T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ŞİFRELEME İŞLEMLERİ İÇİN FPGA İLE YÜKSEK KAPASİTELİ ÇARPMA DEVRESİ TASARIMI

Kenan BAYSAL

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Tez Danışmanı: Yrd. Doç. Dr. Deniz TAŞKIN

EDİRNE-2015

T.Ü. Fen Bilimleri Enstitüsü onayı

Prof. Dr. Mustafa ÖZCAN
Fen Bilimleri Enstitüsü Müdürü

Bu tezin Yüksek Lisans tezi olarak gerekli şartları sağladığını onaylarım.

Prof. Dr. Yılmaz KILIÇASLAN
Anabilim Dalı Başkanı

Bu tez tarafımca okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

Yrd. Doç. Dr. Deniz TAŞKIN
Tez Danışmanı

Bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından Bilgisayar Mühendisliği Anabilim Dalında bir Yüksek Lisans tezi olarak oy birliği ile kabul edilmiştir.

Jüri Üyeleri	İmza
Yrd. Doç. Dr. Deniz TAŞKIN	
Yrd. Doç. Dr. Tarık YERLİKAYA	
Yrd. Doç. Dr. Erdinç UZUN	

T.Ü. FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ YÜKSEK LİSANS PROGRAMI
DOĞRULUK BEYANI

İlgili tezin akademik ve etik kurallara uygun olarak yazıldığını ve kullanılan tüm literatür bilgilerinin kaynak gösterilerek ilgili tezde yer aldığını beyan ederim.



12/06/2015

Kenan BAYSAL

Yüksek Lisans Tezi
Şifreleme İşlemleri İçin Fpga İle Yüksek Kapasiteli Çarpma Devresi Tasarımı
T.Ü. Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

ÖZET

Bilgisayar ile şifrelemede yüksek güvenlik sağlayan algoritmalar, çok büyük boyutta veriler ile çalışmaktadır. Yüksek kapasiteli verilerin gerçek zamanlı olarak işlenebilmesi için gelişmiş donanım yapılarına ihtiyaç duyulmaktadır. Günümüzde ihtiyaç duyulan donanım yapısı, Alanda Programlanabilir Kapı Dizileri (FPGA) kullanılarak FPGA öncesi eski tasarım yöntemlerine göre daha kolay tasarlanabilmektedir. Son on yılda, FPGA hızları, kapasiteleri ve tasarım araçları iyileştirilmiştir. Bu sayede yüksek kapasite veri işleyebilen donanımlar düşük maliyetler ile tasarlanıp üretilebilmektedir.

Bu tez çalışmasının amacı FPGA ile şifreleme algoritmalarında da kullanılabilecek yüksek kapasitede veri işleyebilen yüksek hızlı bir aritmetik ünitenin bileşenlerini oluşturmaktır.

Bu tez çalışmasında, çarpma işlemi algoritmaları incelenmiş, yüksek hızlı çarpma devresini oluşturan yüksek kapasiteli toplama devresi ve başvuru tabloları Çok Yüksek Hızlı Birleşik Devreler İçin Donanım Tanımlama Dili (VHDL) kullanılarak tasarlanmıştır. Tasarlanan devre ISE Design Suite 14.7 yazılımı ile sentezlenmiştir. Benzetim sonuçları ModelSIM ve ISIM programlarından kullanılarak elde edilmiştir.

Yıl : 2015
Sayfa Sayısı : 75
Anahtar Kelimeler : FPGA, VHDL, look up table, çarpma, toplama

Master's Thesis
High Capacity Multiplier Unit Design with FPGA for Cryptographic Operations
Trakya University Institute of Natural Sciences
Department of Computer Engineering

ABSTRACT

Algorithms providing high security in computer encryption are processed with high capacity data. Advanced hardware structures are needed for real time processing of high capacity data. Nowadays, needed hardware structure can be designed easier than old design methods using Field Programmable Gate Arrays (FPGA). In the last decade, speed, capacity and design tools of FPGAs are improved. Thus, high capacity data processing hardware can be designed and manufactured with lower costs.

The aim of this thesis study is to build components of a high capacity arithmetic unit that can process high capacity data which can also be used in encryption algorithms with FPGA.

In this thessis work, multiplication algorithms are studied, high capacity adder circuit and look up tables which together constitute high speed multiplication circuit are designed with Very High Speed Integrated Circuit Hardware Description Language (VHDL). Designed circuit is synthesized with ISE Design Suite 14.7 software. Simulation results are obtained using ModelSIM and ISIM softwares.

Year : 2015
Number of Pages : 75
Keywords : FPGA, VHDL, look up table, multiplication, adder

TEŞEKKÜR

Yüksek lisans eğitimim boyunca bana destek olan, çalışmalarımın her aşamasında yakın ilgi gösteren, bilgi birikimi ile katkı sağlayan ve önerileri ile beni yönlendiren, Yrd. Doç. Dr. Deniz TAŞKIN'a teşekkürlerimi sunarım.

Tezin hazırlanmasında önerileri ile katkıda bulunan iş arkadaşım Öğr. Gör. Dr. Murat Olcay ÖZCAN'a teşekkür ederim.

Sonuçların değerlendirilmesinde ve tezin düzenlenmesinde önerileri ile katkıda bulunan Yrd. Doç. Dr. Uğur ÇİNİ'ye teşekkür ederim.

Beni yetiştiren, hayatım boyunca aldığım kararlarımı destekleyen ve yardımcı olan annem Sema BAYSAL ve babam Mehmet BAYSAL'a teşekkür ederim.

İÇİNDEKİLER

ÖZET i
ABSTRACT ii
TEŞEKKÜR iii
İÇİNDEKİLER iv
SİMGELER ve KISALTMALAR vi
ŞEKİLLER LİSTESİ vii
TABLOLAR LİSTESİ ix
BÖLÜM 1: GİRİŞ 1
BÖLÜM 2: ALANDA PROGRAMLANABİLİR KAPI DİZİLERİ (FPGA) 3
2.1. FPGA Kullanım Alanları 4
2.1.1. Havacılık Alanı 4
2.1.2. Süper Bilgisayar Alanı 4
2.1.3. Görüntü İşleme Alanı 5
2.1.4. Yüksek Kapasiteli Veri Şifreleme Alanı 5
2.2. Programlanabilir Mantık Aygıtları (Programmable Logic Device -PLD) Çeşitleri 5
2.2.1 SPLD (Simple Programmable Logic Device) 6
2.2.1.1. Programlanabilir Mantık Dizileri - PLAs 6
2.2.1.2. Programlanabilir Dizi Mantık – PAL 7
2.2.2. CPLD 8
2.3. FPGA 9
2.3.1. FPGA ile Tasarım 10
2.3.2. FPGA Türleri 11
2.3.2.1. SRAM Tabanlı 12
2.3.2.2. Anti Sigorta Tabanlı 13
2.3.2.3. EEPROM / FLASH Tabanlı 13
2.3.3. FPGA İç Yapısı 14
2.3.3.1. Mantık Blokları 15
2.3.3.2. Matris Anahtarlamalı Yönlendirme Yolları 16
2.3.3.3. Giriş/Çıkış Blokları 17

2.3.3.4. Özel Amaçlı İşlev Blokları 18
2.3.3.4.1. Bellek Blokları 18
2.3.3.4.2. Sayısal Sinyal İşleme (DSP) Blokları 18
2.3.3.4.3. Dahili İşlemci Bloğu 18
BÖLÜM 3: ÇOK YÜKSEK HIZLI TÜMLEŞİK DEVRE DONANIM TANIMLAMA DİLİ - VHDL 19
3.1. VHDL Tasarım Yapısı 21
3.2. VHDL Kod Yapısı 21
3.3. VHDL İle 16 Bitlik Aritmetik Lojik Ünite Tasarımı 28
BÖLÜM 4: ÇARPMA İŞLEMİ ALGORİTMALARI 33
4.1. Sıralı Kaydır/Topla Çarpma 34
4.2 Booth Algoritması 34
4.3. Wallace Ağacı 36
4.4. Dizin Çarpma 38
4.3.Karatsuba Çarpma Algoritması 39
BÖLÜM 5: LOOK UP TABLOSU KULLANARAK YÜKSEK KAPASİTELİ ÇARPMA DEVRESİ TASARIMI 41
5.1. Look-Up Tablosu ile Çarpım 41
5.2. n Bit Yüksek Hızlı LUT Çarpma Devresi Tasarımı 43
5.3. 32 Bit Yüksek Hızlı LUT Çarpma Devresinin VHDL ile Gerçekleştirilmesi 45
5.3.1. 64 Bit Yüksek Hızlı Toplayıcı Devresi Tasarımı 45
5.3.2. Look Up Tablosu Tasarımı 49
5.3.3. Yüksek Hızlı LUT Çarpma Devresi Tasarımı 52
5.3.4. 32 Bit Yüksek Hızlı LUT Çarpma Devresinin FPGA Uygulaması 56
BÖLÜM 6: SONUÇLAR 59
EKLER 61
EK-A 32 bit Yüksek Hızlı LUT Çarpma Devresi Vivado programı ile elde edilen RTL şeması 61
EK-B Ana program TestBench VHDL kodları 62
EK-C Sentez Raporu Özeti 64
KAYNAKLAR 71
ÖZGEÇMİŞ 74
BİLİMSEL ÇALIŞMALAR 75

SİMGELER ve KISALTMALAR

ALU	: Arithmetic Logic Unit
BRAM	: Block Random Access Memory
CLK	: Clock
CMOS	: Complemantary Metal Oxide Semiconductor
CPLD	: Complex Programmable Logic Device
DSP	: Digital Signal Processing
DR	: Data Register
EPROM	: Erasable Programmable Read Only Memory
FF	: Flip Flop
FPGA	: Field Programmable Gate Array
GAL	: Gate Array Logic
G/Ç	: Giriş- Çıkış
JTAG	: Joint Test Action Group
LSB	: Least Significant Bit
LUT	: Look Up Table
MSB	: Most Significant Bit
MUX	: Multiplexer
PAL	: Programmable Array Logic
PCB	: Printed Circuit Board
PLA	: Programmable Logic Array
PLD	: Programmable Logic Device
PROM	: Programmable Read Only Memory
RSA	: Rivest Shamir Adleman
RTL	: Register Transfer Level
SPLD	: Simple Programmable Logic Device
SRAM	: Static Random Access Memory
TTL	: Transistor-Transistor Logic
VHDL	: Very High Speed Integrated Circuit Hardware Description Language

ŞEKİLLER LİSTESİ

Şekil 2.1. Amiga Grafik işlemcisi ilk örnek tasarım modeli 3
Şekil 2.2. Programlanabilen Mantık 5
Şekil 2.3. Programlanmamış PLA iç yapısı 6
Şekil 2.4. Programlanmış PLA iç yapısı ve devre fonksiyonu 7
Şekil 2.5. Programlanmamış PAL içyapısı 7
Şekil 2.6. CPLD Mimarisi 8
Şekil 2.7. FPGA blok yapısı 9
Şekil 2.8. 2010 yılı FPGA üreticileri ve pazar payları 11
Şekil 2.9. FPGA'da kullanılan SRAM yapısı 12
Şekil 2.10. Temel Mantık Bloğu İç Yapısı 15
Şekil 2.11. Örnek bir mantık devresinin mantık bloğu içerisinde gerçekleştirilmesi. 16
Şekil 2.12. Matris Anahtarlamalı Yönlendirme Yolları genel yapısı 17
Şekil 2.13. Altera Cyclone FPGA'nın G/Ç blok yapısı 17
Şekil 3.1. VHDL tasarım aşamaları 20
Şekil 3.2. Temel VHDL Kod Yapısı 21
Şekil 3.3. Örnek AND - OR Mantık devresi 24
Şekil 3.4. AND - OR devresinin ISE programında uygulanması ile oluşan bileşen yapısı 26
Şekil 3.5. AND - OR devresinin ISE programında oluşan RTL şeması 26
Şekil 3.6. AND - OR devresinin ISE programı ile elde edilmiş teknoloji şeması 27
Şekil 3.7. And - Or devresinin ISIM programında görüntülenen benzetim sonuçları 27
Şekil 3.8. Aritmetik Lojik Ünitenin komşu birimlerle olan bağlantısı 28
Şekil 3.9. 16 bit Aritmetik mantık biriminin ISIM benzetim sonuçları 32
Şekil 4.1. Temel çarpma işlemi 33
Şekil 4.2. Sıralı / kaydır topla yöntemi 34
Şekil 4.3. 8 bit iki sayının çarpımında kısmi parçaların elde edilmesi 37
Şekil 4.4. Wallace ağacı 8 bit uzunluğunda iki sayının çarpımı blok yapısı 37
Şekil 4.5. 4 bitlik iki sayının dizin çarpım yöntemiyle çarpımı blok şeması 38
Şekil 4.6. Geleneksel çarpma ile Karatsuba çarpma algoritmasının karşılaştırılması 40

Şekil 5.1. 6 bit iki sayının 3 bit look-up tablosu ile çarpma işlemi.....42
Şekil 5.2. Kaydırma ve toplama işlemi.....44
Şekil 5.3. VHDL ile oluşturulan 32 Bit Çarpma devresinin temel blok şeması.....45
Şekil 5.4. 64 Bit Toplayıcı devresinin ISE programı ile elde edilmiş RTL şeması.....47
Şekil 5.6. 1024 Bit toplayıcı ModelSIM benzetim sonuçları.....48
Şekil 5.5. 64 Bit Toplayıcının ISIM programı ile elde edilen benzetim sonuçları.....48
Şekil 5.7. Look Up Tablosunun ISIM ile elde edilen benzetim sonuçları.....50
Şekil 5.8. LUT devresinin ISE programı ile elde edilmiş RTL şeması.....51
Şekil 5.9. 32 Bit LUT Çarpma devresinin blok şeması.....52
Şekil 5.10. Geçerli duruma göre giriş çıkış ve sinyal bağlantı durumları.....55
Şekil 5.11. 32 bit LUT Çarpma Devresinin ISIM programı ile elde edilmiş benzetim sonuçları.....56
Şekil 5.12. 32 bit LUT Çarpma devresinin ModelSIM benzetim sonuçları.....57
Şekil 5.13. Kaydediciden kaydediciye devrenin zamanlama analizi şema görüntüsü....58

TABLOLAR LİSTESİ

Tablo 2.1. Programlanabilir aygıtlar kapasite ve teknoloji yönünden karşılaştırılması . 10
Tablo 2.2. Üretici firmalar ve tercih ettikleri üretim teknolojileri 11
Tablo 2.3. Programlama teknolojilerinin karşılaştırılması 14
Tablo 3.1. Temel VHDL Terimleri 22
Tablo 3.2. VHDL Entity Yapısı 23
Tablo 3.3. VHDL Mimari Yapısı 23
Tablo 3.4. AND ve OR kapılarının işlevlerinin tanımlandığı alt programların VHDL kodları 24
Tablo 3.5. AND-OR Devresinin VHDL Kodları 25
Tablo 3.6. Aritmetik Lojik İşlemler ve seçim kodları 28
Tablo 3.7. Giriş ve Çıkış portları kısaltmaları ve bit uzunluukları 29
Tablo 3.8. Giriş Çıkış portlarının VHDL'de entity yapısının altında tanımlanması 29
Tablo 3.9. Aritmetik Lojik Ünite VHDL Kod Yapısı 30
Tablo 3.10. 16 bit işlemcinin bileşenlerinin tanımlanması ve portlarının birbirine bağlanması 31
Tablo 5.1. 3 bitlik iki sayının çarpımı Look Up Tablosu 41
Tablo 5.2. 1024 bit veri girişi için farklı parça uzunluklarında look-up tablosu ve adım sayısı 45
Tablo 5.3. 64 Bit Toplayıcı devresinin VHDL kodları g/ç bağlantıları ve bit genişlikleri 46
Tablo 5.4. Bir bitlik tam toplayıcı VHDL kodları 46
Tablo 5.5. 64 Bitlik tam toplayıcının VHDL kodları 47
Tablo 5.6. Look up tablosunun oluşturulması için kullanılan pseudo kodları 50
Tablo 5.7. Virtex 5 xc5vlx50t-2ff1136 üzerinde mantık parçaları kullanımı 51
Tablo 5.8. LUT ile genişletilebilir çarpma devresi kod yapısı 54
Tablo 5.9. Çeşitli FPGA donanımları ile 32 bit LUT çarpma devresinin performansı .. 58
Tablo 5.10. Virtex 5 xc5vlxt-2ff1136 üzerinde güç tüketim analizi 58
Tablo 6.1. Zaman Karmaşıklığı Karşılaştırması 59

BÖLÜM 1

GİRİŞ

Bilgisayarların sayısal işlem gücünün fiziksel sınırlarına yaklaşması ile birlikte, işlemcilerin komut yapısı ve bu komutları işleme teknikleri önem kazanmıştır [1]. Temelde işlemcilerin gerçekleştirdiği işlemler aritmetik ve mantık komutlarına dayanmaktadır [2]. Toplama, çarpma gibi aritmetik işlemlerinin gerçekleştirilme hızları, işlemcinin hızına ve bilgi işleme kapasitesine doğrudan etki etmektedir.

Son yıllarda sanayi, savunma ve finans sektörlerinin iletişim ve iş yükünü sanal ortama taşıması ile birlikte bilgisayarda güvenlik ve şifreleme geçmişte olduğundan daha fazla önem kazanmıştır. Geliştirilen en son teknoloji güvenlik donanım ve yazılımları siber saldırılar karşısında zaman zaman yetersiz kalmaktadır. Güvenliğin kusursuz bir şekilde sağlanabilmesi için daha yüksek kapasitede veri işleyebilen özelleştirilmiş donanımların geliştirilmesi gerekmektedir. Sayısal işaret işleme sistemlerindeki kapasite ve karmaşıklığın artması ile birlikte tasarımcılar gelişmiş bilgisayar tasarım araçlarına ihtiyaç duymaktadır [3].

Geleneksel donanım tasarım yöntemlerinde, transistorun bilgisayar işlemci tasarımında kullanılması ile ilk örnek işlemci tasarımları bir devre kartı üzerine yerleştirilen transistorların kablolar ağı ile birbirine bağlanması şeklinde yapılmıştır. Bu basit yöntem, işlemci kapasitelerin artması ile birlikte karmaşık bir hal almış, ilk örnek tasarımı gerçekleştirmek zorlaşmıştır. Tasarım seviyesi transistor seviyesinden mantık kapıları seviyesine yükseldiyse de zamanla içinden çıkılmaz bir hal almıştır. Gerçekleştirilen ilk örnek tasarım devre kartı istenilen sonucu vermediği takdirde devre üzerinde birbiri ile bağlı olan kablolarda köklü bağlantı değişiklikleri yapılması gerekmekteydi. Yapılan değişikliklere rağmen istenilen sonuç elde edilememişse devrenin tekrar oluşturulmaktaydı. Bu durum hem para hem zaman kaybına sebep olmaktaydı. Üstelik bu işlemlerin karmaşıklığı, tasarım yapacak veya mevcut tasarım üzerinde geliştirme yapabilecek mühendis yetiştirmeyi de zorlaştırmaktaydı.

Transistor kullanılarak gerçekleştirilen elektronik devre tasarımlarında FPGA (Field Programmable Gate Arrays: FPGA) kullanılmaya başlanması ile birlikte karmaşık devre tasarımları daha kolay ve hızlı yapılabilir hale gelmiştir. 1980'li yılların ortalarında Altera, Xilinx, Actel gibi firmaların FPGA üretimine başlaması ve daha kolay kullanılabilir tasarım ve benzetim araçları geliştirmesi ile birlikte daha alt seviyede tasarımcılar düşük bütçeler ile FPGA donanımlarını programlamaya başlamıştır.

Donanım tasarımı için yapısal ve davranışsal olarak transistorlar arasındaki bağlantıları oluşturan, analiz edebilen ve benzetim yapabilen bir programlama dili geliştirilebilmesi için çalışmalar yapılmıştır. Bu süreçte birçok ticari firma kendi ihtiyacına göre donanım tanımlama dili geliştirmiştir. FPGA için IEEE standartlarında Verilog ve VHDL (Very High Speed Integrated Circuit Hardware Description Language) geliştirilen diller arasında en yaygın olarak kullanılanlarıdır. Verilog Gateway Design Automationadında özel bir firma tarafından, VHDL ise Amerika Savunma Bakanlığının isteği üzerine geliştirilmiştir. Yazım biçimlerinde ve yapılarında farklılık olmasına rağmen bu iki dilin sınırları birbirlerine yakındır [4].

Bu tezde, yüksek hızlı aritmetik ünitenin alt birimlerini oluşturmak amacıyla, bilgisayarın veri işleme kapasitesi ve hızında önemli bir rolü olan yüksek hızlı çarpma yöntemleri incelenmiştir. Yüksek hızlı genişletilebilir toplayıcı devresi tasarımı yapılmıştır. Yüksek hızlı toplayıcı devresi kullanılarak başvuru tabloları (Look Up Table: LUT) yöntemi ile genişletilebilir yüksek hızlı LUT çarpma devresi tasarımı yapılmıştır. 32x32 bit uzunluğunda LUT çarpma devresi VHDL ile modellenerek FPGA donanımında uygulanmıştır.

BÖLÜM 2

ALANDA PROGRAMLANABİLİR KAPI DİZİLERİ (FPGA)

1970li yıllarda entegre devreli silikon yapılı bilgisayarların tasarımı, mantıksal sistemler ile TTL mantık yongalarından oluşan PCB (printed circuit board) kartlar üzerinde inşa edilmiştir. Fonksiyonların boyutunun artması ve karmaşıklaşması sebebiyle kurulan mantık devrelerinin sayısının artması tasarımın geliştirilmesini kısıtlayıcı bir etken olmuştur. Fakat geleneksel tasarım tekniğinde en önemli sorun, mantık devreleri sayısının artmasının tasarım hızının yavaşlamasına sebep olmasıdır. Tasarımcılar Karnough haritaları veya Quine - McCluskey tekniklerini kullanarak devre sadeleştirme yolunu seçmiştir [5]. FPGA öncesi geleneksel tasarım aşamasına örnek olarak Amiga grafik işlemcisinin tasarım aşaması gösterilebilir. İlk örnek tasarımın ön tasarım işlemleri öncelikle kâğıt üzerinde gerçekleştirilip, PCB kartların üzerinde entegreleri Şekil 2.1'de görüldüğü biçimde kablolara birbirine bağlanmaktadır [6]. İlk örnek tasarım modelinin tasarlandığı biçimde çalışması halinde tasarımdan üretim aşamasına geçiliyordu. Ancak tasarımda bir hata ile karşılaşılması durumunda bütün tasarım aşamaları ilk aşamadan itibaren tekrar ediliyor, bu da para ve zaman kaybına sebep oluyordu [7].



Şekil 2.1. Amiga Grafik işlemcisi ilk örnek tasarım modeli

Bu tür tasarım sorunlarına çözüm olarak Alanda Programlanabilir Kapı Dizileri geliştirilmiştir. FPGA'lar mantık blokları ile bu bloklar arasındaki değiştirilebilir ara bağlantılardan oluşan sayısal tümleşik devrelerden oluşmaktadır. "Alanda programlanabilir" diye isimlendirilmesinin sebebi, üretimin ardından kullanılacağı alanda isteğe yönelik tasarlanabilir özelliğinin olmasıdır. Bir başka tanıma göre FPGA'lar, programlanabilirliği "antifuse", "EPROM" veya "SRAM" teknolojisini temel alan, ana bileşenler arasında programlanabilir bağlantı sağlayan, basit yapıştırıcı mantık teknolojileridir [8].

2.1. FPGA Kullanım Alanları

Teknolojilerindeki sınırlı gelişmelerden dolayı 90'lı yıllara kadar kısıtlı alanlarda kullanılmıştır. Fakat kapasitelerindeki artış ile birlikte endüstriyel alanda devre tasarımı ve seri üretimi ile ilgili gömülü sistem tasarımlarında başarılı sonuçlar elde edilmiştir. Programlanabilir yapısı sayesinde çeşitli algoritmalara düşük maliyet ile donanım altyapısı sağlayarak gömülü sistem tasarım çalışmalarını hızlandırmıştır. Moore yasasının birçok elektronik endüstri alanında öngördüğü gibi FPGA alanında da kullanılan transistor sayısı her 18 ayda bir yaklaşık iki katına çıkarken üretim maliyetinde bir değişiklik olmamıştır [9, 10]. Bununla birlikte bu teknoloji elektronik ve bilgisayar endüstrisinin her alanında yaygın olarak kullanılmaya başlamıştır.

2.1.1. Havacılık Alanı

Maliyet, performans ve esneklik bakımından havacılık alanında sağladığı avantajlar sebebiyle sıklıkla tercih edilmektedir. F-35 Joint Strike Fighter, Boeing 787 Dreamliner ve NASA'nın Mars Rover araçlarında FPGA'lar kokpit görüntü sistemleri, uçuş yönetimi, silah sistem kontrol ve uçuş radarı gibi alanlarda kullanılmıştır [11, 12, 13].

2.1.2. Süper Bilgisayar Alanı

Bilim ve mühendislik alanında standart bilgisayar teknolojilerinin yetersiz kalmasıyla SRC Computers, Cray ve SGI gibi yüksek kapasiteli bilgisayar üreten şirketler ürünlerinde performans arttırmak için programlanabilir donanımları tercih etmişlerdir. Örnek olarak Cray firması geliştirdiği XD1 modelinde 6 adet yüksek

kapasiteli Xilinx FPGA'yı 12 adet x86 işlemci ile birleştirerek bir kasa içerisine yerleştirmiştir [14].

2.1.3. Görüntü İşleme Alanı

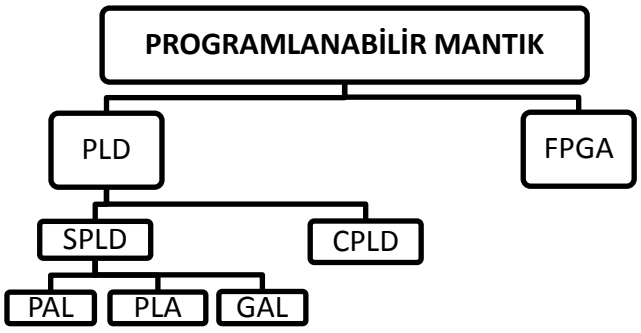
FPGA'lar yapıları gereği, yüksek hızda karmaşık sinyal işleme uygulamaları için uygundur. Görüntü işleme ve yüz tanımlama gibi uygulamaların algoritmaları genellikle büyük çapta matris işlemleri içermektedir. Bu algoritma yapısı FPGA'ların donanım yapısına uygundur [15].

2.1.4. Yüksek Kapasiteli Veri Şifreleme Alanı

Şifrelemede kullanılan yüksek bit kapasiteli kaydırma, toplama veya çoğullama işlemleri için FGPA etkili bir uygulama donanımıdır. FPGA ile, yüksek kapasiteli veri işleme algoritmaları bütünüyle devre üzerinde gerçekleştirilebilir. Modüler çarpma içeren RSA şifreleme gibi açık anahtar şifreleme teknikleri için uygun yapıdadır [16].

2.2. Programlanabilir Mantık Aygıtları (Programmable Logic Device -PLD) Çeşitleri

Programlanabilen mantık aygıtları (PLD), gelişim süreci içerisinde çeşitlere ayrılmıştır. Üretim teknikleri, kullanım alanları, programlama mantığı, programlanabilen mantık aygıtının kullandığı teknoloji gibi sebepler bu çeşitlenmeye sebep olmuştur. Birkaç birleşik işlev içeren küçük ölçekli aygıtlardan, binlerce transistordan oluşan karmaşık yüksek kaliteli aygıtlara kadar oldukça geniş bir çeşit yelpazesi mevcuttur. Şekil 2.2'de görüldüğü gibi programlanabilen mantık aygıtlarını iki ana bölümde incelemek mümkündür [17].



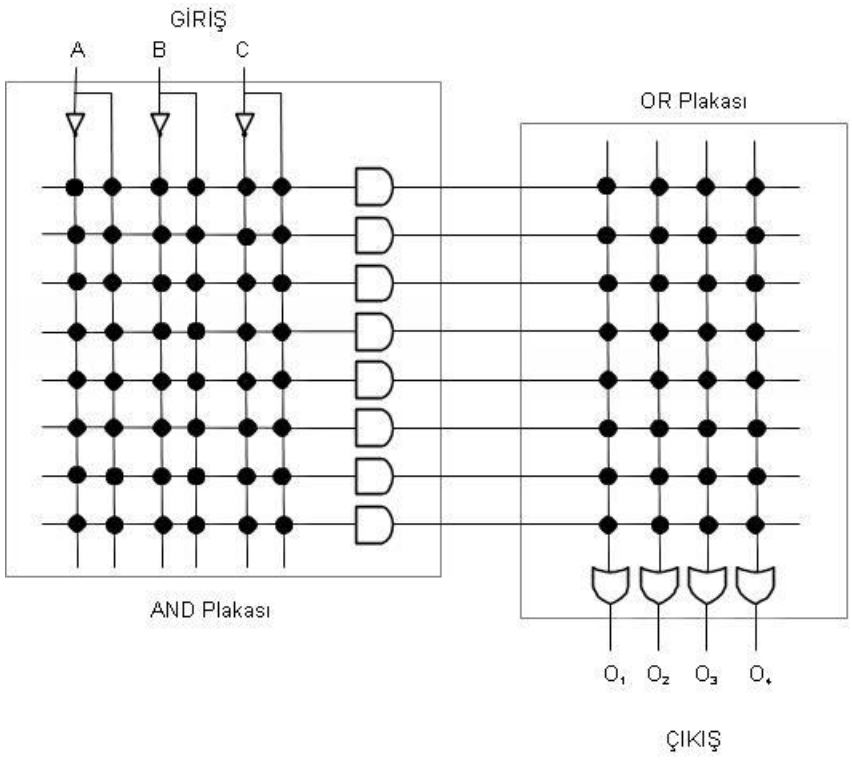
Şekil 2.2. Programlanabilen Mantık

2.2.1 SPLD (Simple Programmable Logic Device)

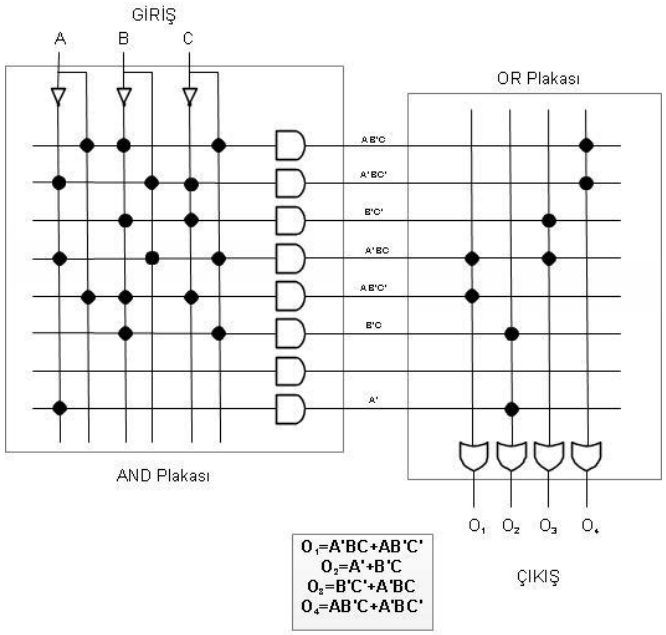
Programlanabilir mantık aygıtların öncü modelleri olarak basit birleşik mantık fonksiyonları içeren, kapasite olarak en düşük, fiyat olarak en ucuz programlanabilen entegre devre modelleridir. SPLD'leri oluşturan hücrelerin her birisinin bir diğer hücre ile doğrudan bağlantısı vardır. PLA, PAL ve GAL yaygın olarak bilinen ve kullanılan çeşitleridir [18].

2.2.1.1. Programlanabilir Mantık Dizileri - PLAs

AND ve OR mantık kapı dizilerinden oluşan düşük kapasiteli, alanda programlanabilir aygıtlardır. PROM (Programmable Read Only Memory) aygıtlarının hız ve giriş kısıtlamalarına çözüm olarak üretilen PLA'lar, PROM'un yapısına oldukça benzer bir yapıdan oluşmaktadırlar [19]. PLA'ların büyüklüğü içerdiği AND ve OR tabakalarının sayısına göre belirlenmektedir. Şekil 2.3'te programlanmamış bir PLA iç yapısı görülmektedir. Şekil 2.4'te ise programlanmış bir PLA'nın yapısı ve devre çıkış fonksiyonları görülmektedir [20].



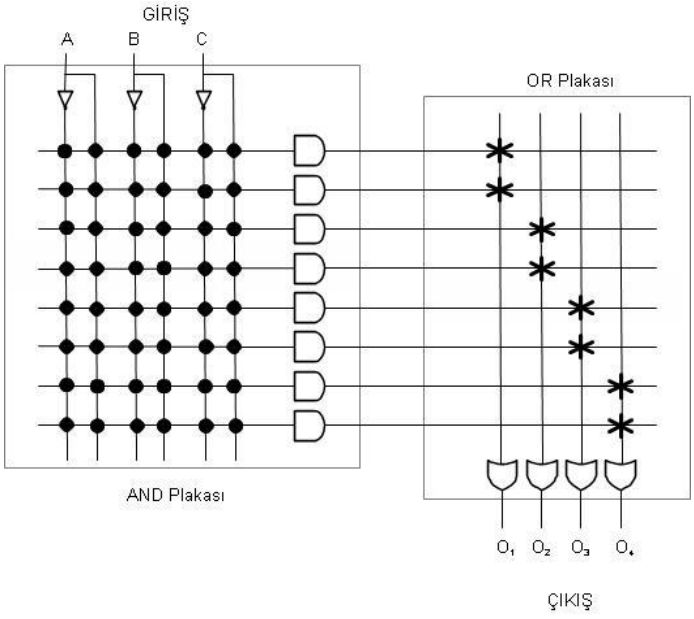
Şekil 2.3. Programlanmamış PLA iç yapısı



Şekil 2.4. Programlanmış PLA iç yapısı ve devre fonksiyonu

2.2.1.2. Programlanabilir Dizi Mantık – PAL

PLA yapısı temel alınarak geliştirilmiş, mevcut yapıyı daha hızlı hale getirmiştir. PLA' dan farklı olarak OR tabakaları önceden programlanmıştır. PLA'lardan farklı olarak programlanabilir yapısında Flip/Flop içermektedir [19].

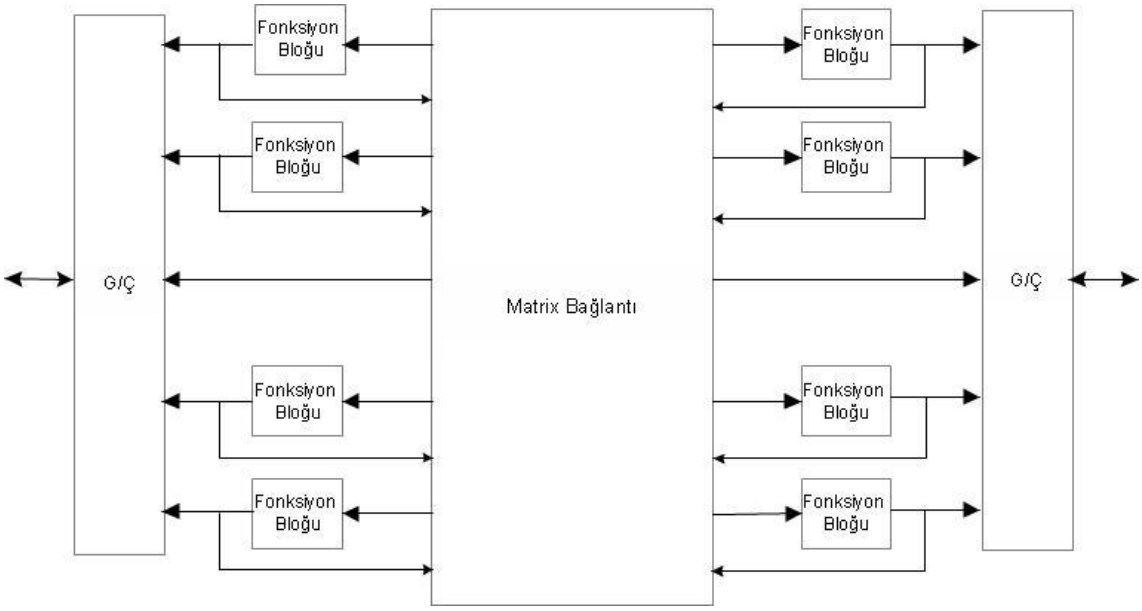


Şekil 2.5. Programlanmamış PAL içyapısı

2.2.2. CPLD

SPLD'ler ile benzer bir yapıda olan, çok sayıda birbiri ile bağlantılı PAL'ın tek bir yonga içerisinde birleşik olarak yer alması ile geliştirilmiş karmaşık programlanabilir mantık aygıtlarıdır. SPLD'lere göre kapasite yönünden en az 2 kat en fazla 64 kat üstündürler. SPLD benzeri yüzlerce hücre bloklarından oluşur ve her blok kendi içerisinde aynı fonksiyon hücrelerinin gruplandırılmasıyla meydana gelmektedir.

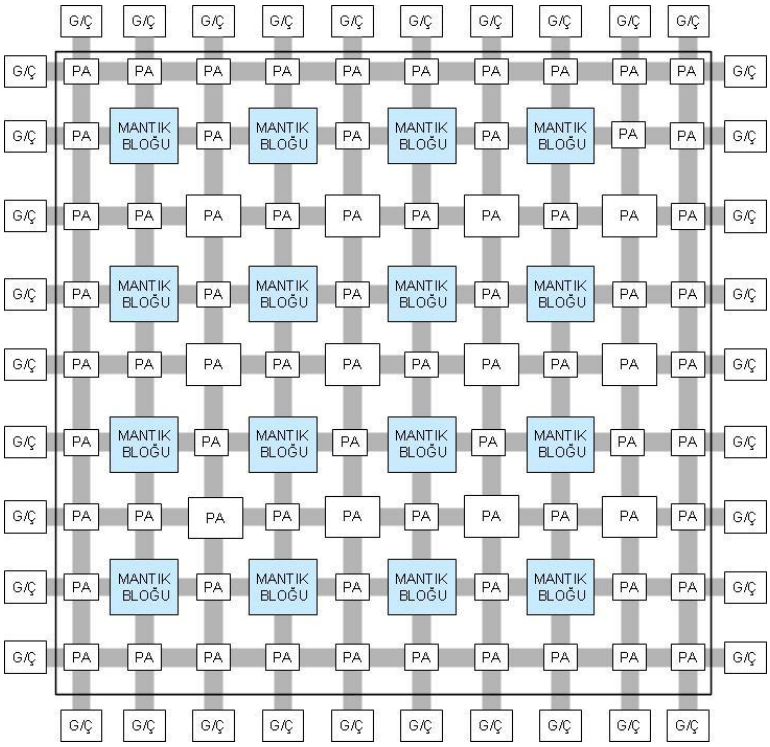
Şekil 2.6'da görüldüğü gibi bütün bloklar birbirlerine programlanabilir matris yollar ile bağlantılıdır [21]. CPLD'lerin hızları bloklar arasındaki bağlantı teknolojilerine göre farklılık göstermektedir. FPGA'lar ile temel mimari ve yarı iletken teknolojileri bakımından farklılık göstermektedir. FPGA'lardan farklı olarak, kalıcı biçimde programlanabilmektedirler ve harici bir yapılandırma ROM'u kullanılmasına ihtiyaç yoktur.



Şekil 2.6. CPLD Mimarisi

2.3. FPGA

Bir FPGA; Şekil 2.7'de görüldüğü gibi; genel mantık blokları ve bu blokları çevreleyip giriş-çıkış birimleri ile arasında bağlantı sağlayan programlanabilir devre yollarından meydana gelmektedir [22]. Bir FPGA yongası, programcı tarafından yazılan kodlar ile programlanır. Derleyici program yardımı ile yonganın içerisindeki bloklar ve yollar arasında bağlantı kurularak, kodları yazılan devre gerçekleştirilir.



Şekil 2.7. FPGA blok yapısı

FPGA'lar yüksek kapasiteleri, karmaşık mimarileri ve daha fazla kaydedici içermesi ile SPLD'lere ve CPLD'lere göre üstünlük gösterir. Tablo 2.1'de programlanabilir aygıtların kapasite ve teknolojileri bakımından karşılaştırmaları görülmektedir [23]. Programlanabilir aygıtlarda kapasiteyi belirleyen en önemli unsur mantık hücresidir. Tablo 2.1'de verilen değerlerde FPGA'ların CPLD'lere göre en az 10 kat daha fazla kapasiteye sahip olduğu görülmektedir.

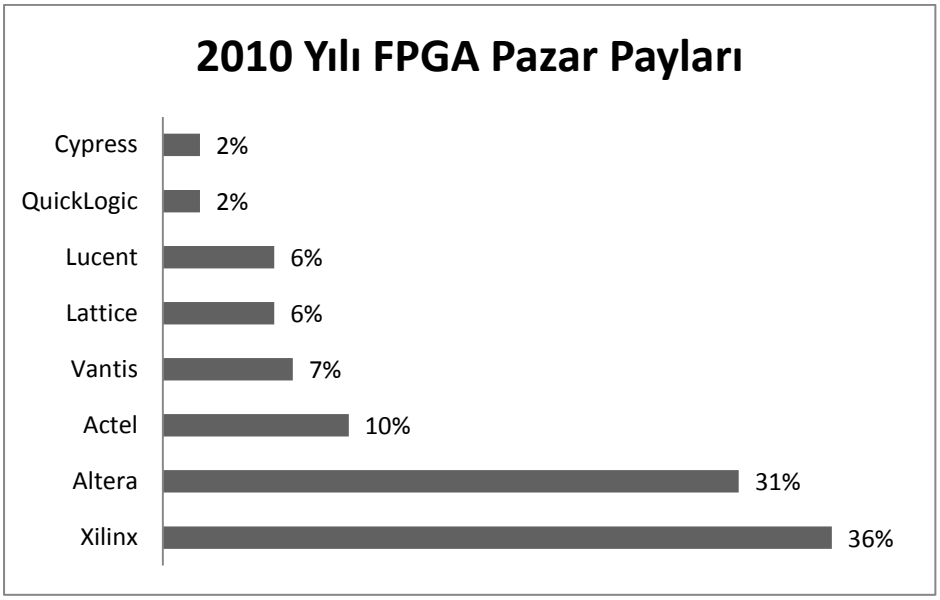
Tablo 2.1. Programlanabilir aygıtlar kapasite ve teknoloji yönünden karşılaştırılması

SPLD	CPLD	FPGA
16-28 arası pin içerir	44-300 arası pin içerir	En az 50 pin içerir
8-24 arası mantık hücresi içerir	32-500 arası mantık hücresi içerir	5000 den fazla mantık hücresi içerir
8-24 Flip Flop içerir	32-500 Flip Flop içerir	5000 den fazla Flip Flop içerir
Yapılandırma teknolojisi olarak EPROM, EEPROM altyapısını kullanır.	Yapılandırma teknolojisi olarak EPROM, EEPROM, FLASH altyapısını kullanır.	Yapılandırma teknolojisi olarak FLASH, EEPROM altyapısını kullanır.
Program enerji kesildikten sonra silinmez, kalıcıdır.	Program enerji kesildikten sonra silinmez, kalıcıdır.	SRAM tabanlı olanlarda enerji kesildikten sonra program silinir.
Program silindikten sonra tekrar programlanabilir.	Tekrar programlanabilir.	SRAM tabanlı olanlar tekrar programlanabilir.
Devre dışından özel araçlarla programlanabilir.	Devre içinde programlanabilir.	Devre içinde programlanabilir.
Boyutu küçüktür.	Boyutu ortadır.	Boyutları; orta boy ve büyük boy
-	900-20000 arası eşdeğer kapı içerir	10000'den fazla eşdeğer kapı içerir

2.3.1. FPGA ile Tasarım

Kullanıcı tasarladığı devreyi VHDL veya Verilog gibi donanım tanımlama dillerinden birini kullanarak yazılım diline döker. Projenin gerçekleştirileceği aygıtın türüne uygun olan bir derleyici program ile yazılan komutlar sentezlenir. İsteğe bağlı olarak bir benzetim programında devrenin güvenilirliği incelenir. Doğrudan PC'ye USB veya seri port yardımı ile bağlanabilen FPGA donanımına, sentezlenen komutlar uygun bir ara yazılım ile aktarılır. Yazılımın çalışmasına uygun olan devre FPGA içerisinde gerçekleştirilir ve giriş çıkış birimleri ile kullanıma uygun hale gelir.

Şekil 2.8'de SourceTech411.com web sitesinin yayınladığı 2013 yılında yayınladığı FPGA üreticileri ve 2010 yılı için pazar payları verilmiştir [24].



Şekil 2.8. 2010 yılı FPGA üreticileri ve pazar payları

2.3.2. FPGA Türleri

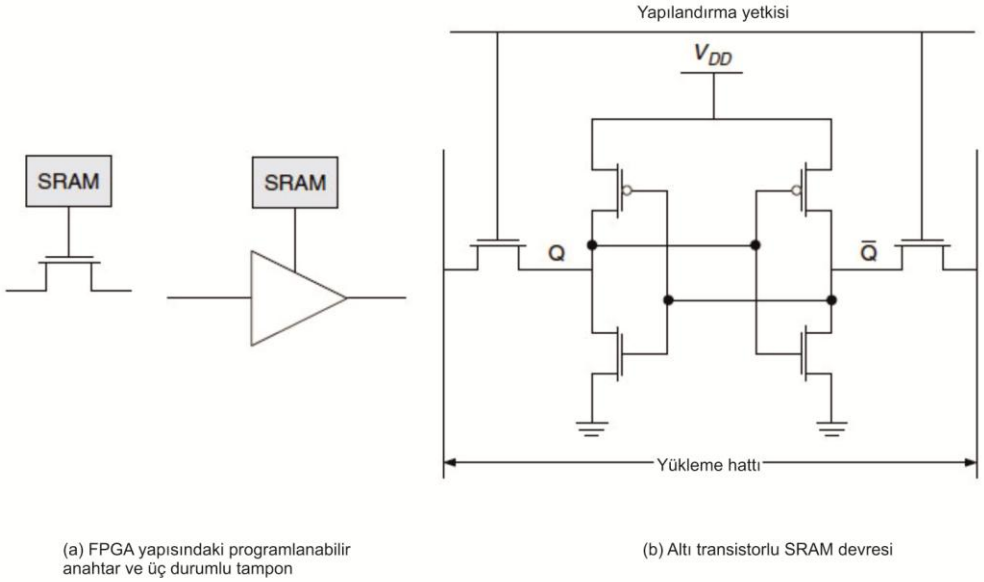
FPGA'lar "tekrar programlanabilir" ve "bir kez programlanabilir" olmak üzere iki farklı türde üretilmektedir. Yapılandırılma biçimlerindeki bu ayrım üretimde farklı yöntemlerin temel alınmasına sebep olmuştur. FPGA'lar temel aldıkları üretim teknolojileri bakımından SRAM tabanlı, Anti-Sigorta tabanlı, EEPROM tabanlı ve Flash tabanlı olarak sınıflandırılabilir. Tablo 2.2'de üretici firmalar ve tercih ettikleri üretim teknolojileri verilmiştir [23].

Tablo 2.2. Üretici firmalar ve tercih ettikleri üretim teknolojileri

Üretici Firma	Teknoloji
Altera	SRAM, FLASH
Actel	Anti-Sigorta
Lattice	SRAM, FLASH
QuickLogic	Anti-Sigorta
Xilinix	SRAM

2.3.2.1. SRAM Tabanlı

FPGA üretiminde en çok tercih teknoloji edilen CMOS tabanlı üretimdir. SRAM tipi olan FPGA'lar defalarca programlanabilmektedir. Ayrıca SRAM tabanlı FPGA'lar devre üzerinde yerleşik durumdayken programlanabilir. FPGA'ların araştırma ve geliştirme alanlarındaki kullanım amacı göz önüne alındığında ilk örnek tasarımı ve geliştirilmesi için en iyi seçenek olarak değerlendirilebilir. SRAM tabanlı bir FPGA'da tipik olarak Look-Up Tablosu (LUT) ve D tipi Flip Flop bulunur. Şekil 2.9'da FPGA'da kullanılan biir SRAM bağlantı yapısı görülmektedir [25].



Şekil 2.9. FPGA'da kullanılan SRAM yapısı

Statik yapısı sebebiyle güç kesildiğinde içerisindeki veriler silinir bu yüzden her açılışta SRAM tabanlı bir FPGA'nın tekrar yapılandırılması gerekir. Bu yapılandırma programın büyüklüğüne bağlı olarak birkaç yüz milisaniye sürer. Her açılışta tekrar yapılandırma verilerinin saklanması için PROM tipi bir bellek içerir. Ancak her türlü kar getiren ürünün kopyalandığı bir pazarın olduğu günümüzde PROM içerisindeki yapılandırma programı rahatlıkla alınıp kopyalanabilir. Buradaki asıl sorun programın alınıp kopyalanmasından ziyade, ters mühendislik uygulanarak mevcut program yapısının geliştirilmesi ve asıl üretimi yapan firmanın haksız rekabete uğramasıdır. Bu

ciddi güvenlik sorununa önlem olarak PROM içersine yeniden yapılandırma programı şifrelenerek yazılır. Şifre anahtarı da FPGA'nın JTAG portu aracılığıyla SRAM tabanlı kaydedici içerisinde barındırılır. Şifre anahtarının SRAM tabanlı kaydedici içerisinden silinmemesi için küçük bir güç kaynağı olarak devreye bir pil ilave edilmesi gerekir. Bu durumda devrenin boyut, ağırlık ve maliyet gibi faktörlerden yana olumsuz etkilenmesine sebep olur.

2.3.2.2. Anti Sigorta Tabanlı

SRAM tabanlı FPGA'ların aksine, yerleşik bulunduğu devre üzerinden ayrı bir başka programlayıcı aygıt ile programlanabilirler. Sistemden güç kesildiğinde içerisindeki program silinmez. Bu sebeple devre üzerinde harici bir hafıza birimi ve güç kaynağı gerektirmez. Olağan yapısından dolayı ortamdaki radyasyondan etkilenmezler. En önemli avantajlarından biriside güvenlik yönündendir. İçerisine yazılan yapılandırma programı FPGA'nın ulaşılamayacak kadar derinlerinde yer alır. SRAM'ın aksine ulaşılması zordur ve kopyalanamaz.

Sistem bekleme durumundayken aynı özelliklerde bir SRAM tabanlı ürüne göre yaklaşık yüzde 20 oranında az güç tüketir. Aktif çalışma durumundayken de yine SRAM'e göre az güç tüketir [26].

Ancak üretim aşamasından sonra programlama için ayrı bir aygıt gerektirmesi ve sadece tek sefer programlanabilmesi gerekçelerinden dolayı tercih olarak SRAM tabanlı mimarini gerisinde kalmıştır.

2.3.2.3. EEPROM / FLASH Tabanlı

SRAM tabanlı mimariye benzer bir yapısı vardır. Devreden ayrı başka bir programlayıcı aygıt aracılığıyla programlanabilir. Ancak bazı çeşitleri devre üzerinde programlanabilmesine rağmen programlama süresi SRAM tabanlı mimariye göre oldukça uzundur. Üzerindeki veri kalıcı olarak saklanır. Sistem aktif duruma getirildiği gibi kullanıma hazırdır.

Program güvenliği için çoklu anahtar ile şifreleme kullanılır. Bu anahtarın bit genişliği 50 bitten fazladır. Bu da programın şifre anahtarının kırılmasını neredeyse imkansız kılmaktadır. Yapısında kullanılan transistor boyutları SRAM içerisinde kullanılan transistorlara göre küçüktür. Bu durumda içerisindeki mantık birimleri

birbirine daha yakın paketlenebilir, bu da paket içerisindeki yollardan kaynaklanan gecikmeleri azaltır.

Tablo 2.3. Programlama teknolojilerinin karşılaştırılması [27]

	SRAM	**Anti-Sigorta**	**EEPROM/FLASH**
Teknoloji	En yüksek düzeyde	Bir veya daha fazla nesil geride	Bir veya daha fazla nesil geride
Tekrar Programlanabilirlik	Var (sistem içerisinde)	Yok	Var (sistem içerisinde veya dışında)
Tekrar Programlama Hızı	Hızlı	-	SRAM'in 3 katı daha yavaş
Geçicicilik (Sistem açılışında yeniden programlanma gerekliliği)	Var	Yok	Yok
Ar-Ge kullanımı	Prototip geliştirmede sık tercih edilir.	Yok	Kabul edilebilir düzeyde kullanılabilir.
Anlık hızlı açılış	Yok	Var	Var
Program güvenliği	Bitstream şifreleme kullanıldığı zaman kabul edilebilir düzeyde	Çok Yüksek	Çok Yüksek
Yeniden yapılandırma hücre boyu	Büyük (6 transistor)	Çok Küçük	Orta-Küçük (2 transistor)
Güç tüketimi	Orta	Düşük	Orta
Hızlı Uygulama Geliştirme zorluğu	Yok	Var	Yok

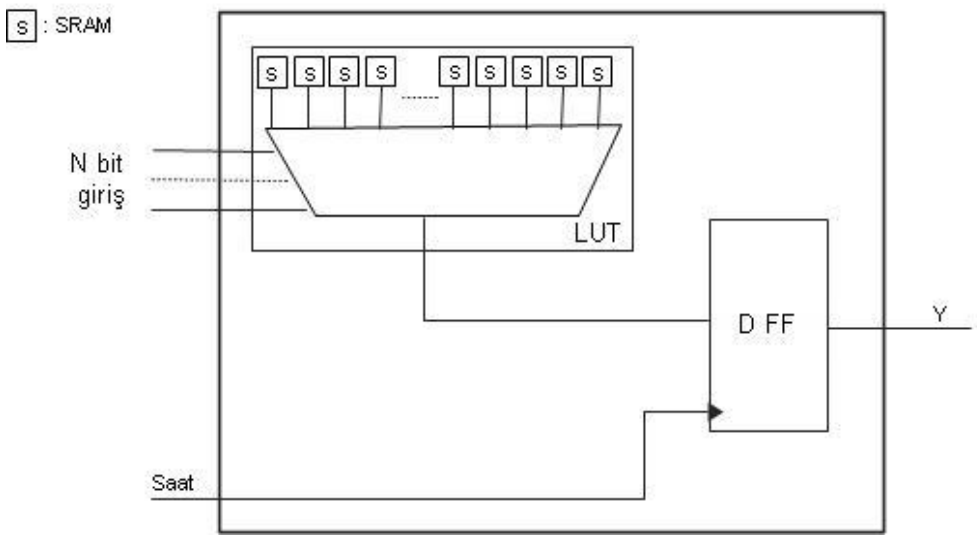
2.3.3. FPGA İç Yapısı

Yaygın olarak kullanılan SRAM tabanlı bir FPGA, yapılandırılabilir türde ortak yapılardan oluşur. Kullanılan yapılar genel mimarisi bakımından üretici firmalara ve aygıt türlerine göre farklılık gösterse de kullanım amacı ve işlevi bakımından genel olarak ortaktır.

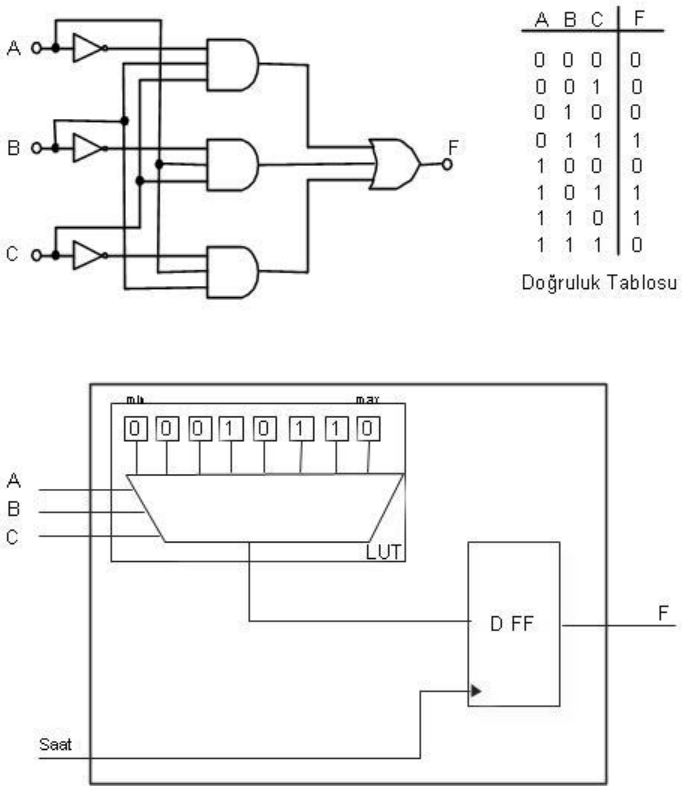
2.3.3.1. Mantık Blokları

Mantık blokları bir FPGA'yı oluşturan ana bileşenlerdir. FPGA'nın programlandığı işlevi yerine getirmesini sağlayan ana devre elemanıdır. Firmalara göre "mantık birimi", "mantık hücresi", "makro hücresi", "dilim" gibi farklı şekillerde isimlendirilebilir, ancak temelde gerçekleştirdiği fonksiyon mimari yapıda farklılık gösterse de aynıdır.

Genel mimari bakımından çoklayıcı (MUX) tabanlı ve Look Up Tablosu (LUT) tabanlı olmak üzere kullanılan iki farklı tip mantık bloğu türü vardır. Günümüz FPGA'larında LUT tabanlı mantık blokları kullanılmaktadır. Tipik bir mantık bloğu N bit girişli Look-Up tablosu ve D tipi Flip Flop'tan oluşur. LUT, basit bir hafıza birimidir ve bu biriminin büyüklüğü FPGA'nın kapasitesini belirleyen en önemli etken unsurdur. Her LUT birimi n bit veya daha az girişli olan herhangi bir mantık işlemini yerine getirebilir.



Şekil 2.10. Temel Mantık Bloğu İç Yapısı



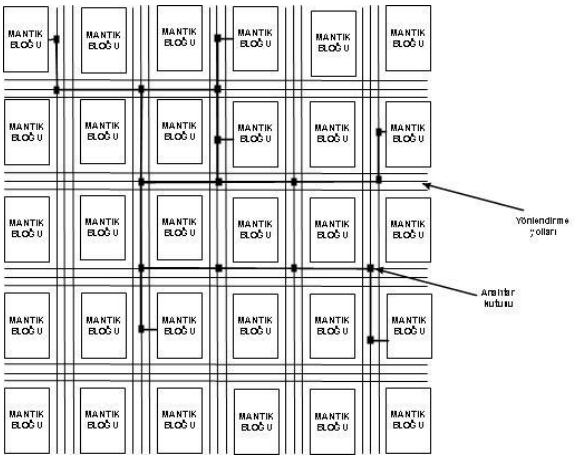
Şekil 2.11. Örnek bir mantık devresinin mantık bloğu içerisinde gerçekleştirilmesi.

Şekil 2.10'da temel mantık bloğu iç yapısı görülmektedir [23]. Şekil 2.11'de örnek olarak verilen bir mantık fonksiyonunun mantık bloğu içerisinde gerçekleştirilmesi görülmektedir. Devrenin doğruluk tablosuna göre Look-Up tablosu içerisindeki SRAM hücreleri içerisine veriler yüklenmiştir.

2.3.3.2. Matris Anahtarlamalı Yönlendirme Yolları

Yatay ve dikey olarak iki boyutlu bir düzlem üzerinde mantık blokları arasındaki bağlantıyı sağlayan bir yapıdır. Şekil 2.12'de görüldüğü gibi her mantık bloğunun etrafını saracak şekilde FPGA'nın içerisinde yer alır [23]. FPGA'nın türüne ve üreticisine göre bu yolların büyüklüğü değişim gösterir. Bağlantı noktalarında bulunan programlanabilir statik anahtarlar sayesinde yüklenen programa göre mantık blokları arasında ihtiyaç duyulan şekilde yönlendirme yapar. İki boyutlu bu matris düzeni içerisinde bu yönlendirmeler 90 veya 180 derece olabilir.

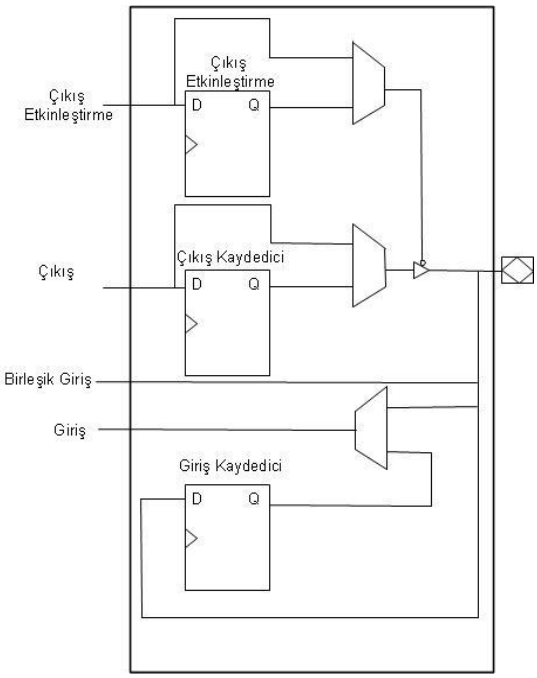
Bloklar arasında yapılan bağlantıda oluşacak gereksiz bir uzunluk veya yanlış yönlendirme, doğrudan FPGA'nın hızını etkileyecek bir unsurdur.



Şekil 2.12. Matris Anahtarlamalı Yönlendirme Yolları genel yapısı

2.3.3.3. Giriş/Çıkış Blokları

Mantık bileşenlerinin harici bileşenler ile bağlantı kurması kullanılan yapılardır. Aygıtın bağlantı pin sayısına denk gelecek kadar bulunur. Bir giriş/çıkış bloğu, kaydediciler, kontrol sinyalleri, mux devreleri ve saat sinyalleri içerir [23].



Şekil 2.13. Altera Cyclone FPGA'nın G/Ç blok yapısı

Şekil 2.13 de Altera firmasının Cyclone ailesine ait bir FPGA'nın giriş-çıkış bloğu görülmektedir [28]. Bir G/Ç bloğu sadece giriş veya sadece çıkış bloğu olarak

programlanabilir. FPGA'nın tamamı yapılandırıldıktan sonra G/Ç blokları yapılandırılır. FPGA'nın harici bağlantı noktasında bulunduğu için oluşabilecek elektrostatik deşarja karşı korumalıdır.

2.3.3.4. Özel Amaçlı İşlev Blokları

Bir FPGA'nın büyük bir bölümünü programlanabilir mantık blokları ve yönlendirme yolları kapsamasına rağmen, transistor teknolojisindeki gelişmeler sayesinde üreticiler, FPGA'nın içerisine ek olarak bazı özel işlevleri yerine getirebilmesi için birleşik devreler yerleştirmiştir. Bunların yaygın olanlarını şunlardır;

2.3.3.4.1. Bellek Blokları

Tasarımların önemli bir kısmında yonga içerisinde bir bellek birimine ihtiyaç duyulmaktadır. Mantık blokları ile bellek birimi oluşturulabildiği halde bu bellek biriminin yetersiz kaldığı durumlar için "Blok RAM" olarak isimlendirilen dahili bellek birimi FPGA içerisine dahil edilmiştir. BRAM miktarı, üreticiye ve FPGA ailesine göre değişim gösterir. Örnek olarak Xilinx Virtex 5 XC5VFX130T FPGA'sı içinde 10.728Kb için 298 adet 36Kb'lık RAM blokları kullanılmıştır [29].

2.3.3.4.2. Sayısal Sinyal İşleme (DSP) Blokları

Sayısal sinyal işleme işlevini içerebilen daha karmaşık devreler oluşturabilmek için birçok FPGA modeline BRAM'lerin yanına DSP blokları yerleştirilmiştir. DSP bloğu içeren FPGA'lar ile toplama, çıkarma, çarpma ve bit seviyesindeki diğer işlemler daha verimli bir şekilde gerçekleştirilebilmektedir.

2.3.3.4.3. Dahili İşlemci Bloğu

Tasarlanan devrelerin birçoğunda işlemciye ihtiyaç duyulmaktadır. Bir devre tasarımında güç ve kaynak tüketimi açısından tasarruf sağlamak için tercih edilen FPGA modelinin dahili işlemci içermesi büyük bir avantaj sağlar. Xilinx Virtex 5 ailesinde IBM PowerPC işlemcileri kullanılmıştır. XC5VFX serisinde 30T ve 70T modellerinde 1 adet, 100T, 130T ve 200T modellerinde ise 2 adet dahili PowerPC işlemcisi bulunmaktadır [29].

BÖLÜM 3

ÇOK YÜKSEK HIZLI TÜMLEŞİK DEVRE DONANIM TANIMLAMA DİLİ - VHDL

VHDL (Very High Speed Integrated Circuit Hardware Description Language), yüksek hızlı tümleşik devrelerin donanım tasarımında kullanılan bir programlama dilidir. İlk olarak Amerikan Savunma Bakanlığı ihtiyacına yönelik olarak geliştirilmeye başlanan yüksek hızlı amaca yönelik tümleşik devre tasarımında kullanılmak üzere geliştirilmiş ve Elektrik & Elektronik Mühendisleri Enstitüsü (IEEE) tarafından 1987 yılında "Std 1076-1987" olarak standartlaştırılmış olan bu dilin, son olarak standartları IEEE tarafından "Std 1076-2008" versiyon ile düzenlenmiştir [30]. Günümüzde yaygın olarak FPGA programlamada kullanılmaktadır.

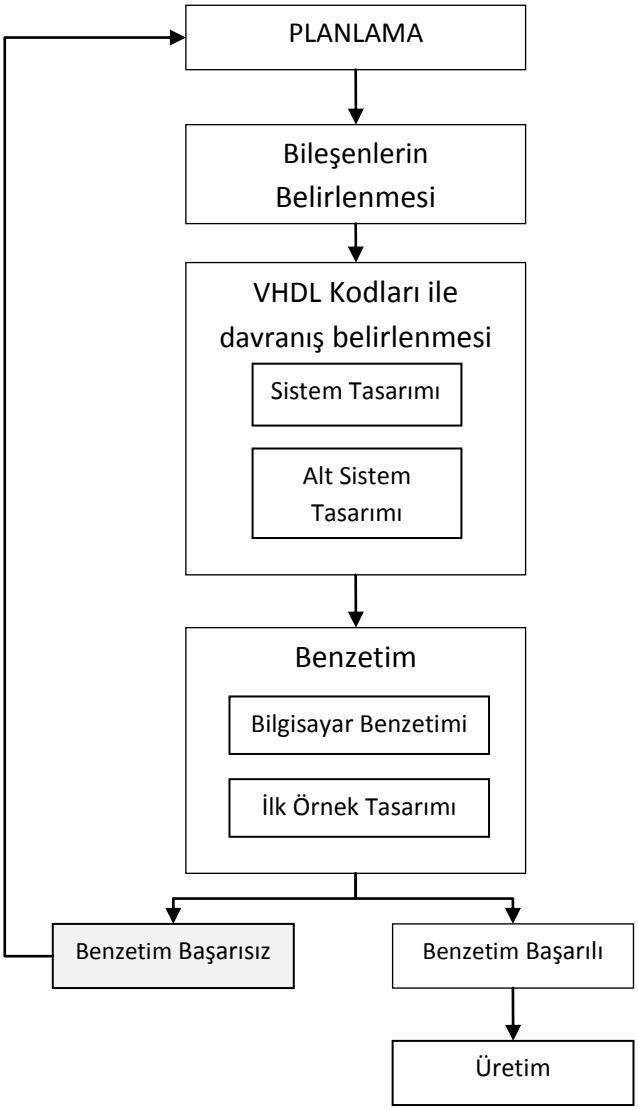
Donanım tanımlama dillerinin ilk örnekleri olarak 70'li yılları sonuna doğru geliştirilmiş olan ISP (Carnegie Mellon University) ve KARL (Kaiserlaustern University) dilleri sayılabilir. Günümüzde ise yaygın olarak kullanılan donanım tasarım dili olarak VHDL'in yanında Verilog HDL'de mevcuttur. VHDL ile Verilog HDL dilleri aynı amaç için kullanılsa da birbirlerinden farklı bir yapıya sahiptir. VHDL, ADA programlama dillerini baz alan bir yapıdadır ve Verilog HDL'e göre daha katı kurallı bir dildir. Verilog HDL ise öğrenmesi ve yazması daha kolay C programlama diline yakın bir yapıya sahiptir ancak VHDL'e göre karmaşık devre tasarım sentezleme aşamasında daha fazla kapı kullanılmasına sebep olmaktadır.

Donanım tasarımında kullanılacak olan programlama dili belirlendikten sonraki aşamada, gerçekleştirilecek olan devreden istenilenler ve sistem bileşenleri belirlenir. Bundan sonraki aşama tasarım aşamasıdır. Tasarım aşamasında kod yapısının tasarım yapısı belirlenir. Tasarımda yukarıdan aşağı bir yaklaşım izlenerek, karmaşık olan bütün üst yapıdan daha az karmaşık olan alt yapılara ayrılır.

Tasarım aşamasında kod parçaları adım adım çalıştırılabilir ve tasarımın bütünlüğündeki eksiklikler kolaylıkla tespit edilip giderilebilir. Biçimsel bir modeldeki

kodlama hataları ya da sistemdeki kavramsal hatalar, tasarım aşamasının her adımında, benzetim programı çalıştırılarak bulunabilir. Oluşturulan modeldeki farklı giriş değerlerine göre sistemin verdiği yanıt benzetimle gözlemlenebilir ve analiz edilebilir.

Benzetim aşamasında sentezleme olarak isimlendirilen bir aşama gerçekleştirilir. Bu işlem tasarımda kullanılan dilin daha alt seviyeye dönüştürülmesi olarak açıklanabilir. C programlama dilinden örnek vermek gerekirse bu işlem yazılan kodun makine diline çevrilmesi olarak görülebilir. Donanım tanımlama dilinde bu işlem RTL aşamasına geçiştir. Diğer bir anlamda, kodları yazılan devrenin veri yolları, bellek elemanları ve kontrol birimleri gibi bileşenlerinin bir devre şeması üzerinde görüntülenmesidir.



Şekil 3.1. VHDL tasarım aşamaları

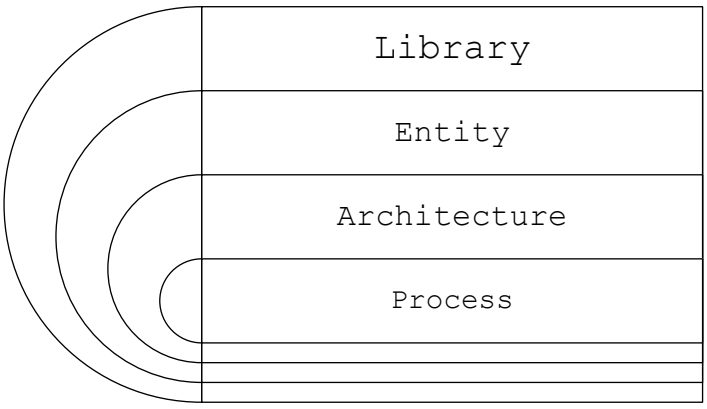
3.1. VHDL Tasarım Yapısı

VHDL tanımlama yapısında birden fazla devre yapısı iç içe oluşturulabilir veya oluşturulan temel bir devre ana proje dosyasında birden fazla defa kullanılabilir. Bunun için VHDL yapısı içerisinde tanımlayacağımız devre; yapısal, veri akışı ve davranışsal biçimlerinde tanımlanabilir.

- Yapısal (Structural): fonksiyon kapılar ve kapı bağlantıları olarak tanımlanarak devre oluşturulur.
- Veri Akışı (Data flow): devredeki kapıların giriş /çıkış sinyal geçişleri tanımlaması yapılarak devre oluşturulur.
- Davranışsal (Behavioral): devredeki kapılar, bu kapıların bağlantı tipleri ve aralarındaki veri akşının göz ardı edildiği devrelerde sadece davranış biçimi tanımlaması yapılarak devre oluşturulur.

3.2. VHDL Kod Yapısı

VHDL kod parçacıkları kullanılan bir çok programın aksine sıralı olarak çalışmaz. Bunun yerine tüm program donanım yapısı üzerinde aynı anda paralel olarak çalışır. VHDL ile tasarlanan FPGA'ların hızlı olmasının en önemli sebeplerinden birisi budur.



Şekil 3.2. Temel VHDL Kod Yapısı

Tablo 3.1. Temel VHDL Terimleri

Entity	Bir tasarımın giriş çıkış gibi arayüz varlıklarının tanımlandığı ana kısımdır. Tasarımın hiyerarşik yapısı içerisinde kullanılacak olan yapılar burada tanımlanmak zorundadır.
Architecture	Entity içerisinde tanımlanan varlıkların benzetimlerinin gerçekleştiği mimari kısımdır. bu kısım içerisinde varlıkların davranışları tanımlanır. Bir tasarım içerisinde birden fazla architecture yapısı olabilir ve bu yapıların bir kısmı davranışsal olarak tanımlanırken diğer kısmı yapısal olarak tanımlanabilir.
Component	Program içerisinde sık kullanılan alt program yapılarının ana program içerisine çağırılarak kullanılmasını sağlar.
Bus	Donanım içerisinde iletişim için kullanılan bir grup sinyal yapısının tanımlanması için kullanılır.
Generic	Entity yapısının içerisine veri tipi tanımlanan bilgiyi göndermek için kullanılan bir yapıdır.
Process	"If" ve "For" gibi sıralı yapıların kullanıldığı işlem parçacıklarının tanımlandığı yapıdır. Bu yapı içerisinde kodlar diğer kodların aksine sıralı olarak çalışır.
Package	VHDL standartlarının dışındaki birimlerin program içerisinde tanımlanmasını sağlayan yapılardır.

Dilin ön tanımlama, varlık (entity) olarak belirltilen kısma devrenin giriş çıkış ve birimleri tanımlanır. “Architecture” olarak belirtilen ana mimari kısmında devrenin yapısı tanımlanır. Her “entity” kısmı için bir “architecture” tanımlaması yapılır. Yine bu ana mimari kısmında daha önceden oluşturulmuş tekrarlanacak devre blokları varsa “component” tanımlaması ile projeye dâhil edilir. Mimari yapı davranışı tanımlanmaya başladıktan sonra “port map” olarak tanımlanan kısımda giriş çıkış bağlantı uçları tayin edilir.

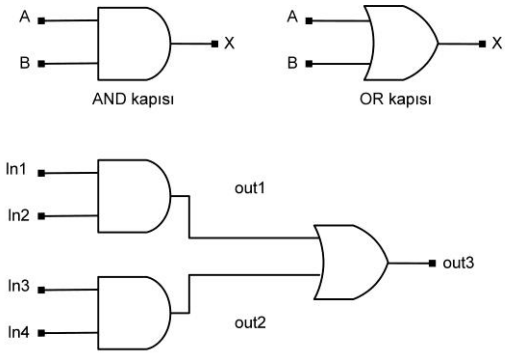
Tablo 3.2. VHDL Entity Yapısı

```
Entity proje_ismi is
      port(
          "giriş çıkış port boyut ve biçimlerini tanımlama"
            );
end
```

Tablo 3.3. VHDL Mimari Yapısı

```
architecture "mimari ismi" of "tanımlayıcı ismi" is
   component "dahil edilecek devre bloğu ismi"
     port(
       "devre bloğunun giriş-çıkış birimleri tanımlaması"
     );
   End component;
begin
   port map(
     "projeye dahil edilen devre bloğunun giriş çıkış birimlerinin
     atanması"
   );
   process("işleme dahil edilecek birimler)
     begin
          "devre davranış yapısı"
   End process;
….
   process(……
   ……..
   End process;
End architecture;
```

VHDL'in mimari yapısı basit bir örnek olarak Şekil 3.3'de AND ve OR mantık kapılarından oluşan bir devre ile açıklanmıştır. Devrenin VHDL kodları Tablo 3.2 ve Tablo 3.3'te verilmiştir. Bu devreyi oluşturan kapı bileşenleri ayrı birer program olarak tanımlanmış, daha sonra ana program içerisinde alt program olarak çağırılmış ve kullanılmıştır.



Şekil 3.3. Örnek AND - OR Mantık devresi

Tablo 3.4. AND ve OR kapılarının işlevlerinin tanımlandığı alt programların VHDL kodları

```
Entity and_gate is
     port(
        A: in bit,
        B: in bit,
        X: out bit);
End entity and_gate;
architecture mimari of and_gate is
     begin
        X <= A and B;
End architecture mimari;
```

```
Entity or_gate is
     port(
        A: in bit,
        B: in bit,
        X: out bit);
End entity or_gate;
architecture mimari of or_gateis
     begin
        X <= A and B;
End architecture mimari;
```

Tablo 3.5. AND-OR Devresinin VHDL Kodları

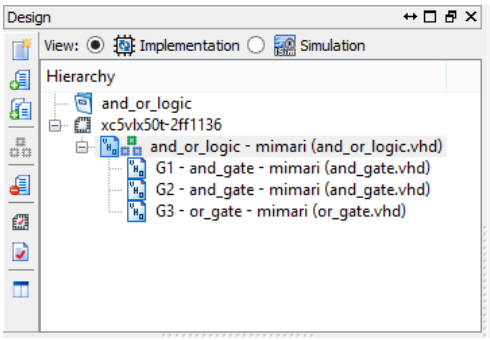
```
Entity and_or_logic is
port(
   In1, In2, In3, In4:    in bit;
   Out3:   out bit);
End and_or_logic;
```

```
architecture mimari of and_or_logic is
   componentand_gateis
      port(
         A: in bit;
         B: in bit;
         X: out bit);
   End component;
   component or_gate is
      port(
         A: in bit;
         B: in bit;
         X: out bit);
   End component;

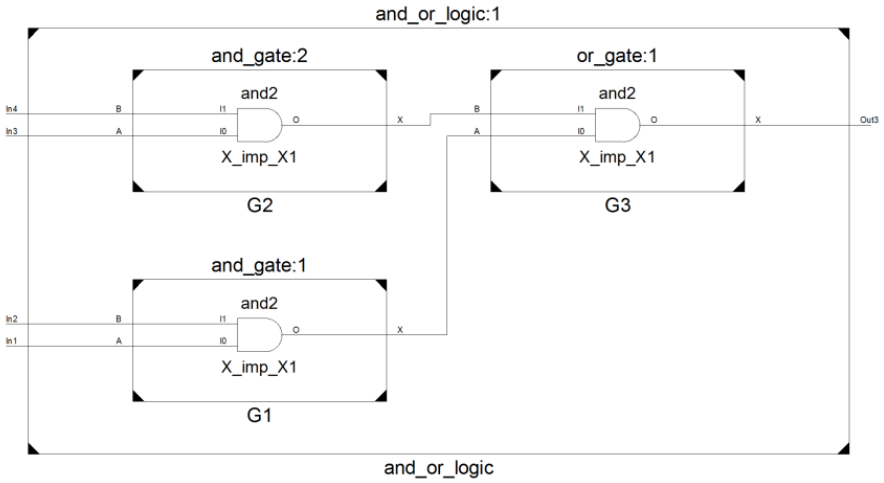
signal out1, out2: bit;
```

```
begin
G1: and_gate port map(
   A=>In1,
   B=>In2,
   X=>out1);
G2:and_gate port map(
   A=>In3,
   B=>In4,
   X=>out2);
G3: or_gate port map(
   A=>out1,
   B=>out2,
   X=>out3);
End architecture;
```

Tablo 3.4 ve Tablo 3.5'te verilen VHDL kodlarının, Xilinx FPGA modülünün VHDL derleyici programı olan ISE ile oluşturulan bileşen yapısı Şekil 3.4'te, RTL şeması ise Şekil 3.5'te, teknoloji şeması ise Şekil 3.6'da verilmiştir.

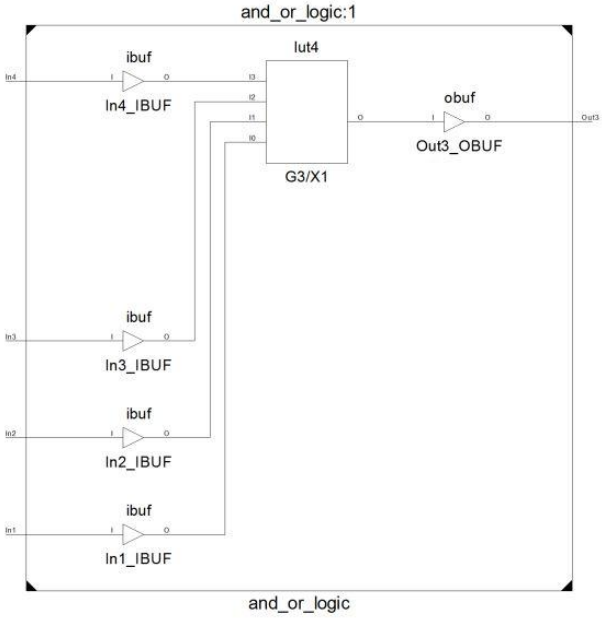


Şekil 3.4. AND - OR devresinin ISE programında uygulanması ile oluşan bileşen yapısı



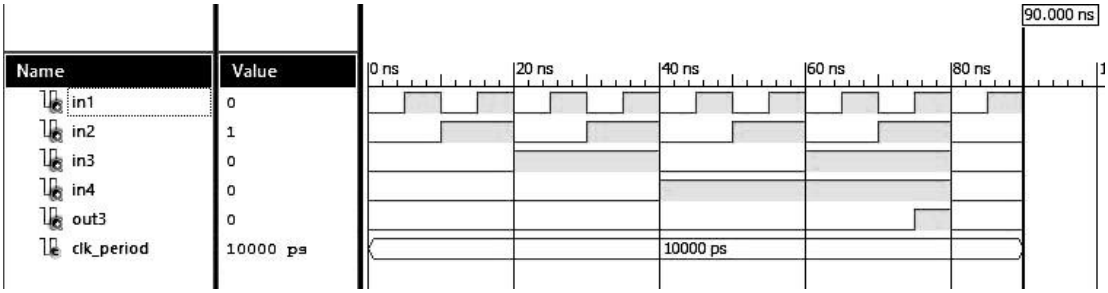
Şekil 3.5. AND - OR devresinin ISE programında oluşan RTL şeması

RTL şeması, oluşturulan devrenin kapı seviyesindeki görüntüsüdür. HDL sentezinin ardından elde edilen bu görüntüde tasarım; çarpıcı, sayıcı, toplayıcı veya mantık kapıları gibi genel semboller ile ifade edilir. Teknoloji şemasında ise ise tasarımın mimari yapısı görüntülenir. Bu aşama, tasarımın hedef aygıta uygun bir biçimde yerleştirildiği kısımdır. Tasarım bu görüntüde uygulanacağı aygıta uygun LUT, G/Ç tamponları, taşıyıcı gibi bileşenler ile ifade edilir.



Şekil 3.6. AND - OR devresinin ISE programı ile elde edilmiş teknoloji şeması

ISE programında sentezlemesi yapılan VHDL kodunun benzetim işlemi için ISIM programı kullanılmıştır. Projeye dahil edilen VHDL dilinde yazılmış bir test kodunun yürütülmesiyle Şekil 3.7'de görülen And - Or devresinin benzetim sonuçları elde edilmiştir.

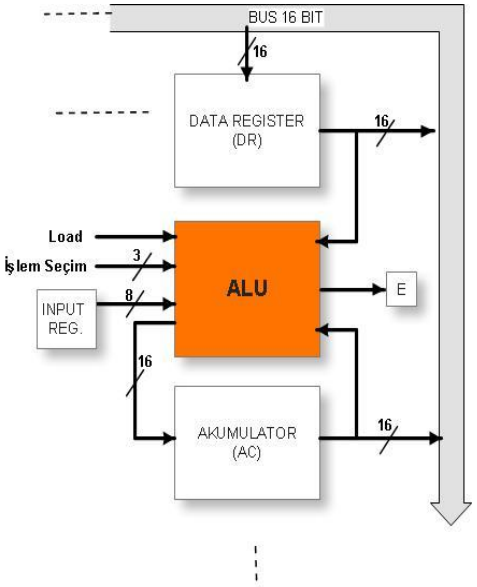


Şekil 3.7. And - Or devresinin ISIM programında görüntülenen benzetim sonuçları

VHDL ile yapılabilecek karmaşık tasarımlar, basit kod parçaları ile pratik olarak gerçekleştirilebilir. Bu duruma en iyi örnek olarak 16 bitlik bir mikroişlemcinin temel birimi olan aritmetik ve lojik ünitesinin tasarımı gösterilebilir. Bu örnek VHDL'in temel yapısını daha iyi kavramak için faydalı bir örnek olarak görülebilir.

3.3. VHDL İle 16 Bitlik Aritmetik Lojik Ünite Tasarımı

Tasarlanacak olan 16 bitlik aritmetik lojik ünitenin gerçekleştireceği işlemler ve işlemlerin seçim kodları Tablo 3.6'da, aritmetik lojik üniteye bağlı olan birimler ise gösterilmiştir. Şekil 3.8'de gösterilmiştir.



Şekil 3.8. Aritmetik Lojik Ünitenin komşu birimlerle olan bağlantısı

Tablo 3.6. Aritmetik Lojik İşlemler ve seçim kodları

İşlem Seçim	Açıklama
000	AC'ü çıkışa aktar
001	AC AND DR
010	AC + DR
011	DR'i, AC'ye aktar
100	NOT AC
101	INPUT Reg.'i AC'ye aktar
110	AC'yi bir sağa kaydır
111	AC'yi bir sola kaydır

Tablo 3.7'de verilen bit uzunluklarına göre giriş ve çıkış portlarının VHDL yapısı içerisinde tanımlanması gerçekleştirilir. Tablo 3.8'de "entitiy" kısmında giriş çıkış portlarının VHDL olarak tanımı yapılmıştır.

Tablo 3.7. Giriş ve Çıkış portları kısaltmaları ve bit uzunluukları

Kısaltma	Giriş / Çıkış	Bit Uzunluğu	Açıklama
sec	Giriş	3 bit	İşlem seçme girişleri
ac	Giriş	16 bit	Akümülatör veri girişi
dr	Giriş	16 bit	Data Register veri girişi
inp	Giriş	8 bit	Input Register veri girişi
ld	Giriş	1 bit	Load girişi
alu_cikis	Çıkış	16 bit	ALU veri çıkışı
e	Çıkış	1 bit	Elde biti

Tablo 3.8. Giriş Çıkış portlarının VHDL'de entity yapısının altında tanımlanması

```
05  entity alu is port (
06  sec        : in std_logic_vector(2 downto 0);
07  ac         : in std_logic_vector(15 downto 0);
08  dr         : in std_logic_vector(15 downto 0);
09  inp        : in std_logic_vector(7 downto 0);
10  ld         : in std_logic;
11  alu_cikis :out  std_logic_vector(15 downto 0);
12  e          :out std_logic);
13  end alu;
```

Programın varlık kısmında entity kalıbı ile oluşturulacak yapının ismi "alu" olarak belirtilmiş ve port tanımlama işlemi gerçekleştirilmiştir. Bir bitten daha fazla bit genişliğine sahip olan giriş/çıkış bağlantıları vektör olarak tanımlanmıştır. Vektör olarak tanımlanan bu bağlantıların bit uzunlukları ve yüksek değerlikli veya düşük değerlikli olan bitleri parantez içerisinde yazılmıştır. Örnek olarak "dr" giriş portunda 15. bit yüksek değerlikli olarak belirtilmiştir. Bunun için "15 downto 0" olarak ifade edilmiştir. Tam tersi bir durum için "15 to 0" yazılarak ifade edilir.

Tablo 3.9. Aritmetik Lojik Ünite VHDL Kod Yapısı

```
14  architecture yapi of alu is
15  signal elde: std_logic_vector(16 downto 0);
16  begin
17  alu_islem : process(ld,elde)
18  variable temp :     std_logic_vector(15 downto 0);
19  begin
20  elde <= "00000000000000000";
21  if ld = '1' then
22  -----AC'yi Aktar-------
      if  sec = "000" then
23      temp := ac;
24      alu_cikis <= temp;
25    end if;
26  -------AC and DR--------
      if  sec = "001" then
27      temp:= ac and dr;
28      alu_cikis <= temp;
29    end if;
30  -------AC+DR------------
      if  sec = "010" then                              -
31      toplayici: for i in 15 downto 0 loop
32        elde(i+1) <= (ac(i) and dr(i)) or (elde(i) and (ac(i) xor
    dr(i)));
33        alu_cikis(i) <= ac(i) xor dr(i) xor elde(i);
34      end loop toplayici;
35      e <= elde(16);
36    end if;
37  -------DR'yi aktar-------
      if  sec = "011" then
38      temp := dr;
39      alu_cikis <= temp;
40    end if;
41  -----AC'yi Tersle--------
      if  sec = "100" then
42      temp := not ac;
43      alu_cikis <= temp;
44    end if;
45  ----INP AC'ye aktar-------
      if  sec = "101" then
46      alu_cikis(7 downto 0)<= inp;
47      alu_cikis(15 downto 8)<= X"00";
48    end if;
49  --Sağa Kaydır-------------
      if  sec = "110" then
50       temp := ac;
51       alu_cikis(15)<= temp(0);
52       for i in 15 downto 1 loop
53       alu_cikis(i-1)<= temp(i);
54       end loop;
55  --Sola Kaydır-------------
     if  sec = "111" then
56       temp := ac;
57       alu_cikis(0)<= temp(15);
58       for i in 0 to 14 loop
59       alu_cikis(i+1)<= temp(i);
60       end loop;
61     end if;
62  end if;
63  end process alu_islem;
64  end yapi;
```

Tablo 3.10. 16 bit işlemcinin bileşenlerinin tanımlanması ve portlarının birbirine bağlanması

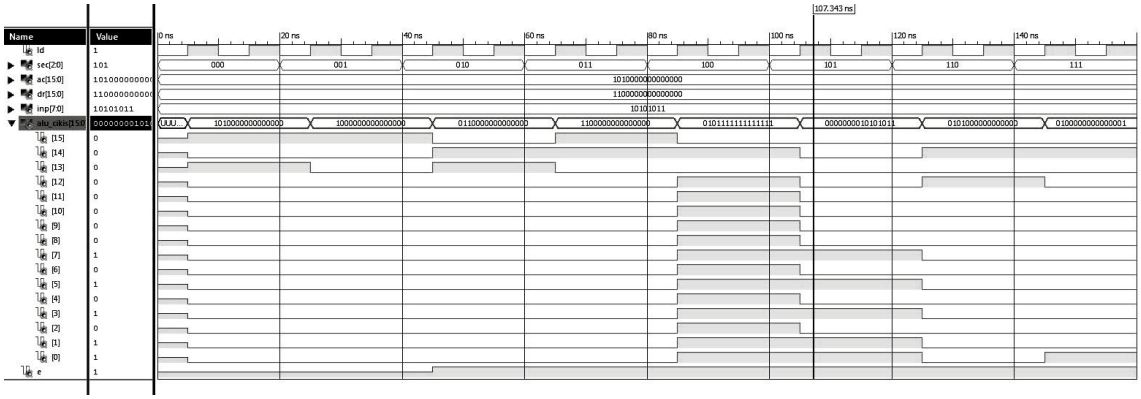
```
…    …
05   entity cpu is port(
06   signal alu_sec: in std_logic_vector(2 downto 0);
07   signal alu_a : in std_logic_vector(15 downto 0);
08   signal alu_d : in std_logic_vector(15 downto 0);
09   signal alu_out : out std_logic_vector(15 downto 0);
10   signal alu_e : out std_logic;
11   signal input : in std_logic_vector(7 downto 0);
…    …
35   );
36   end cpu;
…    ……
59   component alu port(
60   sec     : in std_logic_vector(2 downto 0);
61   ac      : in std_logic_vector(15 downto 0);
62   dr      : in std_logic_vector(15 downto 0);
63   inp     : in std_logic_vector(7 downto 0);
64   ld      : in std_logic;
65   alu_cikis :out  std_logic_vector(15 downto 0);
66   e       :out std_logic);
67   end component;
…    …
…    …
…    …
117  architecture yapi of cpu is
118  Begin
119  u3:    alu port map(
120             sec=>alu_sec,
121             ac=>alu_a,
122             dr=>alu_d,
123             inp=>input,
124             alu_cikis=>alu_out,
125             e=>alu_e);
…    …
…    …
299  end yapi;
```

Tablo 3.9'da Aritmetik lojik ünitenin VHDL kodları verilmiştir. "Architecture" yapısının altında, aritmetik ve lojik ünitenin gerçekleştirmesi istenilen 8 işlemin davranışları tanımlanmıştır. 17 bit uzunluğunda "signal" olarak tanımlanmış "elde" dizisi, aritmetik lojik ünitenin içinde 16 bit boyunca her bitin taşma ucunu bir sonraki bite taşıma işlemi gerçekleştirecek, 17. bit ise "e" elde çıkışına aktaracaktır. Her "process" başlangıcından sonra "elde" birimi sıfırlanacaktır. "ld" girişinde "ld=1" şartı sağlandığında seçme giriş uçlarının 000-111 aralığındaki giriş değerlerine göre işlem seçimi gerçekleştirilir. "temp" değişkeni geçici olarak veri tutma işlemi gerçekleştirir.

16 bitlik bir işlemcinin temel birimi olarak tasarlanan bu aritmetik ve mantık biriminin, işlemcinin VHDL olarak tanımlanmış diğer birimleri ile bağlantı kurabilmesi için bir ana program yazılması ve yazılan diğer birimlerin birer alt program olarak

çağırılıp, ana programda aralarındaki ilişkinin tanımlanması gerekir. Tablo 3.10'da "cpu" olarak isimlendirilmiş ana programda aritmetik lojik ünite "component" anahtar sözcüğü ile çağırılmış ve tanımlanması yapılmıştır.

Şekil 3.9'da tasarlanan aritmetik mantık biriminin ISIM programı kullanılarak oluşturulan benzetim sonuçları verilmiştir. Bu benzetim işleminde Akümülatör girişine "1010000000000000" değeri, data yazacı girişine ise "1100000000000000" değeri verilmiştir. İşlem seçim girişlerine göre sırasıyla sonuçlar ALU çıkışında görülmektedir.

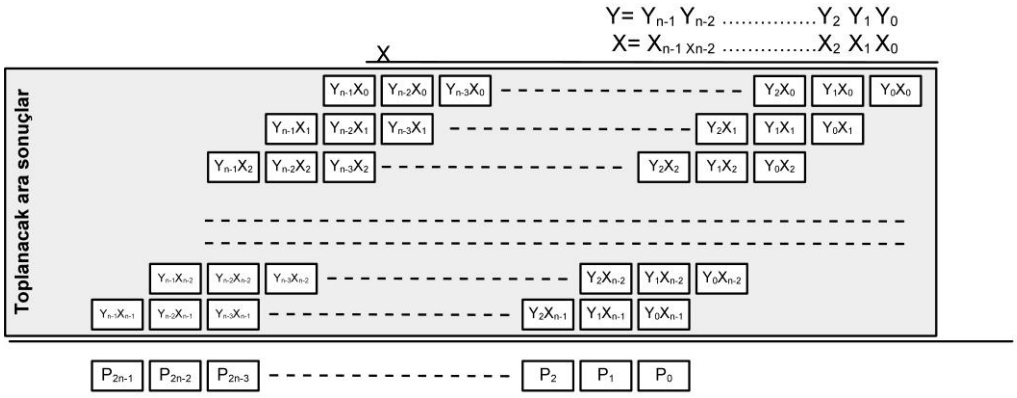


Şekil 3.9. 16 bit Aritmetik mantık biriminin ISIM benzetim sonuçları

BÖLÜM 4

ÇARPMA İŞLEMİ ALGORİTMALARI

Bilgisayarda gerçekleştirilen aritmetik işlemlerin temelinde toplama işlemi vardır. Şifreleme, şifre çözme algoritmaları ve sinyal işleme gibi birçok bilimsel uygulamanın temeli olan çarpma işlemi de temelde kaydırma ve toplama işlemi esasına göre gerçekleştirilmektedir. Çarpma işlemi algoritmalarının yapısı, özellikle bilimsel programlarda yüksek kapasiteli veriler ile çalışıldığı zaman, bilgisayarın performansını etkileyen en önemli etkendir. Bir bilimsel programın yaklaşık %9'u çarpma işleminden oluşur [31]. Bu sebepten dolayı bilgisayar teknolojisindeki gelişmeleri takip eden yıllar içerisinde çok çeşitli çarpma algoritmaları geliştirilmiştir. Verinin boyutuna ve işleneceği algoritmanın uygunluğuna göre seçilecek olan çarpma algoritması bilgisayarın işlemci performansı arttırılmadan işlem hızını arttırabilir.



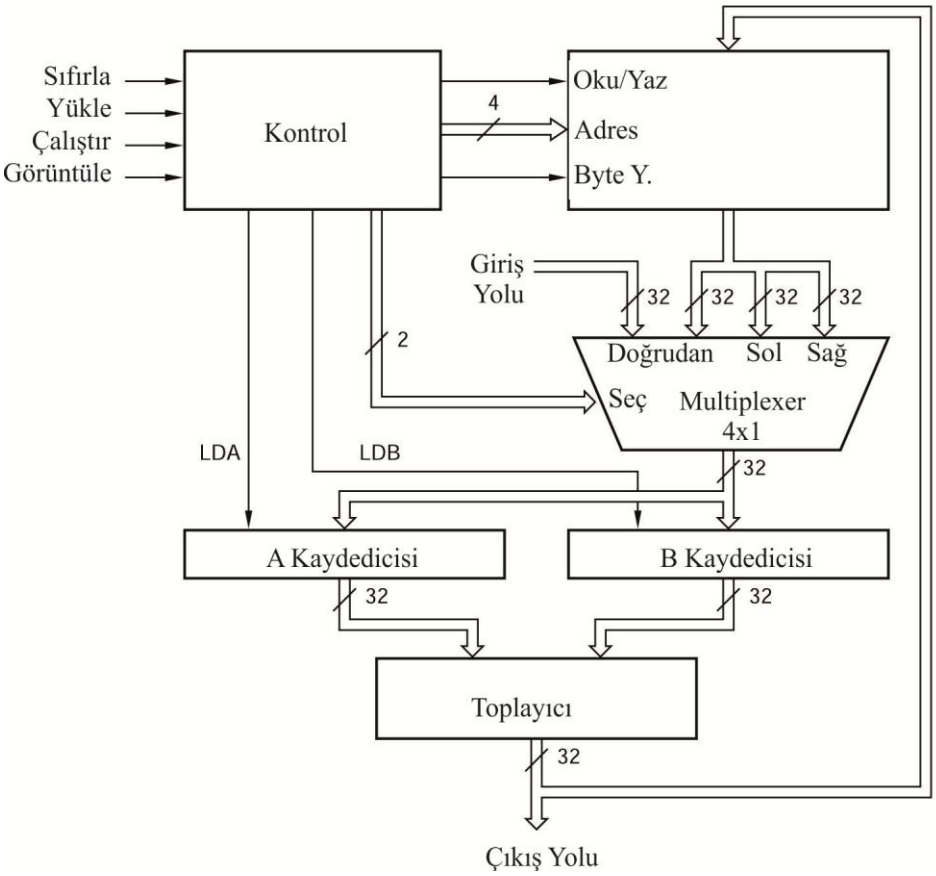
Şekil 4.1. Temel çarpma işlemi

Sıradan bir çarpma işlemi için n bit X ve Y sayıları için Şekil 4.1'de gösterilmiştir. Buna göre n bit uzunluğundaki bir X sayısı ve m bit uzunluğundaki bir Y sayısı için çarpma işlemi aşağıdaki denklem ile ifade edilebilir [2].

$$P(m+n) = X(n)Y(m) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} x(i)y(j)2^{i+j} \quad (4.1)$$

4.1. Sıralı Kaydır/Topla Çarpma

Bu yöntemde çarpanın en düşük değerlikli bit değerinden başlanarak, bit değerinin 1 veya 0 olma durumuna göre her saat vuruşunda çarpılan sayının sola kaydırılarak toplanması yolu ile sonuç elde edilir. Bu yöntem oldukça basit mantık üzerine kurulu olmasına karşılık veri boyutunun fazla olması durumunda işlem performansı olumsuz etkilenmektedir.



Şekil 4.2. Sıralı / kaydır topla yöntemi [20]

4.2 Booth Algoritması

Booth algoritması ile işaretli iki tabanındaki, sayılar 2'nin tersine alma işlemi ile birbirine bitişik bitler karşılaştırılarak toplama-kaydırma esasına göre çarpma işlemi yapılmaktadır.

Booth algoritmasında, m ve r değerlerine göre önceden tanımlanmış A ve S değerlerinin tekrar eden işaretsiz toplama işlemi ile P sonucuna ulaşılır. "x" ve "y"

çarpan ve çarpılan sayıların bit uzunlukları olarak tanımlanır ve çarpma işlemini gerçekleştirmek için şu adımlar izlenir;

A ve S değerleri ile P'nin başlangıç değerleri tanımlanır. Bu sayıların bit uzunlukları x+y+1 kadar olmalıdır.

A değeri için; en yüksek değerlikli bitten başlanarak m sayısı ile doldurulur, geri kalan bitler "0" değeri alır.

S değeri için; en yüksek değerlikli bitten başlanarak m sayısına 2'nin tersi işlemi uygulanarak elde edilen sayı değeri ile doldurulur. Geri kalan bütün bitler "0" değeri alır.

P başlangıç değeri için; en yüksek değerlikli bitten başlanarak, sayıların toplam bit uzunluklarının yarısı kadar olan bit uzunluğu "0" değeri alır. Diğer yarısı r değeri ile doldurulur, geriye kalan bir bit ise "0" değeri alır.

P sayısının son iki bitine bakılarak şu işlemler gerçekleştirilir;

1. 00 ise; P değerini bir bit sağa kaydır
2. 11 ise; P değerini bir bit sağa kaydır
3. 01 ise; P=P+A, P değerini bir bit sağa kaydır. (toplamda taşma değeri önemsenmez)
4. 10 ise; P=P+S, P değerini bir bit sağa kaydır. (toplamda taşma değeri önemsenmez.)

Bu işlemler y kadar tekrar eder. Döngü sonucunda elde edilen değerin en düşük değerlikli biti silinir ve sonuç elde edilir [32, 33].

Örnek olarak $m \times r$ işleminde $m = 3, r = (-6)$ değerleri için ikili sayı tabanında 4 bitlik değerleri elde edilir.

$$m = 0011, \qquad r = 1010$$

m çarpılan değerin ikinin tümleyenine göre tersi alınır.

$$m_T = 1101$$

A, S ve P değerleri belirlenir

$$A = 0011\ 0000\ 0$$
$$S = 1101\ 0000\ 0$$
$$P = 0000\ 1010\ 0$$

P sayısının başlangıç değerinden başlayarak, en önemsiz son iki bitine y döngüsü kadar bakılarak toplama ve kaydırma işlemleri yapılır.

- 0000 101**0 0** → P'nin sonraki döngü değeri için bir bit sağa kaydırılır.
- 0000 010**1 0** → P'nin sonraki döngü değeri için P+S işlemi yapılır ve bir bit sağa kaydırılır.

$$P + S = 1101\ 0101\ 0 \textit{ (bir bit sağa kaydırılır.)}$$
$$P = 1110\ 1010\ 1$$

- 1110 101**0 1** → P'nin sonraki döngü değeri için P+A işlemi yapılır ve bir bit sağa kaydırılır.

$$P + A = 0001\ 1010\ 1 \textit{ (bir bit sağa kaydırılır.)}$$
$$P = 0000\ 1101\ 0$$

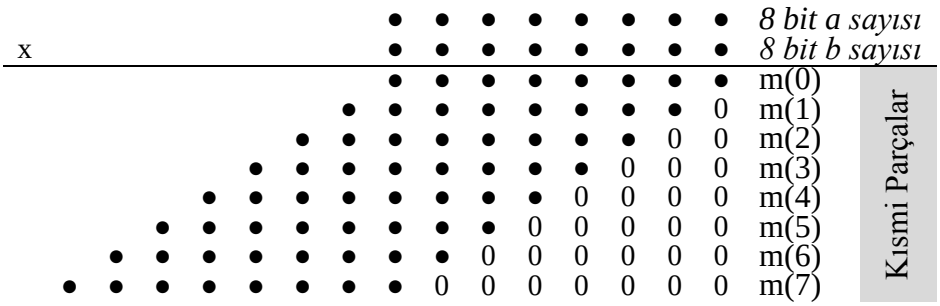
- 0000 110**1 0** → P'nin sonraki döngü değeri için P+S işlemi yapılır ve bir bit sağa kaydırılır.

$$P + S = 1101\ 1101\ 0 \textit{ (bir bit sağa kaydırılır.)}$$
$$P = 1110\ 1110\ 1$$

En düşük değerlikli bit silinir ve işlem sonucu olarak 11101110 değeri elde edilir.

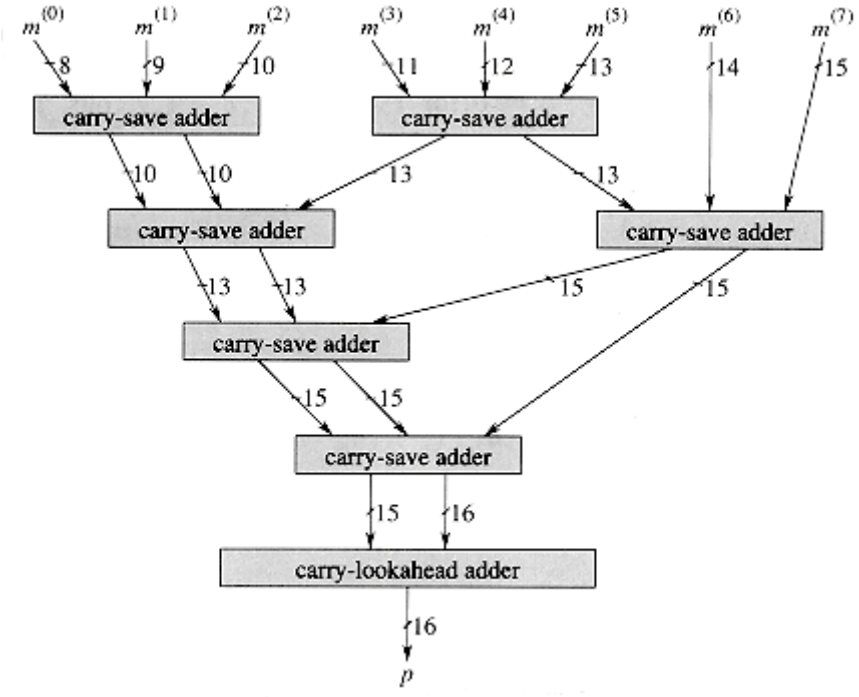
4.3. Wallace Ağacı

Wallace ağacı çarpımı işaretsiz iki integer sayının donanım seviyesinde çarpımında tercih edilebilecek etkili bir çarpma yöntemidir. 1964 yılında bilgisayar bilimcisi olan Chris Wallace tarafından geliştirilmiştir [34].



Şekil 4.3. 8 bit iki sayının çarpımında kısmi parçaların elde edilmesi

Wallace ağacı metodunda, kısmi parçalar son aşamada toplanacak olan son iki kısmi parça satırına ulaşıncaya kadar bir ağaç yapısı biçiminde toplanır. Bu algoritmanın hızı, işlenecek olan bit sayısı ile ters orantılıdır. Kullanılmadığı için boşa harcanan alan ve karmaşık yapısı bu algoritmanın başlıca sorunlarındandır [35]. Donanım yapısında hızlı taşıma mantığına uygun bir yapıda değildir. Ayrıca modern FPGA'ların içerisinde yer alan taşıma yapısı kadar hızlı değildir [36].



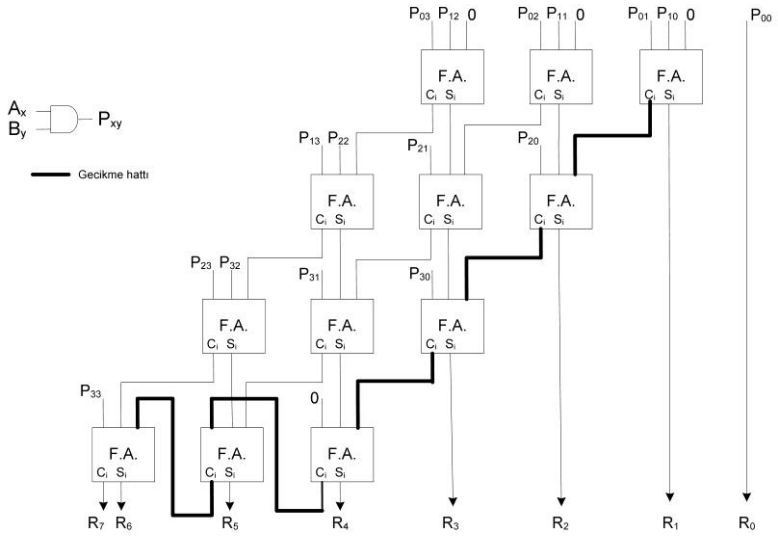
Şekil 4.4. Wallace ağacı 8 bit uzunluğunda iki sayının çarpımı blok yapısı

Şekil 4.3'te 8 bitlik iki sayının çarpımından kısmi parçaların elde edilişi görülmektedir. Elde edilen bu kısmi parçalar ile Wallace ağacı yöntemine göre gerçekleştirilmiş olan çarpma işleminin blok şeması Şekil 4.4'te verilmiştir [37]. Her parçanın bit uzunluğu hat üzerinde belirtilmiştir. Şekilde 4.4'te görüldüğü gibi her taşı-kaydet toplayıcısı (CSA - Carry Save Adder) girişinde 3 adet kısmi parçaya işlem uygulayarak girişindeki en yüksek bit sayısı kadar uzunluğu olan iki adet kısmi parçayı bir alt basamaktaki CSA girişine aktarır. Bu işlem en sonda iki adet kısmi parça kalana kadar devam eder. Son iki kısmi parça kaydır-ileri aktar (CLA - Carry Lookahead Adder) tipi bir toplayıcı ile sonucu üretir.

4.4. Dizin Çarpma

Paralel dizin yapısına sahip, toplama kaydırma esasına dayanan, yaygın bir çarpım algoritmasıdır. Kısmi sonuç değerleri, çarpan sayının sırasıyla birer bitlik basamaklarının çarpılan sayı ile çarpımından elde edilir ve bit sırasına göre kaydırılarak toplanır. Algoritma olarak yavaş bir yapıya sahip olmasına rağmen paralel bir Şekilde ilerleyen dizin yapısının düzenli oluşu ve donanım üzerindeki yerleşim kolaylığı sebebiyle tercih edilmektedir.

Dizi çarpma yöntemiyle çarpımı gerçekleştirilen iki adet 4 bitlik sayının çarpım devresinin blok şeması Şekil 4.5'te verilmiştir [38]. Gecikmenin hesaplanabilmesi için bilinmesi gereken kritik yol Şekil 4.5'te gecikme hattı olarak soldan sağa doğru vurgulanarak belirtilmiştir.



Şekil 4.5. 4 bitlik iki sayının dizin çarpım yöntemiyle çarpımı blok şeması

4.3.Karatsuba Çarpma Algoritması

Yüksek boyutta iki sayının çarpımı için kullanılan etkili bir çarpma yöntemidir. Çarpılacak olan sayılar alt gruplara bölünür. Bu alt grupların çarpım sonuçlarının toplanması ile sonuç elde edilir.

n bit uzunluğunda a ve b sayıları n>m olmak üzere şu şekilde ifade edilebilir [39];

$$a = a_1 \mathrm{B}^m + \mathrm{a}_0 \tag{4.2}$$

$$b = b_1 \mathrm{B}^m + \mathrm{b}_0 \tag{4.3}$$

(4.2) ve (4.3) denklemlerinde, B herhangi bir taban, m ise bu tabanın herhangi bir üssüdür. Bu denklemler kullanılarak ab çarpım sonucu şu şekilde yazılabilir;

$$ab = (a_1 B^m + a_0)(b_1 B^m + b_0) \tag{4.4}$$

$$ab = a_1 b_1 B^{2m} + (a_1 b_0 + a_0 b_1) B^m + a_0 b_0 \tag{4.5}$$

$$\alpha = a_1 b_1 \tag{4.6}$$

$$\beta = a_1 b_0 + a_0 b_1 \tag{4.7}$$

$$\gamma = a_0 b_0 \tag{4.8}$$

$$\alpha B^{2m} + \beta B^m + \gamma \tag{4.9}$$

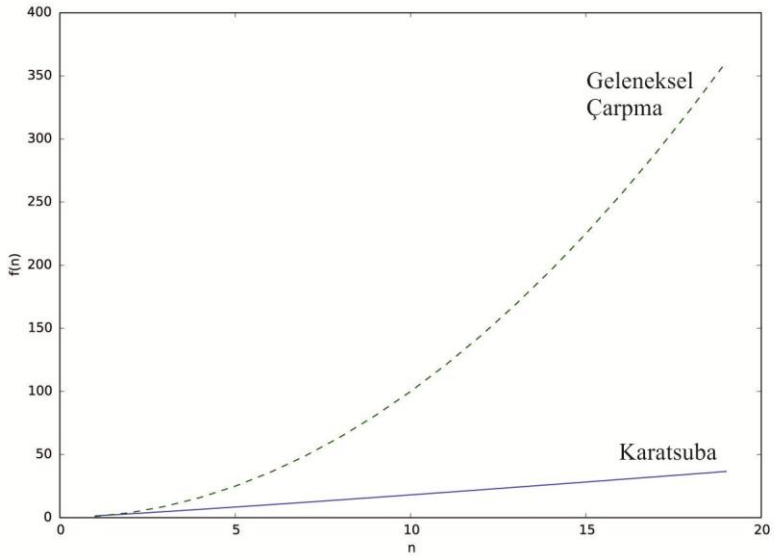
(4.7) denkelminde ifade edilen β değeri α ve γ cinsen şu şekilde yazılabilir;

$$\beta = (a_1 + a_0)(b_1 + b_0) - a_1 b_1 - a_0 b_0 \tag{4.10}$$

$$\beta = (a_1 + a_0)(b_1 + b_0) - \alpha - \gamma \tag{4.11}$$

(4.2) ve (4.3) denkelmelrinde alt bölümlere ayrılan iki sayının çarpma işlemini hızlandırmak için, sayının uzunluğuna göre (4.6) (4.7) ve (4.8) denklemlerindeki çarpım sonuçları karatsuba yöntemi tekrar edilerek elde edilebilir. Bu yöntem yüksek

uzunlunluktaki rakamların çarpma işleminde büyük avantaj sağlamaktadır [40]. Geleneksel çarpma yöntemi ile $O(n^2)$ olan işlem uzunluğu Karatsuba yöntemi ile $O(n^{\log(2)3})$ olmaktadır [39]. Şekil 4.6'da geleneksel çarpma yöntemi ile Karatsuba çarpma yönteminin karşılaştırılması görülmektedir [39]. Burada n sayı uzunluğunu f(n) ise n basamaklı iki sayının çarpımı için işlem uzunluğunu ifade etmektedir.



Şekil 4.6. Geleneksel çarpma ile Karatsuba çarpma algoritmasının karşılaştırılması [39]

BÖLÜM 5

LOOK UP TABLOSU KULLANARAK YÜKSEK KAPASİTELİ ÇARPMA DEVRESİ TASARIMI

5.1. Look-Up Tablosu ile Çarpım

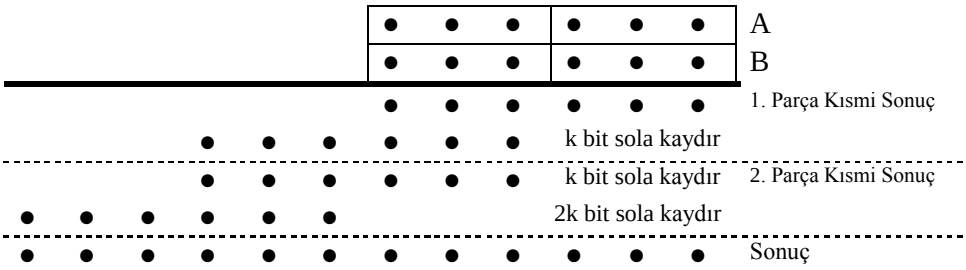
Güçlü hafıza birimlerinin bulunduğu donanım yapılarında kullanıma uygun, etkili ve hızlı bir çarpma yöntemidir. Temel olarak look-up tablosu çarpıcıları, giriş birimlerinin ihtimalleri kadar çarpım sonucu içeren blok hafıza birimleri kullanarak çarpma işleminin sonucu elde edilir. Bu yöntem ile gerçek anlamda bir çarpma işlemi yapılmadığından, çok daha kısa bir süre içinde sonuca ulaşılır. Fakat bu yöntemin en büyük dezavantajı, işlem yapılan sayının boyutu arttıkça look-up tablosunun boyutu da katlanarak artmasıdır. Hafıza birimi içerisinde bütün çarpım ihtimalleri bulunduğundan hiç kullanılmama durumuna rağmen bazı veriler hafızada yer kaplayabilmektedir. Fakat şifreleme, şifre çözme veya görüntü işleme gibi sürekli ve değişken bir sinyal akışının bulunduğu durumlarda gerçek zamanlı çarpma performansı oldukça yüksektir.

Tablo 5.1. 3 bitlik iki sayının çarpımı Look Up Tablosu

A	B	Sonuç	A	B	Sonuç	A	B	Sonuç	A	B	Sonuç
000	**000**	000000	010	**000**	000000	100	**000**	000000	110	**000**	000000
000	001	000000	010	001	000010	100	001	000100	110	001	000110
000	010	000000	010	010	000100	100	010	001000	110	010	001100
000	011	000000	010	011	000110	100	011	001100	110	011	010010
000	100	000000	010	100	001000	100	100	010000	110	100	011000
000	101	000000	010	101	001010	100	101	010100	110	101	011110
000	110	000000	010	110	001100	100	110	011000	110	110	100100
000	111	000000	010	111	001110	100	111	011100	110	111	101010
001	**000**	000000	011	**000**	000000	101	**000**	000000	111	**000**	000000
001	001	000001	011	001	000011	101	001	000101	111	001	000111
001	010	000010	011	010	000110	101	010	001010	111	010	001110
001	011	000011	011	011	001001	101	011	001111	111	011	010101
001	100	000100	011	100	001100	101	100	010100	111	100	001100
001	101	000101	011	101	001111	101	101	011001	111	101	100011
001	110	000110	011	110	010010	101	110	011110	111	110	101010
001	111	000111	011	111	010101	101	111	100011	111	111	110001

Tablo 5.1'de 3 bitlik iki adet binary sayının look-up tablosu verilmiştir. Tabloda muhtemel 64 adet sonuç görülmektedir. Çarpma işleminde look up tablosunun giriş veri uzunluğunun k-bit olduğu durumda look up tablosunun içerdiği muhtemel sonuç miktarı 2^{2k} olacaktır. Ancak tabloya dikkatlice bakıldığında A, B girişlerinden en az birisi sıfır olduğunda sonuç değerlerinin sıfır değeri taşıdığı görülmektedir. Bu durum look-up tablosunda yer işgal etmeye gerek kalmaksızın doğrudan çıkışa sıfır değeri gönderilebilir. Böylelikle hafıza biriminden giriş bit uzunluğunun durumuna göre $2^{2k}/(2.2^k - 1)$ 'de biri kadar tasarruf etmek mümkün olur.

Yüksek veri boyutlarında look-up tablosu ile doğrudan çarpma işlemi yapmak, look-up tablosunun hafıza biriminde kapladığı alan sebebiyle kullanışlı olmayabilir. Bu durumda kısmi look-up tablosu kullanılarak standart çarpma işleminin süresi kısaltılabilir.



Şekil 5.1. 6 bit iki sayının 3 bit look-up tablosu ile çarpma işlemi

Şekil 5.1'de iki adet 6 bit sayının çarpma işleminin 3 bit bir look-up tablosu kullanılarak gerçekleştirilmesi görülmektedir. Look-up tablosunun bit uzunluğuna göre sayı parçalara bölünerek parçaların look-up tablosu sonucuna göre bölümler halinde kısmi sonuçlar elde edilmektedir. Kısmi sonuçlar elde edilirken look-up tablosunun bit uzunluğuna ve adım sayısına göre k-bit sola kaydırma işlemi yapılmaktadır. Bu şekilde standart kaydırma toplama biçiminde çarpma işlemi gerçekleştirilirken, çarpma işlemini fiziki olarak gerçekleştirip sonuç elde etmek yerine sonuç değerine doğrudan look up tablosundan ulaşılmaktadır. Bu yöntemle k-bit uzunluğu kadar işlem döngüsünden tasarruf edilmiş olur.

5.2. n Bit Yüksek Hızlı LUT Çarpma Devresi Tasarımı

n bit uzunluğundaki A ve B sayılarını çarpmak için sayılar *k* bit uzunluğundaki $\frac{n}{k}$ adet parçaya bölünür. i $\leq \frac{n}{k}$ ve j $\leq \frac{n}{k}$ olmak üzere, her *t* zamanında A_j ve B_i sayı parçaları look-up tablosu kullanılarak kısmi sonuçlar elde edilir. Kısmi sonuçlar toplanarak çarpım sonucuna ulaşılır. Çarpım devresinde kullanılacak olan toplama devresi, bir bitlik bir toplayıcı devresinin uygun şekilde çoğaltılarak genişletilmesi ile elde edilmiştir. Bunun sağladığı avantaj ile toplama işlemi, 2n bit uzunluğundaki bir toplama işlemini bir saat vuruşu adımında gerçekleştirebilmektedir.

Genişletilebilir toplama devresi kullanılarak gerçekleştirilen look up tablosu ile çarpma işleminde, işlem adım sayısı (*t*) ile çarpım ve çarpılan sayıların bit uzunluğu (*n*) ve parça bit uzunluğu (*k*) arasındaki ilişki (5.1)'deki eşitlik ile ifade edilebilir.

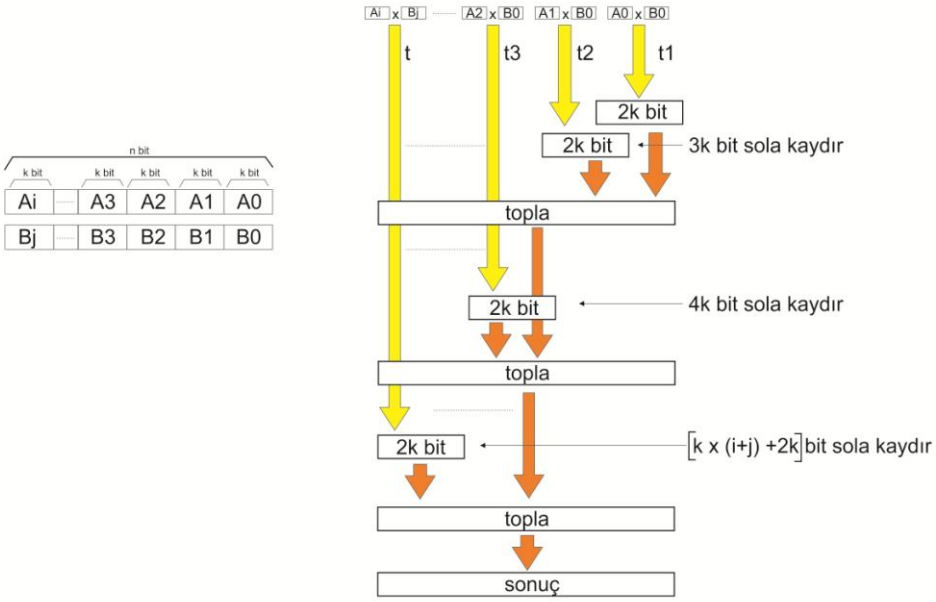
$$t = 2^{2[\log_2(n) - \log_2(k)]} \tag{5.1}$$

Look up tablosunun boyutu, çarpma devresinin uygulanacağı FPGA yongasının kapasitesi dikkate alınarak hesaplanması gerekir. Look up tablosunun ihtimal sayısı (5.2)'deki gibi ifade edilebilir.

$$LUT\ ihtimal\ sayısı\ =\ 2^{2k} \tag{5.2}$$

Girişlerden herhangi birisinin sıfır olduğu durumda çıkış değerinin diğer girişin durumu gözetilmeksizin sıfır olacağı düşünülürse ihtimal sayısı (5.3)'de görüldüğü gibi azaltılabilir.

$$LUT\ ihtimal\ sayısı\ =\ 2^{2k} - 2.2k - 1 \tag{5.3}$$



Şekil 5.2. Kaydırma ve toplama işlemi

Şekil 5.2'de her A_j ve B_i parçasının t_a anlık durumuna göre toplama işleminden önce kaç bit sola kaydırma işlemi yapıldığı gösterilmiştir.

$t_a \leq t$ olmak üzere, t_a anlık durumuna göre, LUT girişlerine uygulanacak k bit uzunluğunda, n bitlik A çarpılan verisinin j parçası ile n bitlik B çarpan verisinin i parçası arasındaki ilişki (5.4) ve (5.5)'teki gibi ifade edilebilir.

$$j = t_a \left(mod \ \frac{n}{k} \right) \tag{5.4}$$

$$i = \left[t_a - t_a \left(mod \ \frac{n}{k} \right) \right] \left(mod \ (\frac{n}{k} - 1) \right) \tag{5.5}$$

Buna göre t_a durumunda A ve B verisinin çarpılan parçaları (5.6) ve (5.7)'deki gibi ifade edilebilir.

$$St_a \rightarrow \ A_j \ x \ B_i \tag{5.6}$$

$$St_a \rightarrow A\left[t_a\left(mod\ \frac{n}{k}\right)\right]\ x\ B\left[t_a - t_a\left(mod\ \frac{n}{k}\right)\left(mod\ (\frac{n}{k} - 1)\right)\right] \quad (5.7)$$

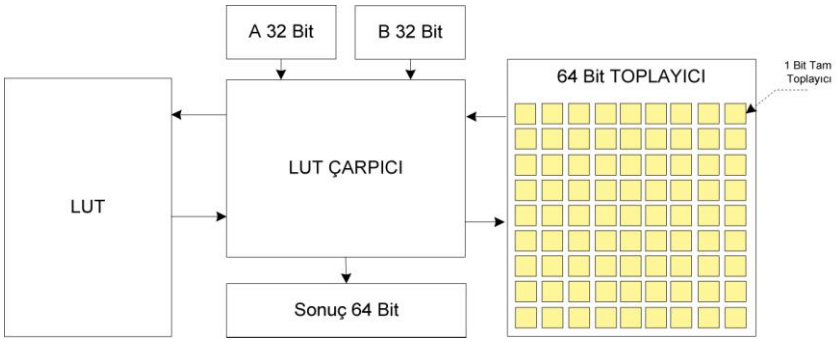
Tablo 5.2. 1024 bit veri girişi için farklı parça uzunluklarında look-up tablosu ve adım sayısı

k	8	32	128	512
LUT	65536	1.8×10^{19}	1.15×10^{77}	1.79×10^{308}
Saat döngüsü	16384	1024	64	4

5.3. 32 Bit Yüksek Hızlı LUT Çarpma Devresinin VHDL ile Gerçekleştirilmesi

Şekil 5.3'te görüldüğü gibi VHDL ile oluşturulacak olan LUT çaprma devresinin 3 temel kısmı vardır.

- Bir bitlik toplayıcıların genişletilmesi ile oluşan 64 bitlik bir toplayıcı,
- Look up tablosundan oluşan ROM,
- Giriş verilerini parçalara bölüp ilgili parçaların çarpım sonucunu look up tablosundan okuyan ve toplayıcıya gönderen temel devre,



Şekil 5.3. VHDL ile oluşturulan 32 Bit Çarpma devresinin temel blok şeması

5.3.1. 64 Bit Yüksek Hızlı Toplayıcı Devresi Tasarımı

Çarpma işleminin temelinde çoklu toplama işlemi olması sebebiyle, bir çarpma devresi tasarımında toplama devresinin hızı oldukça önemlidir. Standart bir toplama işleminde basamak sayısı arttıkça toplama işleminin süresi de artmaktadır. Yüksek kapasiteli sayıların çarpımını hızlı bir şekilde yapabilmek için tasarlanan bu çarpma devresinde standart bir toplama işlemi devresi, yüksek veri boyutlarında istenilen

sonucu verememektedir. Yüksek verilerin toplanmasında karşılaşılan bu sorunu çözmek için, giriş verilerinin boyutuna uygun olarak bir bitlik tam toplayıcıların uygun şekilde genişletilmesi ile tasarlanan toplayıcı devresi kullanılmıştır.

Tasarlanan bu toplayıcı devresinin en büyük avantajı giriş ve çıkışı tek saat vuruşunda birbirine paralel olarak bağlanmış bir bitlik tam toplayıcı devreleri sayesinde toplayabilmesidir. Tasarlanmış olan bu devrenin diğer avantajı ise boyutunun istenilen kapasitede genişletilebilir olmasıdır. Kapasitesi, içerisine yerleştirilecek FPGA donanımının boyutuna bağlı olarak, ihtiyaç duyulan miktarda genişletilebilirken toplama işlemi yine aynı sürede, yani bir saat vuruşunda yapılabilmektedir.

Tasarlanan toplama devresi işlem süresi sabit kalacak şekilde işleyebildiği veri miktarı genişletilebildiği için, çarpma devresini boyutu da istenilen seviyede arttırılabilir.

Tablo 5.3. 64 Bit Toplayıcı devresinin VHDL kodları g/ç bağlantıları ve bit genişlikleri

	Kısaltma	Giriş / Çıkış	Bit Genişliği	Açıklama
Bir Bit Tam Toplayıcı	A_BIT	Giriş	1 Bit	Bir Bit Tam Toplayıcı A girişi
	B_BIT	Giriş	1 Bit	Bir Bit Tam Toplayıcı B girişi
	C_BIT	Giriş	1 Bit	Bir Bit Tam Toplayıcı Elde girişi
	COUT	Çıkış	1 Bit	Bir Bit Tam Toplayıcı Elde çıkışı
	SUM_BIT	Çıkış	1 Bit	Bir Bit Tam Toplayıcı Sonuç çıkışı
64 Bit Toplayıcı	A	Giriş	64 Bit	64 Bit toplayıcı A girişi
	B	Giriş	64 Bit	64 Bit toplayıcı B girişi
	SUM	Çıkış	64 Bit	64 Bit toplayıcı Sonuç çıkışı

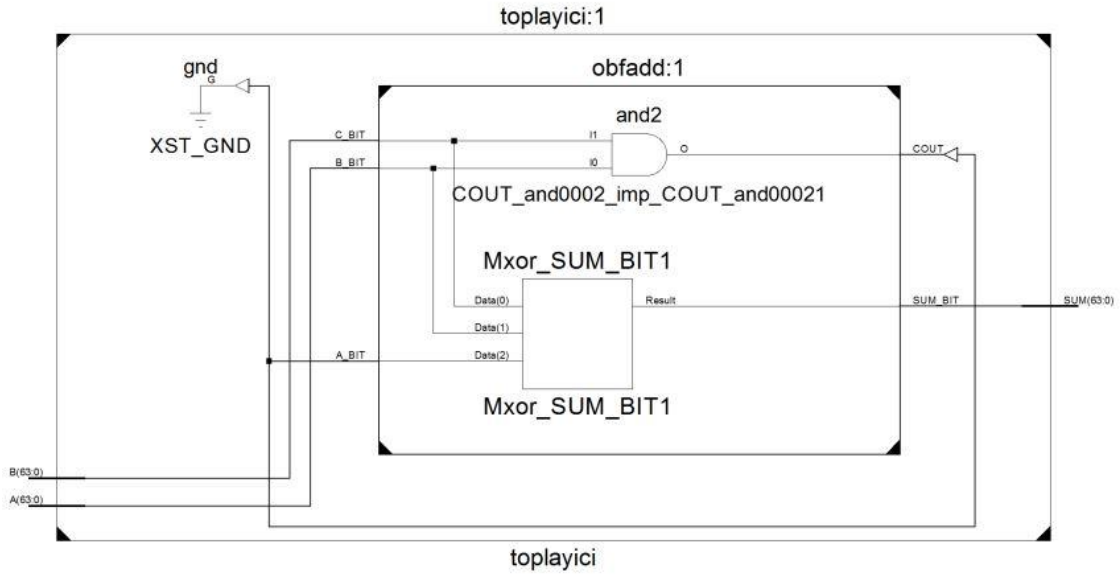
Tablo 5.3'de 64 bitlik toplayıcının VHDL kodları giriş ve çıkış bağlantıları kısaltmaları verilmiştir. 32 bit çarpma devresinin işlem kapasitesi elde basamağına ihtiyaç duymadığı için 64 bit toplayıcıda elde giriş çıkışları kullanılmamıştır.

Tablo 5.4. Bir bitlik tam toplayıcı VHDL kodları *(obfadd.vhd)*

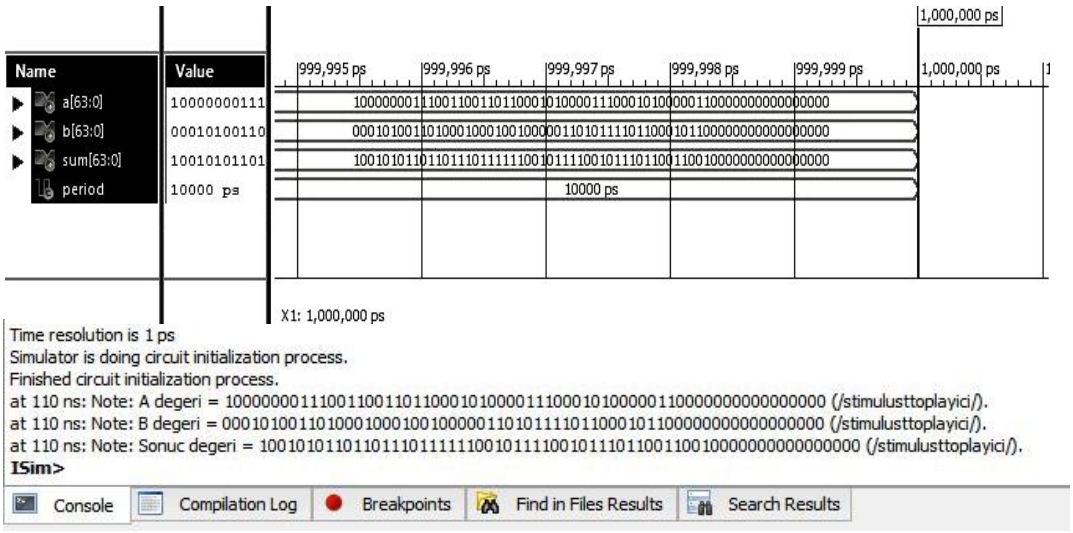
```
26  process(A_BIT, B_BIT, C_BIT)
27  begin
28        SUM_BIT      <=     (A_BIT xor B_BIT) xor C_BIT;
29        COUT   <=     (A_BIT and B_BIT) or (A_BIT and C_BIT) or (B_BIT and
          C_BIT);
30  end process;
```

Tablo 5.5. 64 Bitlik tam toplayıcının VHDL kodları (*toplayici.vhd)*

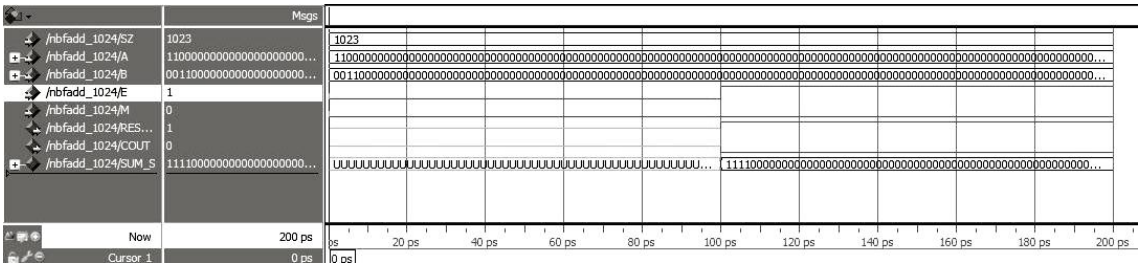
```
26  architecture struct of toplayici is
27  component obfadd
28  port(
29        A_BIT,B_BIT,C_BIT   : in std_logic;
30       SUM_BIT, COUT        : out std_logic);
31  end component;
32  signal T      :      std_logic_vector(64 downto 0);
33  begin
34  T(0) <= '0';
35  FA: for i in 0 to 63 generate
36  obfadd_i:     obfadd port map (T(i), A(i), B(i), SUM(i), T(i+1));
37  end generate;
38  end struct;
```



Şekil 5.4. 64 Bit Toplayıcı devresinin ISE programı ile elde edilmiş RTL şeması



Şekil 5.5. 64 Bit Toplayıcının ISIM programı ile elde edilen benzetim sonuçları



Şekil 5.6. 1024 Bit toplayıcı ModelSIM benzetim sonuçları

Şekil 5.6'da aynı yöntemle oluşturulan 1024 bit bir toplayıcının benzetim sonuçları görülmektedir. Şekil 5.5'te verilen 64 bit toplayıcının benzetim sonuçları ile karşılaştırıldığında, ikisinde de sonucun bir saat vuruşu kadar geçen sürede elde edildiği görülmektedir. Toplayıcı devresi, yerleştirileceği FPGA donanımının kapasitesi yeterli olduğu sürece, toplama işlemi için geçen süre değişmeden, giriş boyutları istenilen miktarda arttırılabilmektedir. Bu durum ile tasarlanan çarpma devresi boyutları istenilen boyutlarda genişletilebilir.

5.3.2. Look Up Tablosu Tasarımı

Look up tablosu, çarpan ve çarpılan sayı değerlerinin önceden hesaplanan bütün olasılık sonuç değerlerini saklamak için kullanılmıştır. Tasarlanan çarpma devresinde bu devre FPGA üzerinde en fazla alan kaplayan kısımdır. Bu sebeple, look up tablosu tasarımı yapılırken, kullanılacak olan FPGA donanımının kapasitesini göz önünde bulundurmak gerekmektedir.

Giriş değerlerinin bit uzunluğuna göre tablonun ihtimal sayısı, denklem (5.2) ve denklem (5.3)'te ifade edilmiştir. Buna göre, 32 bit iki adet binary sayıyı tek parça halinde çarpmak için 32 bit giriş uzunluğunda bir LUT tasarlanmak istenirse LUT ihtimal sayısı (5.9)'da ifade edilmiştir.

$$LUT\ ihtimal\ sayısı = 2^{2.32} = 2^{64} \tag{5.8}$$

$$LUT\ ihtimal\ sayısı \cong 1{,}8448\ x\ 10^{19} \tag{5.9}$$

Kullanılacak olan Xilinix Virtex 5 XC5VLX50T FPGA donanımının kapasitesine uygun olarak, 32 bit uzunluğundaki girişler $k=8$ bit uzunluğunda parçalara bölünebilir. Bu durumda LUT ihtimal sayısı denklem (5.10)'da ifade edilmiştir.

$$LUT\ ihtimal\ sayısı = 2^{2.8} = 2^{16} = 65536 \tag{5.10}$$

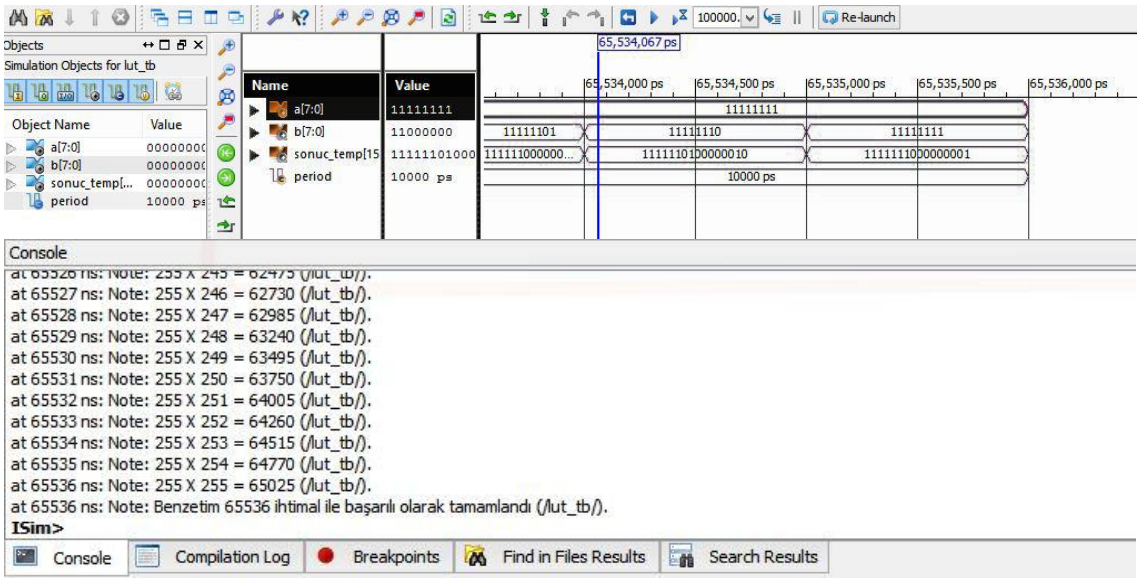
Bu durumda çarpım sonucunu elde edebilmek için gerekli olan süre denklem (5.1)'den;

$t = 2^{2[\log_2(32) - \log_2(8)]} = 2^{2[5-3]} = 2^4 = 16$ adım olarak bulunur. (5.11)

Look up tablosunun oluşturulması işlemi tasarımcı tarafından yazılamayacak kadar büyük fazla olduğu için 32 bit çarpma devresinin look up tablosu C++ ile yazdırılmıştır.

Tablo 5.6. Look up tablosunun oluşturulması için kullanılan pseudo kodları

```
begin
        LUT.txt dosyasını aç
        LUT.txt<= "case B is"
        while(B < 2^2k){
                LUT.txt <= "when " & binary(B) & "=>"
                LUT.txt <= "  case A is"
         while(A < 2^2k){
                Sonuç = A * B
                LUT.txt <= "    when" & binary(A) & "=> sonuc_temp <=" & binary(Sonuç) & ";"
                A'yı 1 arttır
          }
         LUT.txt <= " end case;"
         B'yi 1 arttır
        }
        LUT.txt <= " end case;"
        LUT.txt dosyasını kapat
end
```

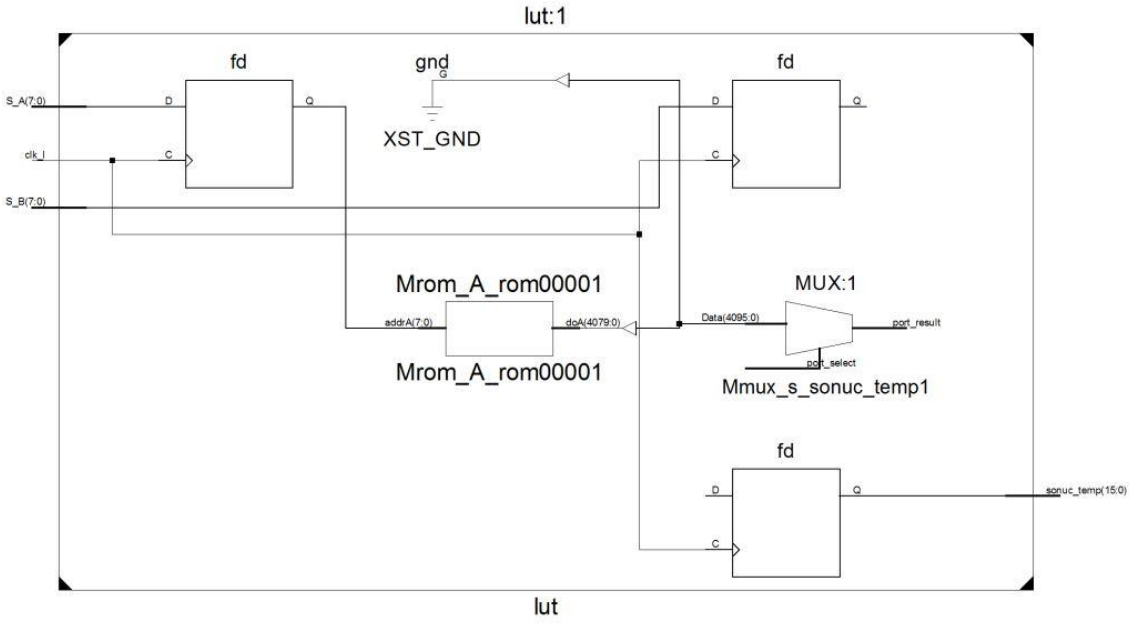


Şekil 5.7. Look Up Tablosunun ISIM ile elde edilen benzetim sonuçları

Şekil 5.7'de VHDL ile oluşturulan look up tablosunun benzetim sonuçları görülmektedir. Devrenin k bit uzunluğundaki girişlerine uygulanması muhtemel 2^{2k}

ihtimalin hepsi 2^{2k} sürede test edilmiştir. Benzetim sonucuna göre devrenin rastgele giriş değerlerine göre bir saat vuruşunda sonuç verdiği görülmektedir.

LUT devresi ile fiziksel bir çarpma işlemi yapılmadığı için girişlerine uygulanan değerlere göre herhangi bir ek adresleme devresine ihtiyaç duymadan sonuç verebilmektedir.



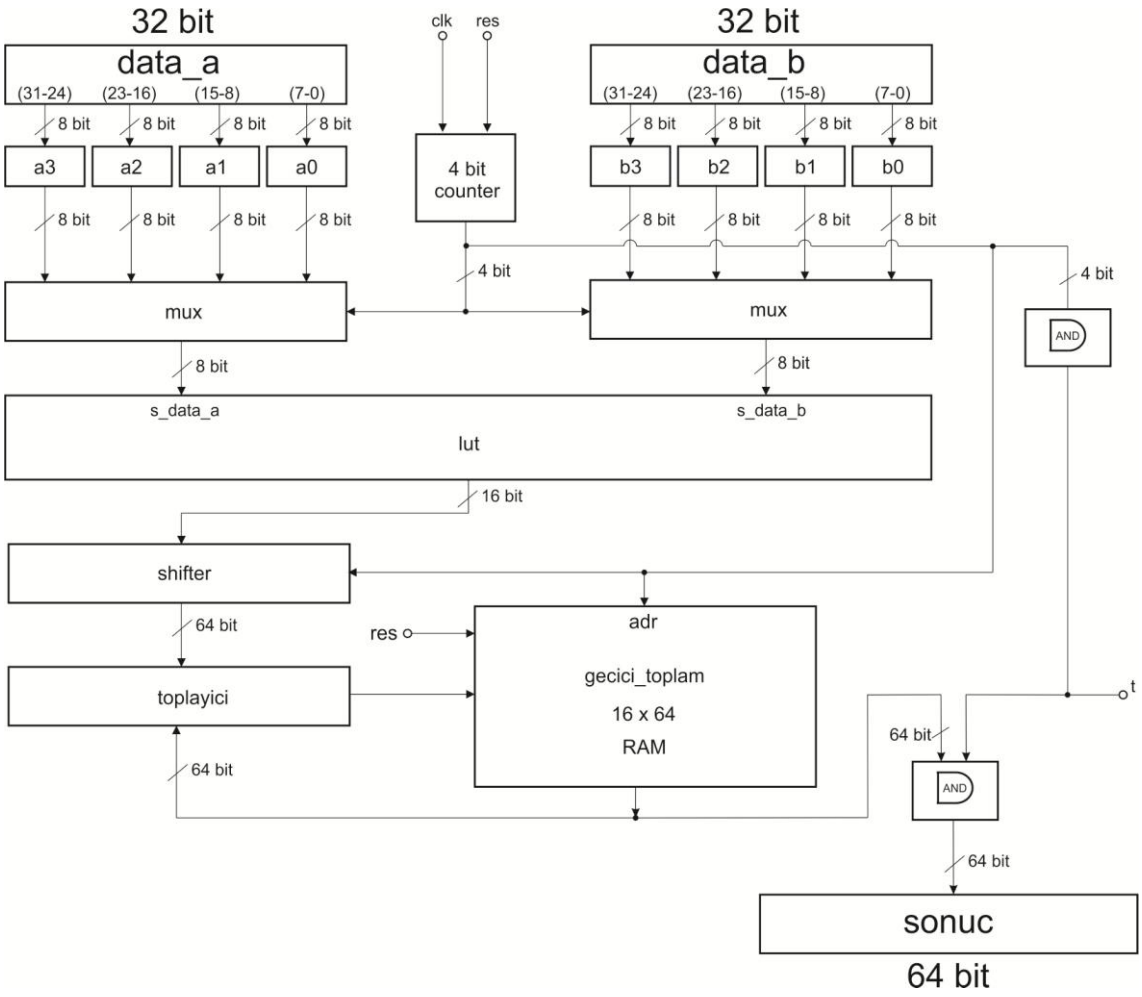
Şekil 5.8. LUT devresinin ISE programı ile elde edilmiş RTL şeması

Tablo 5.7. Virtex 5 xc5vlx50t-2ff1136 üzerinde mantık parçaları kullanımı

Mantık Parçaları Kullanımı	Kullanılan	Mevcut	Yüzde yerleşim
Kaydedici parça sayıları	33	28,800	1%
LUT parça sayıları	6,699	28,800	23%
Kullanılan parça sayıları	1,731	7,200	24%
Bağlı G/Ç sayıları	33	480	6%

5.3.3. Yüksek Hızlı LUT Çarpma Devresi Tasarımı

32 bit iki sayıyı çarpmak için tasarlanmış çarpma devresinde 8 bit girişe sahip look-up tablosu, look-up tablosundan gelen değerleri her adımda toplayacak olan 64 bitlik toplayıcı devresi kullanılmıştır.



Şekil 5.9. 32 Bit LUT Çarpma devresinin blok şeması

Denklem (5.11)'de ifade edildiği gibi 32 bit giriş ve 8 bit parça veri uzunluğuna sahip iki sayının çarpma işlemi 16 adımda gerçekleştirilmektedir. Her saat vuruşunda bir adım işlem süresi geçmektedir. Buna göre adım sayısı, her saat vuruşunda artan bir adet 4 bit yukarı sayıcı ile ifade edilebilir. Sayıcıdan gelen adım sayısına göre *A* ve *B* verisinin hangi parçalarının *lut* girişine uygulanacağı ve hangi bit dizisinin toplanacağı belirlenir.

Tablo 5.8. 32 bit LUT Çarpma devresi giriş çıkış bağlantı açıklamaları

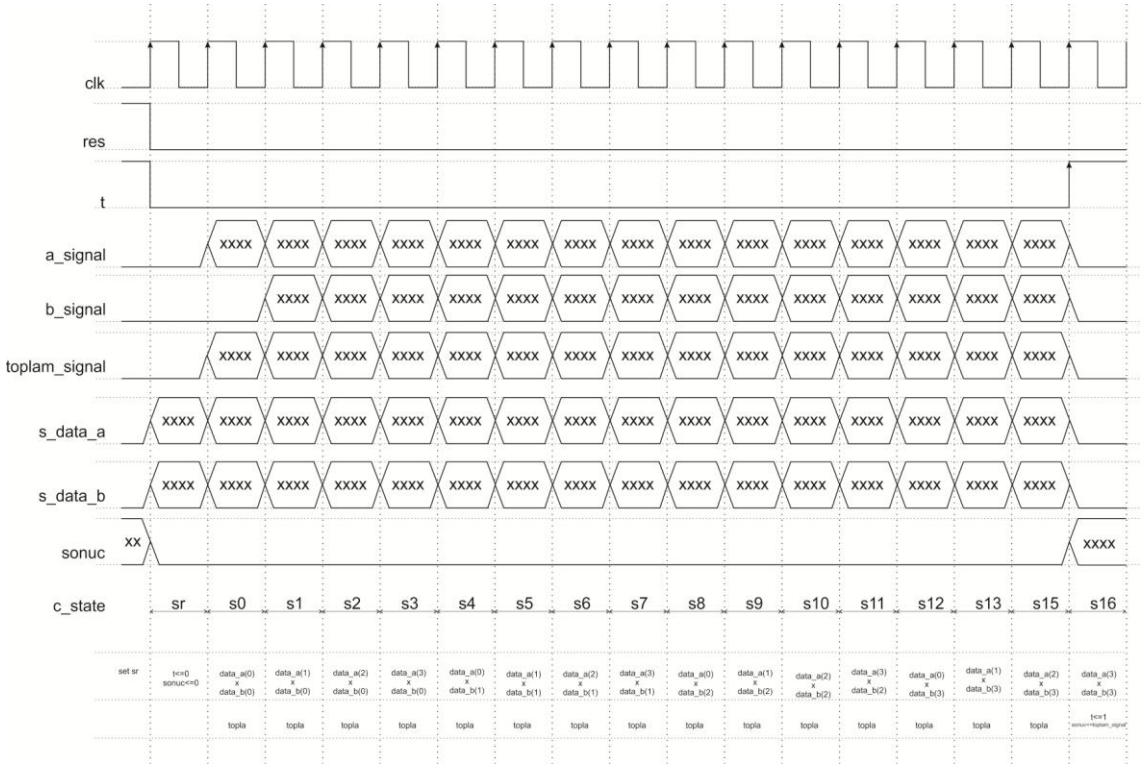
Dosya adı	Tür	Kısaltma	Tip	Bit genişliği	Açıklama
lut_mult.vhd	**Ana Program**	res	Giriş	1 bit	Reset girişi
		clk	Giriş	1 bit	Saat sinyali girişi
		data_a	Giriş	32 bit	A sayı girişi
		data_b	Giriş	32 bit	B sayı girişi
		sonuc	Çıkış	64 bit	Çarpım sonucu
		t	Çıkış	1 bit	İşlem durum sonucu t=1 ise sonuç t=0 ise işlem devam ediyor
		c_state	Signal	State type	Geçerli durum (sr, s0-s16)
		n_state	Signal	State type	Sonraki durum (sr, s0-s16)
		s_data_a	Signal	8 bit	Lut bileşeni A girişi için ayrılmış sinyal bağlantısı
		s_data_b	Signal	8 bit	Lut bileşeni B girişi için ayrılmış sinyal bağlantısı
		lut_sonuc	Signal	16 bit	Lut bileşeni sonuc_temp çıkışı için ayrılmış sinyal bağlantısı
		a_signal	Signal	64 bit	Toplayıcı bileşeni A girişi için ayrılmış sinyal bağlantısı
		b_signal	Signal	64 bit	Toplayıcı bileşeni B girişi için ayrılmış sinyal bağlantısı
		toplam_signal	Signal	64 bit	Toplayıcı bileşeni SUM çıkışı için ayrılmış sinyal bağlantısı
lut.vhd	Component	A	Giriş	8 bit	A sayı girişi
		B	Giriş	8 bit	B sayı girişi
		sonuc_temp	Çıkış	16 bit	Çarpım sonuç çıkışı
toplayici.vhd	Component	A	Giriş	64 bit	A sayı girişi
		B	Giriş	64 bit	B sayı girişi
		SUM	Çıkış	64 bit	Toplama sonuç çıkışı

Tablo 5.8'de devrenin ana ve alt bileşenlerinin giriş çıkış bağlantıları ve ana devrenin bileşenler ile bağlantısını sağlayan sinyal bağlantıları verilmiştir.

Tablo 5.9'da görüldüğü gibi çarpma devresinin kod yapısı oluşturulurken (5.1), (5.2), (5.4), (5.5), (5.6), (5.7) ifadeleri dikkate alınmıştır.

Tablo 5.8. LUT ile genişletilebilir çarpma devresi kod yapısı

```
case c_state is
when sr =>
   sonuc <=(others=>'0');
   t<='0';
   b_signal <= (others=>'0');
   a_signal <= (others=>'0');
   s_data_a <= data_a(k downto 0);
   s_data_b <= data_b(k downto 0);
   n_state  <= S0;
when Sta =>
   s_data_a<=data_a [k.ta(mod n/k)+(k-1) downto k.ta(mod n/k)];
   s_data_b<=data_b [k.[ta-ta(mod n/k)](mod (n/k-1))+(k-1) downto k.[ta-ta(mod n/k)](mod (n/k-1))] ;
   a_signal[ k(i+j)+2k downto k(i+j) ] <= lut_sonuc;
   b_signal <= toplam_signal;
   t<='0';
   n_state<= Sta + 1;
when Sta(max) =>
   sonuc <= toplam_signal;
   t<='1';
   b_signal <= (others=>'0');
   a_signal <= (others=>'0');
   s_data_a <= (others=>'0');
   s_data_b <= (others=>'0');
   n_state  <= sr;
when others =>
   b_signal <= (others=>'0');
   a_signal <= (others=>'0');
   s_data_a <= (others=>'0');
   s_data_b <= (others=>'0');
   sonuc    <= (others=>'0');
   t<='0';
end case;
```



Şekil 5.10. Geçerli duruma göre giriş çıkış ve sinyal bağlantı durumları

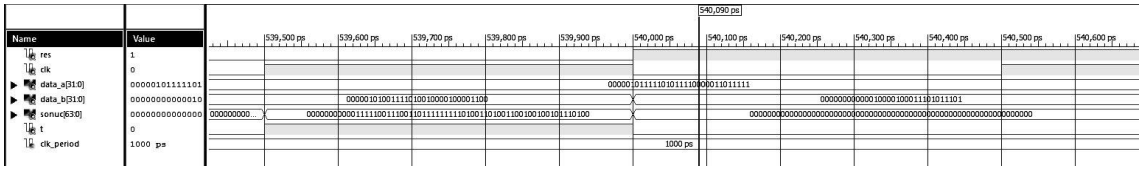
Tasarlanan devrenin durum zamanlaması Şekil 5.10'daki gibi olmaktadır. "c_state" devrenin o an için geçerli durumunu ifade etmektedir. Şekil 5.10'da "s_data_a" ve "s_data_b" olarak ifade edilen sinyaller, look-up devresinin girişleridir. "sr" durumundan itibaren her durumda "data_a" ve "data_b"nin hangi parçalarının look-up devresinin girişine uygulanacağı gösterilmiştir. "a_signal" ve "b_signal" ile ifade edilen sinyal bağlantıları toplayıcı devresinin giriş bağlantılarıdır. "s0" durumundan itibaren look-up devresinin çıkışı "a_signal" toplayıcı devresinin girişine aktarılmaktadır. "toplam_signal" olarak ifade edilen toplayıcı devresinin çıkış sinyal bağlantısıdır. "s0" durumunda a_signal" girişindeki değişime kadar, "s16" durumunda "a_signal" ve "b_signal" girişlerinin lojik sıfır olmasından dolayı çıkışı lojik sıfırdır. "s0" durumundan itibaren "s16"ya kadar her durumda toplayıcı devresinin bir önceki sonucu "b_signal" giriş değerini belirlemektedir. Bu yüzden "b_signal" girişindeki değişim "s1" durumunda olmaktadır.

5.3.4. 32 Bit Yüksek Hızlı LUT Çarpma Devresinin FPGA Uygulaması

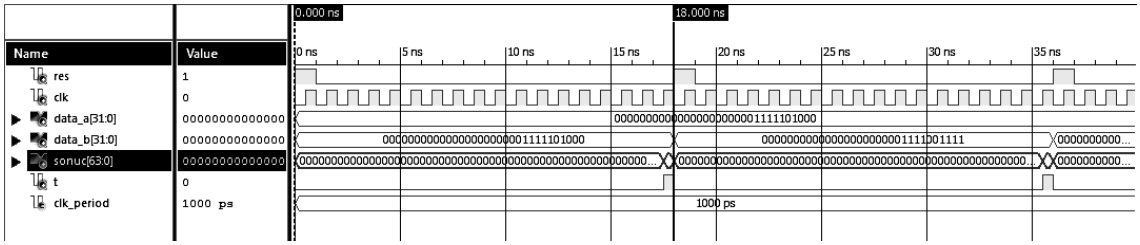
Çarpma devresinin sentezlenmesi ve benzetiminin yapılması aşamasında farklı donanım yapıları üzerinde devre yapısını denemek ve farklı ortamlarda benzetim sonuçları elde edebilmek için Quartus II v9.0 - v14.1 Web Edition, ISE Design Suite v.14.7 ve Vivado v2014.14 programları kullanılmıştır.

EK-B'de programın VHDL test kodları verilmiştir. Devrenin data_a ve data_b girişlerine rastgele giriş değerleri verilerek sonuç değeri gözlemlenmiştir. Şekil 5.11 (a)'da data_a ve data_b girişlerine göre alınan sonuç değeri görülmektedir. Şekil 5.10 (b)'de reset girişinin lojik 0 olmasının ardından 16 döngüde t çıkışının lojik 1 olduğu ve sonuç değerinin elde edildiği görülmektedir. Diğer durumlarda t çıkışı lojik 0 olmaktadır ve sonuç çıkışının tüm bitleri lojik 0 sıfır değeri vermektedir. Şekil 5.11 (c)'de testbench programına göre data_a girişindeki değişim döngüsü görülmektedir.

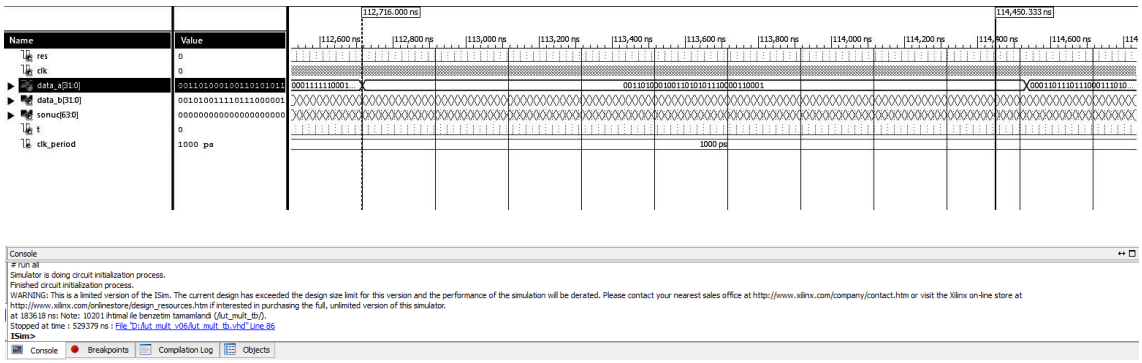
(a)



(b)

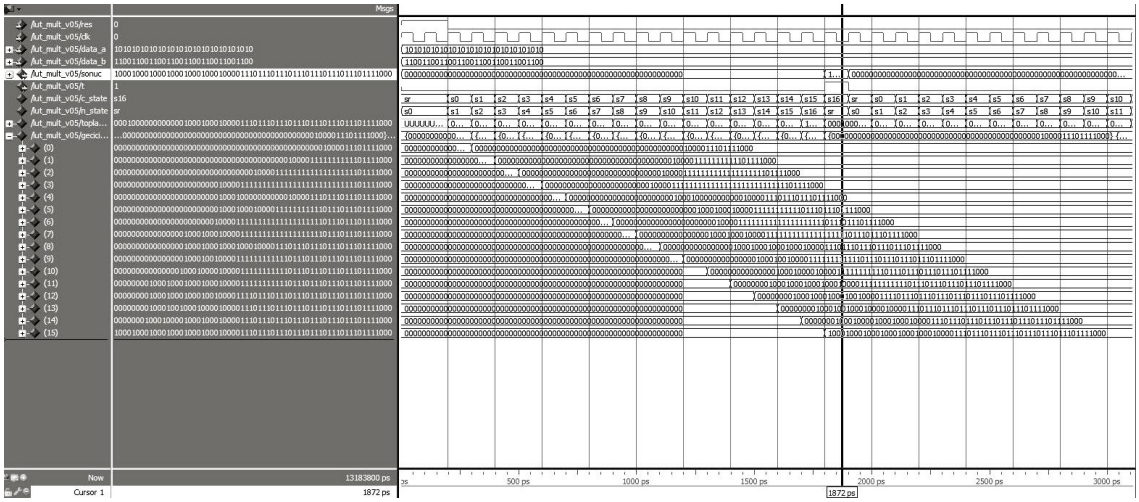


(c)



Şekil 5.11. 32 bit LUT Çarpma Devresinin ISIM programı ile elde edilmiş benzetim sonuçları

Şekil 5.12'deki benzetim sonuçları ModelSIM programı ile elde edilmiştir. Bu benzetim sonuçlarında ise *c_state* durumunun aldığı değerlere göre her adımda toplama işleminin sonuçları görülmektedir. Bu sonuçlara göre gerçek zamanlı bir veri akışında çarpma işleminin davranışı gözlemlenebilir.

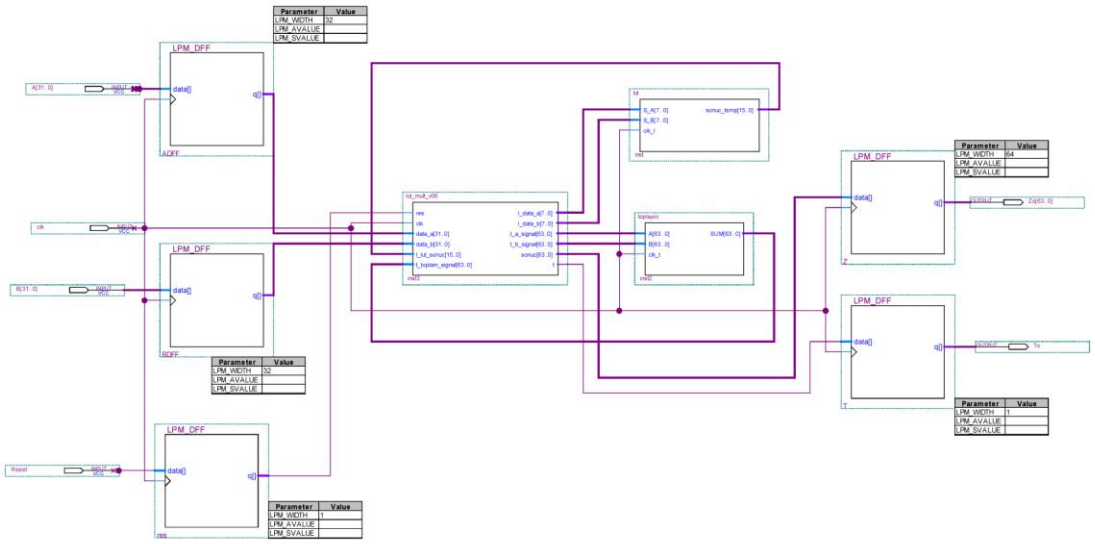


```
VSIM 3> force -freeze sim:/lut_mult_v05/res 1 0, 0 {200 ps}
VSIM 4> force -freeze sim:/lut_mult_v05/clk 1 0, 0 {50 ps} -r 100
VSIM 5> force -freeze sim:/lut_mult_v05/data_a 10101010101010101010101010101010 0
VSIM 6> force -freeze sim:/lut_mult_v05/data_b 11001100110011001100110011001100 0
VSIM 7> run -all
```

Şekil 5.12. 32 bit LUT Çarpma devresinin ModelSIM benzetim sonuçları

Devrenin yüksek hızlardaki performansını, kullanılan FPGA yongası ve devrenin FPGA yongası üzerindeki yerleşimi belirlemektedir. FPGA içerisindeki transistorların veri iletim gecikmeleri yüksek hızlarda devrenin performansını olumsuz etkilemektedir. Buna göre devrenin maksimum frekansı, giriş ile çıkış arasındaki verinin izleyeceği en uzun mesafe ile belirlenmektedir. Şekil 5.13'te görüldüğü gibi zamanlama analizi yapılırken devrenin giriş ve çıkışlarına kaydediciler eklenmiştir. Kaydediciden kaydediciye verinin ulaşım zamanına göre maksimum frekans ölçülmüştür.

Tablo 5.10'da ISE, Quartus ve Vivado yazılımları ile çeşitli FPGA yongaları üzerinde yapılan sentezlemeler ile maksimum frekans değerleri görülmektedir.



Şekil 5.13. Kaydediciden kaydediciye devrenin zamanlama analizi şema görüntüsü

Tablo 5.9. Çeşitli FPGA donanımları ile 32 bit LUT çarpma devresinin performansı

Program	**ISE 14.7**			**Quartus II 14.1**			**Vivado v2014.4**	
FPGA	Virtex 5 xc5vlx50t-2ff1136	Virtex 6 xc6vlx75t-2ff784	Kintex 7 xc7k70t-fbg676	Cyclone IV ep4cgx150df31I7ad	Cyclone V 5cgxfc7d7f27c8	Cyclone V 5cgxfc7d7f31c8	Kintex 7 xc7k160tffg676-2l	Artix 7 xc7a200tffg1156-3
RR (ns)	1,498	1,171	0,995	1,975	2,763	1,640	1,409	1,592
Fmax (Mhz)	667,557	853,97	1005,03	506,33	361,93	609,76	709,72	628,14

Tablo 5.10. Virtex 5 xc5vlxt-2ff1136 üzerinde güç tüketim analizi

	Güç Tüketimi (W)	**Lojik Güç Tüketimi (W)**	**Sinyal Güç Tüketimi (W)**	**Yaz Boz Sayısı**	**LUT Sayısı**	**G/Ç Sayısı**
lut_mult_v06	0,06044	0,00330	0,05714	20	244	-
lut	0,10619	0,06373	0,04246	0	6408	-
toplayici	0,00138	0,00073	0,00065	0	33	-
obfadd x 64	0,00123	0,00124	0	0	62	-
G/Ç	0,304	0	0	0	0	131
Toplam	0,47324	0,06900	0,10025	20	6747	131

BÖLÜM 6

SONUÇLAR

Bu tez çalışmasında look-up tabloları kullanılarak yüksek hızlı çarpma devresi tasarımı yapılmıştır. 8 bit iki sayının önceden hesaplanan tüm çarpım sonuçlarını tutan look-up devresi ile 32 bit uzunluğunda iki sayının çarpma işlemini gerçekleştiren mantık devresi Virtex 5 xc5vlx50t FPGA donanımı üzerinde sentezlenmiştir. ModelSIM ve ISIM üzerinde yapılan benzetim sonuçlarında 32 bit iki sayının çarpma işleminin, kısmi parçalara ayırarak yapılan çarpma işlemi 16 döngüde sonucunun elde edildiği görülmüştür. Kısmi sonuçları toplama işlemini gerçekleştirmek için tasarlanan yüksek hızlı genişletilebilir toplama devresi, kısmi sonuçları bir döngüde toplayabildiği için devrenin hızlı bir şekilde sayısal işlem yapabilmesini sağlamaktadır. ISE programı ile yapılan zaman analizlerinde devrenin maksimum frekans değeri 667,557 Mhz olarak hesaplanmıştır.

Gerçekleştirilen bu devre aritmetik ünite alt birimi veya şifreleme uygulamalarında devre alt birimi olarak kullanılabilir. Gerçek zamanlı sayısal sinyal işleme için yüksek kapasitelerde hız avantajı sağlayabilmektedir. Üstelik devre look up tabloları kullandığından fiziksel bir çarpma işlemi gerçekleştirmediği için devrenin güç tüketimi de az olmaktadır.

Tablo 6.1. Zaman Karmaşıklığı Karşılaştırması

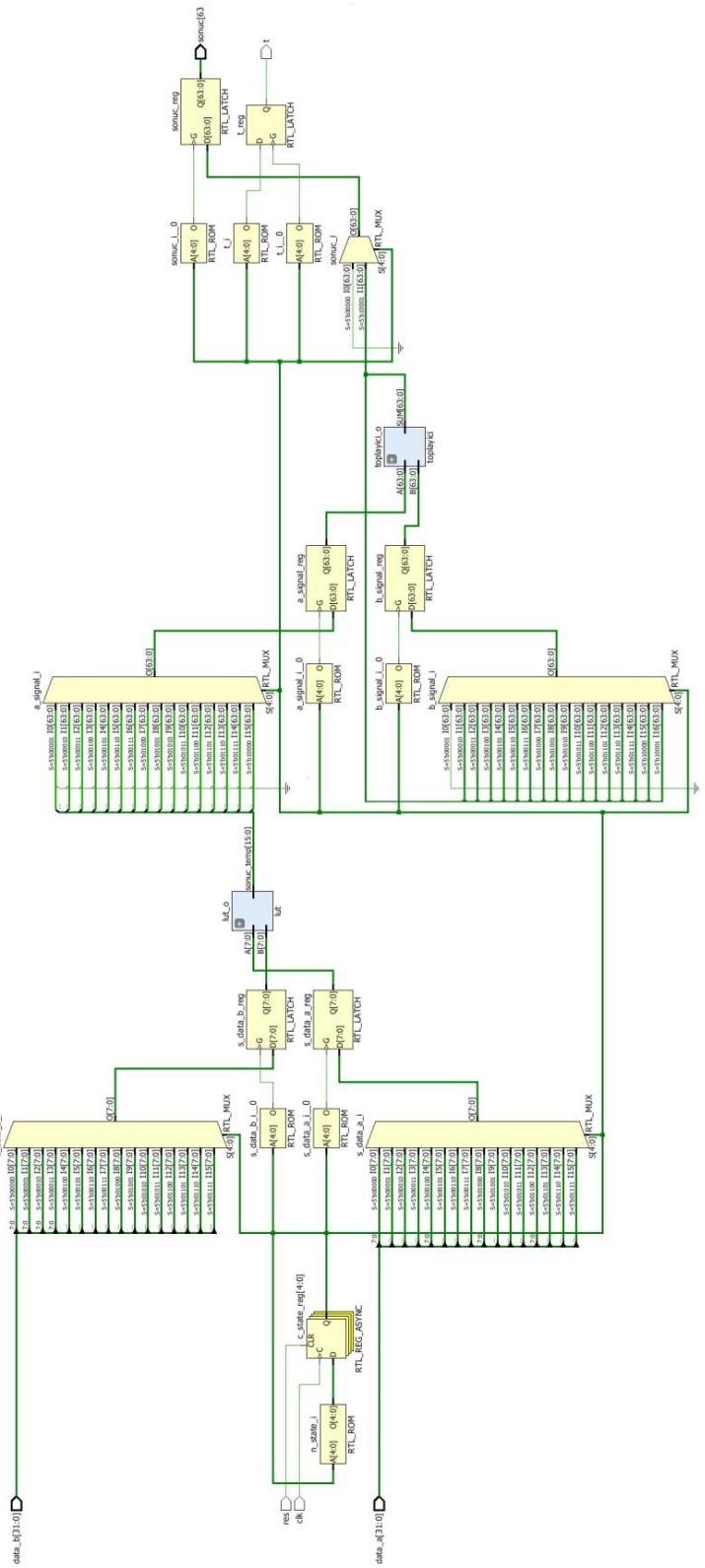
	Zaman Karmaşıklığı	***32 Bit iki sayı için adım sayısı***
Standart Çarpma	$O(n^2)$	1024
Karatsuba	$O(n^{1,585})$	243
Kaydır/Topla	$O(n)$	32
Yüksek Hızlı LUT	$O(2^{2(log_2(n)-log_2(k))})$	16

Gerçekleştirilen devrede sayıların parçalara ayrılmasında kurulan algoritma Karatsuba algoritmasına benzerlik gösterse de, parçaların çarpımı ve kısmi sonuçların toplama işlemi aşamasında farklılık göstermektedir.

Çarpma devresinin kullanılacağı birime göre LUT parçaları genişletilerek tekrar look up tablosu oluşturulabilmektedir. Ancak burada devrenin en büyük sorunu LUT tablosunun boyutu olmaktadır. Önümüzdeki yıllarda FPGA kapasitesinin artması ile birlikte daha yüksek boyutlarda look-up tabloları oluşturularak çarpma işlemi daha kısa sürede gerçekleştirilebilir.

EKLER

EK-A 32 bit Yüksek Hızlı LUT Çarpma Devresi Vivado programı ile elde edilen RTL şeması



EK-B Ana program TestBench VHDL kodları

```
LIBRARY ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;
use ieee.math_real.all;
use ieee.std_logic_arith.all;

library UNISIM;
use UNISIM.Vcomponents.all;


ENTITY lut_mult_tb IS
END lut_mult_tb;

ARCHITECTURE behavior OF lut_mult_tb IS

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

    COMPONENT lut_mult_v05
    PORT(
         res : IN  std_logic;
         clk : IN  std_logic;
         data_a : IN  std_logic_vector(31 downto 0);
         data_b : IN  std_logic_vector(31 downto 0);
         sonuc : OUT  std_logic_vector(63 downto 0);
         t : OUT  std_logic
        );
    END COMPONENT;


--Inputs
   signal res : std_logic := '0';
   signal clk : std_logic := '0';
   signal data_a : std_logic_vector(31 downto 0) := (others => '0');
   signal data_b : std_logic_vector(31 downto 0) := (others => '0');

--Outputs
   signal sonuc : std_logic_vector(63 downto 0);
   signal t : std_logic;

-- Clock period definitions
   constant clk_period : time := 1 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)
   uut: lut_mult_v05 PORT MAP (
          res => res,
          clk => clk,
          data_a => data_a,
          data_b => data_b,
          sonuc => sonuc,
          t => t
        );

-- Clock process definitions
clk_process :process
   begin
                clk <= '0';
                wait for clk_period/2;
                clk <= '1';
                wait for clk_period/2;
end process;
```

```
res_process :process
   begin
               res <= '1';
               wait for clk_period;
               res <= '0';
               wait for clk_period*17;
   end process;

-- Stimulus process
 stim_proc: process

        variable seed1, seed2,seed3,seed4: positive;
        variable rand, rand2: real;
        variable range_of_rand: real:= 1000000000.0;
        variable int1, int2: integer;
        variable z:integer:=0;
        variable logv1, logv2: std_logic_vector(31 downto 0);

 begin

 for i in 0 to 100 loop
               uniform(seed1,seed2,rand);
               int1:=integer(rand*range_of_rand);
               logv1:=conv_std_logic_vector(int1,32);
               data_a <= logv1;

        for j in 0 to 100 loop
                       uniform(seed3,seed4,rand2);
                       int2:=integer(rand2*range_of_rand);
                       logv2:=conv_std_logic_vector(int2,32);
                       data_b <= logv2;
                       wait for clk_period*18;
                       z:=z+1;
        end loop;


end loop;

        report integer'image(z) & " ihtimal ile benzetim tamamlandi";

        wait;

end process;

END;
```

EK-C Sentez Raporu Özeti

```
Release 14.7 - xst P.20131013 (nt64)
Copyright (c) 1995-2013 Xilinx, Inc.  All rights reserved.


Total REAL time to Xst completion: 0.00 secs
Total CPU time to Xst completion: 0.22 secs

Total REAL time to Xst completion: 0.00 secs
Total CPU time to Xst completion: 0.22 secs

=========================================================================
*                            HDL Analysis                               *
=========================================================================
Analyzing Entity <lut_mult_v06> in library <work> (Architecture <struct>).
Entity <lut_mult_v06> analyzed. Unit <lut_mult_v06> generated.

Analyzing Entity <lut> in library <work> (Architecture <beh>).
Entity <lut> analyzed. Unit <lut> generated.

Analyzing Entity <toplayici> in library <work> (Architecture <struct>).
Entity <toplayici> analyzed. Unit <toplayici> generated.

Analyzing Entity <obfadd> in library <work> (Architecture <struct>).
Entity <obfadd> analyzed. Unit <obfadd> generated.


=========================================================================
*                            HDL Synthesis                              *
=========================================================================


=========================================================================
HDL Synthesis Report

Macro Statistics
# ROMs                                                 : 1
 256x4080-bit ROM                                      : 1
# Registers                                            : 1
 18-bit register                                       : 1
# Multiplexers                                         : 1
 16-bit 256-to-1 multiplexer                           : 1
# Xors                                                 : 64
 1-bit xor3                                            : 64

=========================================================================

=========================================================================
*                       Advanced HDL Synthesis                          *
=========================================================================


=========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# ROMs                                                 : 1
 256x4080-bit ROM                                      : 1
# Registers                                            : 18
 Flip-Flops                                            : 18
# Multiplexers                                         : 1
 16-bit 256-to-1 multiplexer                           : 1
# Xors                                                 : 64
 1-bit xor3                                            : 64

=========================================================================

=========================================================================
Final Register Report

Macro Statistics
# Registers                                            : 20
 Flip-Flops                                            : 20

=========================================================================

=========================================================================
*                            Final Report                               *
=========================================================================
Final Results
RTL Top Level Output File Name     : lut_mult_v06.ngr
Top Level Output File Name         : lut_mult_v06
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : No

Design Statistics
# IOs                              : 131

Cell Usage :
# BELS                             : 10094
#      GND                         : 1
#      INV                         : 8
#      LUT2                        : 149
#      LUT3                        : 403
#      LUT4                        : 127
#      LUT5                        : 459
#      LUT6                        : 5685
#      MUXF7                       : 2431
#      MUXF8                       : 830
#      VCC                         : 1
# FlipFlops/Latches                : 20
#      FDC                         : 19
#      FDP                         : 1
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 130
#      IBUF                        : 65
#      OBUF                        : 65
=========================================================================
```

```
Device utilization summary:
---------------------------

Selected Device : 5vlx50tff1136-2


Slice Logic Utilization:
 Number of Slice Registers:              20  out of  28800     0%
 Number of Slice LUTs:                 6831  out of  28800    23%
    Number used as Logic:              6831  out of  28800    23%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:   6838
   Number with an unused Flip Flop:    6818  out of   6838    99%
   Number with an unused LUT:             7  out of   6838     0%
   Number of fully used LUT-FF pairs:    13  out of   6838     0%
   Number of unique control sets:         1

IO Utilization:
 Number of IOs:                         131
 Number of bonded IOBs:                 131  out of    480    27%

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:                1  out of     32     3%


=========================================================================
TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
------------------
------------------------------------+-------------------------+-------+
Clock Signal                        | Clock buffer(FF name)   | Load  |
------------------------------------+-------------------------+-------+
clk                                 | BUFGP                   | 20    |
------------------------------------+-------------------------+-------+

Asynchronous Control Signals Information:
----------------------------------------
------------------------------------+-------------------------+-------+
Control Signal                      | Buffer(FF name)         | Load  |
------------------------------------+-------------------------+-------+
res                                 | IBUF                    | 20    |
------------------------------------+-------------------------+-------+

Timing Summary:
---------------
Speed Grade: -2

   Minimum period: 0.766ns (Maximum Frequency: 1305.909MHz)
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: 106.455ns
   Maximum combinational path delay: 106.037ns

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 0.766ns (frequency: 1305.909MHz)
  Total number of paths / destination ports: 20 / 20
-------------------------------------------------------------------------
Delay:               0.766ns (Levels of Logic = 0)
  Source:            c_state_16 (FF)
  Destination:       c_state_17 (FF)
  Source Clock:      clk rising
  Destination Clock: clk rising

  Data Path: c_state_16 to c_state_17
                              Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDC:C->Q             19   0.396   0.370  c_state_16 (c_state_16)
     FDC:D                    -0.022          c_state_17
    ----------------------------------------
    Total                      0.766ns (0.396ns logic, 0.370ns route)
                                       (51.7% logic, 48.3% route)


Total memory usage is 744912 kilobytes
```

```
Release 14.7 Map P.20131013 (nt64)
Xilinx Mapping Report File for Design 'lut_mult_v06'

Design Information
------------------
Command Line   : map -intstyle ise -p xc5vlx50t-ff1136-2 -w -logic_opt off -ol
high -t 1 -register_duplication off -global_opt off -mt off -cm area -ir off -pr
off -lc off -power off -o lut_mult_v06_map.ncd lut_mult_v06.ngd lut_mult_v06.pcf

Target Device  : xc5vlx50t
Target Package : ff1136
Target Speed   : -2
Mapper Version : virtex5 -- $Revision: 1.55 $
Mapped Date    : Mon Apr 27 13:02:40 2015

Design Summary
--------------
Number of errors:      0
Number of warnings:    0
Slice Logic Utilization:
  Number of Slice Registers:                   20 out of  28,800    1%
    Number used as Flip Flops:                 20
  Number of Slice LUTs:                     6,855 out of  28,800   23%
    Number used as logic:                   6,790 out of  28,800   23%
      Number using O6 output only:          6,790
    Number used as exclusive route-thru:       65
  Number of route-thrus:                       65
    Number using O6 output only:               65
```

```
Slice Logic Distribution:
  Number of occupied Slices:                2,337 out of   7,200   32%
  Number of LUT Flip Flop pairs used:       6,870
    Number with an unused Flip Flop:        6,850 out of   6,870   99%
    Number with an unused LUT:                 15 out of   6,870    1%
    Number of fully used LUT-FF pairs:          5 out of   6,870    1%
    Number of unique control sets:              1
    Number of slice register sites lost
      to control set restrictions:              0 out of  28,800    0%

IO Utilization:
  Number of bonded IOBs:                      131 out of     480   27%

Specific Feature Utilization:
  Number of BUFG/BUFGCTRLs:                     1 out of      32    3%
    Number used as BUFGs:                       1

Average Fanout of Non-Clock Nets:               11.17

-------------------

Section 3 - Informational
-------------------------
INFO:MapLib:562 - No environment variables are currently set.
INFO:LIT:244 - All of the single ended outputs in this design are using slew
   rate limited output drivers. The delay on speed critical single ended outputs
   can be dramatically reduced by designating them as fast outputs.
INFO:Pack:1716 - Initializing temperature to 85.000 Celsius. (default - Range:
   0.000 to 85.000 Celsius)
INFO:Pack:1720 - Initializing voltage to 0.950 Volts. (default - Range: 0.950 to
   1.050 Volts)
INFO:Map:215 - The Interim Design Summary has been generated in the MAP Report
   (.mrp).
INFO:Pack:1650 - Map created a placed design.

Section 4 - Removed Logic Summary
---------------------------------
   2 block(s) optimized away

Section 5 - Removed Logic
-------------------------

Optimized Block(s):
TYPE                    BLOCK
GND                     XST_GND
VCC                     XST_VCC1

To enable printing of redundant blocks removed and signals merged, set the
detailed map report option and rerun map.
```

Release 14.7 par P.20131013 (nt64)
Copyright (c) 1995-2013 Xilinx, Inc. All rights reserved.
Place and Route Report

```
KENANBAYSAL-PC::  Mon Apr 27 13:03:27 2015

par -w -intstyle ise -ol high -mt off lut_mult_v06_map.ncd lut_mult_v06.ncd
lut_mult_v06.pcf


Constraints file: lut_mult_v06.pcf.
Loading device for application Rf_Device from file '5vlx50t.nph' in environment F:\Xilinix\14.7\ISE_DS\ISE\.
   "lut_mult_v06" is an NCD, version 3.2, device xc5vlx50t, package ff1136, speed -2

Initializing temperature to 85.000 Celsius. (default - Range: 0.000 to 85.000 Celsius)
Initializing voltage to 0.950 Volts. (default - Range: 0.950 to 1.050 Volts)


Device Utilization Summary:

   Number of BUFGs                           1 out of 32      3%
   Number of External IOBs                 131 out of 480    27%
      Number of LOCed IOBs                   0 out of 131     0%

   Number of Slices                       2337 out of 7200   32%
   Number of Slice Registers                20 out of 28800   1%
      Number used as Flip Flops             20
      Number used as Latches                 0
      Number used as LatchThrus              0

   Number of Slice LUTS                   6855 out of 28800  23%
   Number of Slice LUT-Flip Flop pairs    6870 out of 28800  23%


Overall effort level (-ol):   High
Router effort level (-rl):    High


+---------------------+--------------+------+------+------------+-------------+
|        Clock Net    |   Resource   |Locked|Fanout|Net Skew(ns)|Max Delay(ns)|
+---------------------+--------------+------+------+------------+-------------+
|           clk_BUFGP | BUFGCTRL_X0Y0| No   |    6 |  0.042     |  1.485      |
+---------------------+--------------+------+------+------------+-------------+


------------------------------------------------------------------------------------------------------
  Constraint                                |    Check    | Worst Case |  Best Case | Timing |   Timing
                                            |             |    Slack   | Achievable | Errors |    Score
------------------------------------------------------------------------------------------------------
  TS_clk = PERIOD TIMEGRP "clk" 4 ns HIGH 5 | SETUP       |     2.871ns|     1.129ns|       0|           0
  0%                                        | HOLD        |     0.294ns|            |       0|           0
                                            | MINPERIOD   |     2.502ns|     1.498ns|       0|           0
------------------------------------------------------------------------------------------------------


All constraints were met.


Generating Pad Report.

All signals are completely routed.

Peak Memory Usage:  517 MB

Timing: Completed - No errors found.
```

```
--------------------------------------------------------------------------------
Release 14.7 Trace  (nt64)
Copyright (c) 1995-2013 Xilinx, Inc.  All rights reserved.
PostMap Static Timing Report

F:\Xilinix\14.7\ISE_DS\ISE\bin\nt64\unwrapped\trce.exe -intstyle ise -v 3 -s 2
-n 3 -fastpaths -xml lut_mult_v06_preroute.twx lut_mult_v06_map.ncd -o
lut_mult_v06_preroute.twr lut_mult_v06.pcf -ucf lut_mult_v06.ucf

Design file:              lut_mult_v06_map.ncd
Physical constraint file: lut_mult_v06.pcf
Device,package,speed:     xc5vlx50t,ff1136,-2 (PRODUCTION 1.73 2013-10-13, STEPPING level 0)
Report level:             verbose report

Environment Variable      Effect
--------------------      ------
NONE                      No environment variables were set
--------------------------------------------------------------------------------

Clock to Setup on destination clock clk
---------------+---------+---------+---------+---------+
               | Src:Rise| Src:Fall| Src:Rise| Src:Fall|
Source Clock   |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
---------------+---------+---------+---------+---------+
clk            |    0.943|         |         |         |
---------------+---------+---------+---------+---------+

--------------------------------------------------------------------------------
Release 14.7 Trace  (nt64)
Copyright (c) 1995-2013 Xilinx, Inc.  All rights reserved.
Post Par Static Timing Report

F:\Xilinix\14.7\ISE_DS\ISE\bin\nt64\unwrapped\trce.exe -intstyle ise -v 3 -s 2
-n 3 -fastpaths -xml lut_mult_v06.twx lut_mult_v06.ncd -o lut_mult_v06.twr
lut_mult_v06.pcf -ucf lut_mult_v06.ucf

Design file:              lut_mult_v06.ncd
Physical constraint file: lut_mult_v06.pcf
Device,package,speed:     xc5vlx50t,ff1136,-2 (PRODUCTION 1.73 2013-10-13, STEPPING level 0)
Report level:             verbose report

Environment Variable      Effect
--------------------      ------
NONE                      No environment variables were set
--------------------------------------------------------------------------------

================================================================================
Timing constraint: TS_clk = PERIOD TIMEGRP "clk" 4 ns HIGH 50%;
For more information, see Period Analysis in the Timing Closure User Guide (UG612).

 20 paths analyzed, 20 endpoints analyzed, 0 failing endpoints
 0 timing errors detected. (0 setup errors, 0 hold errors, 0 component switching limit errors)
 Minimum period is   1.498ns.
--------------------------------------------------------------------------------

Paths for end point c_state_14 (SLICE_X22Y63.CX), 1 path
--------------------------------------------------------------------------------
Slack (setup path):     2.871ns (requirement - (data path - clock path skew + uncertainty))
  Source:               c_state_13 (FF)
  Destination:          c_state_14 (FF)
  Requirement:          4.000ns
  Data Path Delay:      1.094ns (Levels of Logic = 0)
  Clock Path Skew:      0.000ns
  Source Clock:         clk_BUFGP rising at 0.000ns
  Destination Clock:    clk_BUFGP rising at 4.000ns
  Clock Uncertainty:    0.035ns

  Clock Uncertainty:          0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter (TSJ):  0.070ns
    Total Input Jitter (TIJ):   0.000ns
    Discrete Jitter (DJ):       0.000ns
    Phase Error (PE):           0.000ns

  Maximum Data Path: c_state_13 to c_state_14
    Location             Delay type         Delay(ns)  Physical Resource
                                                       Logical Resource(s)
    -------------------------------------------------  -------------------
    SLICE_X22Y63.BQ      Tcko                  0.375   c_state<15>
                                                       c_state_13
    SLICE_X22Y63.CX      net (fanout=12)       0.719   c_state<13>
    SLICE_X22Y63.CLK     Tdick                 0.000   c_state<15>
                                                       c_state_14
    -------------------------------------------------  ---------------------------
    Total                                      1.094ns (0.375ns logic, 0.719ns route)
                                                       (34.3% logic, 65.7% route)
--------------------------------------------------------------------------------
Paths for end point c_state_4 (SLICE_X18Y62.AX), 1 path
--------------------------------------------------------------------------------
Slack (setup path):     2.873ns (requirement - (data path - clock path skew + uncertainty))
  Source:               c_state_3 (FF)
  Destination:          c_state_4 (FF)
  Requirement:          4.000ns
  Data Path Delay:      1.105ns (Levels of Logic = 0)
  Clock Path Skew:      0.013ns (0.147 - 0.134)
  Source Clock:         clk_BUFGP rising at 0.000ns
  Destination Clock:    clk_BUFGP rising at 4.000ns
  Clock Uncertainty:    0.035ns

  Clock Uncertainty:          0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter (TSJ):  0.070ns
    Total Input Jitter (TIJ):   0.000ns
    Discrete Jitter (DJ):       0.000ns
    Phase Error (PE):           0.000ns

  Maximum Data Path: c_state_3 to c_state_4
    Location             Delay type         Delay(ns)  Physical Resource
                                                       Logical Resource(s)
    -------------------------------------------------  -------------------
    SLICE_X19Y63.DQ      Tcko                  0.375   c_state<3>
                                                       c_state_3
    SLICE_X18Y62.AX      net (fanout=5)        0.739   c_state<3>
    SLICE_X18Y62.CLK     Tdick                -0.009   c_state<7>
                                                       c_state_4
    -------------------------------------------------  ---------------------------
    Total                                      1.105ns (0.366ns logic, 0.739ns route)
                                                       (33.1% logic, 66.9% route)
```

```
--------------------------------------------------------------------------------
Paths for end point c_state_6 (SLICE_X18Y62.CX), 1 path
--------------------------------------------------------------------------------
Slack (setup path):     2.904ns (requirement - (data path - clock path skew + uncertainty))
  Source:               c_state_5 (FF)
  Destination:          c_state_6 (FF)
  Requirement:          4.000ns
  Data Path Delay:      1.061ns (Levels of Logic = 0)
  Clock Path Skew:      0.000ns
  Source Clock:         clk_BUFGP rising at 0.000ns
  Destination Clock:    clk_BUFGP rising at 4.000ns
  Clock Uncertainty:    0.035ns

  Clock Uncertainty:          0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter (TSJ):  0.070ns
    Total Input Jitter (TIJ):   0.000ns
    Discrete Jitter (DJ):       0.000ns
    Phase Error (PE):           0.000ns

  Maximum Data Path: c_state_5 to c_state_6
    Location             Delay type         Delay(ns)  Physical Resource
                                                       Logical Resource(s)
    -------------------------------------------------  -------------------
    SLICE_X18Y62.BQ      Tcko                  0.375   c_state<7>
                                                       c_state_5
    SLICE_X18Y62.CX      net (fanout=19)       0.686   c_state<5>
    SLICE_X18Y62.CLK     Tdick                 0.000   c_state<7>
                                                       c_state_6
    -------------------------------------------------  ---------------------------
    Total                                      1.061ns (0.375ns logic, 0.686ns route)
                                                       (35.3% logic, 64.7% route)

--------------------------------------------------------------------------------
Hold Paths: TS_clk = PERIOD TIMEGRP "clk" 4 ns HIGH 50%;
--------------------------------------------------------------------------------
Paths for end point c_state_0 (SLICE_X19Y63.AX), 1 path
--------------------------------------------------------------------------------
Slack (hold path):      0.294ns (requirement - (clock path skew + uncertainty - data path))
  Source:               c_state_17_1 (FF)
  Destination:          c_state_0 (FF)
  Requirement:          0.000ns
  Data Path Delay:      0.283ns (Levels of Logic = 0)
  Clock Path Skew:      -0.011ns (0.134 - 0.145)
  Source Clock:         clk_BUFGP rising at 4.000ns
  Destination Clock:    clk_BUFGP rising at 4.000ns
  Clock Uncertainty:    0.000ns

  Minimum Data Path: c_state_17_1 to c_state_0
    Location             Delay type         Delay(ns)  Physical Resource
                                                       Logical Resource(s)
    -------------------------------------------------  -------------------
    SLICE_X18Y63.BQ      Tcko                  0.345   c_state_17_1
                                                       c_state_17_1
    SLICE_X19Y63.AX      net (fanout=1)        0.128   c_state_17_1
    SLICE_X19Y63.CLK     Tckdi       (-Th)     0.190   c_state<3>
                                                       c_state_0
    -------------------------------------------------  ---------------------------
    Total                                      0.283ns (0.155ns logic, 0.128ns route)
                                                       (54.8% logic, 45.2% route)

--------------------------------------------------------------------------------
Paths for end point c_state_1_1 (SLICE_X18Y63.AX), 1 path
--------------------------------------------------------------------------------
Slack (hold path):      0.407ns (requirement - (clock path skew + uncertainty - data path))
  Source:               c_state_0 (FF)
  Destination:          c_state_1_1 (FF)
  Requirement:          0.000ns
  Data Path Delay:      0.441ns (Levels of Logic = 0)
  Clock Path Skew:      0.034ns (0.157 - 0.123)
  Source Clock:         clk_BUFGP rising at 4.000ns
  Destination Clock:    clk_BUFGP rising at 4.000ns
  Clock Uncertainty:    0.000ns

  Minimum Data Path: c_state_0 to c_state_1_1
    Location             Delay type         Delay(ns)  Physical Resource
                                                       Logical Resource(s)
    -------------------------------------------------  -------------------
    SLICE_X19Y63.AQ      Tcko                  0.345   c_state<3>
                                                       c_state_0
    SLICE_X18Y63.AX      net (fanout=4)        0.286   c_state<0>
    SLICE_X18Y63.CLK     Tckdi       (-Th)     0.190   c_state_17_1
                                                       c_state_1_1
    -------------------------------------------------  ---------------------------
    Total                                      0.441ns (0.155ns logic, 0.286ns route)
                                                       (35.1% logic, 64.9% route)

--------------------------------------------------------------------------------
Paths for end point c_state_12 (SLICE_X22Y63.AX), 1 path
--------------------------------------------------------------------------------
Slack (hold path):      0.410ns (requirement - (clock path skew + uncertainty - data path))
  Source:               c_state_11 (FF)
  Destination:          c_state_12 (FF)
  Requirement:          0.000ns
  Data Path Delay:      0.447ns (Levels of Logic = 0)
  Clock Path Skew:      0.037ns (0.490 - 0.453)
  Source Clock:         clk_BUFGP rising at 4.000ns
  Destination Clock:    clk_BUFGP rising at 4.000ns
  Clock Uncertainty:    0.000ns

  Minimum Data Path: c_state_11 to c_state_12
    Location             Delay type         Delay(ns)  Physical Resource
                                                       Logical Resource(s)
    -------------------------------------------------  -------------------
    SLICE_X19Y62.DQ      Tcko                  0.345   c_state<11>
                                                       c_state_11
    SLICE_X22Y63.AX      net (fanout=5)        0.292   c_state<11>
    SLICE_X22Y63.CLK     Tckdi       (-Th)     0.190   c_state<15>
                                                       c_state_12
    -------------------------------------------------  ---------------------------
    Total                                      0.447ns (0.155ns logic, 0.292ns route)
                                                       (34.7% logic, 65.3% route)

--------------------------------------------------------------------------------
```

```
Component Switching Limit Checks: TS_clk = PERIOD TIMEGRP "clk" 4 ns HIGH 50%;
--------------------------------------------------------------------------------
Slack: 2.502ns (period - min period limit)
  Period: 4.000ns
  Min period limit: 1.498ns (667.557MHz) (Tbgper_I)
  Physical resource: clk_BUFGP/BUFG/I0
  Logical resource: clk_BUFGP/BUFG/I0
  Location pin: BUFGCTRL_X0Y0.I0
  Clock network: clk_BUFGP/IBUFG
--------------------------------------------------------------------------------
Slack: 2.946ns (period - (min low pulse limit / (low pulse / period)))
  Period: 4.000ns
  Low pulse: 2.000ns
  Low pulse limit: 0.527ns (Trpw)
  Physical resource: c_state<7>/SR
  Logical resource: c_state_4/SR
  Location pin: SLICE_X18Y62.SR
  Clock network: res_IBUF
--------------------------------------------------------------------------------
Slack: 2.946ns (period - (min high pulse limit / (high pulse / period)))
  Period: 4.000ns
  High pulse: 2.000ns
  High pulse limit: 0.527ns (Trpw)
  Physical resource: c_state<7>/SR
  Logical resource: c_state_4/SR
  Location pin: SLICE_X18Y62.SR
  Clock network: res_IBUF
--------------------------------------------------------------------------------


All constraints were met.


Data Sheet report:
-----------------
All values displayed in nanoseconds (ns)

Clock to Setup on destination clock clk
---------------+---------+---------+---------+---------+
               | Src:Rise| Src:Fall| Src:Rise| Src:Fall|
Source Clock   |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
---------------+---------+---------+---------+---------+
clk            |    1.129|         |         |         |
---------------+---------+---------+---------+---------+


Timing summary:
---------------

Timing errors: 0  Score: 0  (Setup/Max: 0, Hold: 0)

Constraints cover 20 paths, 0 nets, and 27 connections

Design statistics:
   Minimum period:   1.498ns{1}   (Maximum frequency: 667.557MHz)


------------------------------------Footnotes-----------------------------------
1)  The minimum period statistic assumes all single cycle delays.

Analysis completed Mon Apr 27 13:06:50 2015
--------------------------------------------------------------------------------

Trace Settings:
-------------------------
Trace Settings

Peak Memory Usage: 403 MB
```

```
--------------------------------------------------------------------------
|                        Xilinx XPower Analyzer                          |
--------------------------------------------------------------------------
| Release                  | 14.7 - P.20131013 (nt64)                   |
| Command Line             | Generated from Graphical User Interface |
--------------------------------------------------------------------------

1.  Settings
1.1.  Project
-------------------------------------------------
|                  Project                      |
-------------------------------------------------
| Design File               | lut_mult_v06.ncd |
| Settings File             | NA               |
| Physical Constraints File | lut_mult_v06.pcf |
| Simulation Activity File  | NA               |
| Design Nets Matched       | NA               |
| Simulation Nets Matched   | NA               |
-------------------------------------------------
1.2.  Device
-------------------------------------------------
|                   Device                      |
-------------------------------------------------
| Family           | Virtex5                    |
| Part             | xc5vlx50t                  |
| Package          | ff1136                     |
| Temp Grade       | Commercial                 |
| Process          | Typical                    |
| Speed Grade      | -2                         |
| Characterization | PRODUCTION,v1.63,12-10-08 |
-------------------------------------------------
1.3.  Environment
-----------------------------------------
|               Environment               |
-----------------------------------------
| Ambient Temp (C)      | 50.0            |
| Use custom TJA?       | No              |
| Custom TJA (C/W)      | NA              |
| Airflow (LFM)         | 250             |
| Heat Sink             | Medium Profile  |
| Custom TSA (C/W)      | NA              |
| Board Selection       | Medium (10"x10") |
| # of Board Layers     | 8 to 11         |
| Custom TJB (C/W)      | NA              |
| Board Temperature (C) | NA              |
-----------------------------------------
```

1.4. Default Activity Rates

Default Activity Rates	
FF Toggle Rate (%)	12.5
I/O Toggle Rate (%)	12.5
Output Load (pF)	5.0
I/O Enable Rate (%)	100.0
BRAM Write Rate (%)	50.0
BRAM Enable Rate (%)	50.0
DSP Toggle Rate (%)	12.5

2. Summary

2.1. On-Chip Power Summary

On-Chip Power Summary				
On-Chip	Power (mW)	Used	Available	Utilization (%)
Clocks	6.58	1	---	---
Logic	69.00	6855	28800	24
Signals	100.24	3696	---	---
IOs	304.19	131	480	27
Static Power	564.79			
Total	1044.80			

2.2. Thermal Summary

Thermal Summary	
Effective TJA (C/W)	1.5
Max Ambient (C)	83.4
Junction Temp (C)	51.6

2.3. Power Supply Summary

Power Supply Summary			
	Total	Dynamic	Static Power
Supply Power (mW)	1044.80	480.01	564.79

Power Supply Currents				
Supply Source	Supply Voltage	Total Current (mA)	Dynamic Current (mA)	Quiescent Current (mA)
Vccint	1.000	585.29	180.50	404.79
Vccaux	2.500	68.20	6.20	62.00
Vcco25	2.500	115.60	113.60	2.00

3. Details

3.1. By Hierarchy

By Hierarchy	Power (mW)	Logic Power (mW)	Signal Power (mW)	# FFs	# LUTs
Hierarchy total	169.24	69.00	100.24	20	6747
lut_mult_v06	60.44 / 169.24	3.30 / 69.00	57.14 / 100.24	20	244 / 6747
lut_o	106.19	63.73	42.46	0	6408
toplayici_o	1.38 / 2.61	0.73 / 1.97	0.65	0	33 / 95
FA[17].obfadd_i	0.04	0.04	0.00	0	2
FA[49].obfadd_i	0.04	0.04	0.00	0	2
FA[47].obfadd_i	0.04	0.04	0.00	0	2
FA[23].obfadd_i	0.04	0.04	0.00	0	2
FA[21].obfadd_i	0.04	0.04	0.00	0	2
FA[45].obfadd_i	0.04	0.04	0.00	0	2
FA[51].obfadd_i	0.04	0.04	0.00	0	2
FA[41].obfadd_i	0.04	0.04	0.00	0	2
FA[29].obfadd_i	0.04	0.04	0.00	0	2
FA[11].obfadd_i	0.04	0.04	0.00	0	2
FA[55].obfadd_i	0.04	0.04	0.00	0	2
FA[31].obfadd_i	0.04	0.04	0.00	0	2
FA[39].obfadd_i	0.04	0.04	0.00	0	2
FA[37].obfadd_i	0.04	0.04	0.00	0	2
FA[53].obfadd_i	0.04	0.04	0.00	0	2
FA[19].obfadd_i	0.04	0.04	0.00	0	2
FA[15].obfadd_i	0.04	0.04	0.00	0	2
FA[25].obfadd_i	0.04	0.04	0.00	0	2
FA[9].obfadd_i	0.04	0.04	0.00	0	2
FA[33].obfadd_i	0.04	0.04	0.00	0	2
FA[43].obfadd_i	0.04	0.04	0.00	0	2
FA[13].obfadd_i	0.04	0.04	0.00	0	2
FA[27].obfadd_i	0.04	0.04	0.00	0	2
FA[8].obfadd_i	0.04	0.04	0.00	0	2
FA[35].obfadd_i	0.04	0.04	0.00	0	2
FA[57].obfadd_i	0.04	0.04	0.00	0	2
FA[59].obfadd_i	0.04	0.04	0.00	0	2
FA[61].obfadd_i	0.03	0.03	0.00	0	2
FA[2].obfadd_i	0.03	0.03	0.00	0	2
FA[6].obfadd_i	0.03	0.03	0.00	0	2
FA[4].obfadd_i	0.03	0.03	0.00	0	2

3.2. By Clock Domain

By Clock Domain : 1	Power (mW)	Freq (MHz)	Buffer	Buffer Enable (%)	Enable Signal	Fanout	Slice Fanout
clk_BUFGP/IBUFG							
Logic:							
clk_BUFGP/BUFG	3.87	250.00	BUFG	NA	NA	NA	NA
Nets:							
clk_BUFGP	2.62	250.00	BUFG	NA	NA	20	6
clk_BUFGP/IBUFG	0.09	250.00	NA	NA	NA	1	1
Total	6.58						

Analysis completed: Mon Apr 27 14:26:07 2015

KAYNAKLAR

[1] KEYES R.W., ***Physical Limits of Silicon Transistors and Circuits***, Institute of Physics Publishing,CN:68, SN:12, s2701-s2746, 2005

[2] PARHAMI B., ***Computer Arithmetic Algorithms and Hardware Designs***, Oxford University Press, New York USA, 2000

[3] ORS S. B., ***Design of Multiplier Blocks for DSP Applications Using VHDL***, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, 1999

[4] SMITH D. J., ***VHDL & Verilog Compared & Contrasted - Plus Modeled Example Written in VHDL, Verilog and C***, DAC '96 Proceedings of the 33rd annual Design Automation Conference, ISBN: 0-89791-779-0, s771-s776, 1996

[5] WOODS R., McALLISTER J., LIGHTBODY G., YI Y., ***FPGA-based Implementation of Signal Processing Systems***, John Wiley and Sons Publication, West Sussex U.K., 2008

[6] http://www.floodgap.com/retrobits/ckb/secret/cbm-rlorr-agdaph.jpg, Erişim Tarihi: 01.05.2015

[7] TAŞKIN D., BAYSAL K., TOPÇUBAŞI N., ***VHDL ile Mikroişlemci Tasarımı ve Eğitimde Uygulanabilirliği***, Akademik Bilişim, Malatya, s1-s6, 2011

[8] MAXFIELD C., ***The Design Warrior's Guide to FPGA***, Elsevier, Massachusets USA, 2004

[9] ***Introducing Innovations at 28 nm to Move Beyond Moore's Law***, Altera Corp., 2012

[10] WOODS R., MCALLISTER J., LIGHTBODY G., YI Y., ***FPGA-based Implementation of Signal Processing Systems***, Wiley, UK, 2008

[11] http://www.militaryaerospace.com/articles/print/volume-15/issue-9/productapplications/product-applications.html, Erişim Tarihi: 01.05.2015

[12] http://www.design-reuse.com/news/16992/ttp-controller-ip-altera-cost-cyclone-fpga-families-aerospace-applications.html, Erişim Tarihi: 01.05.2015

[13] http://www.militaryaerospace.com/articles/2005/01/fpga-processors-keep-mars-rovers-moving.html, Erişim Tarihi: 01.05.2015

[14] ***Cray X1 DataSheet***, Cray Inc., 2004

[15] DRAPER B. A., BEVERİDGE J. R., BÖHM A. P. W., ROSS C., CHAWATHE M., ***Accelerated Image Processing on FPGAs***, IEEE Transactions on Image Processing, CN:12, SN:12, s1543 - s1551, 2003

[16] HUFFMIRE T., IRVINE C., NGUYEN T. D., ***Handbook of FPGA Design Security,*** Springer, NewYork U.S.A., 2010

[17] FLOYD T. L., ***Digital Fundamentals***, Pearson Prentice Hall, New Jersey USA, 2006

[18] http://en.wikipedia.org/wiki/Simple_programmable_logic_device, Erişim Tarihi: 01.05.2015

[19] GROUT I., ***Digital Systems Design with FPGAs and CPLDs***, Elsevier, USA, 2008

[20] WAKERLY J. F., ***Digital Design Principles & Practices***, Prentice Hall, USA, 1999

[21] ZEIDMAN B., ***Designing With FPGA's and CPLD's***, CMP Books, Kansas U.S.A., 2002

[22] KUON I., TESSIER R., ROSE J., ***FPGA Architecture: Surveyand Challenges***, Foundations and Trends in Electronic Design Automation, CN:2, SN:2, s135 - s253, 2008

[23] HARDING B., COFER R.C., ***Rapid System Prototyping with FPGA's***, Elsevier, Massachusets USA, 2006

[24] http://sourcetech411.com/2013/04/top-fpga-companies-for-2013, Erişim Tarihi: 01.05.2015

[25] HASSAN H., ANIS M., ***Low-Power Design of Nanometer FPGAs***, Morgan Kaufmann Publisher, USA, 2010

[26] MAXFIELD C., ***FPGAs: Instant Access***, Newnes Massachusets USA, 2008

[27] MAXFIELD C., ***FPGAs World Class Designs***, Newnes Massachusets USA, 2009

[28] ***Cyclone FPGA Family Data Sheet***, Altera Co., 2013

[29] ***Virtex Family-5 Overview Product Specification***, Xilinix, February 2009

[30] ***IEEE Standard VHDL Language Reference Manual Std 1076***, The Institute of Electrical and Electronics Engineers, New York USA, 2000

[31] ÖZBEY R. S., SERTBAŞ A., ***Klasik Çarpma Algoritmalarının Donanımsal Simülasyonları ve Performans Değerlendirimi***, ELECO 2004 (Inter. Conf. on Electrical and Electronics Engineering), Bursa, 2004

[32] BOOTH A. D., ***A Signed Binary Multiplication Technique***, Q. J. Mech. Appl. Math. Oxford University Press, 1951

[33] http://en.wikipedia.org/wiki/Booth's_multiplication_algorithm, Erişim Tarihi: 01.05.2014

[34] WALLACE C. S., ***A Suggestion for a Fast Multiplier***, The Institute of Electrical and Electronics Engineers, 1964

[35] KUAMR M. R., RAO G. P., ***Design and Implementation of 32*32 Bit HighLevel Wallace Tree Multiplier***, International Journal of Innovative Research & Studies, CN:2, SN:8, s445 - s452, 2013

[36] MIKUCKI J., ***An IP Core Generator for Circuits Performing Arithmetic Multiplication***, Politechnika Slaska Wydzial Automatyki - Electroniki Informatyki, Yüksek Lisans Tezi, 2011

[37] CORMEN T. H., LEISERSON C. E., RIVEST R. L., STEIN C., ***Introduction to Algorithms (1st ed.)***, MIT Press and McGraw-Hill, UK, 1990

[38] Al-Khalili A.J., ***Digital Design and Synthesis Lecture Notes***, Concordia University, 2013

[39] KNEUSEL R. T., ***Numbers and Computers***, Springer, USA, 2015

[40] MISHRA S., PRADHAN M., ***Implementation Of Karatsuba Algorithm Using Polynomıal Multiplication***, Indian Journal of Computer Science and Engineering, CN:3, SN:1, s88 - s93, 2012

ÖZGEÇMİŞ

12 Temmuz 1984'te Hayrabolu'da doğdum. 2008 yılında Sakarya Üniversitesi Teknik Eğitim Fakültesi Elektronik ve Bilgisayar Öğretmenliği Bölümünden mezun oldum. 2011 yılı Kasım ayından beri Namık Kemal Üniversitesi Hayrabolu Meslek Yüksek Okulu'nda Öğretim Görevlisi olarak çalışmaktayım.

BİLİMSEL ÇALIŞMALAR

1. Uluslararası kongre, sempozyum, panel gibi bilimsel toplantılarda sunularak, programda yer alan tam metin olarak yayınlanan bildiriler

BAYSAL K. TAŞKIN D., OZCAN M. O., SERT E., *High Capacity Adder & Subtractor Circuit Design Using FPGA,* International Scientific Conference Unitech‟12, Gabrovo, 2012

2. Uluslararası kongre, sempozyum, panel gibi bilimsel toplantılarda sunularak, programda yer alan özet metin olarak yayınlanan bildiri veya posterler

BAYSAL K., TAŞKIN D., BULUT O., *Testing a VHDL Design Using Open Source VHDL Verification Methodology,* International Conference on Engineering and Natural Science ICENS 2015, Skopje, 2015

3. Ulusal hakemli dergilerde yayımlanan makaleler

TAŞKIN D., BAYSAL K., SERT E., TAŞKIN C., ÖZCAN M. O., *VHDL Kullanılarak FPGA ile Yüksek Kapasiteli Toplayıcı Ünite Tasarımı*, Journal of Engineering Sciences, Trakya Üniversitesi, Edirne, 2013

4. Ulusal kongre, sempozyum, panel gibi bilimsel toplantılarda sunularak, programda yer alan tam metin olarak yayınlanan bildiriler

BAYSAL K., TAŞKIN D., SERT E., TOPÇUBAŞI N., *VHDL Kullanarak FPGA ile Yüksek Kapasiteli Tam Çıkarıcı Devre Tasarımı*, Akademik Bilişim'13, Antalya, 2013

D. Taşkın, K. Baysal, N. Topçubaşı, *VHDL ile Mikroişlemci Tasarımı ve Eğitimde Uygulanabilirliği*, İnönü Üniversitesi Akademik Bilişim’11, Malatya, 2011