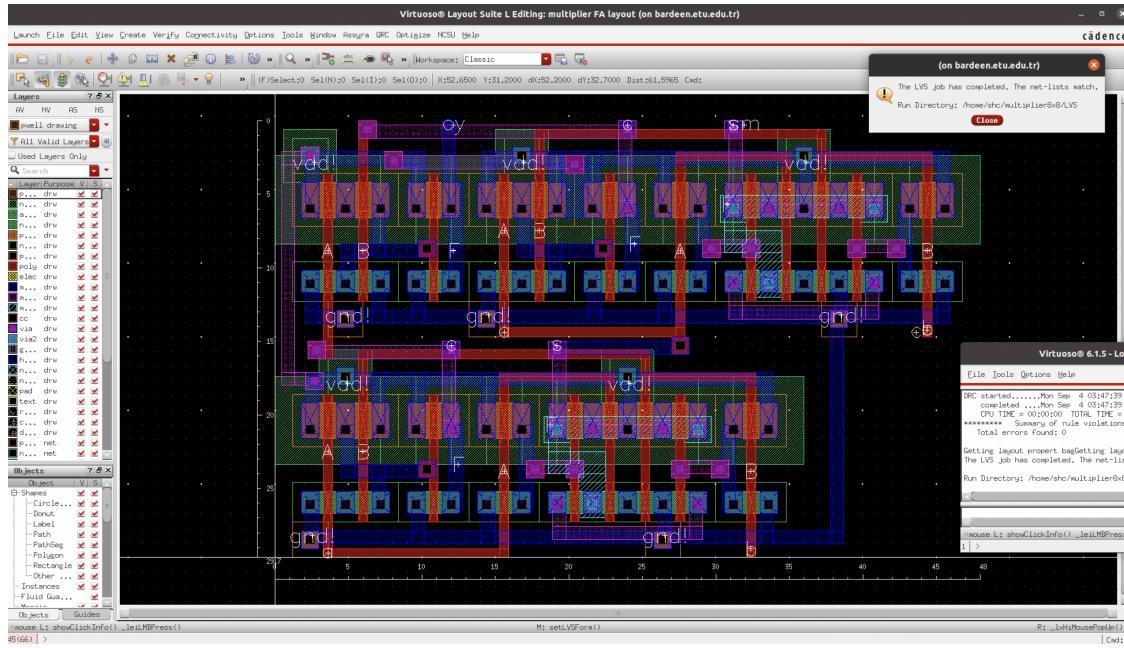
4.8.1 Full Adder Sematik ve Sembol

Full Adder tasarımda 2 Half Adder modülü ve 1 or kapısı vardır.



4.8.2 Full Adder Layout ve Extracted

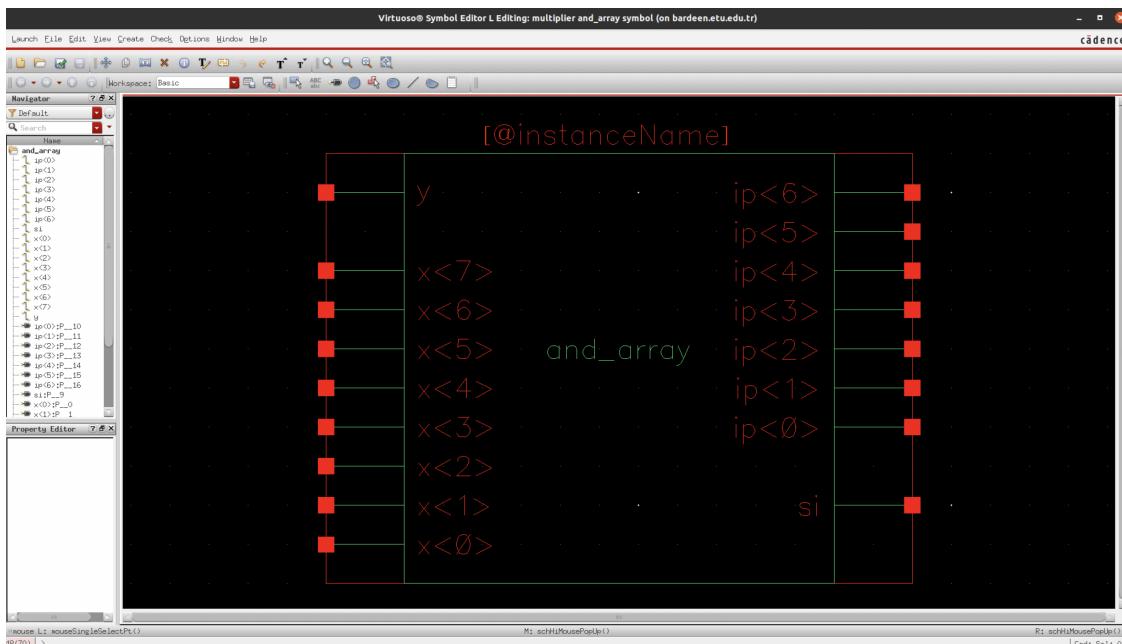
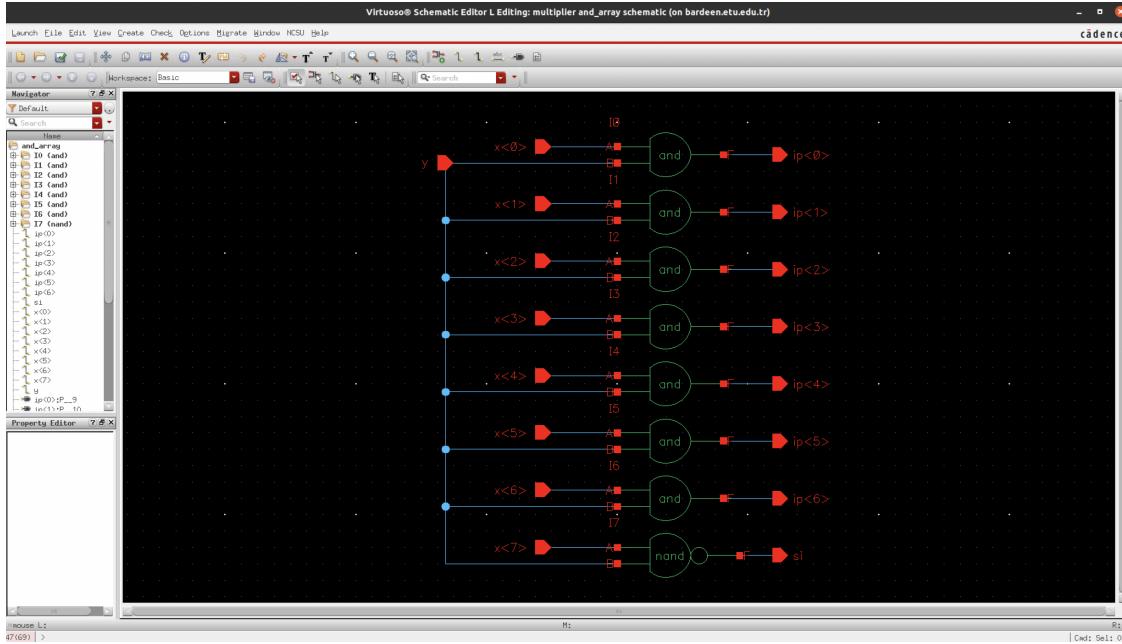
Full Adder layoutu dikeyde 29.7um, yatayda 48um uzunluğundadır ve 1425.6um^2 alana sahiptir.



4.9 And Array

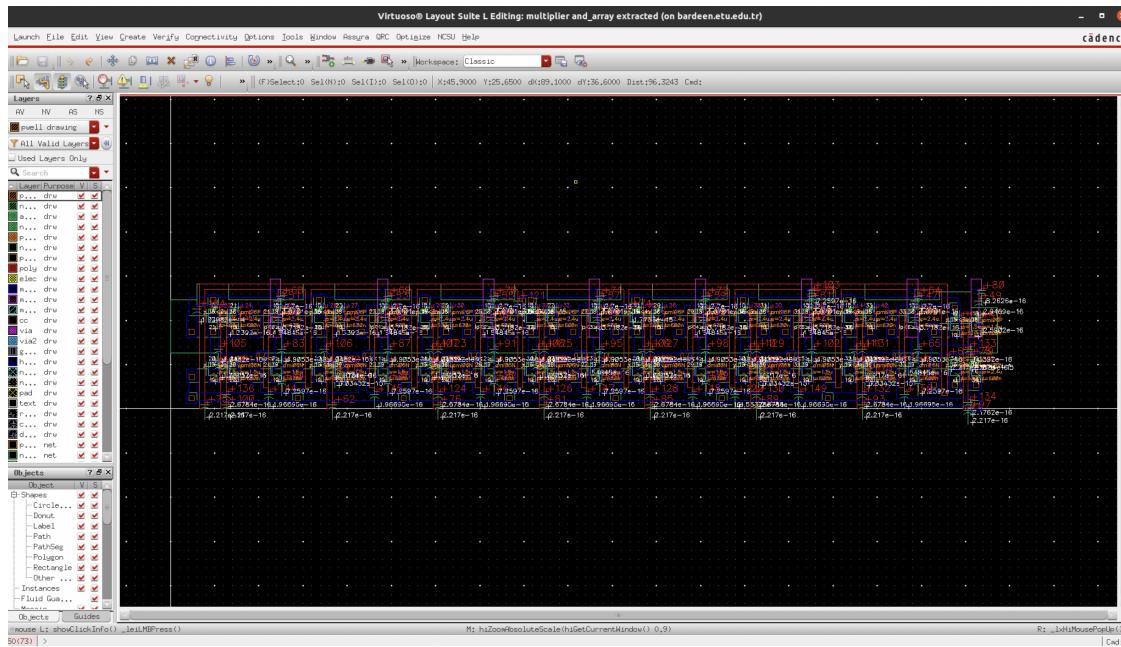
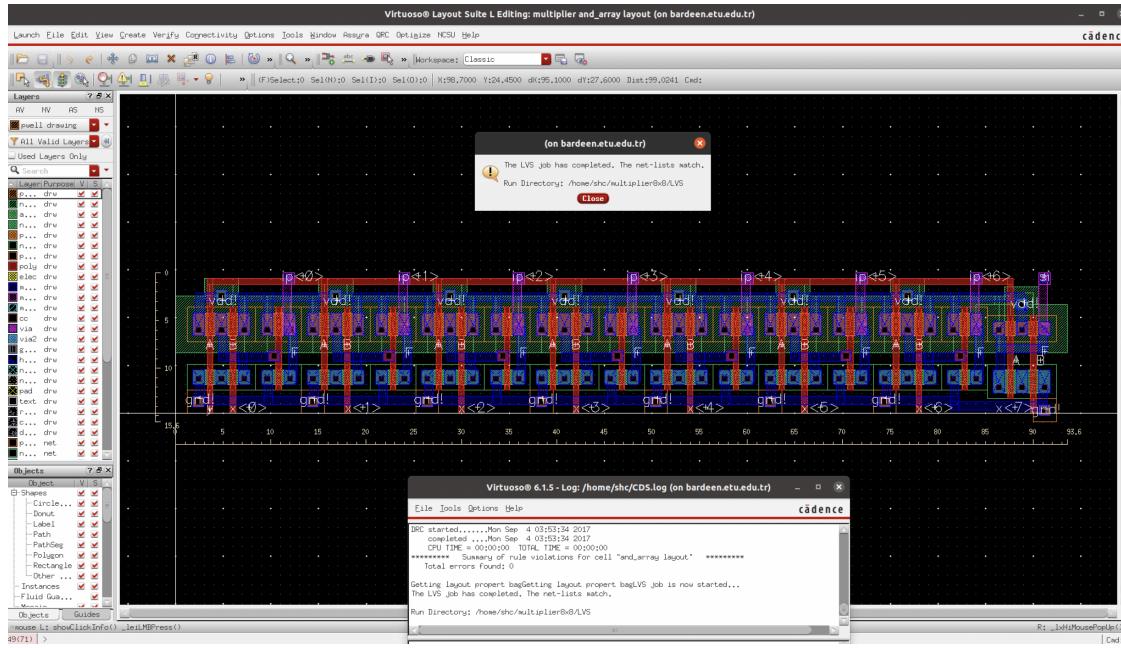
4.9.1 And Array Şematik ve Sembol

And Array tasarımında 7 and ve 1 nand kapısı vardır.



4.9.2 And Array Layout ve Extracted

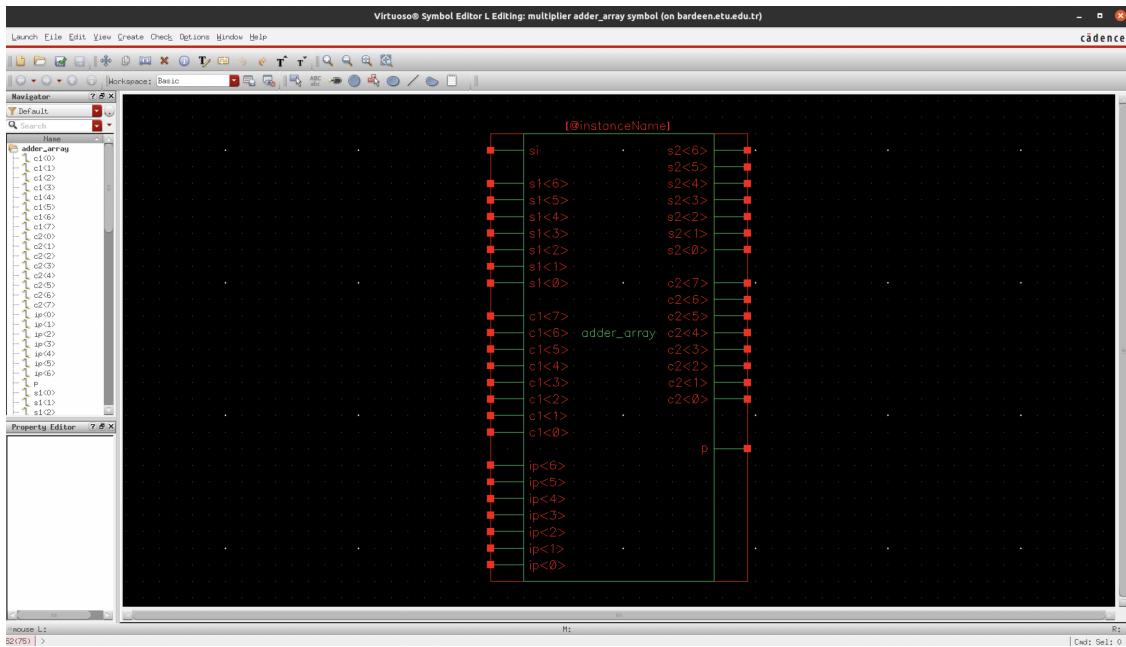
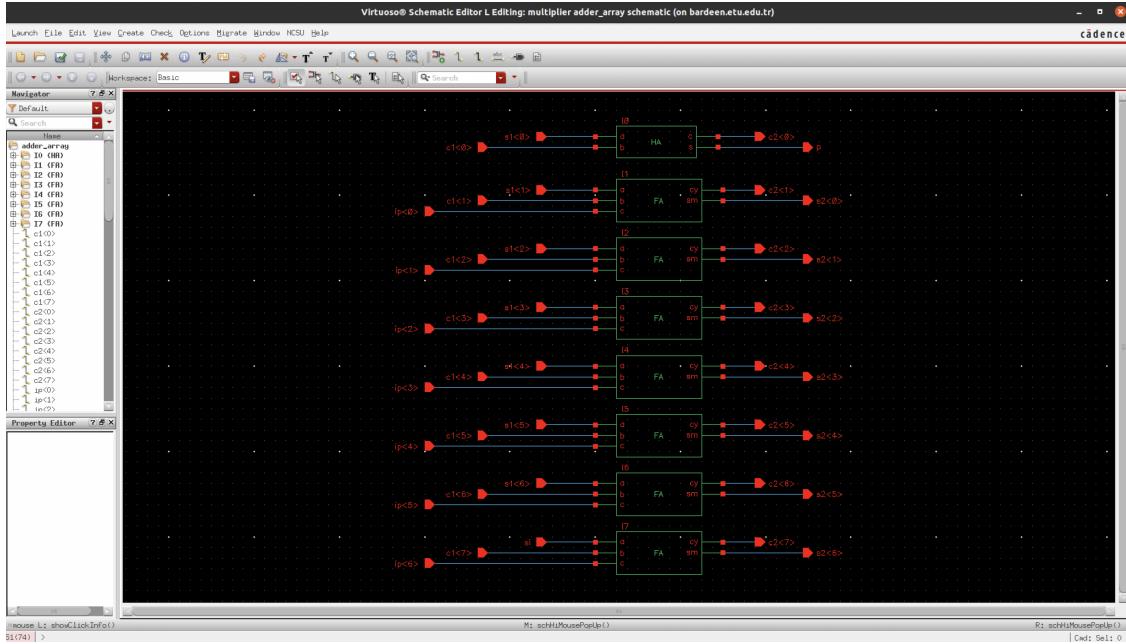
And Array layoutu dikeyde 15.6um, yatayda 93.6um uzunluğundadır ve 1460.16um^2 alana sahiptir.



4.10 Adder Array

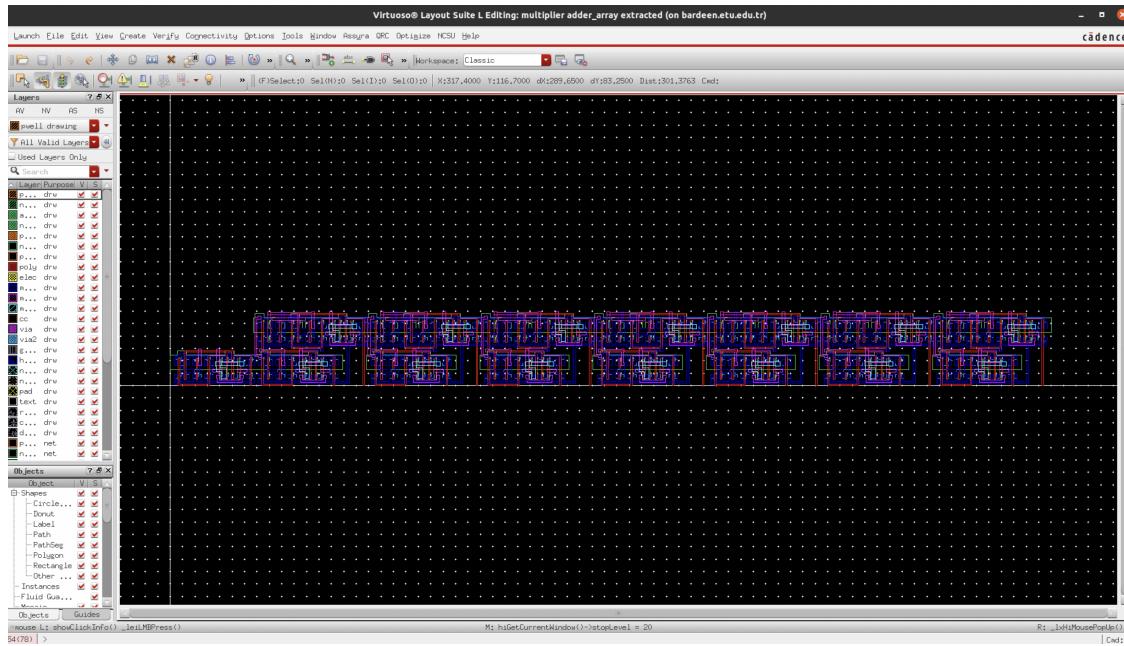
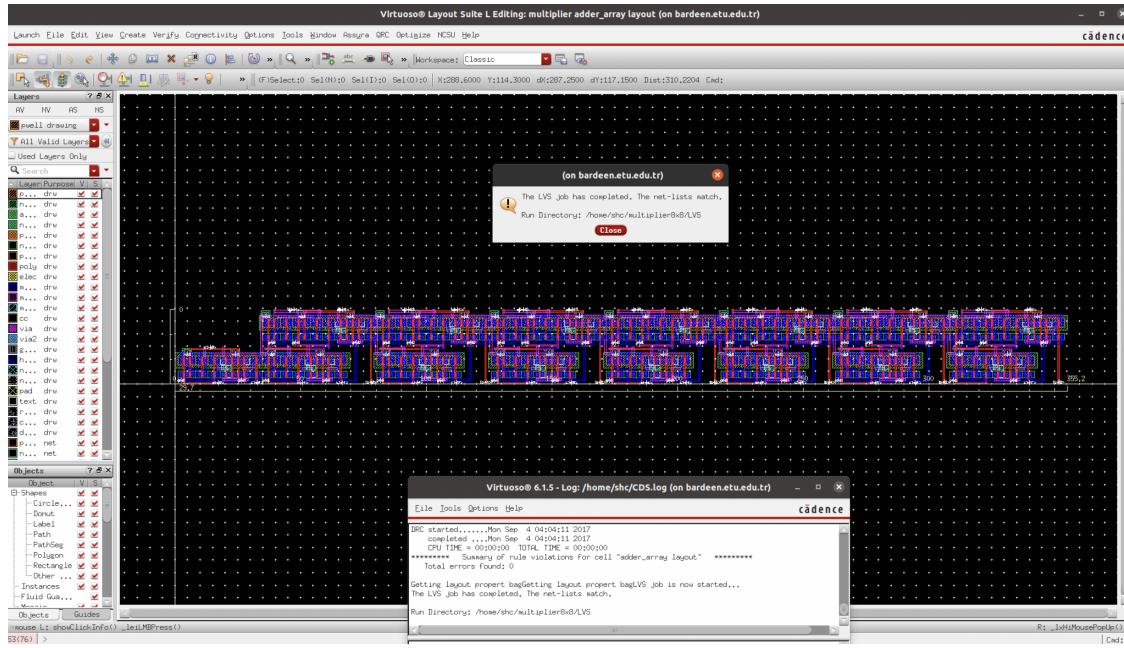
4.10.1 Adder Array Şematik ve Sembol

Adder Array tasarımda 7 Full Adder ve 1 Half Adder modülü vardır.



4.10.2 Adder Array Layout ve Extracted

Adder Array layoutu dikeyde 29.7um, yatayda 355.2um uzunluğundadır ve 10549.44um^2 alana sahip-tir.



5 Hybrid (Booth+Wallace) ve Wallace Tree Çarpıcı Devrelerinin Karşılaştırılması

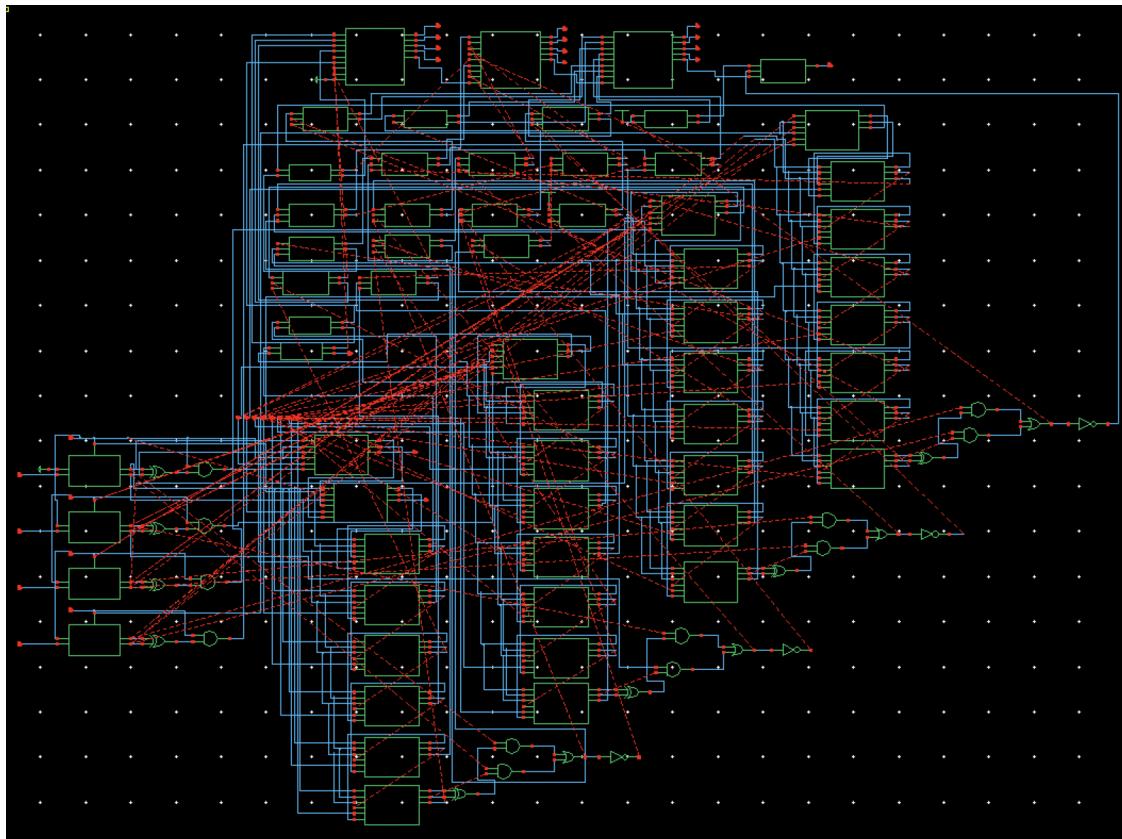
Olabildigince hızlı çarpıcı algoritmalarının kullanıldığı devreler araştırılmış ve bulunan devrelerden görece en hızlı olan Hybrid çarpıcıdan tasarıma başlanmıştır fakat ne yazık ki gecikme süresi az olan hızlı çarpıcıların devre yapısı oldukça karmaşıktır. Bu sebeple layoutunun elle çizilmesi imkansızı yakın olan Hybrid çarpıcı devresinden vazgeçilmiş, gecikme süresi çok da farklı olmayan ama daha çizilebilir olan Wallace Tree devresine geçilmiştir.

5.1 Hybrid Çarpıcı Devresi

Aşağıdaki linkte bulunan kapı seviyesindeki verilog kodu alt modüller parça parça çizildikten sonra Virtuoso'da şematiğe dökülmüştür.

[https://github.com/pareddy113/Design-of-various-multiplier-Array-Booth-Wallace
/blob/master/Hybrid%20multiplier/Hybrid%20multiplier.v](https://github.com/pareddy113/Design-of-various-multiplier-Array-Booth-Wallace/blob/master/Hybrid%20multiplier/Hybrid%20multiplier.v)

Aşağıda Virtuoso'da çizilen Hybrid Çarpıcı Devresi şematiği görülebilir.



Şekilden görüleceği üzere Hybrid çarpıcı devresi oldukça karmaşıktır ve layoutunun elle çizilemeyeceği düşünülüp bu devreden vazgeçilmiştir.

Ayrıca bu devrenin bir başka kullanılmama sebebi ise -128×-128 üç değer çarpım işleminde yanlış değer (0) vermesi. (Bu çizimden değil, verilog kodunda da böyle, sanırım kodu yazan kişi bu

algoritmada bir yeri kaçırmış ya da booth algoritmasında sayının önüne ekstra bir bit gerekmesinden dolayı olabilir.)

Yine bir diğer sebep ise bu karmaşıklığa rağmen transient analiz yükselme ve düşme zamanlarında, Wallace Tree devresinden sadece 0.002us kadar iyidir, bu kadar küçük bir fark için daha karmaşık bir layoutu çizmeye değer olmayıp vazgeçilmiştir.

6 Referanslar

- [1] Rani, Sona, et al. "Performance Analysis of Different 8x8 Bit CMOS Multiplier using 65nm Technology." International Journal of Computer Applications 148.13 (2016). <https://www.ijcaonline.org/archives/volume148/number13/rani-2016-ijca-911049.pdf>
- [2] Jeevitha, M., and R. Muthaiah. "VLSI Based Combined Multiplier Architecture." Journal of Artificial Intelligence 6.2 (2013): 145. <https://scialert.net/fulltext/?doi=jai.2013.145.153>
- [3] <https://github.com/SubZer0811/VLSI/tree/master/verilog/32bit-wallace-multiplier>
- [4] <https://github.com/KiranThomasCherian/VLSI-and-Computer-Architecture/tree/main/Verilog/32%20bit%20Wallace%20multipiler>
- [5] Singh, Khurajam Nelson, and H. Tarunkumar. "A review on various multipliers designs in VLSI." 2015 Annual IEEE India Conference (INDICON). IEEE, 2015. https://www.researchgate.net/publication/304230444_A_review_on_various_multipliers_designs_in_VLSI
- [6] <https://github.com/pareddy113/Design-of-various-multiplier-Array-Booth-Wallace>
- [7] Senthilpari, C., Ajay Kumar Singh, and K. Diwakar. "Low power and high speed 8x8 bit multiplier using non-clocked pass transistor logic." 2007 International Conference on Intelligent and Advanced Systems. IEEE, 2007. <https://ieeexplore.ieee.org/document/4658609>
- [8] Balamurugan, S., et al. "FPGA design and implementation of truncated multipliers using bypassing technique." Proceedings of the International Conference on Advances in Computing, Communications and Informatics. 2012. https://www.researchgate.net/publication/230688600_FPGA_Design_and_Implementation_of_Truncated_Multipliers_Using_Bypassing_Technique
- [9] Kumawat, Pradeep Kumar, and Gajendra Sujediya. "Design and Analysis of 8x8 Wallace Tree Multiplier using GDI and CMOS Technology." International Journal of Advanced Engineering Research and Science 4.7: 237204. https://ijaers.com/uploads/issue_files/3%20IJAERS-JUL-2017-9-Design%20and%20Analysis%20of%208x8%20Wallace%20Tree.pdf