İSTANBUL TEKNİK ÜNİVERSİTESİ
ELEKTRİK – ELEKTRONİK FAKÜLTESİ

BİLGİSAYAR ARİTMETİK ALGORİTMALARININ FPGA ÜZERİNDE GERÇEKLENMESİ

BİTİRME ÖDEVİ

Shenay RAMEZANI
040030907

Bolümü: Elektronik Mühendisliği
Programı: Elektronik Mühendisliği

Danışmanı: Yrd. Doç. Dr. Berna ÖRS YALÇIN

OCAK 2011

ÖNSÖZ

Bitirme ödevim boyunca bilgilerinden faydalandığım ve bana çok değerli zamanını ayırarak yardımcı olan Sayın Yrd. Doç. Dr. Berna ÖRS YALÇIN'a ve eğitimim süresi boyunca bana destek olan aileme sonsuz saygı ve teşekkürlerimi sunarım.

Mayıs 2008 Shenay RAMEZANI

İÇİNDEKİLER

ÖZET **iv**
SUMMARY **v**

1. GİRİŞ 1

2. ÇOK HIZLI TÜMLEŞİK DEVRE DONANIM TANIMLAMA DİLİ (VHDL) **3**
2.1 Giriş 3
2.2 VHDL'in Avantajları 2
2.3 Tasarım 4
2.4 Benzetim 5

3. TOPLAMA DEVRELERİ **6**
3.1. Yarım Toplayıcı 7
3.2. Tam Toplayıcı 9
3.3. Elde Zincirli Toplayıcı 12
3.4. Elde Öngörülü Toplayıcı 14
3.5. Koşullu Toplayıcı 17
3.6. Elde Seçici Toplayıcı 23
3.7. Toplayıcı Devreleri'nin Karşılaştırılması 27

4. Çarpma Devreleri **22**
4.1. Ardışıl Çarpıcı 23
4.2. Dizin Çarpıcı 24
4.3. Çarpıcı Devreleri'nin Karşılaştırılması 26

5. Sonuçlar ve Tartışma **27**

KAYNAKLAR **23**

ÖZGECMİŞ **29**

ÖZET

Bu çalışmada, temel aritmetik işlem olarak pek çok kullanım alanı bulunan toplama ve çarpma işlemleri için geliştirilmiş klasik algoritmalar incelenmiştir. Bu algoritmalara dayanarak geliştirilen aritmetik devrelerin tasarımı, VHDL kullanılarak XILINX ISE 11 programı ile gerçekleştirilmiştir. Ele alınan aritmetik toplama ve çarpma devrelerinin gecikme ve alan değerleri ve benzetimleri elde edilmiş ve devrelerin performanslarının kıyaslanabilmesi için, tablo halinde verilmiştir. Bu tablodaki sonuçlar yardımıyla, daha sonra yapılacak tasarımlarda gereken performans göz önünde bulundurularak hangi çarpma veya toplama devresinin uygun olduğuna karar verilebilir.

Birinci bölümde konu hakkında genel bilgiler verilmiştir.

İkinci bölümde devreleri tasarlamada kullanılan ÇOK HIZLI TÜMLEŞİK DEVRE DONANIM TANIMLAMA DİLİ olan VHDL (Very high speed integrated circuit Hardware Description Language) hakkında kısa bir tanımlama yapılmıştır.

Üçüncü bölümde temel toplama üniteleri hakkında bilgi verilmiştir ve Elde Zincirli Toplama, Elde Öngörülü Toplama, Koşullu Toplama ve Elde Seçici Toplama algoritmalarına göre toplama devreleri tasarlanmıştır. Tasarlanan bu devrelerin performanslarını karşılaştırmak amaçlı, gecikme ve alan bilgileri tablo olarak verilmiştir.

Dördüncü bölümde çarpma devreleri incelenmiştir. Ardışıl Çarpma ve dizin çarpma algoritmalarına göre çarpma devreleri tasarlanmıştır. XILINX ISE 11 programı kullanılarak benzetimleri gerçeklenmiştir. Gecikme ve alan bilgileri elde edilmiştir.

Beşinci ve son bölümde ise önceki bölümlerden elde edilen sonuçlar tartışılmıtır.

SUMMARY

In this project, classical algorithms developed for basic arithmetic operations; addition and multiplication which have many applications, are inspected. Arithmetic circuits are designed according to different algorithms using VHDL and XILINX ISE 11. Delay and area information about designed arithmetic circuits and their simulations have been achieved and given as a table in order to compare their performances. The designer may use this table in future designs to decide about which addition or multiplication circuit to use.

In chapter one, general information abut topic is given.

In chapter two, there is a breif introduction about VHDL (Very high speed integrated circuit Hardware Description Language) which is used in this project to design arithmetic circuits.

In chapter three, after giving some information about basic addition units, adder circuits based on ripple carry, carry look ahead, conditional sum and carry select adder algorithms are designed. Area and delay information of these circuits are given as a table in order to compare their performances.

In chapter four, multiplier curcuit based on sequetntial and array multiplier algorithms are designed. Area and delay information of designed multiplier circuits are calculated and their simulation is generated using XILINX ISE 11.

In chaptre five which is the last chapter, the results achieved in previous chapters are discussed.

1. GİRİŞ

1947'de transistörün icat edilmesi ile başlayan ve tümdevre teknolojilerinin ortaya çıkması ile hız kazanan "katı ortam elektroniği" yayılganlık ve üretkenlik bakımından teknoloji tarihinde benzeri görülmemiş bir konum kazanmıştır. Daha çok "mikroelektronik" adı ile anılan bu geniş ve dinamik teknoloji alanı, dünyada son 50 yılda üretilmiş olan yeniliklerin ve sağlanmış olan gelişmelerin büyük çoğunluğunun kaynağdır [1].

Çok büyük ölçekli tümdevre (VLSI-very large scale integration) tasarım yöntemleri hızla gelişen elektronik endüstrisi için temel bir çözüm haline gelmiştir. VLSI devre tasarım yöntemleri ile tasarlanan tümdevreler, düşük maliyetli olup çok hızlı ve güvenli çalışabilmekte ve çok küçük alanlara sığabildikleri için milyonlarca transistörün tek bir yonga içinde gerçekleştirilmesine olanak sağlamaktadır. CMOS (complomentary metaloxide semiconductor) VLSI devre tasarım yöntemi, sayısal mantık devreleri tasarımına dayandığı için, karmaşık sistemlerin tasarımı ve gerçekleştirilmelerini de kolaylaştırmaktadır, ayrıca CMOS devreler yüksek gürültü marjinine sahip olduklarından daha güvenli çalışabilmektedirler. CMOS devrelerin bir başka önemli özelliği de çok az güç harcamalarıdır. Bütün bu özelliklerin sonucunda CMOS VLSI devre tasarım yöntemleri ve bu teknolojideki gelişmeler, 1971'de ilk mikroişlemcinin üretilmesiyle başlayıp günümüzün süper mikrobilgisayarlarına kadar uzanan ve mikroişlemci tasarımlarında yaşanan çok hızlı ve büyük gelişmelerin itici gücü olmuştur [2].

VLSI devrelerinin optimizasyonu farklı açılardan gerçekleştirilebilir, fakat bazı faktörler birlikte optimize edilemez ve bir faktörün iyileştirirlmesi, diğer faktörün kötüleşmesine sebep olabilir. Günümüzde hız ve alan faktörlerinin optimizasyonu büyük bir önem taşımaktadır [3].

Bu çalışmada FIR filtre, FFT, Konvolüsyon ve diğer önemli sayısal işaret işleme (DSP) yapılarında sıklıkla kullanılan toplama ve çarpma devrelerinin hız ve alan optimizasyonu gerçekleştirilmektedir. Bu amaçla farklı algoritmalara dayanan toplama ve çarpma devreleri, elektronik devrelerin tanımlanmasında kullanılan ve IEEE standart tanımlama dili olan VHDL ile modellenerek FPGA (sahada programlanabilir kapı dizileri) üzerinde gerçeklenmiştir. FPGA içinde, mantık kapıları ve flip-floplardan oluşan lojik birimler, kolon veya matrislerde sıralanmıştır. Birimler arasındaki programlanabilir bağlantı ağları sayesinde lojik birimler diğer lojik birimlerle birleşerek daha çok birim ile daha karmaşık işlemler gerçekleştirebilir.

Sonraki bölümlerde VHDL hakkında bilgi verildikten sonra, farklı algoritmalara göre modellenen toplama ve çarpma devreleri XILING ISE 11 programı kullanılarak FPGA üzerinde gerçeklenerek alan ve gecikme bilgileri elde edilip kıyaslanacaktır.

2. ÇOK HIZLI TÜMLEŞİK DEVRE DONANIM TANIMLAMA DİLİ (VHDL)

2.1. Giriş

Elektronik sistemlerdeki teknolojinin ilerlemesi, tasarım yöntemlerinin de gelişmesini gerektirmiştir. Bu sebeple, geleneksel "kağıt ve kalem kullanarak tasarımı yap" ve "devreyi kurarak denemeleri yap" yöntemlerinin yerini "tanımla ve sentezle" yöntemleri almıştır. Tanımlama ve sentezleme yöntemleri, donanım tanımlama dillerinin (Hardware Description Language) ortaya çıkmasına neden olmuştur. VHDL (Very high speed integrated circuit Hardware Description Language) günümüzdeki donanım tanımlama dillerinin en popüleridir [4].

2.2.VHDL'in Avantajları

VHDL, büyük ölçekli dijital tasarımların dokümantasyonu, doğrulanması ve sentezlenmesi işlemlerini gerçekleştirmek için kullanılır. Aynı VHDL kodları ile bu üç farklı işlemin gerçekleştirebilmesi sayesinde, tasarım süresi ve kolaylığı açısından büyük kolaylıklar sağlamaktadır [5]. VHDL'in avantajları aşağıda verilmiştir.

Güç ve Esneklik: VHDL tasarımları yazılım tabanlı olduğundan dolayı bir çok fonksiyon tek bir donanım ile gerçekleştirilebilir.

Çipten Bağımsız Tasarım: Özel işlevleri yerine getiren bir çok entegrenin VHDL ile gerçeklenmesi mümkün olduğu için tasarımlarımız çipten bağımsızdır.

Taşınabilirlik: Yapılan tasarım bir dosya olduğundan mevcut başka bir tasarım dosyasının içine veya bir çipin içine eklenebilir. Böylelikle yapılan her tasarım farklı bir modül olarak düşünülebilir.

Test Edilebilirlik: Tasarımı yapılan VHDL devreleri emülatörler yardımıyla gerçek zamanlı olarak, simülatörler yardımıyla da sanal olarak test edilebilir.

Piyasaya Hızlı Çıkış ve Düşük Maliyet : Mevcut bir sistem üzerinde yapılması istenen bir değişiklik, donanım olarak değil de yazılım olarak yapıldığından tasarım süresi kısa sürmektedir ayrıca bir çok entegre, VHDL çipleri ile tasarlanabildiğinden tasarım maliyeti düşüktür.

2.3.Tasarım

Sayısal bir sistemin VHDL tanımlaması yapılırken dört temel yapı kullanılır. Bu yapılar entity bölümü, mimari bölümü, konfigürasyon ve paketlerdir. Sayısal bir sistem, modüllerin bir hiyerarşi ile birleştirilmesiyle oluşturulur. Her modül VHDL de bir entity ile ifade edilir. Her entity ile giriş çıkışları ve fonksiyonu tamamen belirlenmiş olan bir donanım yapısı gösterilir. Her tanımlamanın entity, bildirimi ve mimari bölümleri olmak üzere iki bölümü vardır. Entity bildirimi tasarımın dış bağlantılarının, mimari bölümü ise, içeride yapılacak işlemlerin gösterilmesinde kullanılır. Pakette ise pek çok tanımlamada ortak olarak kullanılacak genel bilgiler verilir. Konfigürasyon ile yapısal tanımlamada kullanılacak alt sistemlerin giriş çıkış bilgileri tanımlanır [4],[7]. Tablo 2.1'de entity ve mimari bölümlerinden oluşan bir yarım toplayıcı'nın VHDL kullanılarak tanımlanması yer almaktadır.

Tablo 2. 1: Bir yarım toplayıcı'nın VHDL kullanılarak tanımlanması.

entity	entity half_adder is PORT(a: in std_logic; b: in std_logic; s: out std_logic; c: out std_logic); end half_adder;

mimari	architecture arch of half_adder is
	begin
	s <= a xor b;
	c <= a and b;
	end arch;

2.4.Benzetim

VHDL'de yazılan bir tasarım tanımının, bir VHDL benzetim programından faydalanılarak doğruluğu kontrol edilebilir. Tanımlamanın benzetimini yapabilmek için benzetim programına bir takım sayısal girişleri verilir, program daha önceden belirlenen aralıklarla bu sayısal girişleri tasarımın modeline uygular ve çıkışları üretir. Bu sonuçlar tasarımcı tarafından gözlenerek modelin istenildiği gibi çalışıp çalışmadığına karar verilir [7].

Benzetim, tasarımın her aşamasında kullanılabilir. Tasarımın yüksek seviyelerinde yapılan benzetim, sadece tasarımın fonksiyonel davranışı hakkında bilgi verir. Bu aşamada benzetim çok hızlıdır, fakat benzetim sonuçlarından tasarımın gerçek devre elemanları ile çalışması ve zamanlaması konusunda çok fazla bilgi elde edilemez. Daha düşük tasarım aşamalarına gidildikçe benzetim daha uzun sürecektir, fakat benzetim sonuçlarından tasarımın çalışması ve zamanlaması konusunda daha fazla bilgi elde edilebilir [4].

3. TOPLAMA DEVRELERİ

İki sayının toplamı neredeyse tüm aritmetik ünitelerde en çok kullanılan işlemdir. Toplama devreleri, toplama ve çıkartma işlemlerinin yanısıra, çarpma ve bölme gibi daha karmaşık işlemlerde de kullanılır. Toplama devreleri bir çok elektronik uygulamasında da kullanılmaktadır ve aritmetik ünitelerde hız belirleyici unsurdur. İki tabanındaki iki sayının toplamı, sayısal işaret işlemede, tümleşik devrelerde ve mikroişlemcilerde en temel aritmetik işlem olmaktadır. Toplama devrelerinde üç temel faktör, devrenin hızı, kapsadığı alan ve harcadığı güçtür. Yüksek hızlı paralel toplayıcılar için bir çok algoritma tasarlanmıştır ve genellikle hız ve kapsadıkları alan arasında ters orantı vardır. Sonuç olarak toplama devreleri VLSI devrelerinin en temel yapı bloklarından biri olmaktadır.

Toplama devrelerinde hız optimizasyonu için en temel faktör elde oluşumudur. Oluşan elde mümkün olduğu kadar kısa sürede çıkışa ulaşmalıdır. Bu süre devrenin gecikmesi olarak adlandırılır. Devrenin gecikmesini azaltmak için ödenen bedel ise daha çok lojik kapı kullanmak ve sonuç olarak devrenin kapsadığı alanın artması olacaktır. Bu bölümde temel toplama üniteleri olan Yarım ve Tam Toplaycılar hakkında bilgi verdikten sonra Elde Zincirli Toplama, Elde Öngörülü Toplama, Koşullu Toplama ve Elde Seçici Toplama algoritmalarına göre tasarlanan toplama devrelerini inceleyip, hız ve kapsadıkları alanları kıyaslayacağız. Bu verileri elde etmek için Xilinx Ise 11 programında gerekli VHDL kodlar yazılarak FPGA üzerinde devreler gerçeklenemiştir.

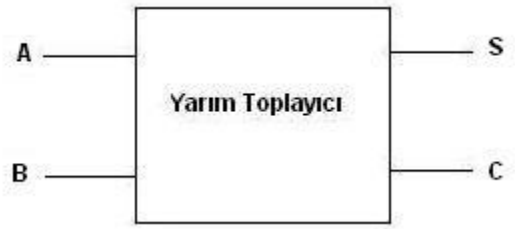
3.1.Yarım Toplayıcı

Toplama devrelerinin en temel elemanı, yarım toplayıcıdır. Bu devre, ikilik tabanında verilen iki tek bitlik sayıyı toplamaktadır. Yarım toplayıcı devresi tek başına çok faydalı olmasa da, daha büyük toplama devrelerinin alt bloğu olarak kullanılır. Devrenin çalışma şeklini daha iyi anlamak için ikilik tabanda toplama kurallarını göz önüne alalım;

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

Yarım toplayıcı devresi, giriş bitlerini toplar ve çıkışı sonuç ve elde biti olarak gösterir. Devrenin blok diyagramı şekil 3.1'de gösterilmektedir [6].

(A) Birinci giriş
(B) İkinci giriş
(S) Toplama biti
(C) Elde biti



Şekil 3. 1: Yarım toplayıcı devresinin blok diyagramı.

Çıkış değerlerinin ifadesi, yarım toplayıcının tablo 3.1'deki doğruluk tablosundan yararlanarak (3.1) ve (3.2) numaralı denklemlerde görüldüğü gibi ifade edilmektedir.

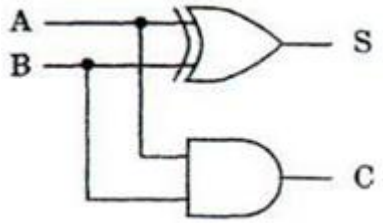
Tablo 3. 1: Yarım toplayıcının doğruluk tablosu.

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$S = A \oplus B$ (3.1)

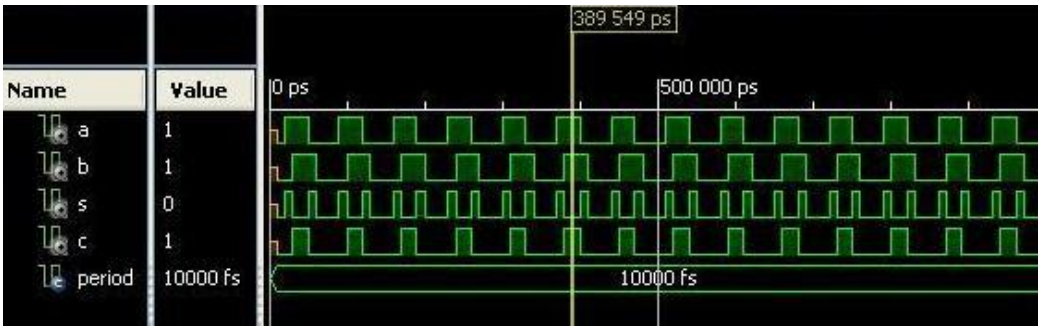
$C = A.B$ (3.2)

Bu sonuçlardan yararlanarak devreyi şekil 3.2'deki gibi tasarlayabiliriz.



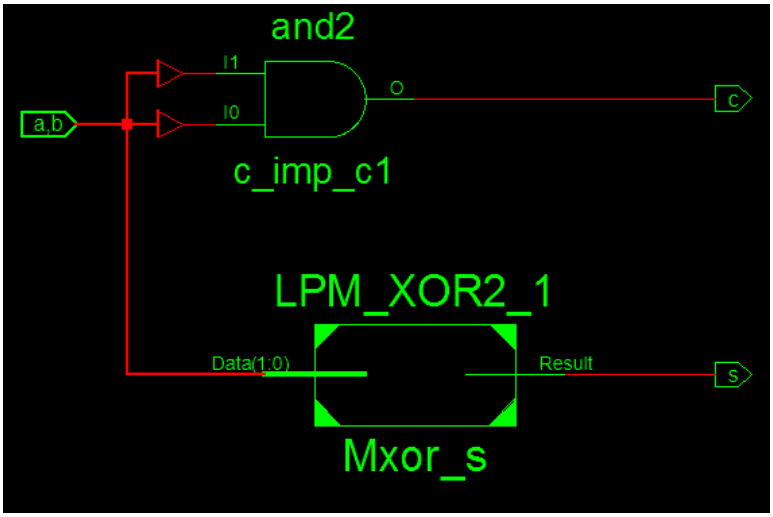
Şekil 3. 2: Yarım toplayıcı devresi.

Xilinx Ise11 programında, devreye uygun VHDL kodlar yazılıp, girişlere rastgele sayısal değerler verilerek devrenin doğru çalışıp çalışmadığını belirlemek için benzetim yapılmıştır. Şekilde 3.3'de görüldüğü gibi, sonuç beklentimize uygun çıkmıştır.



Şekil 3. 3: Yarım toplayıcının Xilinx Ise ile oluşturulmuş benzetimi.

VHDL ile tanımlanan yarım toplayıcının devre şematiği Xilinx Ise11 kullanılarak oluşturulmuş ve şekil 3.4'de görüldüğü gibi beklentilere uygun olarak çıkmıştır.



Şekil 3. 4: Yarım toplayıcının Xilinx Ise ile oluşturulmuş devre şematiği.

3.2.Tam Toplayıcı

Tam toplayıcı, iki tabanında üç biti toplamak için kullanılan kombinezonsal bir devredir. Tablo 3.2'de tam toplayıcı'nın doğruluk tablosu verilmiştir.

Tablo 3. 2: Tam toplayıcı'nın doğruluk tablosu

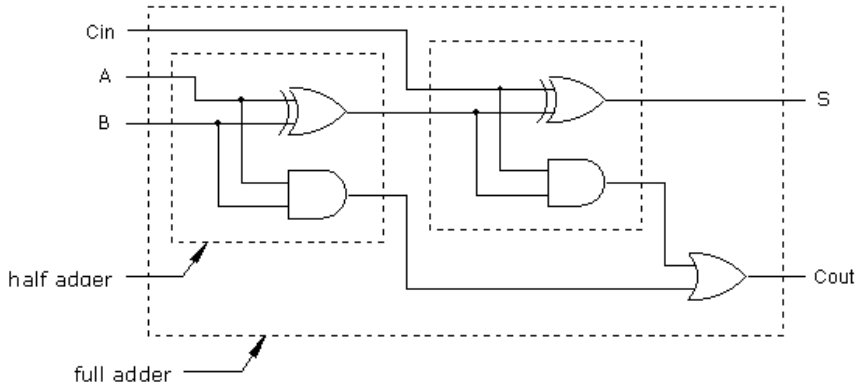
A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Bu doğruluk tablosundan yararlanarak , A ve B giriş biti, Cin bir önceki kattan gelen elde biti, S sonuç ve Cout elde biti olmak üzere tam toplayıcı'nın işlevi (3.3) ve (3.4) numaralı denklemlerde görüldüğü gibi ifade edilmektedir [6].

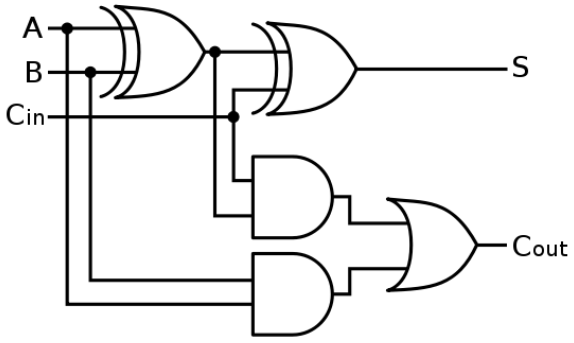
$$S = Cin \oplus (A \oplus B) \quad (3.3)$$

$$Cout = A.B + Cin.(A \oplus B) \quad (3.4)$$

Tam toplayıcı devresi şekil 3.5'de gösterildiği gibi iki paket halinde yarım toplayıcı devresi kullanarak veya şekil 3.6'da gösterildiği gibi (3.3) ve (3.4) numaralı bağlantılara uygun, ayrı bir devre olarak tasarlanabilir.

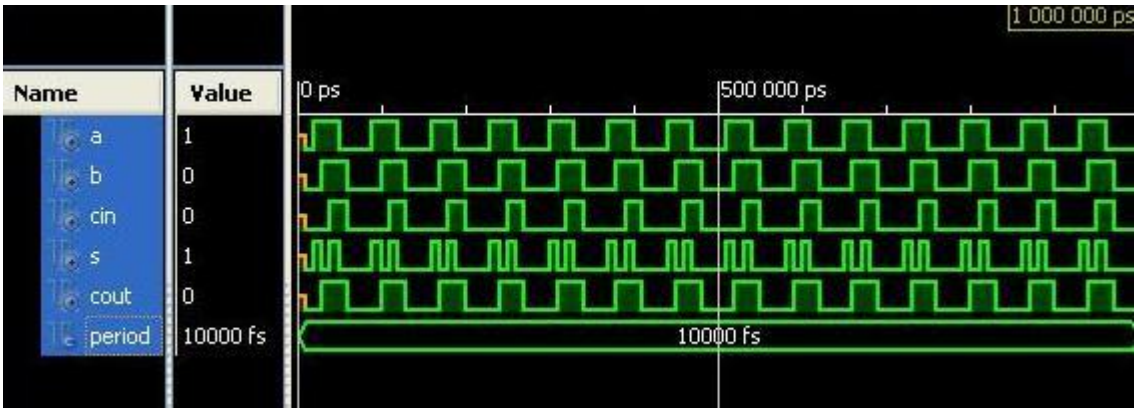


Şekil 3. 5: İki yarım toplayıcı devresinden oluşan tam toplayıcı.



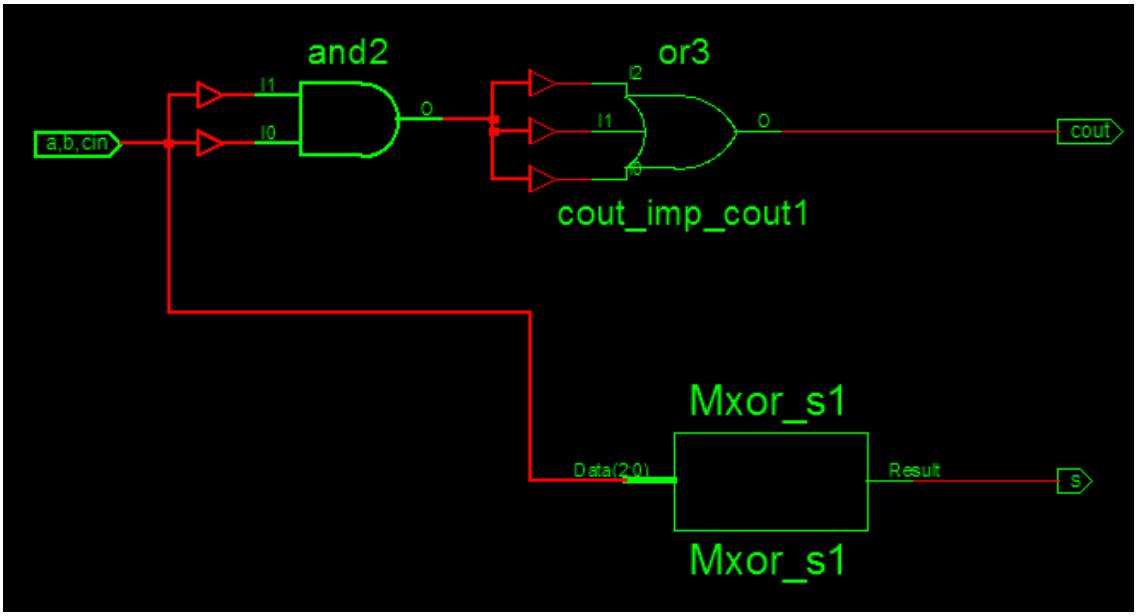
Şekil 3. 6: Tam toplayıcı devresi.

Xilinx Ise11 programında devreye uygun VHDL kodlar yazılıp, girişlere rastgele sayısal değerler verilerek devrenin doğru çalışıp çalışmadığını belirlemek için benzetim yapılmıştır. Şekil 3.7'de de görüldüğü gibi, sonuç beklentilere uygun çıkmıştır.



Şekil 3. 7: Tam toplayıcının Xilinx Ise ile oluşturulmuş benzetimi.

VHDL ile tanımlanan tam toplayıcı'nın devre şematiği, Xilinx Ise11 kullanılarak oluşturulmuş ve şekil 3.8'de görüldüğü gibi beklentilere uygun olarak çıkmıştır.

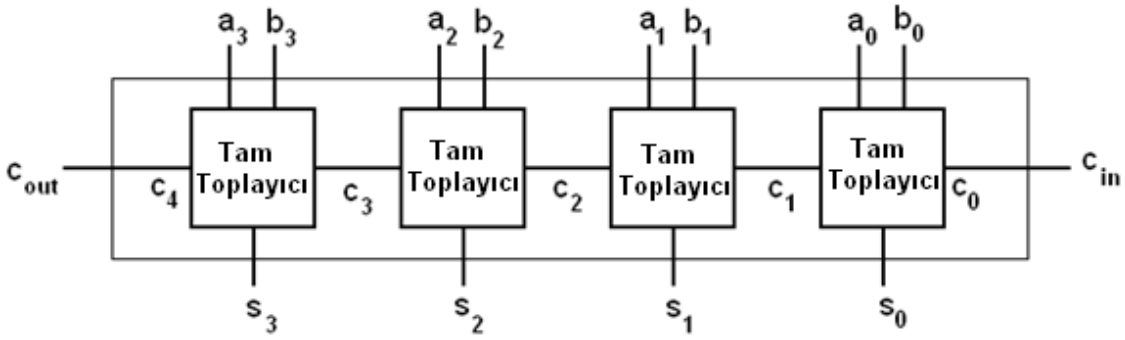


Şekil 3. 8: Tam toplayıcının Xilinx Ise ile oluşturulmuş devre şematiği.

3.3.Elde Zincirli Toplayıcı

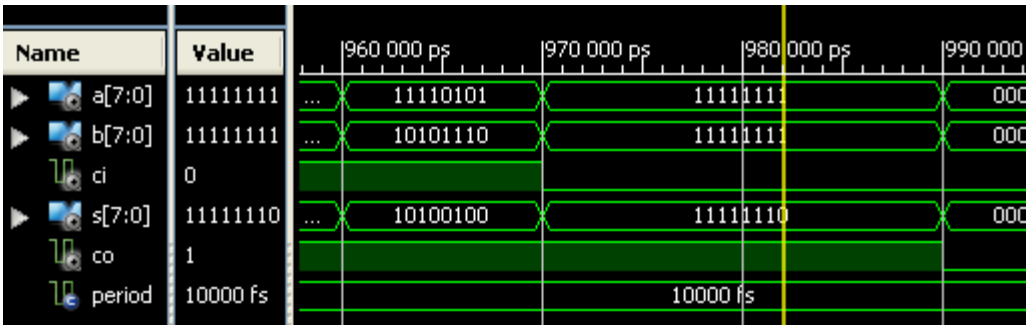
n-bitlik bir toplayıcı, n tane bir bitlik toplayıcıyı kaskad bağlanarak yapılabilir. Bu tür toplayıcılara "Elde Zincirli Toplayıcı" ismi verilmektedir. Şekil 3.9'da 4-bitlik bir elde zincirli toplayıcı devresinin blok diyagramı verilmiştir. Bu devrede, her katın elde çıkış (Cout) sinyali bir sonraki katın elde giriş (Cin) sinyalidir, bu nedenle toplama'nın sonucu olan S sinyali, elde sinyallerinin oluşumuna bağlı olacaktır. Toplamanın sonucunun doğru bir şekilde oluşması için, her katın elde sinyali oluşmuş olmalı ve bu da daha fazla gecikme anlamına gelmektedir.

Elde zincirli toplayıcı, çok basit bir toplama devresidir ve bit uzunluğu fazla olan sayılar için kolayca tasarlanabilir, fakat bit sayısı arttıkça devrenin gecikmesi de orantılı olarak artmaktadır [6].



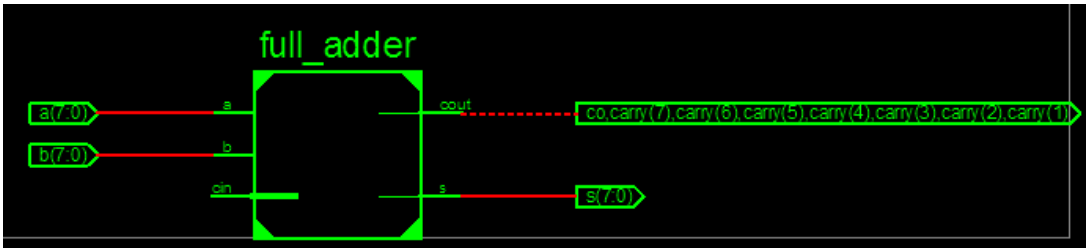
Şekil 3. 9: 4-bitlik elde zincirli toplayıcı.

Xilinx Ise11 programında 8-bitlik elde zincirli toplayıcı devresine uygun VHDL kodlar yazılıp, girişlere rastgele sayısal değerler verilerek devrenin doğru çalışıp çalışmadığını belirlemek için benzetim yapılmıştır. Şekil 3.10'da da görüldüğü gibi, sonuç beklentilere uygun çıkmıştır.

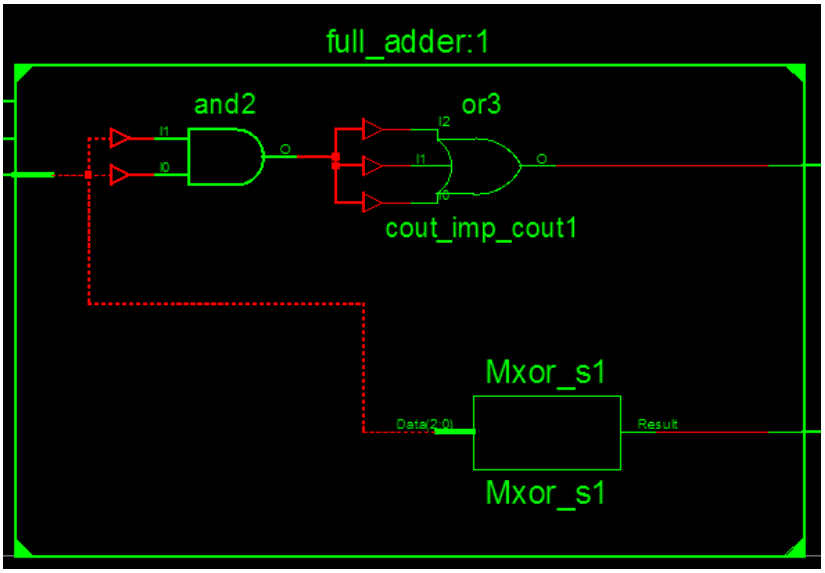


Şekil 3. 10: 8-bitlik elde zincirli toplayıcı devresinin Xilinx Ise ile oluşturulmuş benzetimi.

VHDL ile tanımlanan 8-bitlik elde zincirli toplayıcı'nın devre şematiği, Xilinx Ise11 kullanılarak oluşturulmuştur. Şekil 3.11 ve 3.12'de görüldüğü gibi beklentilere uygun olarak çıkmıştır.



Şekil 3. 11: 8-bitlik ripple carry adder'ın Xilinx Ise ile oluşturulmuş devre şematiği.



Şekil 3. 12: Full_adder (tam toplayıcı) ünitesinin iç yapısı.

3.4.Elde Öngörülü Toplayıcı

Toplama devrelerinde elde oluşumunu hızlandırmak için en çok kullanılan teknik elde öngörülü toplayıcı algoritmasıdır. Bu yöntemde ana fikir bir önceki kattan doğru eldenin gelmesini beklemeden elde sinyallerini giriş bitlerinden üretmektir [8].

Elde öngörülü toplayıcı'nın işleyişini daha iyi anlayabilmek için bir tam toplayıcı ünitesini göz önüne alalım. Elde yaılma sinyali (Pi) ve elde üretme sinyali (Gi), (3.5) ve (3.6) numaralı denklemlerde belirtildiği gibi tanımlanmaktadır;

$$Pi = Ai \oplus Bi \quad (3.5)$$
$$Gi = Ai.Bi \quad (3.6)$$

Görüldüğü gibi yayılma ve üretme sinyallerinin her ikisi de sadece giriş bitlerine bağlıdırlar. Pi ve Gi sinyalleri kullanılarak tam toplayıcıdan oluşan her katın sonuç ve elde çıkışını tekrar yazalım;

$$Si = Pi \oplus Ci \quad (3.7)$$
$$Ci+1 = Gi + Pi.Ci \quad (3.8)$$

(3.7) ve (3.8) eşitliklerinden elde çıkışı (Ci+1) sinyalinin iki durumda 1 olduğu anlaşılmaktadır;

Ai ve Bi girişlerinin her ikisi de 1 olduğunda.

Ai ve Bi girişlerinden herhangi birinin 1 ve elde girişi (Ci) 1 olduğunda.

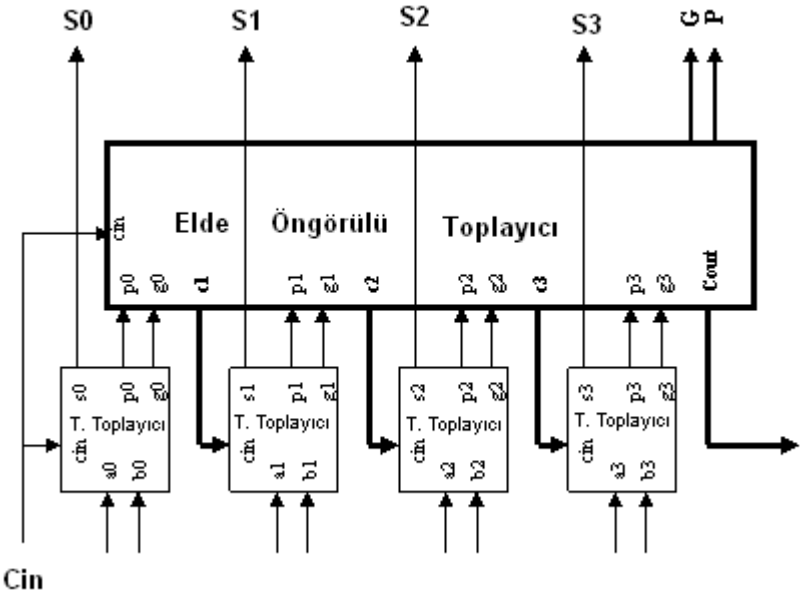
Bu denklemleri 4-bitlik toplayıcıya uygulayalım;

$$C1 = G0 + P0C0 \quad (3.9)$$
$$C2 = G1 + P1C1 = G1 + P1\,(G0 + P0C0) = G1 + P1G0 + P1P0C0 \quad (3.10)$$
$$C3 = G2 + P2C2 = G2 + P2G1 + P2P1G0 + P2P1P0C0 \quad (3.11)$$
$$C4 = G3 + P3C3 = G3 + P3G2 + P3P2G1 + P3P2P1G0 + P3P2P1P0C0 \quad (3.12)$$

Bu ifadelerden C2, C3 ve C4'ün bir önceki katın elde çıkışından bağımsız oldukları anlaşılmaktadır. C0 bilindiği anda C2, C3 ve C4 de eş zamanlı olarak oluşmaktadır. Bu denklemlerin genelleştirilmiş ifadesi ise (3.13) numaralı denklemde vrilmektedir.

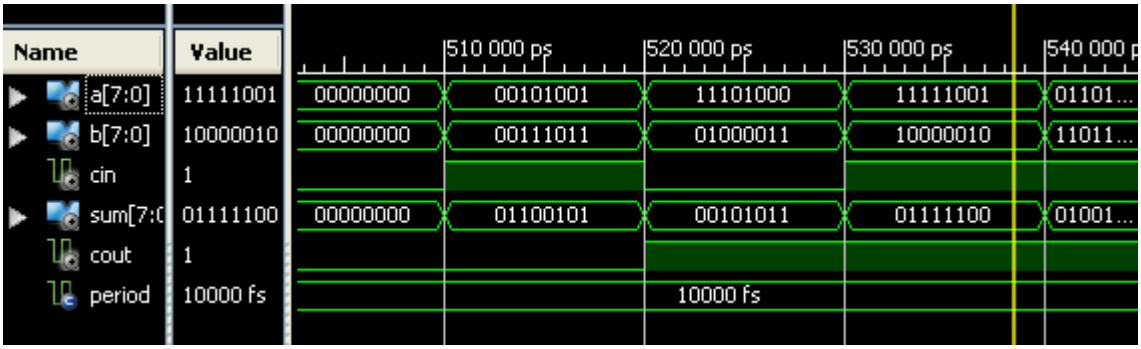
$$Ci+1= Gi + Pi.Gi-1 + Pi.Pi-1.Gi-2 + ... Pi.Pi-1….P2.P1.G0 + Pi.Pi-1 ….P1.P0.C0 \quad (3.13)$$

4-bitlik elde öngörülü toplayıcı'nın yapısı şekil 3.13'de gösterilmiştir [6].



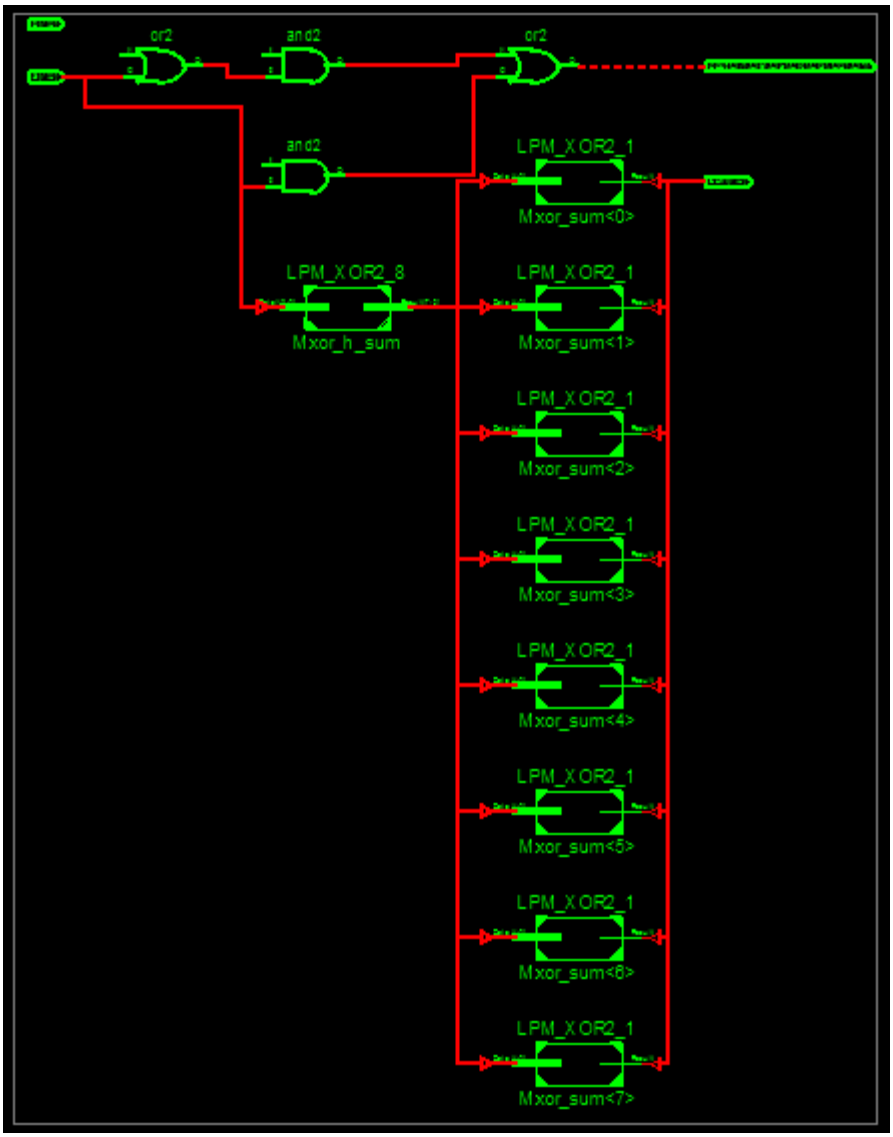
Şekil 3. 13: 4-bitlik elde öngörülü toplayıcı'nın yapısı.

Xilinx Ise11 programında 8-bitlik elde öngörülü toplayıcı devresine uygun VHDL kodlar yazılıp, girişlere rastgele sayısal değerler verilerek devrenin doğru çalışıp çalışmadığını belirlemek için benzetim yapılmıştır. Şekil 3.14'de de görüldüğü gibi, sonuç beklentilere uygun çıkmıştır.



Şekil 3. 14: 8-bitlik elde öngörülü toplayıcı devresinin Xilinx Ise ile oluşturulmuş benzetimi.

VHDL ile tanımlanan 8-bitlik elde öngörülü toplayıcı'nın devre şematiği, Xilinx Ise11 kullanılarak oluşturulmuştur. Şekil 3.15'de görüldüğü gibi beklentilere uygun olarak çıkmıştır.

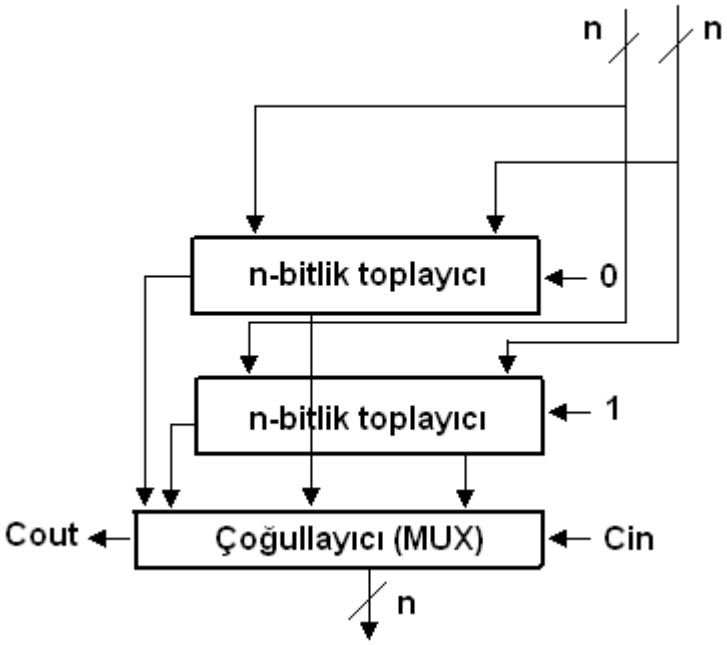


Şekil 3. 15: 8-bitlik elde öngörülü toplayıcı'nın Xilinx Ise ile oluşturulmuş devre şematiği.

3.5.Koşullu Toplayıcı

Toplama işlemini hızlandırmak için kullanılan yöntemlerden bir diğeri ise koşullu toplama yöntemidir. Bu yöntemde şekil 3.16'da görüldüğü gibi verilen girişler için iki farklı sonuç oluşturulmaktadır. Birinci sonucun üretiminde, giriş bitleri toplanırken elde girişi sıfır olarak alınmakta ve ikinci sonuç üretiminde, giriş bitleri toplanırken elde girişi 1 olarak alınmaktadır. Elde girişinin değeri bilindiği anda yapılması gereken doğru sonucu bir çoğullayıcı kullanarak seçmektir [9].

Bu yöntem giriş bitlerinin tamamına sıralı şekilde uygulanırsa sonuç ripple carry adder'dan farklı olmayacaktır ve hala elde bitlerinin bütün katlarda oluşmasını bekliyor olacağız. Bu nedenle giriş bitlerini gruplara böleceğiz ve yöntemi her grup için ayrı olarak uygulayacağız. Böylece her grubun elde oluşumu paralel olarak yapılacaktır ve toplam gecikme azalacaktır. Bu gruplar ayrıca alt gruplara bölünebilmekte ve elde oluşumunun süresi daha da azalabilmektedir. Alt grupların sonucu birleştirilerek grupların sonucu, grupların sonucu da birleştirilerek giriş sayılarının toplama sonucu oluşturulur.

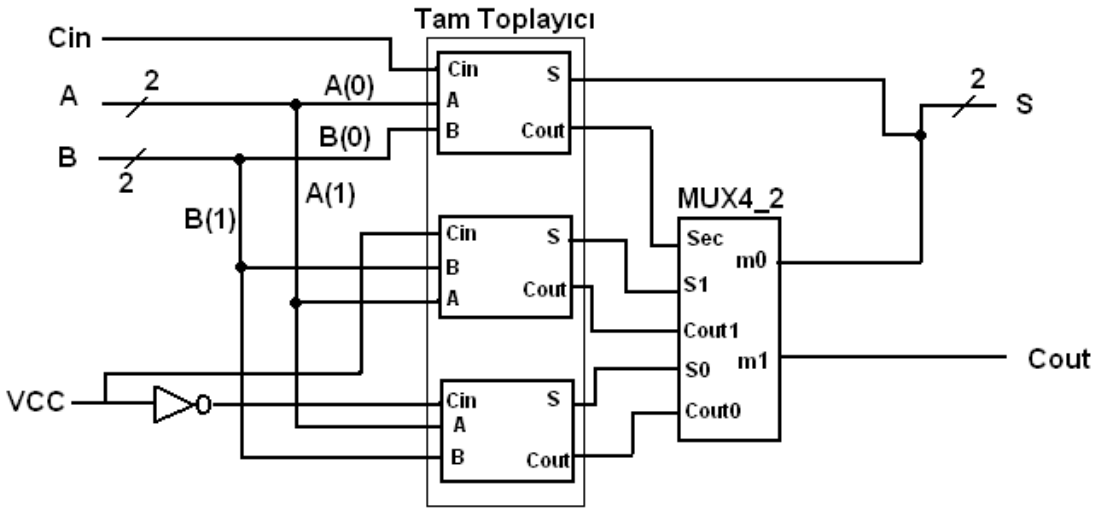


Şekil 3. 16: Koşullu toplama yönteminde çoğullayıcı kullanarak doğru sonucun seçilmesi.

Tablo 3.3'de 8-bitlik iki sayının koşullu toplama yöntemi ile toplanma mantığı gösterilmektedir. Şekil 3.17'de ise 2-bitlik koşullu toplama devresinin blok diyagramı verilmiştir.

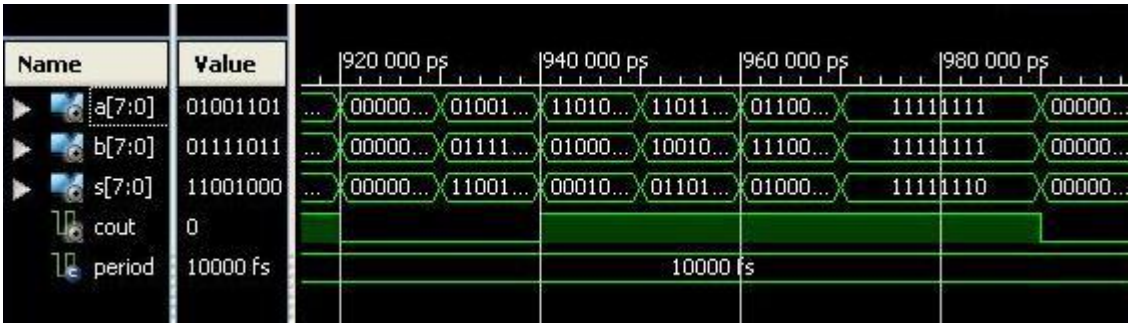
Tablo 3. 3: 8-bitlik iki sayının koşullu toplama yöntemi ile toplamının elde edilmesi.

	i	7	6	5	4	3	2	1	0
	A*i*	1	0	1	1	0	1	1	0
	B*i*	0	0	1	0	1	1	0	1
Aşama 1	S^0i	1	0	0	1	1	0	1	1
	C^0i+1	0	0	1	0	0	1	0	0
	S^1i	0	1	1	0	0	1	0	
	C^1i+1	1	0	1	1	1	1	1	
Aşama 2	S^0i	1	0	0	1	0	0	1	1
	C^0i+1	0		1		1		0	
	S^1i	1	1	1	0	0	1		
	C^1i+1	0		1		1			
Aşama 3	S^0i	1	1	0	1	0	0	1	1
	C^0i+1	0				1			
	S^1i	1	1	1	0				
	C^1i+1	0							
Sonuç		1	1	1	0	0	0	1	1



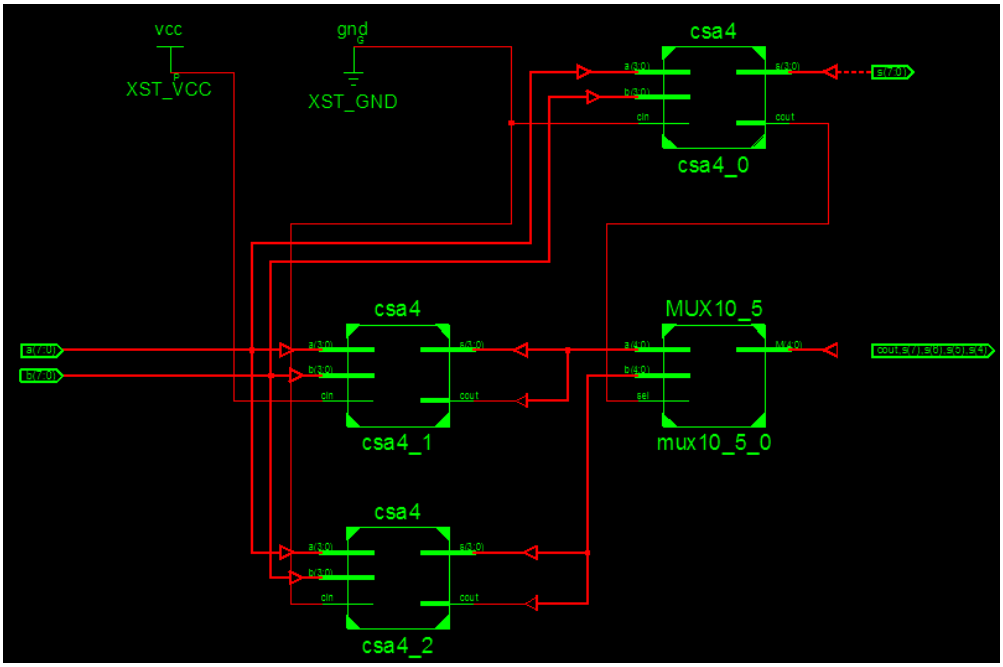
Şekil 3. 17: 2-bitlik koşullu toplayıcı devresinin blok diyagramı.

Xilinx Ise11 programında 8-bitlik koşullu toplayıcı devresine uygun VHDL kodlar yazılıp, girişlere rastgele sayısal değerler verilerek devrenin doğru çalışıp çalışmadığını belirlemek için benzetim yapılmıştır. Şekil 3.18'da görüldüğü üzere, sonuç beklenildiği gibi çıkmıştır.

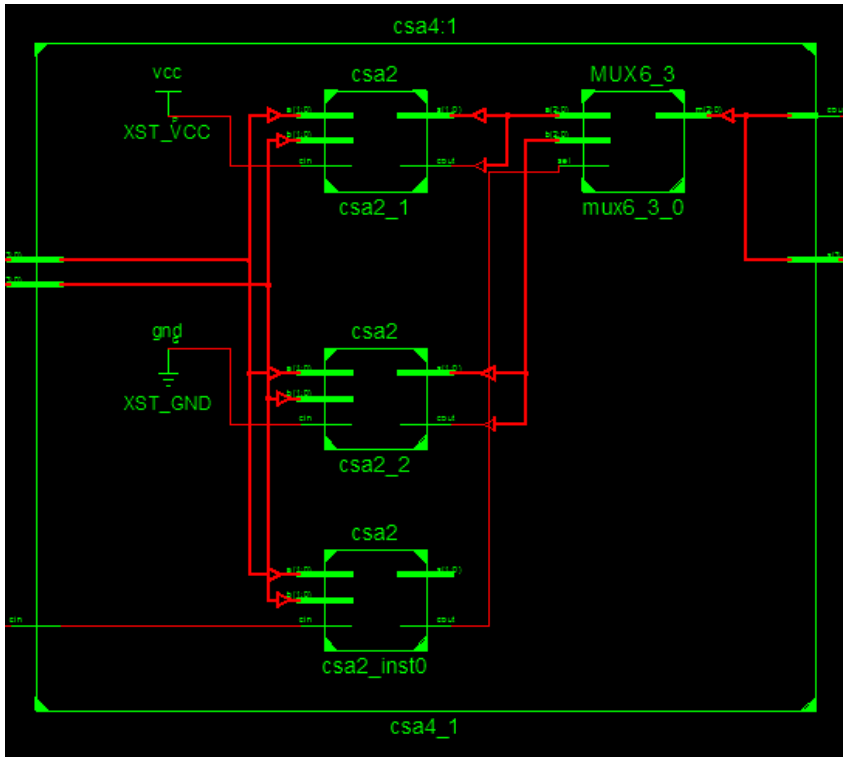


Şekil 3. 18: 8-bitlik koşullu toplayıcı devresinin Xilinx Ise ile oluşturulmuş benzetimi.

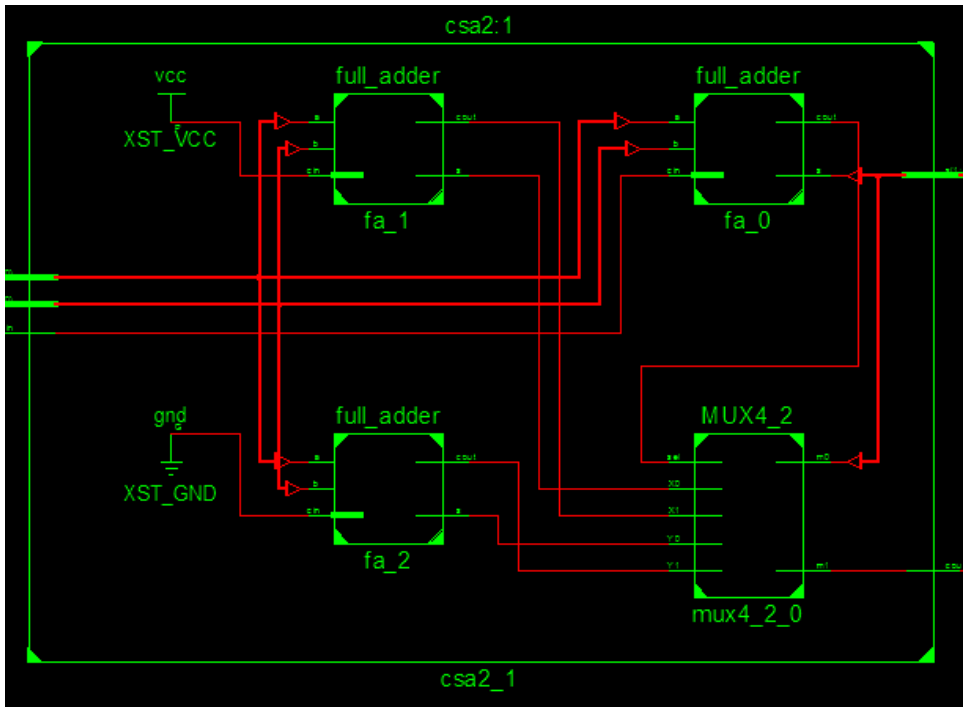
VHDL ile tanımlanan 8-bitlik koşullu toplayıcı'nın devre şematiği, Xilinx Ise11 kullanılarak oluşturulmuştur. Devre şematiği şekil 3.19, 3.20, 3.21, 3.22, 3.23 ve 3.24'de görüldüğü gibi beklentilere uygun olarak çıkmıştır.



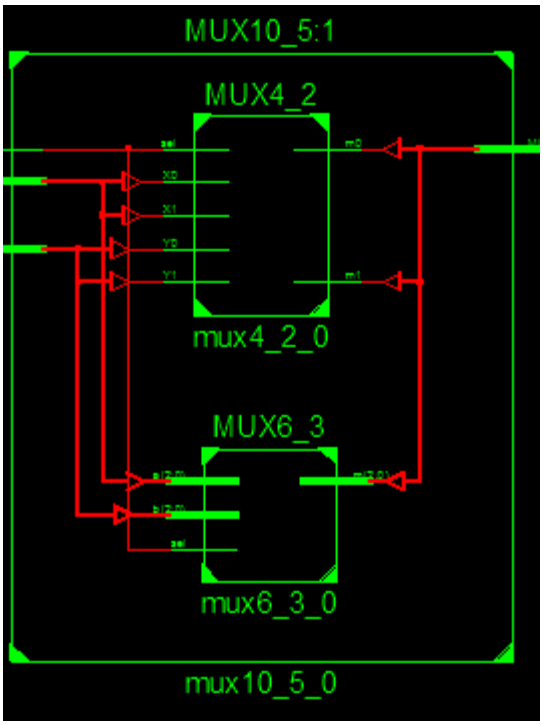
Şekil 3. 19: 8-bitlik koşullu toplayıcı'nın 4-bitlik koşullu toplayıcı 'lardan oluşan Xilinx Ise ile oluşturulmuş devre şematiği.



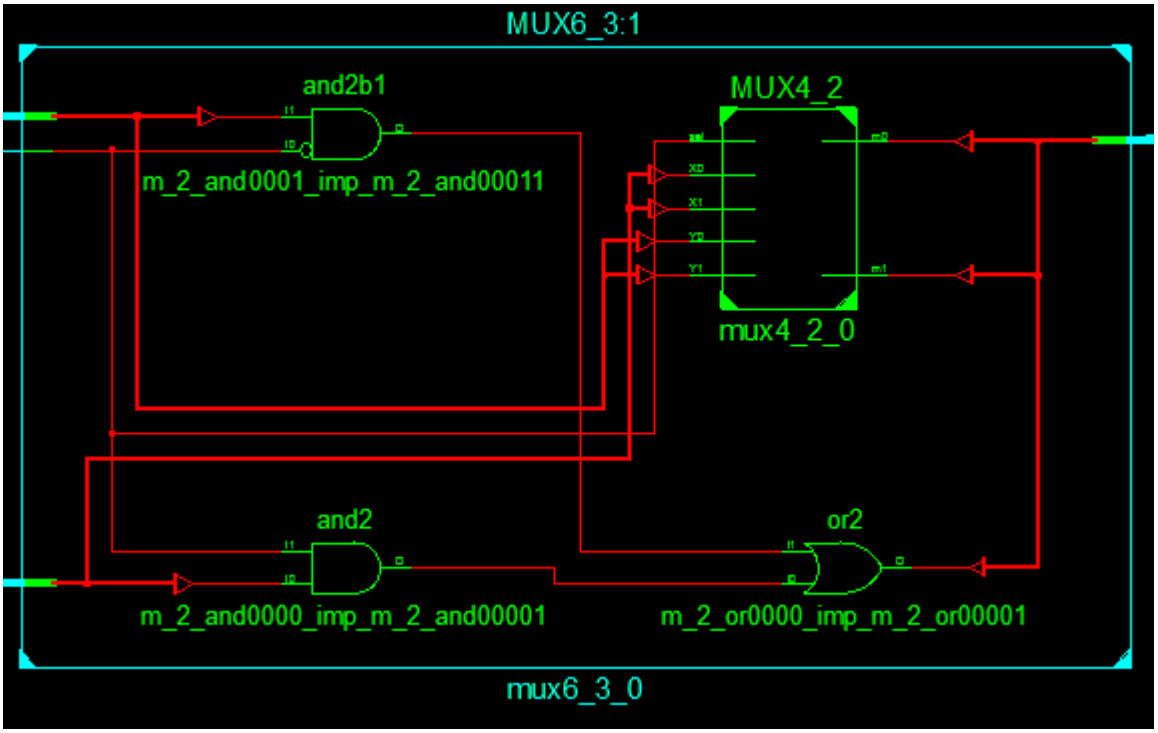
Şekil 3. 20: 4-bitlik koşullu toplayıcı 'nın 2-bitlik koşullu toplayıcı 'lardan oluşan Xilinx Ise ile oluşturulmuş devre şematiği.



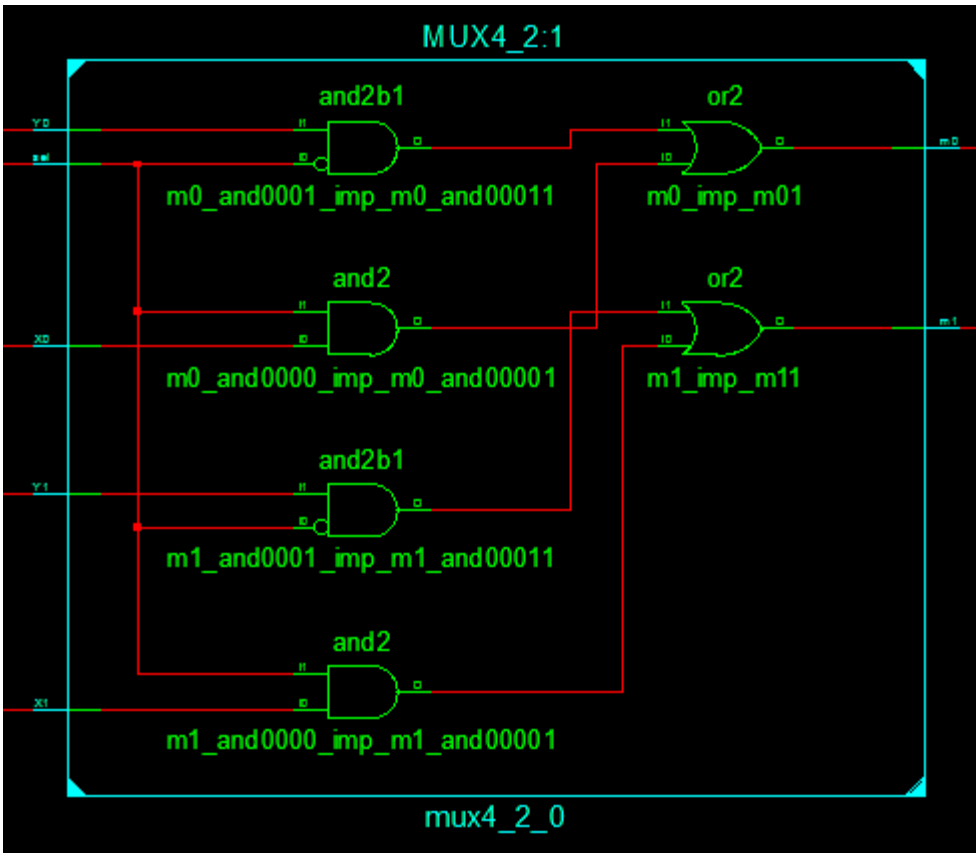
Şekil 3. 21: 2-bitlik koşullu toplayıcı 'nın Xilinx Ise ile oluşturulmuş devre şematiği.



Şekil 3. 22: 10-5'lik çoğulayıcı'nın Xilinx Ise ile oluşturulumuş devre şematiği.



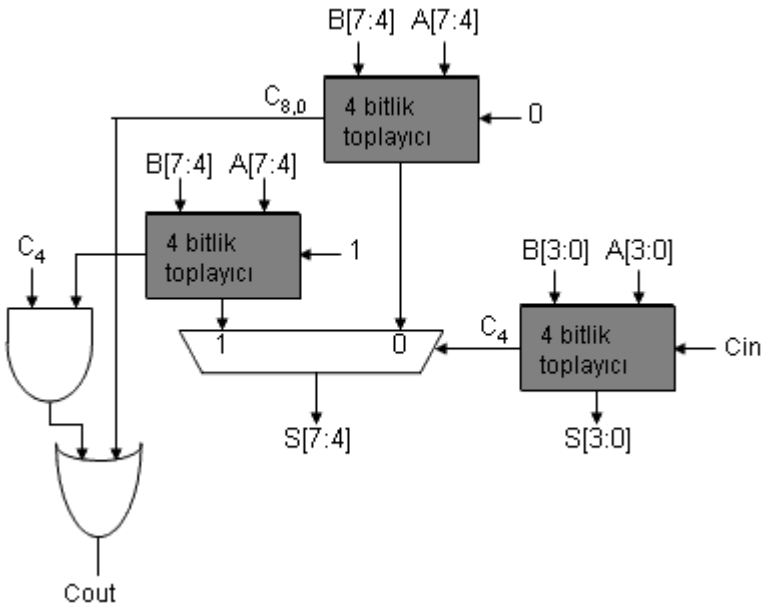
Şekil 3. 23: 6-3'lük çoğulayıcı'nın Xilinx Ise ile oluşturulumuş devre şematiği.



Şekil 3. 24: 4-2'lik çoğulayıcı'nın Xilinx Ise ile oluşturulumuş devre şematiği.

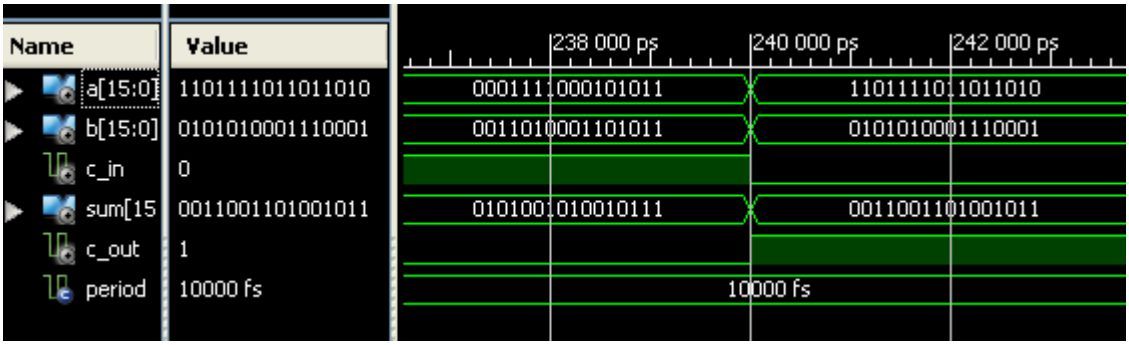
3.6.Elde Seçici Toplayıcı

Paralel toplayıcıların hızını arttırma yönünde bir yaklaşım da elde seçici toplayıcı kullanımıdır. Elde seçici toplayıcıda, giriş bitleri farklı gruplara ayrılırlar. Temel mantık ise bölüm 3.5'de anlatılan koşullu toplamaya benzemektedir. Her grup için iki farklı sonuç ve elde çıkışı oluşturulmaktadır. Sonuçlardan biri elde girişi 1 varsayılarak, diğeri ise elde girişi 0 kabul edilerek hesaplanmaktadır. Grubun elde giriş biti belli olduğunda, doğru sonuç seçilir. Bu yöntemin koşullu toplamadan farkı ise, koşullu toplamada her grup daha küçük alt gruplara ayrılmakta idi, elde seçici toplamada her grup bir elde zincirli toplayıcı ile toplanmaktadır. Şekil 3.25'de 8-bitlik elde seçici toplayıcının blok diyagramı verilmiştir [8],[6].



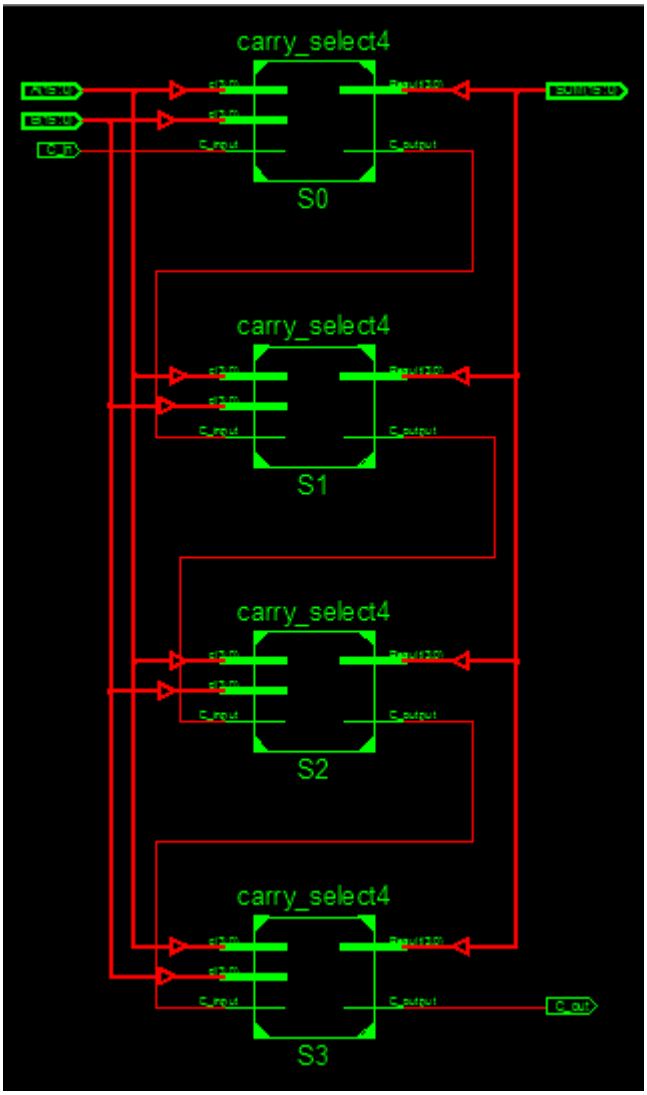
Şekil 3. 25: 8-bitlik elde seçici toplayıc'nın blok diyagramı.

Xilinx Ise11 programında 16-bitlik elde seçici toplayıcı devresine uygun VHDL kodlar yazılıp, girişlere rastgele sayısal değerler verilerek devrenin doğru çalışıp çalışmadığını belirlemek için benzetim yapılmıştır. Şekil 3.26'da görüldüğü üzere, sonuç beklenildiği gibi çıkmıştır.

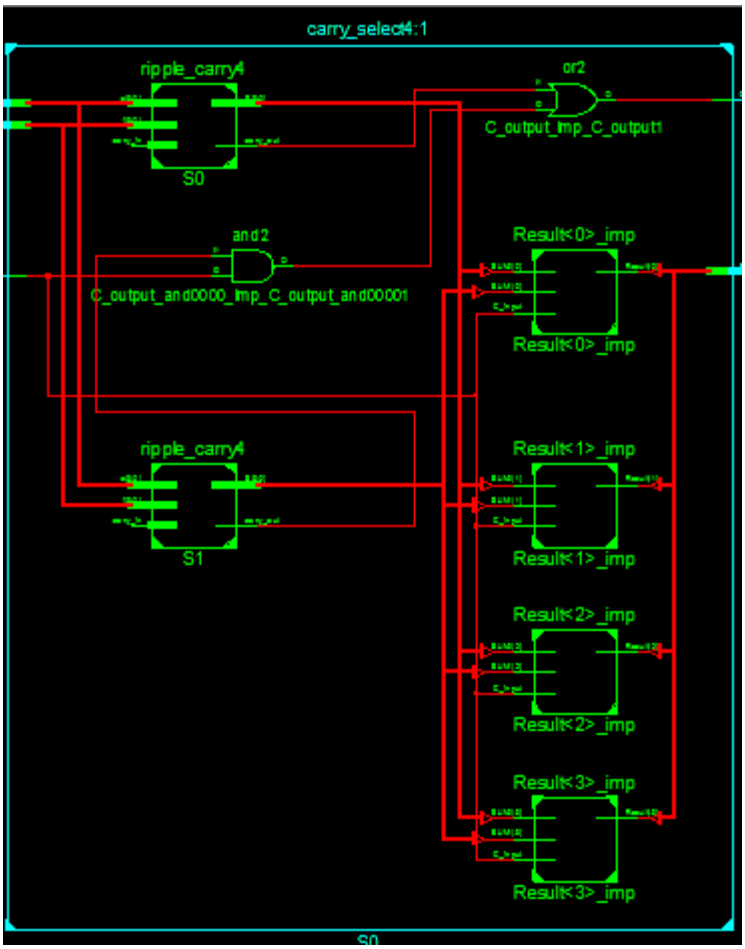


Şekil 3. 26: 8-bitlik elde seçici toplayıcı devresinin Xilinx Ise ile oluşturulmuş benzetimi.

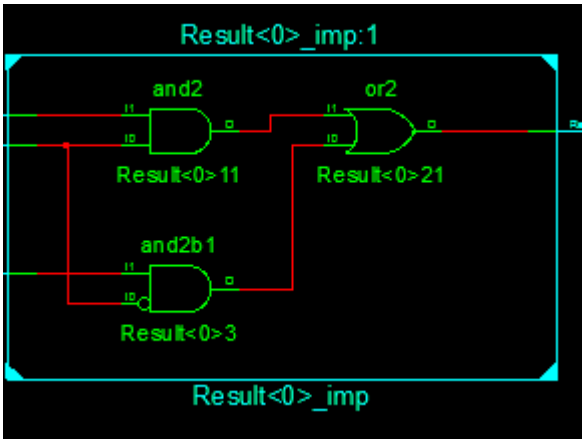
VHDL ile tanımlanan 16-bitlik elde seçici toplayıcı'nın devre şematiği, Xilinx Ise11 kullanılarak oluşturulmuştur. Devre şematiği şekil 3.27, 3.28ve 3.29'da görüldüğü gibi beklentilere uygun olarak çıkmıştır.



Şekil 3. 27: 16-bitlik elde seçici toplayıcı'nın Xilinx Ise ile oluşturulmuş devre şematiği.



Şekil 3. 28: 4-bitlik elde seçici toplayıcı olan carry_select4 ünitesinin iç yapısı.



Şekil 3. 29: Çoğullayıcı olan Result ünitesin iç yapısı.

3.7.Toplayıcı Devrelerinin Karşılaştırılması

Bu çalışmada elde zincirli toplayıcı, elde öngörülü toplayıcı, koşullu toplayıcı ve elde seçici toplayıcı algoritmalarına göre toplama devreleri incelenmiştir. Bu algoritmalara uygun devreler VHDL ile tanımlanmış ve Xilinx Ise 11 programı kullanılarak doğruluğu test edildikten sonra FPGA üzerinde gerçeklenmiştir. Gerçekleme yapıldıktan sonra program tarafından oluşturulan raporlarda her algoritmanın gecikme ve alan bilgileri kontrol edilmiş ve tablo 3.4'de not edilmiştir.

Tablo 3. 4: Farklı toplama algoritmaları'nın gecikme ve alan bilgileri.

Toplayıcı	Gecikme (ns)	Alan (CLB)
Ripple Carry Adder	36.700	37
Carry Look Ahead Adder	28.043	48
Conditional sum adder	18.730	82
Carry Select Adder	23.381	64

İncelen toplayıcı türleri hakkında özet yapılacak olursa, sorulması gereken soru "Hangi toplayıcı nerede kullanılmalı?" sorusudur. Herhangi bir toplayıcı devresi için avantajlı veya dezavantajlı olduğunu söylemek doğru değildir, çünkü bir faktörün iyi olması, diğer bir faktörün kötü olmasına yol açmaktadır. İncelediğimiz algoritmalardan da anlaşıldığı gibi hız ve alan faktörü ters orantılıdır. Daha yüksek performanslı toplama devreleri daha fazla yer kapsarlar. Bu sebepden dolayı, tasarımcı güç, alan ve gecikme gibi faktörleri göz önünde bulundurarak tasarımına uygun algoritmayı seçmelidir.

4. ÇARPMA DEVRELERİ

Çarpma devresi oluşturmanın en temel yöntemi iki pozitif ikilik tabandaki sayının geleneksel yöntemle çarpımından yola çıkmaktır. Bu yöntemde alt alta toplamalar ve kaydırmalar mevcuttur. Örnek verilecek olursa, iki pozitif tamsayı olan $(12)_{10}$ ve $(5)_{10}$ sayılarının çarpımı kaydır ve topla yöntemi ile şöyle gerçekleştirilir:

çarpılan: 1100 : $(12)_{10}$
çarpan: 0101 : $(5)_{10}$
1100
0000
1100
0000
çarpım: 0111100 : $(60)_{10}$

Buradan çarpma işleminin şu iki basamaktan oluştuğu görülür:

1. Kısmi çarpımların elde edilmesi

2. Kısmi çarpımların kaydırılarak toplanması

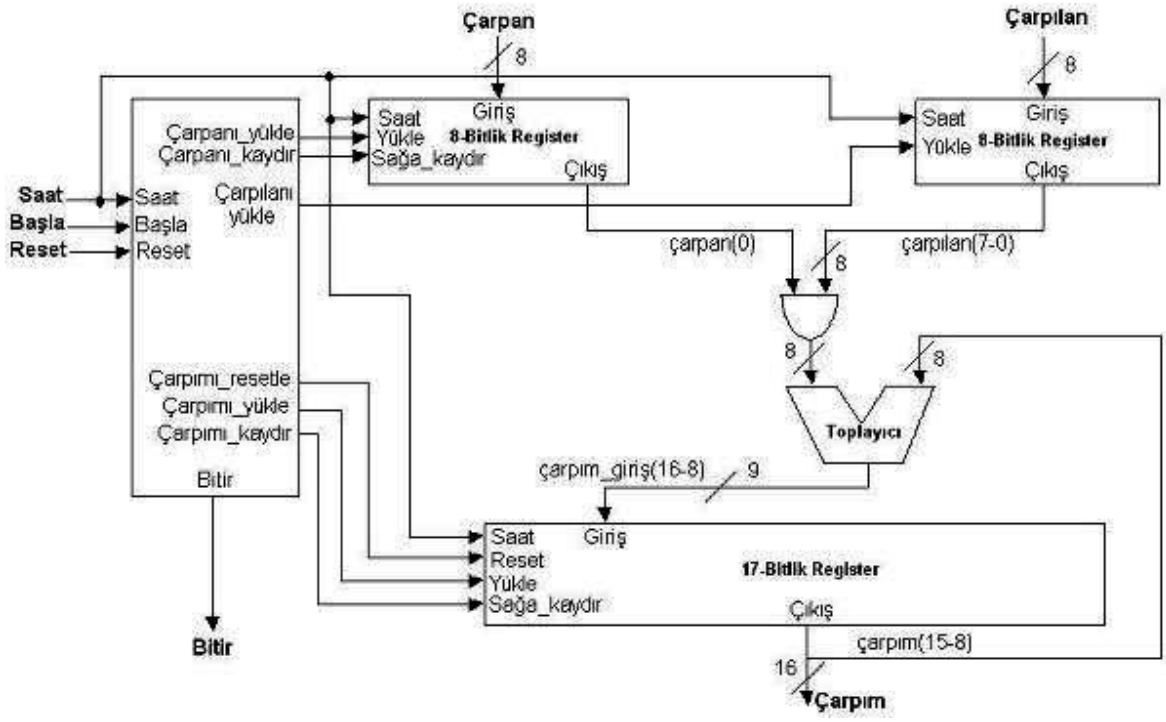
İkilik tabanda çarpama işleminin mantıksal VE işlemine denk olduğu yapılan işlemlerde görülmektedir. Dolayısı ile kısmi çarpımların bulunması işlemi, çarpılanı sıra ile çarpanın bitleri ile mantıksal VE işleminden geçirmektir. Daha sonra kısmi çarpımların sütunları birbirleri ile toplanacaktır ve gerekli ise elde bitleri bir sonraki sütuna ilave edilecektir. Bu işlemleri gerçekleştirmek için seri, paralel ve her ikisi birlikte kullanılan yöntemler geliştirilmiştir.

Bu bölümde işaretsiz sayılar için tasarlanmış ardışıl çarpma ve dizin çarpma algoritmalarını gerçekleştireceğiz.

4.1.Ardışıl Çarpıcı

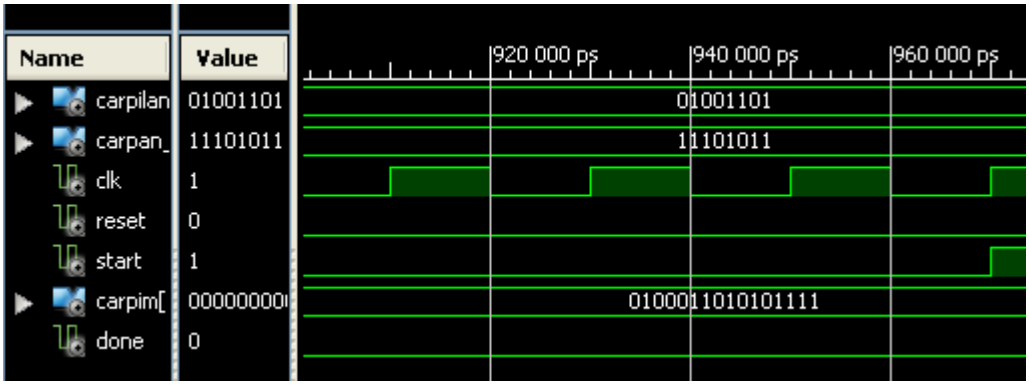
Ardışıl çarpıcı algoritmasının temeli, çarpan sayısının en düşük anlamlı bitinin değerine göre çarpılan sayının toplanarak kısmi toplam olarak değerlendirilmesine dayanmaktadır. Herbir saat çevriminde çarpan sayı bir bit sağa kaydırılır ve bu bitin değeri çarpılan sayının bitleri ile VE kapısından geçirelerek, çıkışı kısmi toplama eklenir ve bu kısmi toplam değeri bir bit sağa kaydırılır. Çarpan sayının tüm bitleri işleme tabi edildikten sonra n giriş sayıların bit uzunluğu olmak üzere, 2n uzunluğunda bir sonuç elde edilir. Şekil 4.1'de 8-bitlik sıralı çarpıcı'nın blok diyagramı gösterilmektedir [9].

Bu diyagramda ilk aşamada çarpan ve çerpılan, 8-bitlik register'da saklanır. Kontrol ünitesinin durumuna göre çarpan'ın en düşük anlamlı biti, çarpılanın ise bütün bitleri çıkışa verilir. Bu register'ların çıkışı VE kapısından geçirilerek 8-bitlik toplama ünitesine gelir. Toplama ünitesinin diğer girişi ise ilk değeri 0 olan 17 bitlik register'ın 15-8 bitlerinden gelmektedir. Toplama ünitesinin çıkışı, 17-bitlik çarpım register'ın 16-8 bitlerine yerleşmektedir. Bir sonraki aşamada kontrol ünitesinden gelen sinyale göre çarpım ve çarpan register'larının içeriği birer bit sağa kaydırılarak aynı işlemler bütün bitler için tekrarlanır ve çarpım sonucu elde edilir.



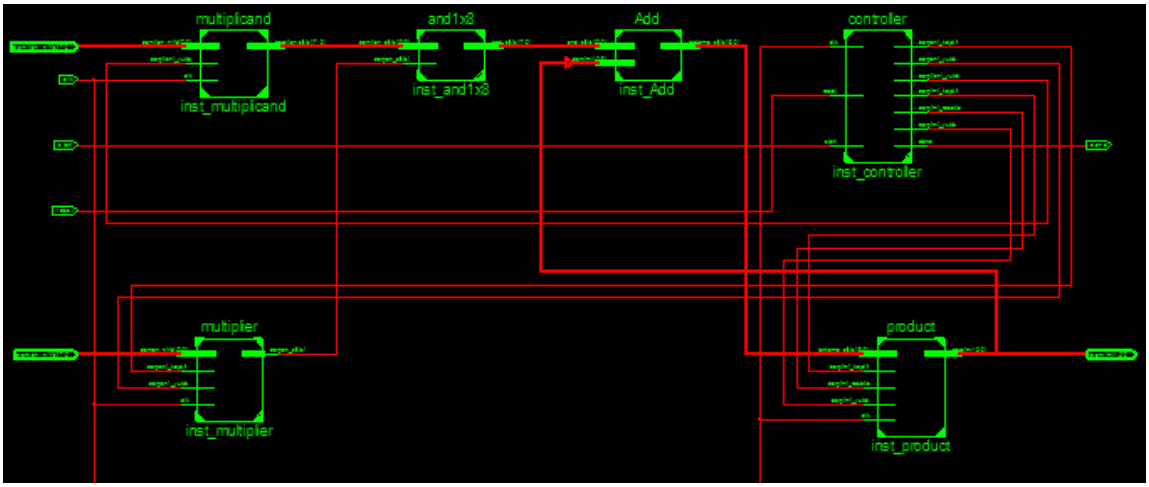
Şekil 4. 1: Ardışıl çarpıcı'nın blok diyagramı.

Xilinx Ise 11 programında 8-bitlik ardışıl çarpıcı devresi VHDL ile tanımlandıktan sonra, girişlere rastgele sayısal değerler verilerek devrenin doğru çalışıp çalışmadığını belirlemek için benzetim yapılmıştır. Şekil 4.2'de görüldüğü gibi, sonuç beklenildiği gibi çıkmıştır.

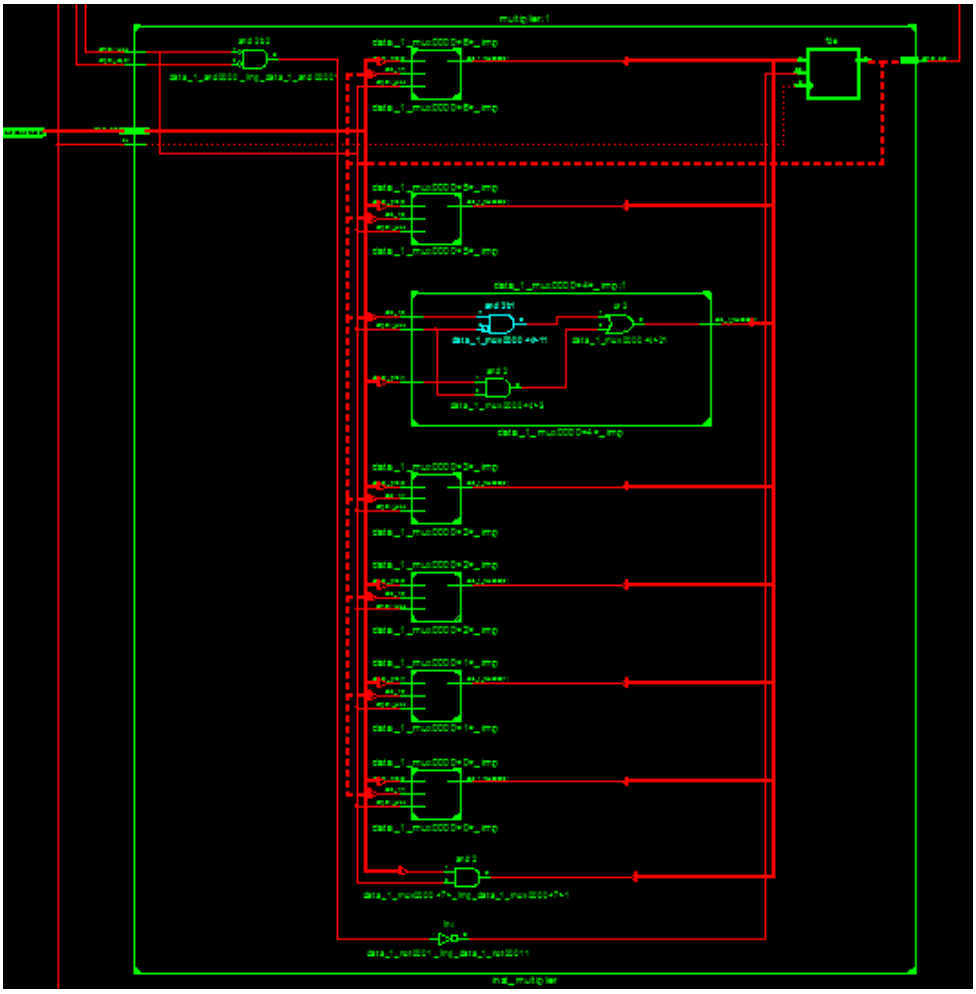


Şekil 4. 2: 8-bitlik ardışıl çarpıcı devresinin Xilinx Ise ile oluşturulmuş benzetimi.

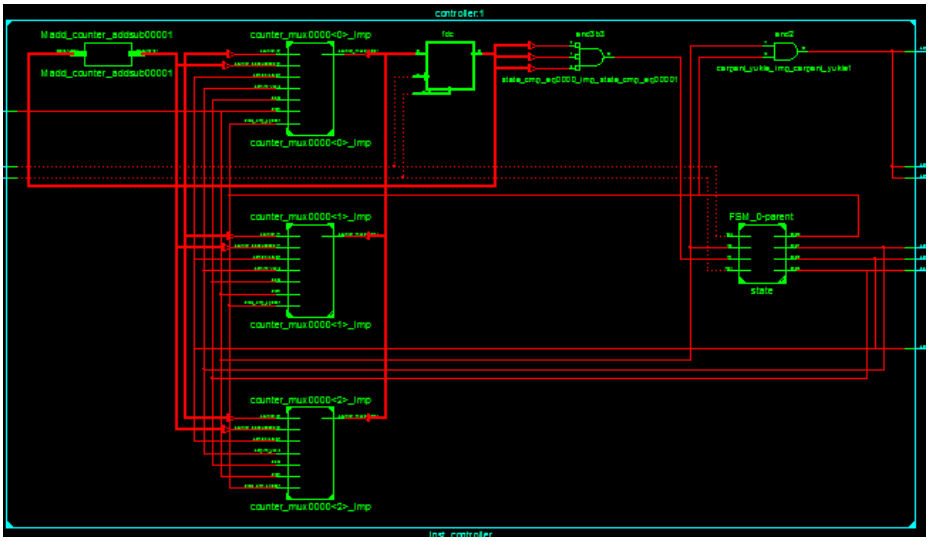
VHDL ile tanımlanan 8-bitlik ardışıl çarpıcı'nın devre şematiği, Xilinx Ise 11 kullanılarak oluşturulmuştur. Devre şematiği şekil 4.3, 4.4, 4.5 ve 4.6'da görüldüğü gibi beklentilere uygun olarak çıkmıştır.



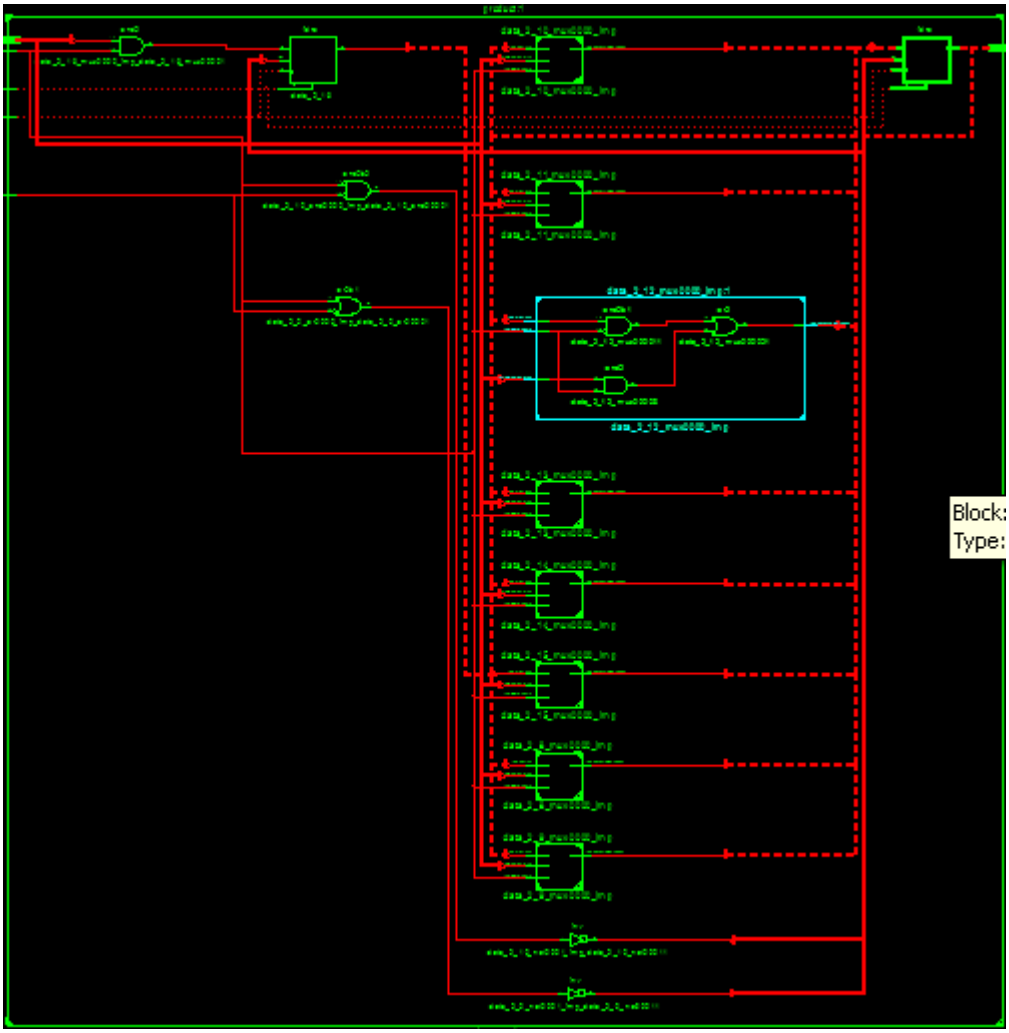
Şekil 4. 3: 8-bitlik ardışıl çarpıcı'nın Xilinx Ise ile oluşturulmuş devre şematiği.



Şekil 4. 4: Multiplier (çarpan) ünitesinin iç yapısı.



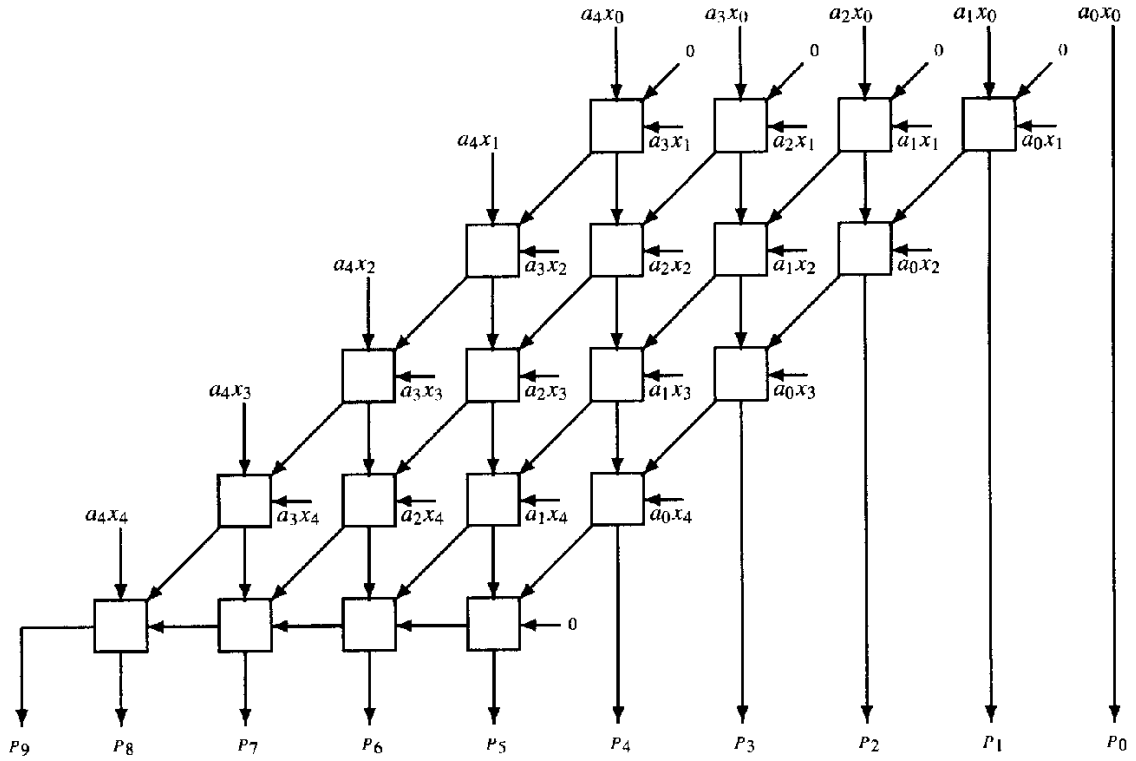
Şekil 4. 5: Controller (kontrol) ünitesinin iç yapısı.



Şekil 4. 6: Product (çarpım) ünitesinin iç yapısı.

4.2.Dizin Çarpıcı

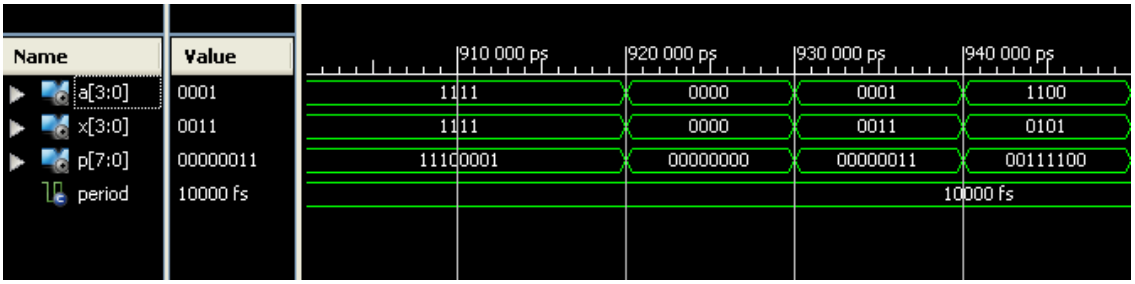
Çarpma işleminde, kısmi çarpımların elde edilişi ve toplama işlemi birleştirilebilir. Böylece bu iki işlemi ayrı ayrı kontrol etmenin yükü azalacak ve çarpım daha hızlı gerçeklenecektir. Şekil 4.7'de 5-bitlik bir dizin çarpıcı'nın blok diyagramı verilmiştir.



Şekil 4. 7: 5-bitlik bir dizin çarpıcı'nın blok diyagramı [6].

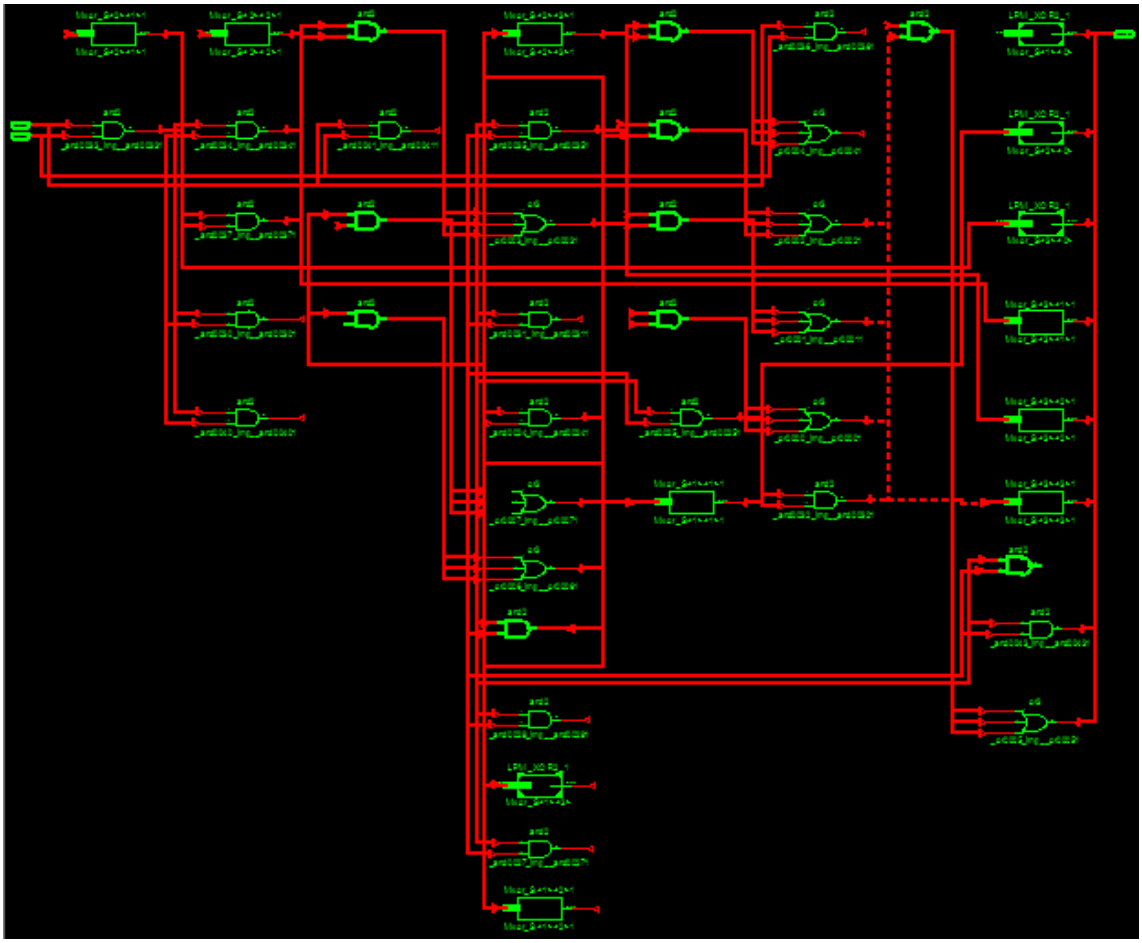
Görüldüğü gibi bu devre özdeş hücrelerden oluşmaktadır. Her bir hücre yeni bir kısmi çarpım oluşturmakta ve daha önce oluşturulan kısmi çarpımlarla toplamaktadır. İlk dört satır'da yatay bir elde yayılımı olmamaktadır. Sadece son satır'da yatay elde yayılımı vardır ve aslında bu satır elde zincirli toplayıcı devresidir ve daha hızlı olan farklı bir toplayıcı ile değiştirilebilir.

Xilinx Ise11 programında 4-bitlik dizin çarpıcı devresine uygun VHDL kodlar yazılıp, girişlere rastgele sayısal değerler verilerek devrenin doğru çalışıp çalışmadığını belirlemek için benzetim yapılmıştır. Şekil 4.8'de görüldüğü üzere, sonuç beklenildiği gibi çıkmıştır.



Şekil 4. 8: 4-bitlik dizin çarpıcı devresinin Xilinx Ise ile oluşturulmuş benzetimi.

VHDL ile tanımlanan 4-bitlik dizin çarpıcı'nın devre şematiği, Xilinx Ise11 kullanılarak oluşturulmuştur. Devre şematiği şekil 4.9'da görüldüğü gibi beklentilere uygun olarak çıkmıştır.



Şekil 4. 9: 4-bitlik dizin çarpıcı'nın Xilinx Ise ile oluşturulmuş devre şematiği.

4.3.Çarpıcı devrelerinin karşılaştırılması

Bu bölümde ardışıl ve dizin çarpma algoritmalarına göre çarpma devreleri incelenmiştir. Bu algoritmalara uygun devreler VHDL ile tanımlanmış ve Xilinx Ise 11 programı kullanılarak doğruluğu test edildikten sonra FPGA üzerinde gerçeklenmiştir. Gerçekleme yapıldıktan sonra program tarafından oluşturulan raporlarda her algoritmanın gecikme ve alan bilgileri kontrol edilmiş ve tablo 4.1'de not edilmiştir.

Tablo 4. 1: Farklı çarpma algoritmaları'nın gecikme ve alan bilgileri.

Çarpıcı (16 bit)	Gecikme (ns)	Alan (CLB)
Ardışıl Çarpıcı	31.421	46
Dizin Çarpıcı	27.672	65

Sonuçlardan dizin çarpıcının ardışıl çarpıcıdan daha hızlı olduğu, fakat daha faza alan kapsadığı kolayca görülmektedir.

5. Sonuçlar ve Tartışma

Bu çalışmada yüksek hızlı toplama ve çarpma yöntemleri incelenmiş ve bu yöntemler kullanılarak tasarlanan toplama çarpma devrelerinin, Çok Hızlı Tümleştirilmiş Devre Tanımlama Dili'ndeki (Very High Speed Integrated Circuit Harware Description Language: VHDL) modelleri yazılmıştır.

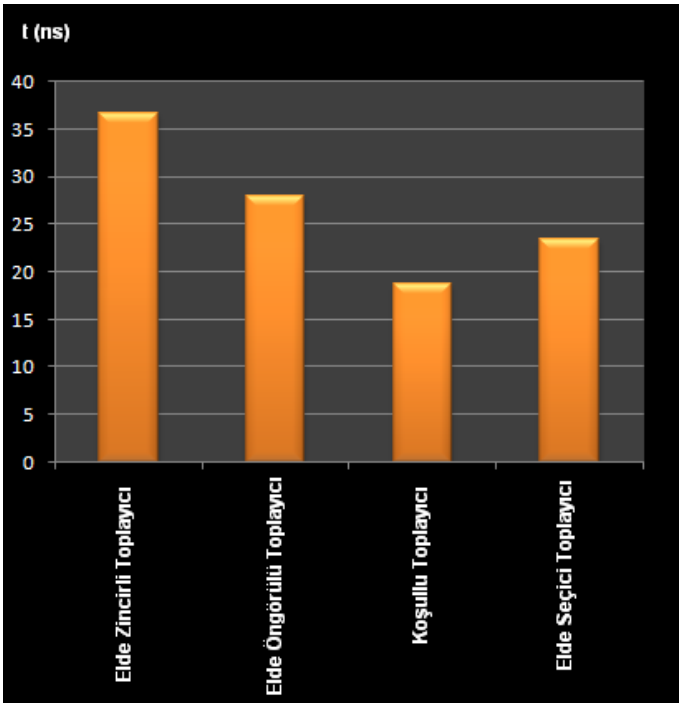
İncelenen toplama yöntemleri şu şekildedir:

1. Elde zincirli toplayızı
2. Elde öngörülü toplayıcı
3. Koşullu toplayıcı
4. Elde seçici toplayıcı

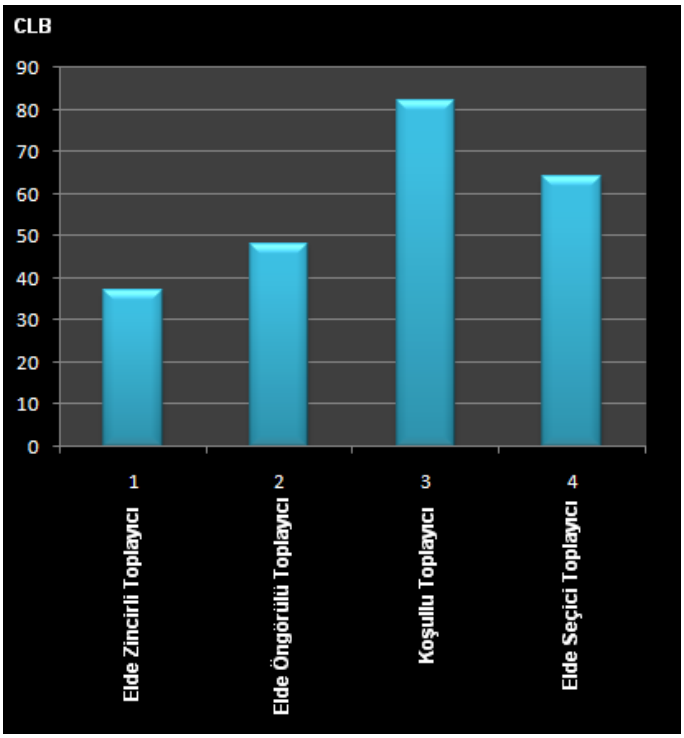
İncelenen çarpma yöntemleri ise şu şekildedir:

5. ardışıl çarpıcı
6. dizin çarpıcı

Bu devrelere ilişkin gecikme ve alan bilgileri Xilinx Ise 11 programı kullanılarak elde edilmiş ve grafiksel olarak çizdirilmiştir. Toplama devrelerine ilişkin gecikme grafiği şekil 5.1'de, alan grafiği de şekil 5.2'de gösterilmektedir.

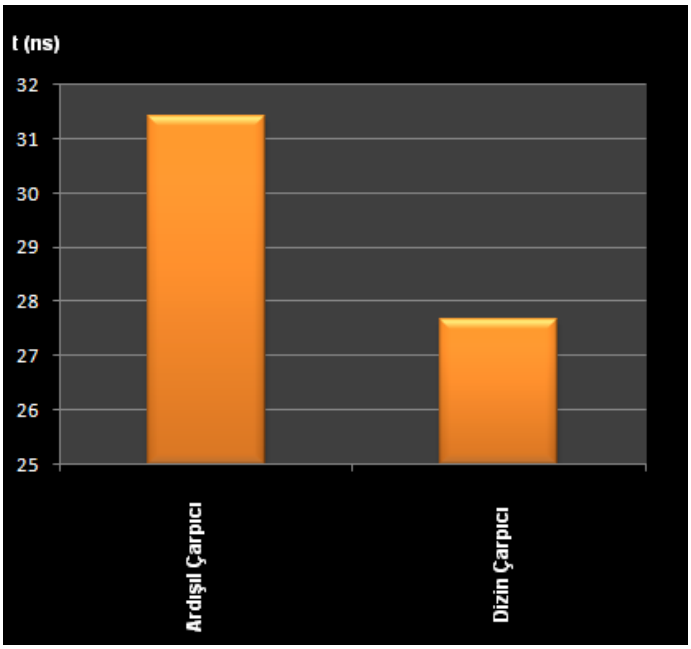


Şekil 5. 1: Toplama devrelerinin gecikme grafiği.

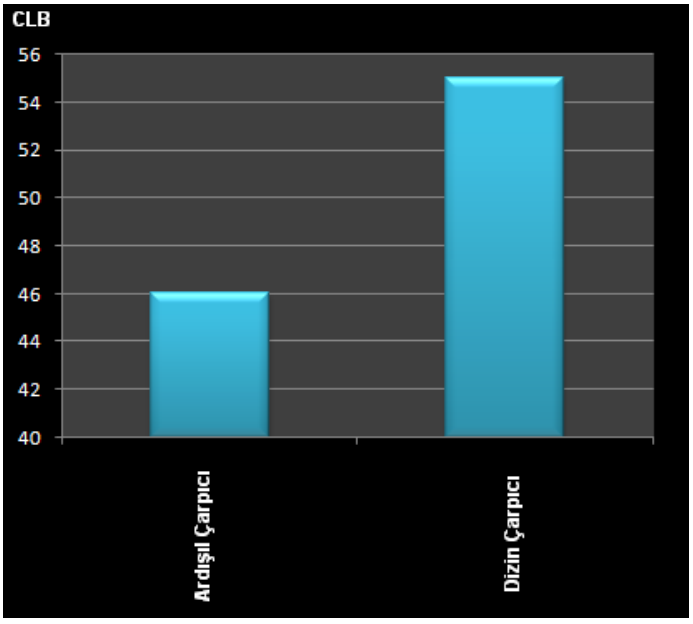


Şekil 5. 2: Toplama devrelerinin alan grafiği.

Çarpma devrelerine ilişkin gecikme grafiği ise şekil 5.3'de, alan grafiği de şekil 5.4'de yer almaktadır.



Şekil 5. 3: Çarpma devrelerinin gecikme grafiği.



Şekil 5. 4: Çarpma devrelerinin alan grafiği.

Grafiklerden de kolayca anlaşıldığı gibi, toplama ve çarpma devrelerinde gecikmeyi azaltmak ve donuca daha kısa bir sürede ulaşmak için, daha fazla sayıal kapı kullanılmalı ve bunun sonucunda da devrenin kapsadığı alan artmaktadır. Tasarımcı belirli bir toplama veya çarpma algoritmasını seçmeden önce, feragat edebileceği alanı bilmeli ve ona göre de daha hızlı veya daha yavaş algoritmayı seçmelidir.

KAYNAKLAR

[1] **Leblebici, D. Akurgal, A., Geray, H., Payzın, E., Sarper, S**., 2004. *Bilgi Ve İletişim Teknolojileri Stratejisi*. Vizyon 2023 Projesi Bilgi Ve İletişim Teknolojileri Strateji Grubu, Işık Üniversitesi, 4-8 s.

[2] **İsmailoğlu, A. N**., 1996, Yarı Özel CMOS VLSI Teknolojisi ile 4 Bit Mikrodenetleyici Tasarımı. *ASELSAN Dergisi*, 32 s, Ankara.

[3] **C. N.Marimuthu, P. Thangaraj**, 2008, Low Power High Performance Multiplier, *Master Thesis,* Engineering College, Perundurai, Anna University, India.

[4] **S. Berna Örs,** 1999, "Design of Multiplier Blocks for DSP Applications Using VHDL", *Master Thesis*, İTÜ, İSTANBUL.

[5] **Ö. Çetin**, 2003, 80C51 mikrodenetleyicilerinde watchdog FPGA mimarileri kullanılarak geliştirimesi, *Yüksek Lisans Tezi*, Sakarya Üniversitesi, Fen Bilimleri Enstitüsü, Sakarya.

[6] **Israel Koren**, 2001, Computer Arithmetic Algorithms, 2nd edition, A. K. Peters Publishers, Canada.

[7] **Peter J. Ashenden**, 2002, The Designer's Guide to VHDL, 2nd edition, Morgan Kaufmann publishers, Adelaide University, Australia.

[8] **Gary W. Bewick**, 1994, Fast Multiplication: Algorithms and Implementation, Master Thesis, Standford University.

[9] **Naofumi Takagi, HirotoYasuura, and ShuzoYajima**, Sept 1985, High-speed VLSI Multiplication Algorithmwith a Redundant Binary Addition Tree. *IEEE Transactions on Computers*,C–34(9).

ÖZGEÇMİŞ

1983 yılında İstanbul'da doğdu. İlkokul ve lise eğitimini İran'da başarıyla tamamladıktan sonra üniversite eğitimi için Türkiye'ye geldi. İlgi alanı olan İstanbul Teknik Üniversitesi, Elektronik Mühendisliğini bölümünü kazandı.