ON TABANINDA RAKAM ÇARPIMINA DAYALI PARALEL ÇARPMA ALGORİTMASININ DONANIM UYGULAMASININ İYİLEŞTİRİLMESİ

IMPROVEMENTS ON THE HARDWARE IMPLEMENTATION OF THE DIGIT MULTIPLICATION BASED PARALLEL DECIMAL MULTIPLICATION ALGORITHM

KENAN BOZDAŞ

Hacettepe Üniversitesi

Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin

ELEKTRİK ve ELEKTRONİK Mühendisliği Anabilim Dalı İçin Öngördüğü

DOKTORA TEZİ

olarak hazırlanmıştır.

2011

Fen Bilimleri Enstitüsü Müdürlüğü’ne,

Bu çalışma jürimiz tarafından ELEKTRİK ve ELEKTRONİK Mühendisliği ANABİLİM DALI’nda DOKTORA TEZİ olarak kabul edilmiştir.

Başkan :
Prof.Dr. H. Selçuk GEÇİM

Üye (Danışman) :
Doç.Dr. Ali Ziya ALKAR

Üye :
Doç.Dr. Atila YILMAZ

Üye :
Yrd.Doç.Dr. Derya ALTUNAY

Üye :
Yrd.Doç.Dr. Oğuz ERGİN

ONAY

Bu tez ..../..../2011 tarihinde Enstitü Yönetim Kurulunca kabul edilmiştir.

..../..../2011

Prof.Dr. Adil DENİZLİ
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRÜ

ON TABANINDA RAKAM ÇARPIMINA DAYALI PARALEL ÇARPMA ALGORİTMASININ DONANIM UYGULAMASININ İYİLEŞTİRİLMESİ

KENAN BOZDAŞ

ÖZ

İnsanoğlu gündelik yaşamında, kendi doğasına en uygun olan on tabanını kullanmaktadır. Ancak sayısal sistemlerde iki tabanı tercih edilmektedir. Bu iki taban arasındaki çevrim işlemleri, hem zaman kaybına hem de hatalara neden olmaktadır. Bu tür hatalar bazı kritik uygulamalarda ciddi sorunlara yol açmıştır. Bu sorunları önlemek için yazılım kütüphaneleri sıklıkla kullanılmaktadır. Ancak günümüzde finansal, ticari ve kullanıcı tabanlı uygulamalar için on tabanında aritmetik işlemlerin donanımla gerçekleştirilmesine duyulan ihtiyaç artmaktadır. Bu tez çalışmasında, on tabanına göre çarpıcılarla ilgili önceden yapılmış olan çalışmalar irdelenmiş ve rakam çarpımına dayalı bir yöntem kullanılarak özgün bir 16 rakamlı paralel çarpıcının donanımsal tasarımı gerçekleştirilmiştir. Yapılan çalışmada, rakam çarpımına dayalı paralel çarpma yönteminde kolon toplamlarının alabileceği en büyük değerin bir eniyileme problemi olarak ifade edilebileceği ortaya konmuştur. Bu problemin çözümü için deneysel bir yönteme dayanan Genetik Algoritma aracı ile benzetim yapılmıştır. Daha sonra, değişken sayısının az olduğu durumlar için kapsamlı yineleme ile problemin kesin çözümü hesaplanmıştır. Elde edilen kapsamlı yineleme sonuçlarının Genetik Algoritma benzetim sonuçlarını doğruladığı gözlenmiştir. Bulunan yeni sınır değerleri, donanım tasarımında kullanılan toplayıcı ve iki tabanından on tabanına çevirici birimlerinde bit sayısının azalmasını sağlamıştır. Bu sayede tasarımın bütünü için alan veya gecikme başarımlarında iyileştirme sağlandığı gözlenmiştir. Bulunan yeni sınır değerlerinin, meydana gelebilecek geçici hataların tespitinde kullanılması ile ilgili yapılan incelemeler ve benzetim sonuçları sunulmuştur.

Anahtar Kelimeler: On tabanında çarpma, Sınırlı kolon toplamı.

Danışman: Doç.Dr. Ali Ziya ALKAR, Hacettepe Üniversitesi, Elektrik ve Elektronik Mühendisliği Bölümü

IMPROVEMENTS ON THE HARDWARE IMPLEMENTATION OF THE DIGIT MULTIPLICATION BASED PARALLEL DECIMAL MULTIPLICATION ALGORITHM

KENAN BOZDAŞ

ABSTRACT

Human beings use base ten in daily life since it is well suited for our nature. However in digital systems, base two is preferred. Conversion operation between these bases is not only time consuming but may lead to rounding errors as well. These rounding errors cause serious problems in some critical applications. In order to prevent these kinds of problems software libraries are often used. Nowadays, however, the need for the hardware support for decimal arithmetic is growing for financial, commercial and user-based applications. In this study, previous works on decimal multipliers are investigated and a unique hardware implementation of a 16 digit parallel decimal multiplier based on digit multiplication is designed. It has been revealed that determining the maximum value of each column sum for a digit multiplication based multiplier is an optimization problem. Genetic Algorithm which is based on a heuristic approach is applied to find the solution to this problem. Subsequently, the exact solution of this problem is found by using the exhaustive iteration method for cases which the number of variables is small. Results of exhaustive iteration and Genetic Algorithm confirm each other. The proposed boundaries for the column sums decrease the number of bits for binary addition and binary to Binary Coded Decimal conversion, leading to improved area and delay performances of the multiplier. Results of analysis and simulations concerning using the proposed boundaries in detection of contingent soft errors are presented.

Keywords: Decimal multiplication, Column sum boundary.

Advisor: Assoc. Prof. Dr. Ali Ziya ALKAR, Hacettepe University, Department of Electrical and Electronics Engineering

TEŞEKKÜR

Bu tez çalışmasına katkılarından dolayı ve bana göstermiş olduğu sabır için, tez danışmanım Sayın Doç. Dr. Ali Ziya ALKAR'a teşekkürlerimi sunarım.

Tez çalışması boyunca fikir ve yönlendirmeleri ile desteklerini hissettiğim tez izleme komitesi üyeleri Sayın Doç. Dr. Atila YILMAZ'a ve Sayın Yrd. Doç. Dr. Derya ALTUNAY'a teşekkür ederim.

Katkılarından dolayı jüri üyeleri Sayın Prof. Dr. Selçuk GEÇİM'e ve Sayın Yrd. Doç. Dr. Oğuz ERGİN'e teşekkür ederim.

Birçok aşamada fikirlerini paylaşan Sayın Dr. Gürhan BULU'ya ve Sayın Sinan ALTIN'a teşekkürler.

Hayatım boyunca benden maddi ve manevi desteğini esirgemeyen ve bugünlere gelmemi sağlayan sevgili aileme teşekkür ederim.

Tezin anlatım dilinin ve yazım hatalarının düzeltilmesi için destek olmasının yanı sıra, desteğine ve sabrına ihtiyaç duyduğum her an yanımda olan sevgili eşim Münevver BOZDAŞ'a içtenlikle teşekkür ederim.

Süreç boyunca stresimi alan ve neşe kaynağım olan, biricik kızım Begüm'e...

İÇİNDEKİLER DİZİNİ

Sayfa

ÖZ i

ABSTRACT ii

TEŞEKKÜR iii

İÇİNDEKİLER DİZİNİ iv

ŞEKİLLER DİZİNİ vii

ÇİZELGELER DİZİNİ ix

SİMGELER VE KISALTMALAR DİZİNİ x

1. GİRİŞ 1

2. SAYI SİSTEMLERİ VE KURAMSAL ALTYAPI 6

2.1. Rakam ve Sayıların Tarihçesi 6

2.1.1. Tarihin İlk Rakamları 7

2.1.2. Sayıların Simgeleştirilmesi ve Taban İlkesinin Keşfi 7

2.1.3. Konumlu Sayılar 8

2.2. Modern Sayı Sistemleri 9

2.2.1. On Tabanına Göre Gösterim 10

2.2.2. İki Tabanına Göre Gösterim 11

2.2.3. On Altı Tabanına Göre Gösterim 11

2.3. Sayı Tipleri 12

2.3.1. Sabit Noktalı Sayılar 12

2.3.2. Kayan Noktalı Sayılar 13

2.4. Aritmetik İşlemler 14

2.4.1. İki Tabanında Aritmetik İşlemler 14

2.4.1.1. Toplama 14

2.4.1.2. Çıkarma 15

2.4.1.3. Çarpma 16

2.4.1.4. Bölme 17

2.4.2. BCD Aritmetik İşlemler 17

2.4.2.1. Toplama 19
2.4.2.2. Çıkarma 20
2.4.2.3. Çarpma 20
2.4.2.4. Bölme 21
2.5. Genetik Algoritma 22
2.5.1. Maliyet İşlevi ve Değişkenlerin Tanımlanması 23
2.5.2. İlk Nüfusun Oluşturulması 24
2.5.3. Her Birey İçin Maliyet Hesaplanması 25
2.5.4. Seçim 25
2.5.5. Eşleştirme (Çaprazlama) 25
2.5.6. Mutasyon 25
2.5.7. Bitiş Kontrol 26
2.6. Hata Tespit Sistemleri 26
3. ON TABANINDA ÇARPMA 29
3.1. Çarpma Yöntemleri 29
3.1.1. Ardışık Toplama ve Kaydırma Yöntemi 29
3.1.2. Paralel Çarpma Yöntemi 35
3.1.2.1. Lang ve Nannarelli Yöntemi 35
3.1.2.2. Vazquez Yöntemi 38
3.1.2.3. Diğer Yöntemler 40
3.2. Kısmi Çarpımların Hesaplanması 42
3.3. Kısmi Çarpımların İndirgenmesi 42
3.4. Elde Aktarımlı Son Toplam 44
4. RAKAM ÇARPIMINA DAYALI PARALEL ÇARPMA 45
4.1. Vedic Yönteminin 4 Rakam İçin Uyarlanması 45
4.2. Vedic Yönteminin 16 Rakam İçin Uyarlanması 48
4.2.1. On Tabanında Rakam Çarpımı 50
4.2.2. Kolon Toplamlarının İndirgenmesi 53
4.2.3. Kolon Toplamlarının Sınırlı Olmasının İncelenmesi 55

4.2.3.1. En Kalabalık Kolonların Toplamı 56
4.2.3.2. EKKT Değerlerinin 16 Rakamlı Sayıların Çarpımında Kullanılması 60
4.2.4. Elde Aktarımlı Toplayıcı ile Son Toplam 62
4.2.4.1. Aktarım ve Üretim Sinyalleri(pg) 62
4.2.4.2. Paralel Önek Elde Hesaplama 64
4.2.4.3. Elde Öncesi Koşulsuz Toplama 65
4.2.4.4. Eldelere Göre Seçim 65
4.3. RDPÇ Yönteminde Piramidin Üç Parçaya Ayrılması 66
4.4. RDPÇ Yönteminin FPGA İncelemesi 67
4.5. RDPÇ Yönteminin Ardışık Düzen İncelemesi 67
4.6. RDPÇ Yönteminde BCD4221 Gösteriminin Kullanılması 69
4.7. RDPÇ Yönteminde BCD4221 Gösteriminde Sayaç Kullanılması 70
5. UYGULAMALAR 72
5.1. Donanım Tasarımı 72
5.2. Hata Tespiti 74
6. SONUÇ VE DEĞERLENDİRMELER 78
KAYNAKLAR DİZİNİ 81
ÖZGEÇMİŞ 87

ŞEKİLLER DİZİNİ

Sayfa

Şekil 2.1. İki Tabanında Toplama 15
Şekil 2.2. İki Tabanında Çıkarma 15
Şekil 2.3. İki Tabanında Çarpma 16
Şekil 2.4. İki Tabanında Tekrarlanan Çıkarma Yöntemi ile Bölme 17
Şekil 2.5. İki Tabanında Kaydır ve Çıkar Yöntemi ile Bölme 18
Şekil 2.6. BCD Toplama 19
Şekil 2.7. BCD Çıkarma 20
Şekil 2.8. BCD Çarpma 21
Şekil 2.9. BCD Bölme 22
Şekil 2.10. Genetik Algoritma Genel Şeması 24
Şekil 3.1. On Tabanında Çarpma İşlemi 29
Şekil 3.2. Ardışık Çarpma Blok Şeması 30
Şekil 3.3. Ardışık Ekle Kaydır Yöntemi İle Çarpma (Erle and Schulte, 2003) . . 32
Şekil 3.4. Kısmi Çarpımların Hesaplanması ve İndirgenmesi (James et al., 2008b) 34
Şekil 3.5. Paralel Çarpma Blok Şeması 35
Şekil 3.6. Lang Yöntemi İle 4 Rakamlı İki Sayının Çarpılması (Lang and Nannarelli, 2006) 36
Şekil 3.7. Paralel Çarpma Yönteminde Ağaç Yapısı (Lang and Nannarelli, 2006) 37
Şekil 3.8. Paralel Çarpma İçin Vazquez Tarafından Önerilen Yapı (Vazquez et al., 2007) 38
Şekil 3.9. BCD4221 Gösteriminde EST 39
Şekil 3.10. BCD4221 Gösteriminde EST İçin Örnek Toplama 39
Şekil 3.11. EST Kullanılarak 15 Rakamın Toplanması 40
Şekil 4.1. Dört Rakam İçin Vedic Algoritması 47
Şekil 4.2. RDPÇ Blok Şeması 48
Şekil 4.3. n Rakam İçin Vedic Algoritması 49

Şekil 4.4. İki Tabanından On Tabanına Çevirici Devresi (Jaberipur and Kaivani, 2007) 50
Şekil 4.5. Düzeltilmiş İki Tabanından On Tabanına Çevirici Devresi 51
Şekil 4.6. Nicoud Yöntemi İle İki Tabanından On Tabanına Çevrim (Dadda, 2007) 54
Şekil 4.7. Sağ Taraftaki En Kalabalık Kolon Toplamı 57
Şekil 4.8. Sol Taraftaki En Kalabalık Kolon Toplamı 59
Şekil 4.9. Elde Aktarımlı Toplayıcı (EAT) 62
Şekil 4.10. Kogge-Stone Paralel Önek Elde Hesabı (Kogge and Stone, 1973) . . 64
Şekil 4.11. Örnek Bir Elde Aktarımlı Toplama 66
Şekil 4.12. Piramidin Üç Parçaya Ayrılması 66
Şekil 4.13. Ardışık Düzende İş Akışı 68
Şekil 4.14. DC "retime" Seçeneği ile Gecikme Dengeleme 68
Şekil 4.15. Kolonlar Arası Elde Aktarımı 69
Şekil 4.16. BCD4221 Gösterimi İçin 8 ve 9 Bitlik Sayaçlar 70
Şekil 4.17. BCD4221 Gösteriminde Sayaç Kullanılarak 17 Rakamın Toplanması 71

ÇİZELGELER DİZİNİ

Sayfa

Çizelge 1.1. Denektaşı Uygulamalarındaki Fonksiyonların Yüzdeleri 4

Çizelge 2.1. On Tabanından İki Tabanına Örnek Bir Çevrim İşlemi 12

Çizelge 2.2. Bazı Sayıların On, İki ve On Altı Tabanlarında Karşılıkları 12

Çizelge 2.3. BCD Rakamların Gösterimi 18

Çizelge 3.1. Lang Yönteminde Çarpan Sayının Rakamlarına Göre Kat Seçimi 36

Çizelge 3.2. Jaberipur Yönteminde Çarpan Sayının Rakamlarına Göre Kat Seçimi 41

Çizelge 4.1. Jaberipur Tarafından Önerilen Devrenin Hatalı Olduğu Durumlar . 51

Çizelge 4.2. $EKKTR_n$ ve Karşılık Gelen Sayı Çiftleri İçin Kapsamlı Yineleme Sonuçları 58

Çizelge 4.3. $EKKTS_n$ ve Karşılık Gelen Sayı Çiftleri İçin Kapsamlı Yineleme Sonuçları 59

Çizelge 4.4. Kolon Toplamlarının Maksimum Değerleri ve Gerekli Bit Sayıları . 63

Çizelge 5.1. On Altı Rakamlı BCD Çarpıcı Sentez Sonuçları 73

Çizelge 5.2. Literatürdeki Çalışmalarla Karşılaştırmalı Sentez Sonuçları 74

Çizelge 5.3. Telco Denektaşı İçin Tek Hata Tespiti 75

Çizelge 5.4. Telco İçin Birden Fazla Konumda Gerçekleşen Hatalı Bit İçin Tespit Sayısı 76

Çizelge 5.5. Dört Rakamlı Tüm Sayılar İçin Hata Tespiti 77

SİMGELER VE KISALTMALAR DİZİNİ

BCD : İkili Kodlanmış Onlu

EAT : Elde Aktarımlı Toplayıcı

EKKT : En Kalabalık Kolon Toplamının En Büyük Değeri

EKKTR : Sağ Taraftaki En Kalabalık Kolon Toplamının En Büyük Değeri

EKKTS : Sol Taraftaki En Kalabalık Kolon Toplamının En Büyük Değeri

EST : Elde Saklayan Toplayıcı

FPGA : Alan Programlanabilir Kapı Dizisi

GA : Genetik Algoritma

RDPÇ : Rakam Çarpımına Dayalı Paralel Çarpma

C_{ij} : Çapraz Rakam Çarpımı

C_{ij}^{O} : Çapraz Rakam Çarpımının Onlar Basamağı

C_{ij}^{B} : Çapraz Rakam Çarpımının Birler Basamağı

CP_i : Kısmi Çarpım

K_t : Kolon Toplamı

K_{tm} : Kolon Toplamının En Büyük Değeri

1. GİRİŞ

İnsanoğlu gündelik yaşamında, kendi doğasına en uygun olan on tabanını kullanmaktadır. Ancak sayısal sistemlerde iki tabanı tercih edilmektedir. Sayısal sistemlerde on tabanında sayılarla aritmetik işlem yapılabilmesi için, on tabanı ile iki tabanı arasında çevrim işlemine ihtiyaç duyulmaktadır. Bu çevrim işlemleri yuvarlama hatalarını barındırmaktadır. Örneğin; 0,2 sayısı iki tabanında $0,\overline{0011}$ şeklinde sonsuz sayıda bitle ifade edilmesi gerekirken, mevcut donanım sınırlamaları ile belirli bir basamağa kadar hassasiyetle ifade edilebilmektedir. Bu tür hatalar gerçek hayatta bazı ciddi sorunlara yol açmıştır. Örneğin; 1991 yılında Patriot füzesi, içindeki zamanlayıcı ve mesafe hesaplamasında meydana gelen yuvarlama hatası nedeniyle SCUD füzesini durdurmayı başaramamıştır[1]. Bu zamanlama hatası, SCUD füzesinin yerinin tespit edilememesine ve hatta sonucunda 28 askerin ölümüne sebep olmuştur.

Tsang tarafından yapılan çalışmada, çeşitli ticari veritabanlarında incelenen bir milyon sütundaki verilerin %55'inin onluk sayılar içerdiği ve ek olarak %43'lük kısmının da on tabanında ifade edilebileceği ortaya konmuştur (Erle, 2008). Bu veriler ışığında ticari uygulamalarda, verilerin onluk sayılarla ifade edilmesi, on tabanında aritmetik işlemleri destekleyen sistemlerin kullanılması ve saklanması önemli bir ihtiyaçtır. Sayısal sistemlerde aritmetik işlemlerin iki tabanında yapılması ciddi bir avantaj getirdiği için on tabanındaki aritmetik işlemlerde meydana gelen hesap hataları yakın zamana kadar ya fazla önemsenmiyordu ya da yazılım kütüphaneleri kullanılarak hatalar en aza indirilmeye çalışılıyordu. IBM tarafından sunulan "decNumber[2]" ve Intel tarafından sunulan "Decimal Floating-Point Math[3]" kütüphaneleri bunlara iki örnektir.

Günümüzde finansal, ticari ve kullanıcı tabanlı uygulamalarda on tabanına göre aritmetik işlemleri gerçekleştiren donanımlara duyulan ihtiyaç artmaktadır (Cowlishaw, 2003). Bu tür uygulamalarda, her sayının iki tabanına göre tam olarak ifade edilememesi sebebiyle veya kullanıcı arayüzlerinde kullanım kolaylığı sağlamak amacıyla on tabanı tercih edilmektedir. Örneğin; internet bankacılığı ile döviz veya hisse senedi alım-satımı esnasında kullanıcı tarafından girilen miktarlar veya sistemde ta-

[1] http://archive.gao.gov/t2pbat6/145960.pdf

[2] http://www.alphaworks.ibm.com/tech/decnumber

[3] http://software.intel.com/en-us/articles/intel-decimal-floating-point-math-library

nımlı birim fiyatlar, ondalık sayılardan oluşabilmektedir. Bu sayılar ile gerçekleştirilen aritmetik işlemler, ya kullanılan makinenin hassasiyetine (precision) bağlı olarak yuvarlama işlemiyle ya da yazılım kütüphaneleri yardımıyla gerçekleştirilmektedir. Bu seçeneklerin kesinlik (accuracy) açısından yetersiz veya yavaş kaldığı durumlar için on tabanında işlem yapan donanımların kullanılması önem arz etmektedir. Çünkü aritmetik işlem donanımları, hem kesinlik hem de hız yönünden yazılıma göre daha iyi bir çözüm sunmaktadır.

Mevcut sistemlerin büyük bir bölümünde, on tabanına göre karşılıkları ile depolanmış sayıların işlenebilmesi için iki tabanına göre karşılıklarına çevrilmesi gerekmektedir. Bu çevrim işlemi hem zaman kaybına hem de hatalara neden olmaktadır. Ortaya çıkan bu sorunların çözümü için, aritmetik işlemleri on tabanına göre gerçekleştiren donanımlar önerilmiş ve uygulanmıştır (Erle and Schulte, 2003), (Erle *et al.*, 2005), (Ueda, 1995), (Busaba *et al.*, 2001), (Lang and Nannarelli, 2006), (Vazquez *et al.*, 2010), (Jaberipur and Kaivani, 2009). On tabanında kayan noktalı sayıların kullanılması ihtiyacının artması ile birlikte, on tabanında aritmetik işlemlerin ve veri tiplerinin standardizasyonu önem kazanmıştır. Bu amaçla IEEE 754-2008 (*IEEE 754-2008 - IEEE Standard for Floating-Point Arithmetic*, 2008) standardı ortaya çıkmıştır. Son yıllarda, özellikle IBM firması tarafından üretilen bazı mikroişlemcilerde on tabanında işlem yapan aritmetik birimler mevcuttur (Busaba *et al.*, 2001), (Eisen *et al.*, 2007), (Schwarz *et al.*, 2009). Ayrıca fikri mülkiyet hakkı Silminds[4] firmasına ait olan çeşitli ondalık işlem birimleri, IP (Intellectual Property) çekirdeği olarak markete sunulmuştur.

Yonga üretiminin ucuzlaması ve donanımın yazılım kütüphanelerine göre çok daha hızlı işlem gerçekleştirmesi etkenleri, on tabanında aritmetik işlem donanımlarına olan ilgiyi arttırmıştır (Erle *et al.*, 2002). Aritmetik işlemleri on tabanında gerçekleştiren donanımlar iki tabanında işlem yürüten donanımlara göre daha fazla alan kaplamakta ve göreceli olarak daha yavaş çalışmaktadır. Bu nedenle, bu sorunları gidermeye yönelik olarak farklı yaklaşımlar önerilmiştir.

Farklı uygulamalarda, farklı aritmetik işlemler gerçekleştirildiğinden, hangi uygulamada ne tür işlemlerin yapıldığının belirlenmesi, yapılan çalışmalara ışık tutacaktır. Bu amaçla Wang (Wang *et al.*, 2007) ve Anderson (Anderson *et al.*, 2009) tarafın-

[4] http://www.silminds.com/decimal-products/ip-core

dan yapılan denektaşı (benchmark) çalışmaları, önemli bilgiler ortaya çıkarmıştır. Yapılan çalışmalarda, on tabanındaki aritmetik işlemlerin en çok kullanıldığı uygulamalardan bazıları seçilmiş ve bu uygulamalarda ihtiyaç duyulan işlemlerin hangi oranlarda kullanıldığı tespit edilmiştir. On tabanında kayan noktalı sayıları içeren SPECjbb2005[5], SPECjAppServer2004[6], TPC-H[7] ve telco[8] gibi denektaşlarının yeterli olmadığını düşünen yazarlar, 4 yeni denektaşı tanımlamışlardır. Tanımladıkları bu denektaşları; bankacılık sistemi, Avro dönüştürücü, risk yönetimi ve vergi hazırlama uygulamalarıdır. Bu uygulamalar, on tabanında aritmetik işlemler için hedef ticari uygulamaları barındırmaktadır. Denektaşı uygulamaları sanal bir işlemci üzerinde koşturularak, uygulamaların işlem profilleri çıkarılmıştır. Bu uygulamalardan elde edilen sonuçların bir kısmı Çizelge 1.1'de verilmiştir.

Çizelge 1.1'de, IEEE standardında (*IEEE 754-2008 - IEEE Standard for Floating-Point Arithmetic, 2008)* belirtilen ondalık fonksiyonların, denektaşı uygulamalarında toplam çalışma süresine göre yüzdesel oranları verilmiştir. Buna göre, uygulamalardan dördünde (Bankacılık, Avro, Risk ve Telco) ondalık fonksiyonlar, çalışma süresinin büyük bir bölümünü (%75,4,% 93,1,% 85,1 ve %78,9) işgal etmektedir. Çarpma işlemi bu dört uygulamanın hepsinde, en çok zaman harcanan ilk üç fonksiyon içinde yer almaktadır.

Bu tez çalışmasında, on tabanına göre çarpıcılarla ilgili önceden yapılmış olan çalışmalar incelenmiş ve 16 rakamlı çarpıcı tasarımı konusunda yoğunlaşılmıştır. Literatürdeki çalışmalarda kısmi çarpımların hesaplanması aşamasında çarpılan sayının belirli katlarının hesaplanması ve daha sonra bu katlardan kısmi çarpımların elde edilmesi yaygın olarak kabul görmüştür. Çarpma işlemi, ardışık ekle-kaydır yöntemi veya paralel yöntemlerle gerçekleştirilebilmektedir. On tabanına göre çarpma yöntemlerinde temel üç ara basamak tanımlanmıştır (Jaberipur and Kaivani, 2009). Bu ara basamaklar: kısmi çarpımların hesaplanması, kısmi çarpımların indirgenmesi ve elde aktarımı ile son toplamdır. On tabanında yapılan işlemlerde İkili Kodlanmış Onlu (Binary Coded Decimal, BCD) gösterimi kullanılmaktadır. Bu gösterimin kullanılması ile aritmetik işlemlerde BCD düzeltme işlemi gereksinimi ortaya çıkmaktadır. Bu gereksinim, üç ara basamağın her birinde fazladan gecikme olmasına sebep

[5]http://www.spec.org/jbb2005/

[6]http://www.spec.org/jAppServer2004/

[7]http://www.tpc.org/tpch/

[8]http://speleotrove.com/decimal/telco.html

Çizelge 1.1. Denektaşı Uygulamalarındaki Fonksiyonların Yüzdeleri

Fonksiyon	Bankacılık	Avro	Risk	Vergi	Telco
Toplama (Add)	10,1	4,5	11,1	7,5	19,0
Karşılaştırma (Compare)	2,8	3,6	-	1,5	-
Kopyalama (Copy)	1,0	0,9	0,1	1,7	-
decimal64FromNumber	-	-	-	2,1	-
decimal64ToNumber	1,3	2,7	0,6	2,8	5,1
Bölme (Divide)	29,9	50,7	3,9	1,1	-
En Büyük (Max)	0,3	-	-	-	-
En Küçük (Min)	-	-	-	0,3	-
Negatif (Minus)	0,0	-	-	0,0	-
Çarpma (Multiply)	13,4	12,5	23,1	1,5	27,5
Yeniden Ölçeklendirme (Rescale)	-	-	18,7	-	24,1
Yuvarlama Kipi (SetRoundMode)	0,7	0,9	1,3	0,0	2,3
Karekök (SquareRoot)	-	-	0,4	-	-
Çıkarma (Subtract)	7,0	6,3	25,6	1,3	-
Tamsayıya Yuvarlama (RoundToInt)	8,5	10,8	-	0,4	-
Sıfırlama (Zero)	0,3	0,3	0,2	13,6	0,9
Toplam Ondalık İşlem (%)	75,4	93,1	85,1	33,9	78,9

olmaktadır. Gecikme başarımının arttırılması, on tabanında aritmetik işlem donanımlarının motivasyon kaynağı olmuştur. Tez kapsamında nihai hedef, 16 rakamlı iki BCD sayının çarpımını yapan paralel çarpıcının donanım tasarımıdır. Bu amaçla Rakam Çarpımına Dayalı Paralel Çarpma (RDPÇ) yöntemi irdelenmiş ve donanım tasarımı gerçekleştirilmiştir. Yapılan çalışmada, RDPÇ yöntemine özgü bir özellik ortaya çıkarılmıştır. Bu özelliğe göre, RDPÇ yöntemi ile yapılan çarpma işleminin ara basamaklarında hesaplanan kolon toplamlarının alabileceği en büyük değer, bir eniyileme problemi olarak ifade edilebilmektedir. Bu eniyileme probleminin çözümü için Genetik Algoritma kullanılarak sınır değerlerine ulaşılmıştır. Ayrıca, değişken sayısının az olduğu bazı durumlar için problemin kesin çözümü kapsamlı yineleme ile elde edilmiştir. Kapsamlı yineleme ve Genetik Algoritma benzetim sonuçlarının birebir uyumlu olduğu gözlenmiştir. Bulunan yeni sınır değerlerinin donanım tasarımında kullanılmasıyla alan ve gecikme başarımı açısından iyileştirme olduğu ortaya

konmuştur. Önerilen sınır değerlerinin, hata tespitinde kullanılabilmesini irdelemek amacıyla benzetim çalışması yapılmış ve elde edilen sonuçlar sunulmuştur.

Bu tez raporu altı bölümden oluşmaktadır. Bölüm 2'de, tez kapsamında irdelenen konulara ait temel bilgiler verilmiştir. Günümüzde kullanmakta olduğumuz rakamların ve sayı sistemlerinin kısa bir tarihçesi anlatılmıştır. Ayrıca modern sayı sistemleri ve bu sistemlerde kullanılan farklı tabanlara göre gösterimlerden bahsedilmiştir. Daha sonra iki ve on tabanında aritmetik işlemler irdelenmiştir. Tez kapsamındaki eniyileme probleminde kullanılan genetik algoritma için genel bilgiler verilmiştir. Son olarak, tez çalışmasının olası uygulama alanlarından biri olan hata tespit sistemleri anlatılmıştır. Bölüm 3'te, on tabanında çarpma algoritmaları ile ilgili literatürde yer alan çalışmalar özetlenmiştir. On tabanında çarpma, ardışık ve paralel çarpma olarak iki farklı yaklaşımla gerçekleştirilebilir. Bu iki yöntemle yapılan çalışmalar ve bu çalışmalarda elde edilen iyileştirmeler yorumlanmıştır. Bölüm 4'te, bu tez kapsamında irdelenen Rakam Çarpımına Dayalı Paralel Çarpma (RDPÇ) yöntemi detaylı bir biçimde incelenmiştir. RDPÇ yönteminin ara basamaklarında ihtiyaç duyulan işlemler tanımlanmış ve bu işlemler için ihtiyaç duyulan birimler irdelenmiştir. RDPÇ yönteminde her bir kolon toplamının en büyük değeri için yapılan çalışmalar ve elde edilen sonuçlar bu bölümde sunulmuştur. Bölüm 5'te, RDPÇ yönteminin donanım uygulamasından elde edilen sonuçlar ve hata tespit sistemlerinde kullanılmasıyla ilgili yapılan çalışmanın sonuçları verilmiştir. Bölüm 6'da, tez kapsamında yapılan çalışmalar ve elde edilen sonuçlar özetlenmiş ve genel değerlendirmeler sunulmuştur. Ayrıca, bu konuda gelecekte yapılması muhtemel çalışmalar için önerilerde bulunulmuştur.

2. SAYI SİSTEMLERİ VE KURAMSAL ALTYAPI

Bu bölümde, tez kapsamında irdelenen konulara ait temel bilgiler ve kuramsal altyapı verilmiştir. Günümüzde kullanmakta olduğumuz sayı sistemlerinin kısa bir tarihçesi anlatılmıştır. Daha sonra iki ve on tabanında aritmetik işlemler irdelenmiştir. Tezde kullanılan genetik algoritma için temel bilgiler verilmiştir. Son olarak, hata tespit sistemleri anlatılmıştır.

2.1 Rakam ve Sayıların Tarihçesi

İnsanoğlunun sayı olgusunu oluşturması, günlerin, hayvanların, tarihlerin, askerlerin, esirlerin, ölülerin veya kayıpların sayısını bilme kaygısıyla ortaya çıkmıştır. Ancak bu ortaya çıkış, sanıldığının aksine ardı ardına gelişmiş bir olgu değildir. Farklı kültürlerde, farklı ihtiyaçlar için, farklı sayılama sistemleri gelişmiştir. Bunların kimi eş zamanlı iken, kimileri daha önceden var olan sayılama sistemlerinden bağımsız sistemlerdir (Ifrah, 1995*a*). Ancak ticaret veya benzeri sebeplerle ortaya çıkan etkileşimler yadsınamaz bir gerçektir.

İnsanoğlunun sayılama kaygısı, her otlak seferinden dönen hayvan sürülerinde, hepsinin sağ salim dönüp dönmediğinden emin olmaktan, alet, yiyecek veya silah stoklarının yeterli olup olmamasına, mal değiş tokuşu yapabilmek için değer biçebilmekten, zamanı ölçmeyi becerebilmeye kadar çeşitlilik göstermiştir. Bunun için insanlık, elindeki birçok aracı kullanmıştır. Araçlar başlangıçta somut, deneysel ve gelişigüzel olmuş, giderek soyutlaşmış, yetkinleşmiş ve hatta bazen gizemli, mitolojik yapılara bürünmüştür. Sonrasında ise genelleştirme biçimi almaya yatkın ve varoluşu sorgulanmayan bir şekilde hayatın vazgeçilmez parçası haline gelmiştir.

Günümüzdeki haliyle sayı saymasını bilmeyen eski çağ toplumlarında sayı adları olarak bir, iki ve çoktan başka bir şey olmamasına rağmen bire bir uygunluğu belirten kemik ve ağaç kertme yolları kullanılmıştır (Ifrah, 1995*a*). Örneğin; ağıla giren her bir hayvan için bir çentik atılarak olması gerekenle bire bir uygun olup olmadığı kontrol edilmiştir. Saymak için kimi toplumlar çakılları veya çomakları üst üste yığma ya da yan yana dizme yolunu kullanmışlardır. Kimileri ise, el ve ayak parmaklarına, kol ve bacak eklemlerine, gözlere, buruna, ağız veya kulaklara yönelerek sayıları betimlemişlerdir (Ifrah, 1995*a*).

2.1.1 Tarihin İlk Rakamları

Tarihin ilk yazılı sayılamasında, sıradan çakılların yerine belirli anlamlar taşıyan farklı biçimlerde, pişmemiş topraktan yapılmış nesneleri kullanılmıştır. Her bir nesnenin boyutu ve şekli onu bir sayılama dizgesinin basamaklarından birinin karşılığı haline getirmiştir. Örneğin; M.Ö. 4000 yıllarında, Arap-İran körfezinin yakınlarında bulunan ve bir İran toprağı olan Elam'da birler basamağı için bir çubuk, onlar basamağı için bir bilye ve yüzler basamağı için bir küre kullanılmıştı. Aynı çağda, Aşağı Mezopotamya'daki Sümer ülkesinin sakinleri benzer bir dizgeyi aynı şekilde kullanmışlardır. Ama bu topluluk onluk yerine altmışlık sayma yöntemini benimsemiş ve yöntemi birkaç farklılıkla uygulamıştır: 1 için küçük bir koni, 10 için bir bilye, 60 için büyük bir koni, 600 için delikli büyük bir koni ve 3600 için bir küre (Ifrah, 1995*a*). Sümerlilerin kullanmış olduğu 60 tabanı, günümüzde saniye, dakika, saat kavramlarının ve ayrıca geometride açıların temelini oluşturmuştur.

Sözlü olarak yayılan bu yöntemin insan belleği ile sınırlı olması sorununu aşmak için bu hesap dizgesinin nesnelerinin kilden yapılmış topların içine koyulması fikri ortaya çıkmıştır. Bu fikir sayesinde, sadece aritmetik işlemler yapma gereksinimi değil, her çeşit mal sayımı ve ticari işlemlerin belgesinin arşivlerde saklanması gereği de kolayca karşılanıyordu. Denetim için, kilden yapılmış topu kırmak yeterliydi. Daha sonraları topun içine konulan nesnelerin simgeleştirilmesi akla geldi ve küçük bir koni küçük bir kertikle, bir bilye küçük bir yuvarlak delikle, büyük bir koni kalın bir kertikle, bir küre bir daireyle betimlendi. Böylece tarihin en eski rakamları olan Sümer rakamları, M.Ö. 3200'e doğru ortaya çıkmış oldu (Ifrah, 1995*a*).

2.1.2 Sayıların Simgeleştirilmesi ve Taban İlkesinin Keşfi

İnsanoğlu, sayıları soyutlamayı başarınca, eski sayısal aletleri (çakıl, çubuk, boncuk kolye, vb) bir kenara bırakarak gerçek sayısal simgeleri kullanmaya başlamıştır. O zamana dek sayılar çoğu kez, çevreleyen doğa ve dünyayla ilişki içinde, algısal terimler aracılığıyla yazılıyordu (bir için güneş veya ay, iki için gözler veya kuşun kanatları, üç için yonca, dört için hayvanın ayakları, vb). Sonra el ve ayak parmakları yardımı ile iki farklı biçimde gösterimler kullanılmıştır. İlk gösterimde, birimi temsil eden bir simge belirlenip, istenilen sayıya ulaşmak için bu sayının içerdiği birim kadar, birimin simgesi yinelenmiştir. Örneğin; 4 sayısı için 4 parmak veya 4 çentik kullanılmıştır. İkinci gösterimde ise, her sayıya özgün bir simge belirlenip, birbiriyle

hiç ilişkisi olmayan simgelerin ardıllığına bakılmaktadır. Örneğin; 1 sayısı için baş parmak, 2 sayısı için işaret parmağı veya 3 için orta parmak kullanılmıştır.

Bu iki gösterimde de büyük sayıların ifadesinde yaşanan zorluklar, "sayıların en az sayıda olanaklı simgeyle nasıl gösterilebileceği" sorusunu beraberinde getirmiştir. Çözüm, belli bir öbeklemeye (onluk, on ikilik, yirmilik ya da altmışlık) ayrıcalık tanımak ve sayıların kurallı dizisini bu temele dayanan sıra düzenli bir sınıflamaya göre düzenlemek olmuştur. Başka bir deyişle, sayılar belirlenen öbeklemeye göre basamaklı bir merdiven şeklinde ifade edilmiş ve bu merdivenin sahanlıklarına birinci basamağın birimleri, ikinci basamağın birimleri, vb gibi adlar verilmiştir (Ifrah, 1995*a*).

2.1.3 Konumlu Sayılar

Geçmişte kullanılan sayı gösterimleri temelde toplama esasına dayanmaktaydı. Örneğin; Mısır hiyeroglif sayılaması, 1'e dikey çizgi, 10'a ters "U", 100'e bir sarmal ve 1000'e bir nilüfer çiçeği betimliyordu. Buna göre 1253 sayısı, 1 nilüfer çiçeği, 2 sarmal, 5 ters "U" ve 3 dikey çizgi ile ifade ediliyordu. Sümerler ise 60 tabanını kullandıklarından, aynı sayıyı ($600x2 + 5x10 + 3x1$) şeklinde çözümleyip 600 için kullanılan simgeden 2 tane, 10 için kullanılan simgeden 5 tane ve 1 için kullanılan simgeden 3 tane kullanıyorlardı.

Romalılar sayıları belirtmek için 7 ayrı harfi kullanmışlardır. Sayılar ilk önceleri sadece toplama işlemi ile elde ediliyordu (XXX=30, DLXIII=500+50+10+1+1+1=563). Ancak daha sonra çıkarma işleminden de yararlanarak daha kısa yazmanın yollarını ortaya koymuşlardır (XC=100-10=90, IX=10-1=9). Roma sayılamasında, diğer sayılamalardan farklı olarak basamak sistemi yoktur. Bu nedenle Roma sayma düzeni aritmetik işlemlere uygun değildir.

Büyük sayıların gösterimi ciddi bir sorun teşkil ettiğinden, bu sorunun çözümü için hem çarpma hem toplama ilkesini kullanarak geliştirilen karmaşık bir ilke ortaya çıkmıştır. Asur-Babil ve Arami dizgelerinde bu ilkeden yararlanılmıştır. Buna göre 600 sayısını ifade etmek için 100 simgesini 6 kere yazmak yerine, bu simgenin yanına 6'nın simgesini koyarak (6*100) işlemini gerçekleştirmişlerdir (Ifrah, 1995*b*).

Bugün sayılamada kullandığımız rakamların ve hesap işlemlerinin kökeni Hindistan'dır. Bu rakamlar, yüzyıllar süren uzun bir göç yolunu izleyerek Avrupa'ya gelmiş

ve geliştirilerek modern Avrupa rakamları oluşturulmuştur. Hint rakamlarının ve sayı sisteminin tüm dünyada tutunmasının ve kullanılmasının tek nedeni, rakamların bulundukları yere göre basamak değerini alması ve dört işlemdeki hesap yapma kolaylığıdır. Anakıta'da Hint matematiği tutunurken, yeni kıta olan Amerika'da Mayaların matematiği daha eskilerden beri vardı. Mayaların rakamları da bulundukları yere göre basamak değerini alıyor ve yirmi tabanına göre yazılıyordu. Oysa Hint matematiğinde sayılar, on tabanına göre yazılarak okunuyordu (Dönmez, 2002). Bugün kullandığımız sayılamada, "3" bir sayısal betimlemede birinci, ikinci ya da üçüncü konumda bulunmasına göre, 3 birim, 3 on ya da 3 yüz değerini taşır. Ancak insanoğlu bu sayılama sistemine erişinceye kadar birçok başarısız veya pek kullanışlı olmayan deneme ve sonuçlarla karşı karşıya kalmıştır. Bugün kullandığımız konumlu sayılamada her bir basamakta yer alan rakamın ifade ettiği değer, kullanılan tabana ve sayının hangi basamakta yer aldığına bağlıdır. Her bir basamağın değeri, sola gidildikçe tabanın üssel katları olarak belirlenmektedir. Bu ilkenin tamsayılar için matematiksel ifadesi Eş. 2.1'de verilmiştir.

$$A = \sum_{i=1}^{n} a_i r^{i-1} \tag{2.1}$$

Burada a_i, sayının rakamlarını ve r kullanılan tabanı ifade etmektedir.

2.2 Modern Sayı Sistemleri

Sayı, birçokluğu belirtmek için kullanılan soyut birimdir. Günümüzde genellikle konumlu sayı sistemi kullanılmaktadır. Buna göre her basamakta yer alan rakam, bulunduğu basamağa ve kullanılan tabana göre bir değer ifade etmektedir. Örneğin; on tabanı için "4" rakamı birler basamağında "dört", onlar basamağında "kırk", yüzler basamağında "dört yüz" değerini ifade etmektedir. Gündelik hayatta insan doğasına en yakın olan on tabanı sıklıkla kullanılmaktadır. Ancak ortaya çıkan ihtiyaçlara göre farklı tabanlar kullanılmaktadır. Taban n olarak tanımlandığında, bu tabanda kullanılabilecek tanımlı rakamlar $\{0, 1, 2, ..., n-1\}$ kümesindedir. En yaygın olarak kullanılan tabanlar $2, 3, 4, 5, 8, 10, 12, 16, 20$ ve 60 tabanlarıdır. Gündelik hesaplamalarda doğal sayma sistemi olarak da adlandırılan on tabanı kullanılmaktadır. İki tabanı ise sayısal elektroniğin gelişimi ile birlikte bu alanda çok sık kullanılmıştır. Her ne kadar bilgisayar çağının ilk yıllarında ENIAC (Goldstine and Goldstine, 1996), UNIVAC (Eckert *et al.*, 1951) ve IBM650 (Trimble, 1986) gibi bazı bilgisayarlarda on tabanı

kullanılmış olsa da, zamanla iki tabanı daha baskın hale gelmiştir. Bunun nedeni, sayısal sistemlerin yapısı gereği tanımlanan "var" ve "yok" veya "açık" ve "kapalı" ifadelerinin iki tabanında gösteriminin bu sistemler için on tabanından çok daha elverişli olmasıdır. İki tabanında kullanılabilecek iki rakam vardır: ′0′ ve ′1′. Ancak genel kullanımda iki tabanına göre gösterimlerde kullanılan karakterler (0 ve 1), bit olarak tanımlanmaktadır. On altı tabanına göre gösterim ile iki tabanına göre gösterim arasındaki ilişki, on altı tabanına göre gösterimin popüler hale gelmesinde büyük rol oynamaktadır.

Eş. 2.1'de verilen ifade tamsayılar için geçerlidir. Ancak aritmetik işlemler sadece tamsayılarla yapılmamaktadır. Aynı zamanda ondalık biçimde ifade edilen sayılar da kullanılmaktadır (Örneğin; 1,657, -0,344, vb.). Genel olarak ondalık bir sayı, ($a_n a_{n-1} \ldots a_1, c_1 c_2 c_3 \ldots$) ile ifade edildiğinde ve kullanılan taban r olduğunda Eş. 2.2'de verilen biçimde ifade edilebilir.

$$(a_n a_{n-1} \ldots a_1, c_1 c_2 c_3 \ldots)_r = \sum_{i=1}^{n} a_i r^{i-1} + \sum_{i=1}^{\infty} c_i r^{-i} \tag{2.2}$$

Burada a_i tamsayı kısımdaki rakamları, c_i ondalık kısımdaki rakamları ve r kullanılan tabanı ifade etmektedir.

2.2.1 On Tabanına Göre Gösterim

Günümüzde sıklıkla kullanmakta olduğumuz on tabanının kökleri Hint ve Arap uygarlıklarına dayanmaktadır. On tabanında kullanılan rakamların Araplardan Avrupa'ya geçtiği düşünülmektedir. Ancak Araplar kullandıkları rakamları Hint rakamları diye adlandırmaktadırlar. Bu sebeple bu rakamlara Hint-Arap rakamları da denmektedir. On tabanında gösterim, insanın on parmağı olmasından hareketle insan doğasına en yatkın olan gösterimdir. Bu sebeple bilgisayar ve sayısal elektronik çağı hariç tutulduğunda en fazla kullanılan tabandır. On tabanına göre gösterimin matematiksel ifadesi Eş. 2.3'te verilmiştir.

$$A = \sum_{i=1}^{n} a_i 10^{i-1} \tag{2.3}$$

Burada a_i, sayının $\{0, 1, \ldots 9\}$ kümesinde yer alan rakamlarını ifade etmektedir.

2.2.2 İki Tabanına Göre Gösterim

İki tabanında gösterimi ilk olarak M.Ö. 200 yıllarında Hintli bir yazarın şiir ve şarkılardaki heceleri uzun ve kısa olarak ayırırken kullandığı tespit edilmiştir[1]. Günümüzde kullandığımız ikili sayma sistemi ise 1703 yılında Leibniz tarafından yapılan çalışmada sunulmuştur[2]. 1853 yılında İngiliz matematikçi Boole (Boole, 1853) önemli bir çalışmaya imza atarak, günümüzde Boole Cebiri diye adlandırılan iki tabanındaki cebirsel ifadeleri tanımlamıştır. Daha sonra Shannon, yaptığı yüksek lisans tezinde (Shannon, 1940) ilk kez Boole Cebirini elektrik röle ve anahtarlarında kullanarak sayısal sistemlerde ikilik sistemin kullanılmasının temelini atmıştır. Sayısal elektronik sistemlerinin hızlı gelişimi ile birlikte iki tabanında gösterim popüler hale gelmiştir. Sayısal sistemlerde temel olarak tanımlanan iki durum, "var" ve "yok" veya "açık" ve "kapalı" olarak tanımlanıp ′0′ ve ′1′ ile ifade edilmiştir. Bu noktadan hareketle kullanılmaya başlanan ikilik sistem, sayıların gösteriminde de kullanılmaya başlanmıştır. Buna göre her bir basamakta yer alan karakter 0 veya 1 ile ifade edilmiş ve her bir basamağın ağırlığı 2'nin üsleri olarak tanımlanmıştır. İki tabanına göre gösterimin matematiksel ifadesi Eş. 2.4'te verilmiştir.

$$A = \sum_{i=1}^{n} a_i 2^{i-1} \tag{2.4}$$

Burada a_i, sayının 0 veya 1 değerini alabilen bitlerini ifade etmektedir.

On tabanına göre ifade edilen bir sayının iki tabanındaki karşılığı, bu sayıyı ardışık olarak ikiye bölerek hesaplanır. Her aşamada kalan, iki tabanına göre gösterimin bitlerini ifade etmektedir. Son aşamada elde edilen bölüm, son biti ifade etmektedir. 38 sayısının iki tabanına çevrimi Çizelge 2.1'de örnek olarak verilmiştir.

Buna göre aşağıdaki eşitlik yazılabilir:
$(38)_{10} = (100110)_2$

2.2.3 On Altı Tabanına Göre Gösterim

On altı tabanında $\{0, 1, ... 9\}$ kümesine ek olarak $\{A, B, ... F\}$ kümesinde yer alan harfler de kullanılmaktadır. On altı tabanına göre gösterimin önemi, iki tabanına göre

[1] http://www.sju.edu/ rhall/Rhythms/Poets/arcadia.pdf
[2] http://www.leibniz-translations.com/binary.htm

Çizelge 2.1. On Tabanından İki Tabanına Örnek Bir Çevrim İşlemi

Sayı	Bölüm	Kalan
38	19	**0**
19	9	**1**
9	4	**1**
4	2	**0**
2	**1**	**0**

gösterimle arasındaki ilişkiden kaynaklanmaktadır. Bu ilişkiye göre iki tabanındaki sayının en sağdan hizalanarak ayrılan 4 bitlik öbeklerin her biri on altı tabanındaki her basamaktaki rakama karşılık gelmektedir. Bu sayede sayısal sistemlerde sıklıkla kullanılan iki tabanındaki bir gösterimin, herhangi bir aritmetik işleme gerek kalmaksızın, daha az karakterle ve kullanıcı tarafından daha kolay anlaşılabilir bir yapıda ifade edilmesi sağlanmıştır. Farklı dört sayının on, iki ve on altı tabanlarındaki karşılıkları Çizelge 2.2'de örnek olarak verilmiştir. On tabanından iki tabanında çevrimde aritmetik işlem gerekirken, iki tabanında ifade edilmiş olan bir sayının on altı tabanındaki karşılığını bulmak için bitleri 4'lü öbeklemek yeterlidir.

Çizelge 2.2. Bazı Sayıların On, İki ve On Altı Tabanlarında Karşılıkları

On Tabanı	İki Tabanı	On Altı Tabanı
11	1011	B
38	10 0110	26
109	110 1101	6D
1356	101 0100 1100	54C

2.3 Sayı Tipleri

2.3.1 Sabit Noktalı Sayılar

Sabit noktalı sayılar, ondalık sayıların belli bir sınırlama ile ifade ediliş biçimidir. Bu sınırlama, bilgisayar sistemlerinde ondalık sayıları bellekte tutmak için akla gelen ilk yöntemdir. Bu yöntemde sayının virgülden önceki ve sonraki kısımları için sabit uzunlukta yer ayrılır. Bu sınırlama, virgülden sonra kaç basamağın anlamlı olduğunu

ve ifade edilebilecek en büyük sayıyı belirler. Örneğin; virgülden sonra sadece iki basamağın anlamlı olduğu varsayıldığında 2, 87 ile 2, 8796 aynı değeri ifade edecektir. Bu sınırlama aritmetik işlem yapılan cihazdan kaynaklanan bir sınırlama olabileceği gibi çok hassas bir aritmetik işlem gerekmemesinden de kaynaklanabilir.

2.3.2 Kayan Noktalı Sayılar

Bilgisayar sistemlerinde sabit noktalı gösterim çok kullanışlı değildir. Çünkü bellekte fazla yer kaplar. Ama sabit noktalı sayılar bir ölçekleme katsayısı ile çarpılarak üssel biçimde de ifade edilebilir. Örneğin; 2, 87 sayısı, 0,287'nin 10 ile çarpılmış hali olarak da ifade edilebilir. Bilgisayar sistemlerinde bu özellikten yararlanılmaktadır; tamsayı kısmı sıfır olacak şekilde normalleştirilir (normalization) ve gerekli ölçekleme katsayısı için üs bulunur. Örneğin; 2,87 sayısı için kesir 0,287 ve üs 1 olarak tanımlanır. Sabit noktalı sayıların getirmiş olduğu kısıtlamaları en aza indiren, saklanabilecek en büyük değeri arttıran ve sayıların gerçek değerlerine en yakın biçimde gösterimini sağlayan bu sisteme "Kayan Noktalı Sayılar" sistemi denir.

Kayan noktalı gösterimde virgülün yerine önceden karar verilir ve bu bilgi bellekte tutulmaz. Örneğin; 2,87 sayısı için normalleştirme yapıldıktan sonra bellekte 287 ve 1 değerleri ilgili alanda saklanır. Virgülün, sıfırdan farklı en anlamlı rakamın hemen solunda olduğu önceden belirlenmiştir. İki tabanında ise virgülün, her zaman sıfırdan farklı en anlamlı rakamın sağında olduğu kabul edilir. Örneğin; iki tabanında 10101,110 sayısı $1{,}0101110 \times 2^4$ biçiminde normalize edilir. Normalize edilmiş bir ondalık sayı Eş. 2.5'te gösterilen şekilde ifade ile edilir.

$$A = \pm K x T^{\pm E} \tag{2.5}$$

Burada K kesiri, E üssü ve T tabanı ifade etmektedir. Bellekte sadece işaret bitleri, K ve E saklanır.

Bu gösterim, çok büyük ve çok küçük değerlerin ifade edilmesi için avantaj sağlamaktadır. Örneğin; sabit noktalı gösterimde n bitlik alanda noktanın yeri belirlenmiş ve anlamlı kısım için x bit, virgülden sonraki kısım içinse geriye kalan (n-x) bit ayrılmıştır. Bu durumda, 8 bitin 4 biti sayının tam kısmını, kalan 4 biti virgülden sonraki kısmını ifade ederse "1011, 1001", "1000, 0001" ve "1111, 1111" şeklindeki sayılar gösterilebilir. Kayan noktalı gösterimde ise daha geniş aralıktaki sayılar üretilebi-

lir. Yani 8 bitlik alanda "1, 0001101", "101, 10011" ve "111001, 01" şeklinde virgülün herhangi bir basamağa gelmesiyle oluşacak tüm sayılar ifade edilebilir.

IEEE754-2008 (*IEEE 754-2008 - IEEE Standard for Floating-Point Arithmetic, 2008)* standardında iki tabanında kayan noktalı sayılar için iki farklı biçim belirtilmiştir. Bunlar, tek ve çift duyarlı (precision) biçimlerdir. Tek duyarlı kayan noktalı sayı 32 bitle ifade edilir. Bu bitlerden ilki işaret bitidir. Sonraki 8 bit üs kısmını içerir ve sonraki 23 bitlik kısım mantisi (anlamlı kısım) tanımlar. Çift duyarlı kayan noktalı sayı biçiminde ise üs 11 bitle ve mantis 52 bitle ifade edilir. Normalleştirilmiş bir sayı Eş. 2.6'da gösterilen şekilde ifade edilir.

$$A = Sx2^{E}x1,M \tag{2.6}$$

Burada S işaret bitini gösterir ve pozitif sayılar için +1, negatif sayılar için -1 değerini alır. Eş. 2.6'da kullanılan E üs kısmını ifade eder ve gerçek üsten 127 çıkarılmış haliyle hafızada saklanır. Bu şekilde elde edilen üs, saptırılmış (biased) üs olarak tanımlanır. İşlem gerçekleştirilirken E'ye 127 eklenerek gerçek üs bulunur. Saptırılmış üs kullanılmasının faydası hafızada üssün işaret bitinin saklanması zorunluluğunu ortadan kaldırmasıdır. Eş. 2.6'da kullanılan M, mantissayı ifade eder. Sayılar normalleştirildiği için "1,M" sayısı [1,2) aralığında bir değer alabilir.

Kayan noktalı sayılarla aritmetik işlemler, sabit noktalı sayılara göre daha zordur, gerçekleştirilmeleri daha uzun sürer ve daha karmaşık donanım gerektirir. Bununla beraber sabit noktalı gösterimin içerdiği ölçeklendirme problemleri yüzünden bilimsel hesaplamalar için kayan noktalı gösterimin kullanılması bir zorunluluktur (Mano, 2003).

2.4 Aritmetik İşlemler

2.4.1 İki Tabanında Aritmetik İşlemler

2.4.1.1 Toplama

İki tabanında toplama işlemi, kağıt üzerinde on tabanında yapılan toplama işlemiyle benzerdir. Ancak burada rakamlar yerine bitler toplanır ve elde oluşursa bir sonraki basamağa aktarılır. İki bitin toplanması durumunda aşağıdaki durumlar ortaya çıkabilir:

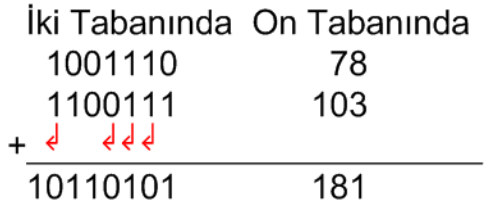
0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 elde=1

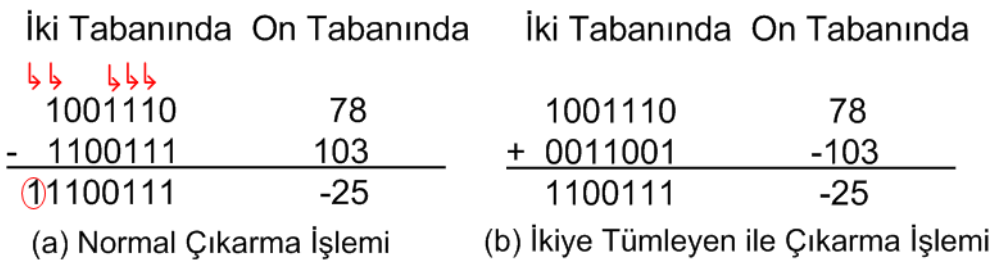
Şekil 2.1'de iki sayının iki tabanında toplanması, on tabanındaki karşılığı ile birlikte örnek olarak verilmiştir. Şekilde gösterilen oklar, hangi basamaklarda elde oluştuğunu belirtmektedir.



Şekil 2.1. İki Tabanında Toplama

2.4.1.2 Çıkarma

İki tabanında çıkarma işlemi, on tabanında yapılan çıkarma işlemi mantığıyla aynıdır. Eğer çıkarılan hane çıkan haneden daha küçükse, yüksek anlamlı haneden ödünç 2 alınır. Ancak 2'ye tümleyen yöntemiyle çıkarma işlemini yapmak çok daha verimli bir yöntemdir. Buna göre önce çıkarılacak sayının tüm bitleri 0'dan 1'e ve 1'den 0'a çevrilir ve daha sonra 1 eklenerek 2'ye tümleyeni bulunur. Buna göre çıkan sayının 2'ye tümleyeni, çıkarılan sayıya eklenerek sonuç elde edilir. Şekil 2.2'de iki yöntem için birer örnek verilmiştir.



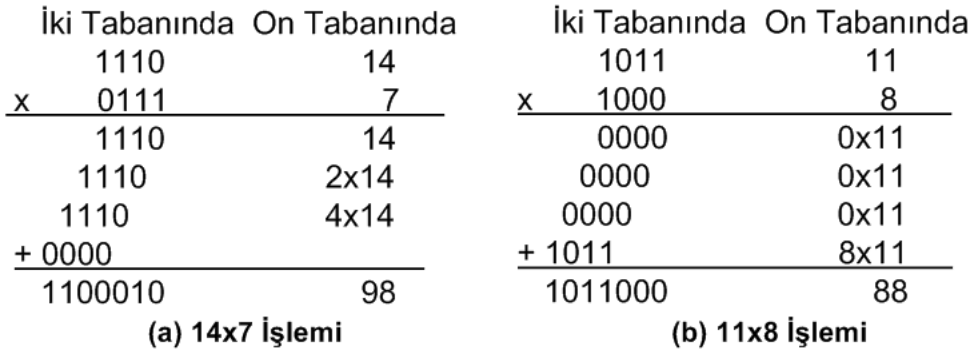
Şekil 2.2. İki Tabanında Çıkarma

Şekil 2.2a'da gösterilen oklar, hangi basamaklardan ödünç alındığını belirtmektedir. Ayrıca en soldaki konumdan (8'inci bit) ödünç alınmış olması, sonucun negatif olduğunu göstermektedir. Sonuçta elde edilen 1100111 değerinin ikiye tümleyeni alındığında 0011001 değerine ulaşılır ve bu değer on tabanında 25'e karşılık gelmektedir.

Şekil 2.2b'de çıkan sayının (1100111) ikiye tümleyeni alındığında 0011001 değerine ulaşılır. Bu değer çıkarılan sayı ile toplandığında yine 1100111 değerine ulaşılır. Burada yapılan toplama işleminde taşma (overflow) olmaması, sonucun negatif olduğunu belirtir. Bu sebeple sonucun ikiye tümleyeninin alınması gerekir ve bu değer on tabanında 25'e denk gelmektedir. Taşma olması durumunda ise sonucun pozitif olduğu anlaşılır ve oluşan taşma eldesi yok sayılır.

2.4.1.3 Çarpma

İki tabanında tanımlı olan rakamların sadece 0 ve 1 olması, çarpma işleminde kolaylık sağlamaktadır. Şekil 2.3a'da ve Şekil 2.3b'de farklı sayıların iki tabanında çarpımları, on tabanındaki karşılıkları ile birlikte örnek olarak verilmiştir. Çarpan sayının ilgili biti 1 olduğunda çarpılan sayının kendisi yazılmaktadır. İlgili bit 0 olduğunda ise sıfır yazılmasına gerek yoktur. Bir sonraki konuma geçilerek kaydırma yapılması yeterlidir; ancak anlaşılır olmasını sağlamak amacıyla Şekil 2.3'te çarpımın sıfır olduğu durumlar da belirtilmiştir.



Şekil 2.3. İki Tabanında Çarpma

2.4.1.4 Bölme

İki tabanında bölme işlemi iki farklı yolla yapılabilir: Tekrarlanan Çıkarma veya Kaydır ve Çıkar. Tekrarlanan Çıkarma yönteminde, bölen sayının bölünen sayıda kaç kere tekrar ettiğinin bulunması mantığı yürütülür. Buna göre, bölünen sayıdan bölen sayı çıkarılır ve bölüm bir arttırılır. Kalan, bölen sayıdan büyük veya bölen sayıya eşit olduğu müddetçe bu işleme devam edilir. Şekil 2.4'te bu yöntemle gerçekleştirilen basit bir bölme işlemi örnek olarak verilmiştir.

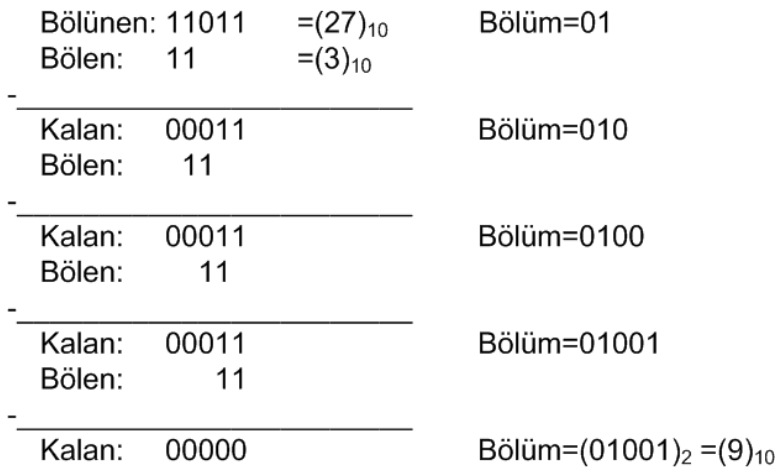
Bölünen:	$(1100)_2 = (12)_{10}$	Bölüm=$(01)_2 = (1)_{10}$
Bölen:	$(11)_2 = (3)_{10}$	
-		
Kalan:	$(1001)_2 = (9)_{10}$	Bölüm=$(10)_2 = (2)_{10}$
Bölen:	$(11)_2 = (3)_{10}$	
-		
Kalan:	$(0110)_2 = (6)_{10}$	Bölüm=$(11)_2 = (3)_{10}$
Bölen:	$(11)_2 = (3)_{10}$	
-		
Kalan:	$(0011)_2 = (3)_{10}$	Bölüm=$(100)_2 = (4)_{10}$
Bölen:	$(11)_2 = (3)_{10}$	
-		
Kalan:	$(0000)_2 = (0)_{10}$	Bölüm=$(100)_2 = (4)_{10}$

Şekil 2.4. İki Tabanında Tekrarlanan Çıkarma Yöntemi ile Bölme

Kaydır ve Çıkar yönteminde ise, bölen sayının bit sayısına eşit olacak şekilde, bölünen sayının en soldaki bitleri öbeklenir. Eğer bu öbek, bölen sayıdan büyük ise, bölüm kısmına 1 yazılır, öbekten bölen çıkarılır ve çıkarma sonucunun sağ tarafına bölünen sayının sıradaki biti iliştirilir. Eğer öbek, bölen sayıdan küçük ise bölüm kısmına 0 yazılır ve bu öbeğin sağ tarafına bölünen sayının sıradaki biti iliştirilir. Bu işlem, bölünen sayının tüm bitleri tamamlanıncaya kadar devam eder. Şekil 2.5'te Kaydır ve Çıkar yöntemiyle gerçekleştirilen bir bölme işlemi örnek olarak verilmiştir.

2.4.2 BCD Aritmetik İşlemler

İki tabanını esas alan sayısal sistemlerde, on tabanında aritmetik işlem yapabilmek için iki farklı yol izlenebilir. İlk yöntemde, on tabanındaki sayılar iki tabanına çevrilir ve gerekli aritmetik işlem yapıldıktan sonra tekrar on tabanına çevrilmesi gerekir.



Şekil 2.5. İki Tabanında Kaydır ve Çıkar Yöntemi ile Bölme

Bu yöntem, çok yoğun işlem gerektiren durumlarda verimsizdir. İkinci yöntemde on tabanındaki sayıların iki tabanında kodlanması gerekmektedir. Bu kodlama için en yaygın olarak benimsenen model İkili Kodlanmış Onlu (Binary Coded Decimal, BCD) kodlamasıdır. BCD olarak kodlanan sayılar ile on tabanında aritmetik işlemlerin yapılması mümkün olmaktadır. Bu kısımda BCD aritmetik işlemler hakkında genel bilgi verilecektir.

Her bir BCD rakam, iki tabanında 4 bitle ifade edilmektedir. Çizelge 2.3'te BCD rakamların karşılıkları verilmiştir. On tabanındaki bir sayının BCD olarak ifade edilişinde sayının, her bir rakamı 4 bitle ifade edilerek yan yana yazılır. Örneğin; 438 sayısı BCD olarak ifade edildiğinde, 0100 0011 1000 olarak elde edilir. BCD aritmetik işlemler yapılırken, sayıların BCD kodlandığından kullanıcı haberdardır; ama işlemi gerçekleştiren birim bundan habersiz olabilir. Başka bir deyişle, 010000111000 sayısının on tabanındaki karşılığı 1080'dir. Ancak kullanıcı bu sayının 438 sayısını ifade ettiğini kabullenerek işlem yapar. Gerekli durumlarda, aritmetik işlemin sonucunda düzeltme ihtiyacı doğabilir.

Çizelge 2.3. BCD Rakamların Gösterimi

BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
Rakam	0	1	2	3	4	5	6	7	8	9

2.4.2.1 Toplama

İki BCD rakam toplandığında toplam, en fazla 18 olabilir. Ancak toplama işlemi iki tabanında yapıldığından ve iki tabanında *{A, B,...,F}* harfleri geçerli birer rakamı ifade ettiğinden, toplama sonucunda düzeltme yapılması gereken durumlar ortaya çıkabilir. Örneğin; 1000 ve 0100 rakamları iki tabanında toplandığında, sonuç 1100 olacaktır. Bu sonuç geçerli bir BCD rakam değildir. Ancak bu rakama 0110 eklendiğinde, toplam sonucu 0010 ve elde 1 olacaktır ve dolayısıyla ulaşılması gereken 12 sayısının BCD gösterimi elde edilmiş olur. Genel kural olarak, iki BCD rakam toplamında, toplam sonucu 9 (1001)'dan büyük olursa veya toplam sonucunda elde oluşursa, sonuca 6 (0110) eklenerek doğru sonuç elde edilir. Şekil 2.6'da farklı rakam ve sayılar için BCD toplama örnekleri verilmiştir. Şekil 2.6a'da iki farklı BCD rakamın toplanması durumunda oluşabilecek 3 durum ve bu durumlarda izlenmesi gereken adımlar gösterilmiştir. Şekil 2.6b'de iki farklı BCD sayının toplanması gösterilmiştir. Şekilde de görüleceği gibi basamaklar arasında elde aktarımı gereken durumlarda birden fazla düzeltme işlemine ihtiyaç duyulmaktadır. Şekil 2.6b'de altı çizili rakamlar 9'dan büyük olduğu için düzeltme gerekmektedir. Düzeltme sonucunda oluşan elde, bir sonraki basamağa aktarılmıştır.

		0010	(2)	1000	(8)
		+ 1000	(8)	+ 1001	(9)
0001	(1)	1010	>9	10001	elde
+ 0101	(5)	+ 0110	(6)	+ 0110	(6)
0110	(6)	10000	(10)	10111	(17)

(a) İki BCD Rakam Toplamı

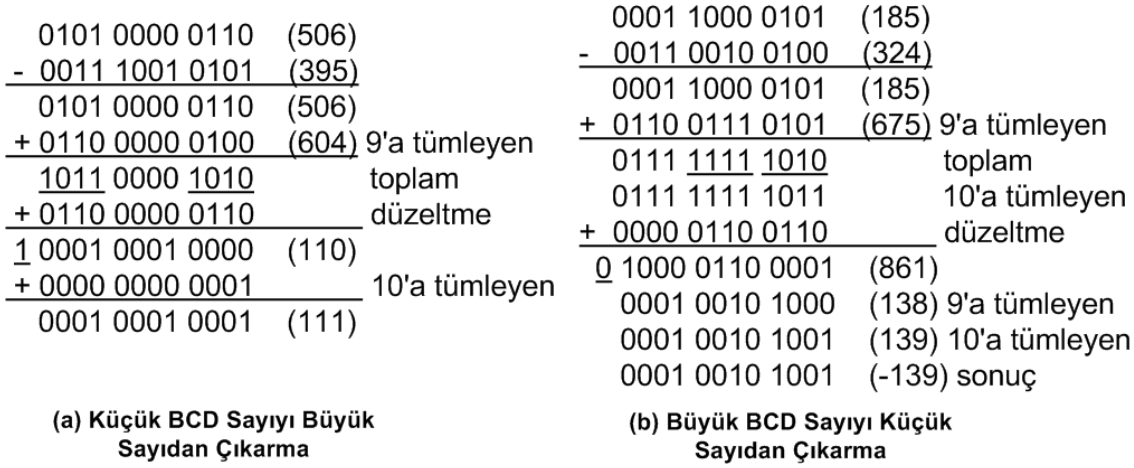
0011 1001 0101 (395)	
+ 0101 0000 0110 (506)	
1000 1001 1011	toplam
+ 0000 0000 0110	düzeltme
1000 1010 0001	toplam
+ 0000 0110 0000	düzeltme
1001 0000 0001 (901)	

(b) İki BCD Sayı Toplamı

Şekil 2.6. BCD Toplama

2.4.2.2 Çıkarma

BCD çıkarma için, çıkan sayının 10'a tümleyeninin, çıkarılan sayıya eklenmesi gerekmektedir. Ancak bunun daha kolay yolu, 9'a tümleyeninin hesaplanması ve toplama sonucuna 1 eklenmesidir. Toplama sonucunda elde oluşursa, elde ihmal edilir ve sonucun pozitif olduğu anlaşılır. Elde oluşmaması durumunda, sonucun negatif olduğu anlaşılır ve doğru sonuca ulaşmak için tekrar 10'a tümleyenin alınması gerekir. Çıkarılan sayı ile çıkan sayının 9'a tümleyeninin toplanması işleminde, Bölüm 2.4.2.1'de belirtilen kurallar geçerlidir. Şekil 2.7'de farklı BCD sayılarla gerçekleştirilen çıkarma işlemleri örnek olarak verilmiştir. Şekil 2.7a'da çıkan sayı çıkarılan sayıdan küçük olduğu için son basamakta oluşan elde ihmal edilmektedir ve sonucun pozitif olduğu anlaşılmaktadır. Şekil 2.7b'de ise çıkan sayı, çıkarılan sayıdan daha büyük olduğu için son basamakta elde oluşmamıştır. Bu nedenle sonucun negatif olduğu anlaşılmaktadır ve 10'a tümleyeni hesaplanmaktadır.

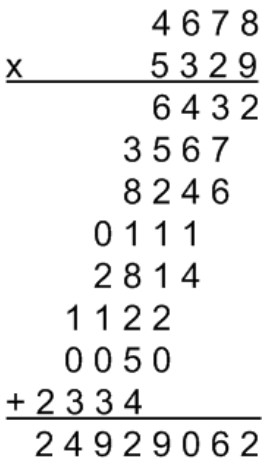


Şekil 2.7. BCD Çıkarma

2.4.2.3 Çarpma

BCD çarpma işlemi iki tabanında yapılan çarpma işleminden daha karmaşık ve zordur. İki tabanında rakamlar, 0 veya 1 değerini alabildiği için kısmi çarpımlar hesaplanırken çarpılan sayının kendisi veya sıfır alınıyordu. BCD çarpma işleminde ise çarpılan sayının {0,1,2,...,9} katlarının hesaplanması gerekmektedir. BCD çarpma yöntemleri ile ilgili detaylı inceleme Bölüm 3'te verilmiştir. Bu kısımda, çarpan sa-

yının rakamlarının, çarpılan sayının rakamları ile sırayla çarpılması yöntemi örnek olarak verilmiştir. Bu yöntemde, rakam çarpımlarının birler ve onlar basamakları alt alta ilgili basamağa yazılarak toplanmıştır. Şekil 2.8'de iki BCD sayının çarpımı örnek olarak verilmiştir. Gösterimin kolay olması amacıyla Şekil 2.8'de çarpma işlemi için on tabanındaki rakamlar kullanılmıştır; ancak bu rakamlar 4 bitlik BCD olarak kodlanmıştır ve bu rakamların toplanması işleminde, Bölüm 2.4.2.1'de belirtilen kurallar geçerlidir.

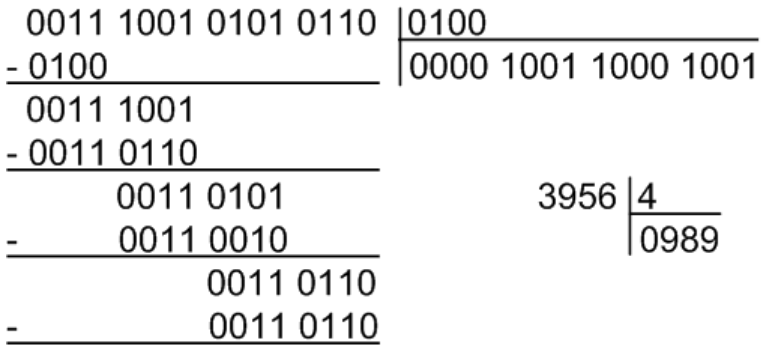


Şekil 2.8. BCD Çarpma

2.4.2.4 Bölme

BCD bölme işlemi için iki tabanında kullanılan yönteme benzer bir yöntem izlenebilir. BCD için farklılık, sayıların 4'er bitlik öbeklere ayrılarak bölme işleminin gerçekleştirilmesiyle ortaya çıkar. Şekil 2.9'da iki BCD sayı için bölme işlemi örnek olarak verilmiştir.

Bölünecek sayılar basamak sayısı olarak birbirine yakınsa veya bu sayılar ondalık sayılardan oluşuyorsa Şekil 2.9'da gösterilen yöntem yetersiz kalmaktadır. Örneğin; 345/658 veya 3.456/0.015 gibi işlemler için Newton-Raphson yineleme yöntemini kullanan (Wang and Schulte, 2004) veya Goldschmidt bölme yöntemini kullanan (Schulte *et al.*, 2007) yöntemler mevcuttur. Bu yöntemlerde temel mantık, paydanın çarpmaya göre tersinin bir yaklaşım ile hesaplanması ve pay ile çarpılmasıdır. Bu işlem, hata sıfır veya kabul edilebilir bir değer oluncaya kadar yinelenir.



Şekil 2.9. BCD Bölme

2.5 Genetik Algoritma

Bilgisayar çağının ilk yıllarından itibaren bilim adamları, füze yörüngeleri ve askeri kodların çözümlemeleri gibi problemlerle uğraşmanın yanı sıra doğadaki olayları da modellemeye çalışmışlardır (beyin sinyalleri, insan öğrenmesinin taklidi, biyolojik evrimin benzetimi, vb). Doğada gözlemlenen evrimsel süreçten faydalanan ve "evrimsel hesaplama" olarak adlandırılan yöntemlerin en göze çarpanı Genetik Algoritma'dır (Mitchell, 1998). Genetik Algoritma (GA), evrimsel bir süreç kullanarak, bu süreç sonunda en iyi sonucu veren çözüme erişmeye çalışan bir arama ve eniyileme yöntemidir. GA ile, geleneksel analitik ve nümerik yöntemlerle çözümü mümkün olmayan veya çok uzun sürede çözüme ulaşılan problemlerde, en iyinin hayata devam etmesi prensibine dayanarak en iyi çözüm aranır.

GA'nın temelleri, 1960'lı ve 1970'li yıllarda Michigan Üniversitesi'nde John Holland ve öğrencileri tarafından yapılan çalışmalarda atılmıştır. Holland, bu çalışmaları daha sonra bir kitapta (Holland, 1975) toplayarak GA'nın teorik çerçevesini oluşturmuştur (Mitchell, 1998).

GA terminolojisinde kullanılan terimler, biyolojik evrimde kullanılan terimlerle örtüşmektedir. Örneğin; arama esnasında kullanılan her değişken "birey (kromozom)", bu değişkenlerin tümü "nüfus" ve her döngüde yer alan bireyler "nesil" olarak adlandırılmaktadır. Genellikle kromozomlar GA programlarında ikili kodlanır. Ayrıca, GA programlarında işleç olarak "seçim", "çaprazlama" ve "mutasyon" tanımlanmaktadır. Seçim işleci, nüfustaki bireyleri tekrar üretim için seçer. Bireylerin seçilme olasılığı, bu bireylerin problem için çözüm olup olmayacağına karar veren "uygun-

luk işlevi (fitness function)" değerine göre değişir. Uygunluk değeri yüksek olan bir bireyin hayatta kalma ve nüfustaki diğer bireyler ile çoğalma olasılığı yüksektir. Çaprazlama işleci, uygun bir konum belirleyerek, iki kromozomun bu konumdan önceki ve sonraki kısımlarını çaprazlar ve sonuçta iki yeni birey oluşturur. GA programlarında, çaprazlama sıklığını belirlemek için çaprazlama olasılığı kullanılır. Mutasyon işleci, programda ikili olarak kodlanan bireylerin rastgele herhangi bir bitinin değerini çevirir; 1 ise 0, 0 ise 1 yapar. Kromozomun parçalarının ne kadar sıklıkla mutasyona uğrayacağı, mutasyon olasılığı ile belirlenir.

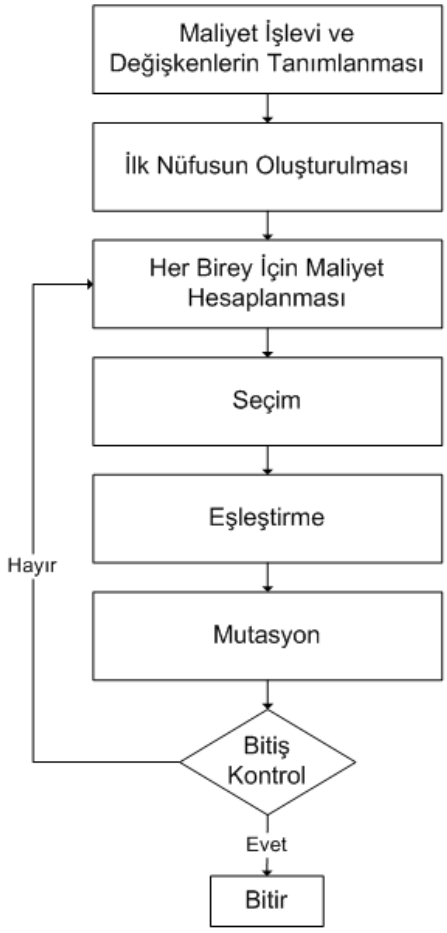
GA, aynı anda birden fazla birey ile işlem yapabildiği için geniş arama uzayına sahip problemlerin çözümünde başarılıdır. Ayrıca uygunluk değeri düşük olan bireyler zamanla nüfustan çıkarıldığı için arama uzayında, çözüm olma olasılığı yüksek bireyler ve bunlardan doğan çocuklar yer alır. Bu sayede, geniş arama uzayına sahip, geleneksel yöntemlerle sonuç alınamayan veya belli bir matematiksel modelle ifade edilemeyen problemlerde etkili ve kullanışlıdır. Ancak GA, her zaman için en iyi çözüme ulaşma garantisi vermez (Haupt and Haupt, 2004).

Basit bir GA'nın genel görünümü ve ara basamakları Şekil 2.10'da belirtilmiştir (Haupt and Haupt, 2004). Bu bölümde ara basamaklar kısaca irdelenecektir.

2.5.1 Maliyet İşlevi ve Değişkenlerin Tanımlanması

Maliyet işlevi (cost function), bir matematiksel işlev, bir deney veya bir oyun olabilir. Örneğin; bir sayısal filtrenin katsayılarında yapılan yuvarlama işlemi (Karaboğa, 1994), bir haberleşme kanalı için kaynak tahsisi (Turgu, 2008), bir anten dizisi tasarımı gibi problemler için kullanılan maliyet işlevi farklı biçimlerde tanımlanabilir. Burada genel olarak amaç, problemdeki girişlerin (değişkenlerin) değiştirilmesiyle istenen maliyet işlevini maksimize veya minimize etmektir. GA terminolojisinde maliyet işlevi ile uygunluk işlevi terimleri aynı anlamda kullanılmaktadır (Haupt and Haupt, 2004). Her bir bireyin uygunluk değeri, problemin uygunluk işlevi ile belirtilen değere göre hesaplanır. Bireyin uygunluk değeri, o bireyin sonraki nesillere aktarılmasını belirleyen bir unsurdur. Bu sebeple her nesildeki her bir birey için uygunluk değeri hesaplanır.

Problemde tanımlanan değişkenler, nüfustaki her bireyin yapısını tanımlar. Başka bir deyişle *n* değişkenli bir problemdeki bireyler *n* boyutlu bir vektörden oluşur. De-



Şekil 2.10. Genetik Algoritma Genel Şeması

ğişkenler farklı veri tiplerinde tanımlanabilirler. Örneğin; ikili (binary), tamsayı, gerçel sayı, karakter, vb. Literatürde sıklıkla ikili kodlama kullanılmaktadır; her bir değişken ’0’ veya ’1’ bitlerini alabildiğinden, bireyler iki tabanında sayılardan oluşmaktadır.

2.5.2 İlk Nüfusun Oluşturulması

Problemin başlangıcında rastgele belirlenen bireyler, çözüm arayışının ilk adımında yer alırlar. Bu bireyler genellikle rastgele sayı üreteçleri ile belirlenirler. Ancak literatürde ilk nüfusun kısmen gelişigüzel oluşturulmasının etkileri araştırma konusu olmuştur (Karcı, 2002). Başlangıçtaki bireylerin sayısı, sonraki nesillerdeki bireylerin sayısını da belirler. Burada birey sayısının kalabalık olması çözüme ulaşmayı kolaylaştırsa da her bir nesilde yapılan arama süresini arttırmaktadır.

2.5.3 Her Birey İçin Maliyet Hesaplanması

Nüfusta yer alan her birey için maliyet (uygunluk) değeri hesaplanır. Bu değere göre bireyin çözüme yaklaşıp yaklaşmadığı belirlenmiş olur ve bireyin sonraki nesillere aktarılıp aktarılmayacağı belirlenir.

2.5.4 Seçim

Bireylerin uygunluk değerleri hesaplandıktan sonra, hangi bireylerin sonraki nesillere aktarılacağının kararı bu aşamada verilir. Uygunluk değerlerine göre bireyler en güçlüden en güçsüze doğru sıralanır. Bu aşamada güçsüz birey olarak tanımlanan bireyler nüfustan çıkarılır. Hangi bireylerin nüfustan çıkarılacağına karar vermek tasarımcının kararına bağlıdır. Ancak az sayıda bireyin hayatta kalmasına izin vermek güçlü bireylerin genlerinin aktarılmasını engelleyebilir. Hayatta kalacak bireylerin sayısını fazla tutmak da güçsüz bireylerin soylarını devam ettirmelerine sebep olabilir. Bu iki durumdan hareketle genellikle, nüfustaki bireylerin yarısı hayatta kalırken diğer yarısı nüfustan çıkarılır.

2.5.5 Eşleştirme (Çaprazlama)

Genlerinin sonraki nesillere aktarılmasına karar verilen bireyler eşleştirilerek yeni bireyler oluşturulur. Burada, güçlü bireyler birden fazla kez ebeveyn olabilir. Hangi bireylerin eşleştirileceğine karar vermek için farklı yöntemler kullanılmaktadır. Bunlardan en basiti, sıradan her bireyin bir sonraki bireyle eşleştirilmesidir. Başka bir yöntem ise hayatta kalan bireyler arasında rastgele eşleştirme yapmaktır. En sık kullanılan yöntemler ise güçlü bireylere daha fazla şans tanıyan ağırlıklandırma yöntemleridir.

2.5.6 Mutasyon

Eşleştirme ile doğan yeni bireylerin genlerini mutasyona tabi tutarak, GA'nın yerel minimum veya maksimum değerlere takılması engellenir. Bu sayede algoritma, arama uzayının farklı kesimlerine de atlamış olur ve kısa zamanda popüler bir kesime yakınsamanın önüne geçmiş olur.

2.5.7 Bitiş Kontrol

Algoritmanın bitişini belirlemek için iki farklı kıstas kullanılabilir. Belirlenen nesil sayısına ulaşılmışsa veya maliyet işlevinde belirlenen seviyenin üstünde iyileştirme olmamışsa algoritma bitirilir.

2.6 Hata Tespit Sistemleri

Sayısal sistemlerde, iletim esnasında veya saklanan verilerde gürültü sebebiyle hata oluşması sıklıkla karşılaşılan bir durumdur. Bu durumlar için gerek iletim protokollerinde, gerekse veri depolama birimlerinde hata tespit ve/veya hata giderme yöntemleri uygulanmaktadır. Örneğin; eşlik biti (parity bit), sağlama toplamı (checksum), çevrimsel artıklık kodlaması (cyclic redundancy check) gibi yöntemler bunlardan bazılarıdır. Eşlik biti yönteminde, saklanan veya iletilen veride yer alan 1'lerin sayısının tek veya çift olduğu bilgisi hesaplanır. Örneğin; ASCII 1001001 (49H) karakterinin iletileceğini varsayalım. Bu karakterdeki 1'lerin sayısı 3'tür. Çift eşlik biti kullanılması durumunda, en soldaki bit eşlik biti olarak kullanılır ve çift sayıda 1 olabilmesi için 1 yapılır. Başka bir deyişle 11001001 bilgisi iletilir. Bu yöntem ile sadece hata tespiti yapılabilmektedir. Hangi bitte hata olduğu bilinemez. Ayrıca birden fazla bitte hata olması durumunda hata tespit edilemeyebilir. Örneğin; 1'lerin sayısının çift olduğu bir veride, iki tane 1 olan bit 0 olursa, 1'lerin sayısı yine çift olacaktır. Eşlik biti kullanılmasının avantajı, tek bit ile ifade edilebilmesi ve "DIŞARAN VEYA (Exclusive OR)" kapıları ile kolayca hesaplanabilmesidir. Sağlama toplamı yönteminde ise gönderilen veya saklanan veriler, eldeler ihmal edilerek toplanır ve toplamın ikiye tümleyeni sağlama bilgisi olarak bu verilere iliştirilir. Örneğin; iletilecek bilgilerin on altılık tabanda 14, 1*A*, 3*E*, 82 ve *FC* olduğunu varsayalım. Bu bilgilerin toplamı 1*EA*'dır. Oluşan elde ihmal edildiğinde kalan *EA* olacaktır ve ikiye tümleyeni 16'dır. Alıcı tarafta, son iletilen bilginin sağlama olduğu bilinir. İletilen tüm veriler (sağlama bilgisi dahil) toplanır. Elde ihmal edildiğinde toplam sıfırdan farklı ise hata oluştuğu anlaşılır. Toplama sağlamasında, eşlik biti yöntemine göre hesaplama daha uzun sürmesine karşın, hata tespit oranı daha yüksektir. Çevrimsel artıklık kodlaması, polinom kodlama olarak da adlandırılır. Bu yöntemde bir polinom üreteci belirlenir ve bu üretecin her bir biti polinomun katsayılarını teşkil ettiği kabul edilir. Örneğin; 110010 şeklinde tanımlanan polinom üreteci için 1, 4 ve 5'inci dereceden elemanların katsayısı sıfırdan farklı olduğundan, polinom $x^5 + x^4 + x^1$ biçiminde tanımlanır.

Hata tespiti için, gönderilecek veya saklanacak veri bu polinoma bölünür ve kalan, denetim verisi olarak gönderilir veya saklanır. Bu hesaplamalar diğer yöntemlere göre daha karmaşıktır ve uzun sürede tamamlanır. Ancak, birçok iletim ve saklama sisteminde kullanılmaktadır. Tüm bu yöntemlerde hata tespit edildikten sonra, hatalı verinin en baştan tekrar gönderilmesi veya saklanması gerekmektedir.

Hata düzeltme yöntemlerinde, hatanın nerede olduğu algılanarak hatasız veriye ulaşmak mümkündür. Bu yöntemlerden en çok bilineni Hamming kodlamasıdır (Mano and Kime, 1999). Bu yöntemin temeli, Hamming uzaklığı diye adlandırılan ve iki kod kelimesi (codeword) arasında, farklı olan bitlerin sayısı olarak tanımlanan parametreye dayanmaktadır. Hamming kodlamasında d adet hatayı algılayabilmek için Hamming uzaklığının $d + 1$ olması gerekmektedir. Benzer şekilde, d adet hatayı düzeltmek için Hamming uzaklığının $2d + 1$ olması gerekmektedir. Sistemdeki kod kelimeleri bu şartı taşıdığında, hatalı olan veriden hatasız kod kelimesini bulmak mümkündür. Örneğin; sistemdeki kod kelimelerin, 0000000000, 0000011111, 1111100000 ve 1111111111 olduğunu varsayalım. Böyle bir kodlama sisteminde herhangi iki kod kelimesi arasındaki farklı bit sayısı en az 5 olduğundan, Hamming uzaklığı 5 olarak tanımlanır. Buna göre bu kod kelimelerinde oluşacak 2 bitlik hata düzeltilebilir. Alıcı tarafına 0000000111 bilgisi geldiğinde, bu bilginin hatalı olduğu ve hatasız bilginin 0000011111 olduğu bulunabilir (Tanenbaum, 1996).

İletim esnasında veya sayısal devrelerde meydana gelen kalıcı bozulmalardan kaynaklanan hataların yanı sıra kozmik parçacıklardan veya ambalaj kaynaklı ışımadan meydana gelen geçici hatalar oluşması da muhtemeldir. Bu hatalar, sunucu işlemcileri için önemli bir sorun teşkil etmektedir. Yarı iletken hafıza ve işlem birimlerinde meydana gelen bu hatalar çoğunlukla geçici hatalardır. Bu tip hatalar tespit edildikten sonra, hatalı işlem tekrarlandığında aynı hatayla karşılaşma olasılığı düşüktür. Bu sebeple birçok güvenilir işlemcide hata tespit ve düzeltme yöntemleri uygulanarak geçici hataların etkisi azaltılmaktadır.

Geçici hatalar birden fazla etki ile oluşabilmektedir. 1970'li yıllarda üretilen rastgele erişimli belleklerde (Random Access Memory, RAM), yonga ambalajındaki radyoaktif malzemelerin zamanla aşınması ile geçici hatalar dikkat çekmeye başlamıştır. Zamanla üretici firmalar radyoaktif malzemeden arınmış ambalajlar üretmeye başladıklarında, geçici hataların başka nedenlerle de oluştuğu anlaşılmıştır. Kozmik ışın

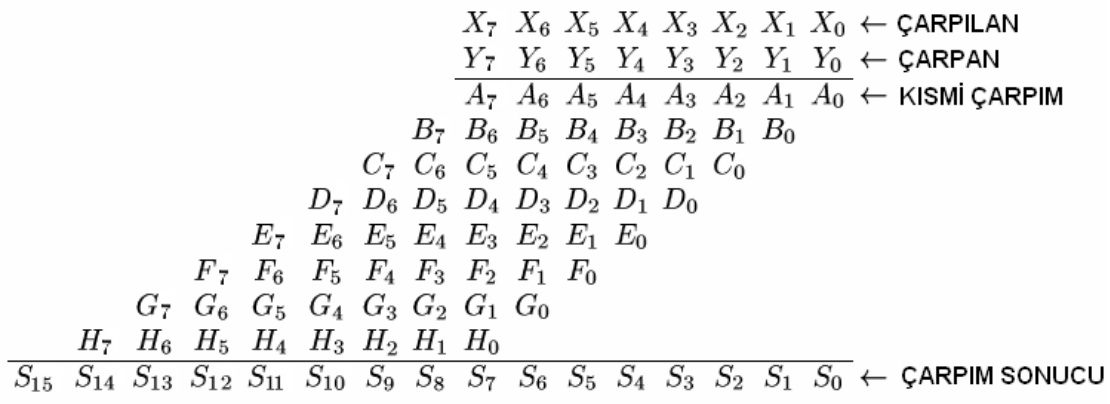
diye adlandırılan ve %95'lik kısmı nötronlardan oluşan etkinin büyük bir kısmı yeryüzüne ulaşmamasına rağmen şiddetli ikincil parçacık yağmurlarına neden olurlar. Nötronlar yüksüz olduklarından herhangi bir devrenin çalışmasına direkt etki etmezler; ama yongadaki bir atomun çekirdeğinde meydana gelen nötron yakalaması, yüklü ikincil parçacıkların oluşmasına neden olur[3].

İletim veya saklama sırasında oluşan hataların yanı sıra işlevsel birimlerde meydana gelebilecek hatalar da ayrı bir araştırma konusudur. Örneğin; toplama veya çarpma işlemi yapılırken hata tespitine karşı dayanıklı tasarımlar gerçekleştirmek mümkündür. Yilmaz tarafından yapılan çalışmada (Yilmaz *et al.*, 2006) mikroişlemcilerdeki çarpıcılar için hata tespiti ve giderilmesi için öneriler sunulmuştur. Yilmaz başka bir çalışmasında (Yilmaz *et al.*, 2007), toplayıcı ve çarpıcı gibi işlevsel birimler için hata tespiti ve giderilmesi için önerilerde bulunmuştur. Mikroişlemcilerin mevcut kaynaklarından yoğun şekilde kullanılmayan birimleri ve/veya alanları hata tespiti için kullanılabilir. Örneğin; mikroişlemcilerde veriler için belirli bir büyüklükte alan ayrılır. Ancak genellikle karşılaşılan veriler daha az bir alanda ifade edilebilmektedir. Bu durumlarda boş kalan alanlar, hassas bilgilerin kopyasının saklanması için elverişlidir. Böylece hata tespiti için asıl veri ile kopyasının karşılaştırılması yeterli olacaktır (Ergin *et al.*, 2006). Ergin başka bir çalışmasında (Ergin *et al.*, 2009), yazmaç yeniden adlandırma (register renaming) aşamasında bağımlılık testi için sistemde mevcut olan karşılaştırıcılardan boş olanlarının hata tespiti için kullanılmasını önermiştir. Sistemde mevcut olan etiket eşleme (tag-match) karşılaştırıcılarının da hata tespiti için kullanılabileceği ortaya konulmuştur (Yalçın and Ergin, 2007).

[3]http://en.wikipedia.org/wiki/Soft_error

3. ON TABANINDA ÇARPMA

On tabanına göre çarpma işleminde çarpılan sayının, çarpanın rakamları ile çarpılması sonucu kısmi çarpımlar elde edilir ve bu kısmi çarpımların ağırlıklarına göre toplanması ile çarpım sonucuna ulaşılır (Şekil 3.1).



Şekil 3.1. On Tabanında Çarpma İşlemi

Çarpma işlemi temelde iki farklı yöntemle gerçekleştirilebilir:

- Ardışık toplama ve kaydırma
- Paralel çarpma

Genel olarak, çarpma işlemi üç ara basamakta gerçekleştirilir:

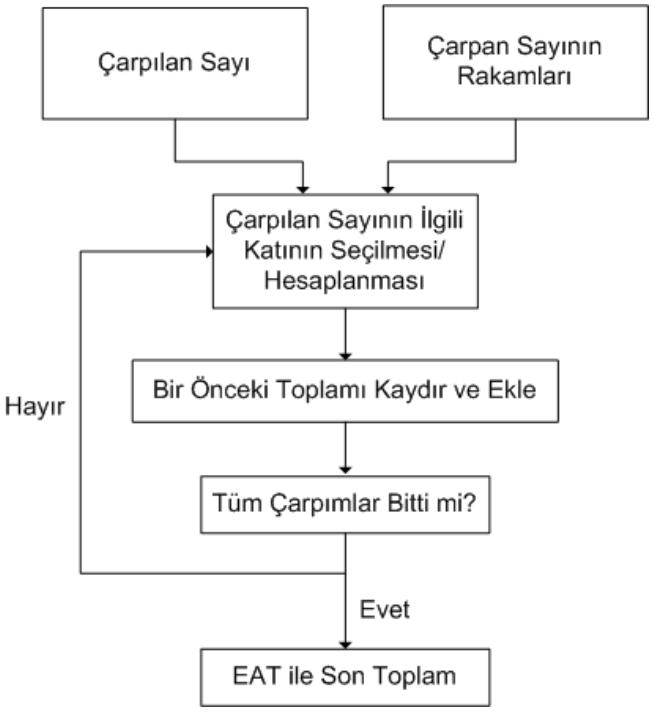
- Kısmi çarpımların hesaplanması
- Kısmi çarpımların hızlı bir şekilde indirgenmesi
- Elde Aktarımlı Toplayıcı (Carry Propagate Adder, EAT) ile son toplam

3.1 Çarpma Yöntemleri

3.1.1 Ardışık Toplama ve Kaydırma Yöntemi

Ardışık çarpma yönteminde, her aşamada kısmi çarpımlar hesaplanır ve kaydırılarak bir önceki toplam sonucuna eklenir (Şekil 3.2). Basit bir yaklaşımla, çarpan

sayının her bir rakamı, çarpılan sayı ile çarpılarak bir çarpım yazmacına yazılır. Her basamakta hesaplanan bu kısmi çarpım, daha önceki çarpım yazmacındaki değerin bir basamak sağa kaydırılmış haliyle toplanır ve sonuç yine çarpım yazmacında saklanır. Çarpan sayının tüm rakamları için bu döngü devam eder. Çarpan sayının rakamları ile çarpılan sayıyı her basamakta çarpmak yerine, çarpılan sayının $2-9$ arasındaki katları bir defaya mahsus olarak hesaplanıp, her basamakta ilgili kat kullanılabilir. Çarpılan sayının tüm katlarının saklanması yerine sadece bazı katların saklanıp, diğer katların sadece gerektiğinde dinamik olarak hesaplanması daha iyi bir çözümdür. Her bir basamakta hesaplanan kısmi çarpımın, daha önceki toplam değerine eklenmesi için ara format kullanılması veya son aşamada ihtiyaç duyulan elde aktarımı gibi konularda literatürde çeşitli çalışmalar mevcuttur. Bu kısımda, literatürde yer alan on tabanına göre ardışık çarpma yöntemleri için geliştirilen bu tür çözümler irdelenecektir.



Şekil 3.2. Ardışık Çarpma Blok Şeması

Bilgisayar tarihinde on tabanında aritmetik işlemlerin popüler olduğu dönemlerde, ardışık çarpma yöntemleri ile ilgili çeşitli çalışmalar patent ve teknik dökümanlarda yer almıştır (Larson, 1973*b*), (Larson, 1973*a*), (Hoffman and Schardt, 1975), (Ohtsuki *et al.*, 1987), (Bradley *et al.*, 1986) ve (Ueda, 1995). Daha sonraki dönemlerde, iki ta-

banında aritmetik işlemlerin daha yaygın olarak kullanılmasıyla birlikte on tabanında yapılan çalışmalar popülerliğini yitirmiştir.

Larson (Larson, 1973*b*) tarafından yapılan çalışma, çarpan ve çarpılan sayıların her bir rakamının çarpılması esasına dayanmaktadır. Bu rakam çarpımları, başvuru tabloları yardımı ile hesaplanmıştır. Hesaplanan kısmi çarpımlar, ardışık şekilde bir önceki basamakta elde edilen sonuca, kaydırılarak EAT ile eklenmiştir. Larson daha sonraki çalışmasında (Larson, 1973*a*), yine tablo yardımı ile rakam çarpımlarını hesaplamış; ancak ara toplamlarda EAT yerine Elde Saklayan Toplayıcı (EST) kullanmıştır.

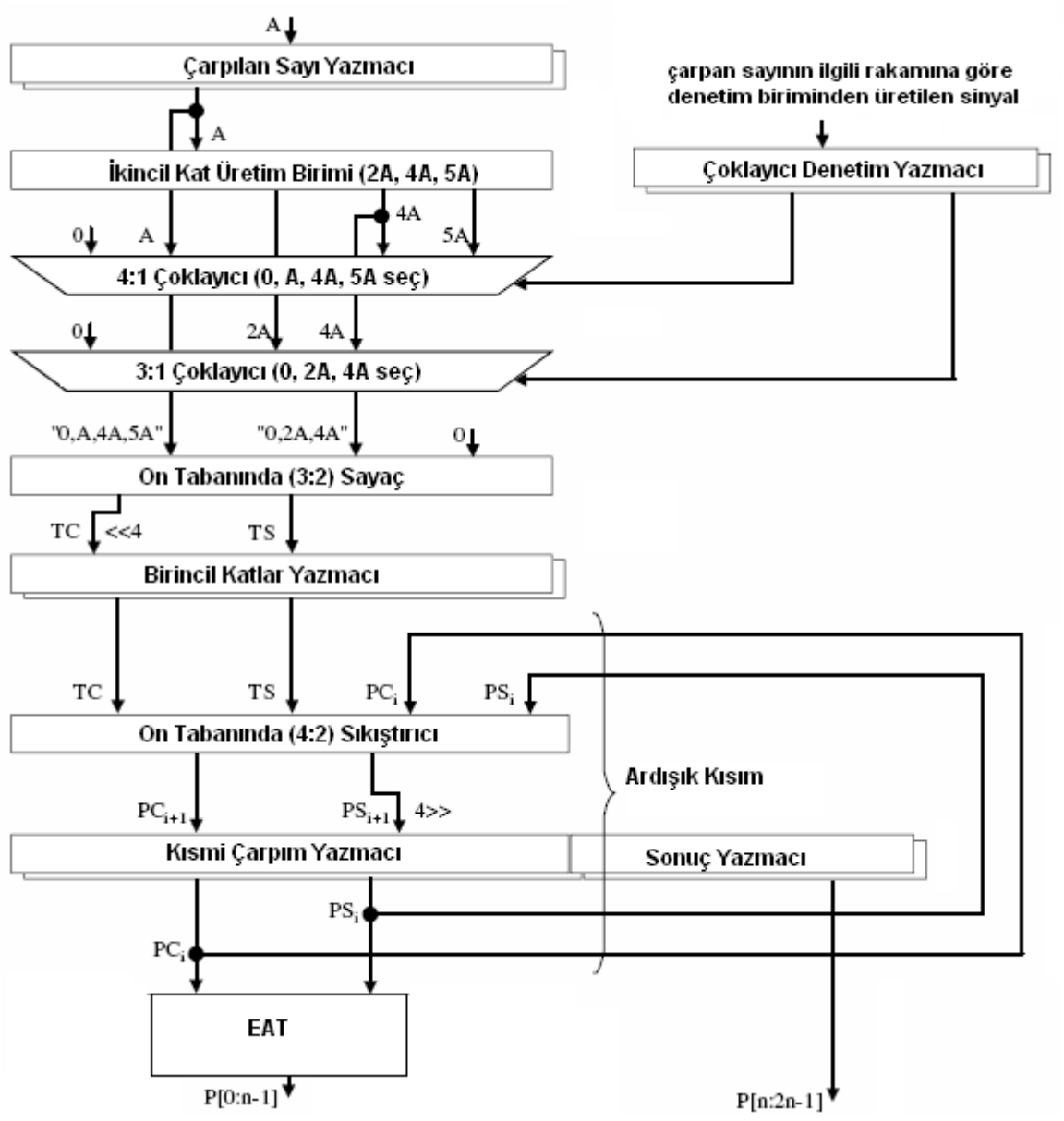
Ohtsuki (Ohtsuki *et al.*, 1987) yaptığı patent çalışmasında, çarpılan sayının $2, 4$ ve 8 katları ile kendisi kullanarak diğer katları hesaplamıştır. Ancak, 7 katını bulmak için sayının kendisini hem 2 katı ile hem de 4 katı ile toplamak gecikme başarımını olumsuz etkilemiştir. Bu çalışmada, hesaplanan kısmi çarpımlar tüm rakamlarına 6 eklenerek döngüye dahil edilmiştir. Ardışık iki kısmi çarpımın toplanmasında elde oluşmaması durumunda ilgili rakamdan 6 çıkarılarak çarpım sonucu elde edilmiştir.

Ueda (Ueda, 1995) tarafından yapılan patent çalışmasında, çarpan sayının her bir rakamı, çarpılan sayının rakamları ile tek tek çarpılarak kısmi çarpımlar elde edilmiştir. Her bir rakam çarpımı, programlanabilir mantıksal dizi (Programmable Logical Array) kullanılarak gerçekleştirilmiştir. Rakam çarpım birimleri, çarpan ve çarpılan sayının her bir rakamına uygun biçimde bağlanarak rakam çarpımları elde edilmiştir ve ardışık toplama işlemi ile çarpım sonucuna ulaşılmıştır.

2000'li yıllara gelindiğinde, yonga üretimlerinin ucuzlamasıyla beraber on tabanında aritmetik işlemlerin gerçekleştirildiği donanımların tasarımı tekrar gündeme gelmiştir. Örneğin, Busaba (Busaba *et al.*, 2001) tarafından yapılan çalışma IBM z900 serisi işlemcilerde kullanılmıştır. Bu çalışmada çarpılan sayının $2, 4, 6$ ve 8 katı hesaplanarak saklanmıştır. Çarpılan sayının tek katları, ihtiyaç duyulduğunda, sayının kendisi ile daha önceden saklanan katları toplanarak elde edilmiştir.

Erle (Erle and Schulte, 2003) tarafından yapılan çalışmada, on tabanında ardışık çarpma yöntemi önerilmiştir. Bu çalışmada, kısmi çarpımların toplanması sırasında EST kullanılarak artık (redundant) biçiminde ara değerler elde edilmiştir. Çarpılacak sayının (A) $2, 3, 4$ ve 8 katı hesaplanarak $\{0, A, 2A, 3A\}$ ve $\{0, 4A, 8A\}$ ikincil kümeleri

oluşturulduktan sonra, çarpan sayının ilgili rakamına göre bu iki kümeden birer eleman alınıp toplanarak kısmi çarpımlar hesaplanmıştır. Örneğin; çarpan sayının ilgili rakamı 7 ise ilk kümeden 3*A*, ikinci kümeden 4*A* alınarak kısmi çarpım elde edilmiştir. Daha sonra, her adımda hesaplanan kısmi çarpımlar, bir önceki toplam sonucuna, ardışık ekle-kaydır yöntemi ile eklenerek toplam ve elde vektörleri elde edilmiştir. Bu aşamada on tabanı için (3:2) sayaç (counter) kullanılmıştır. Son aşamada, bu iki vektör EAT ile toplanarak çarpım sonucuna ulaşılmıştır. Erle, aynı çalışmada daha sonra $\{0, A, 4A, 5A\}$ ve $\{0, 2A, 4A\}$ kümelerini oluşturarak ve ekleme kısmında on tabanı için (4:2) sıkıştırıcı kullanarak kendi tasarımını geliştirmiştir (Şekil 3.3).



Şekil 3.3. Ardışık Ekle Kaydır Yöntemi İle Çarpma (Erle and Schulte, 2003)

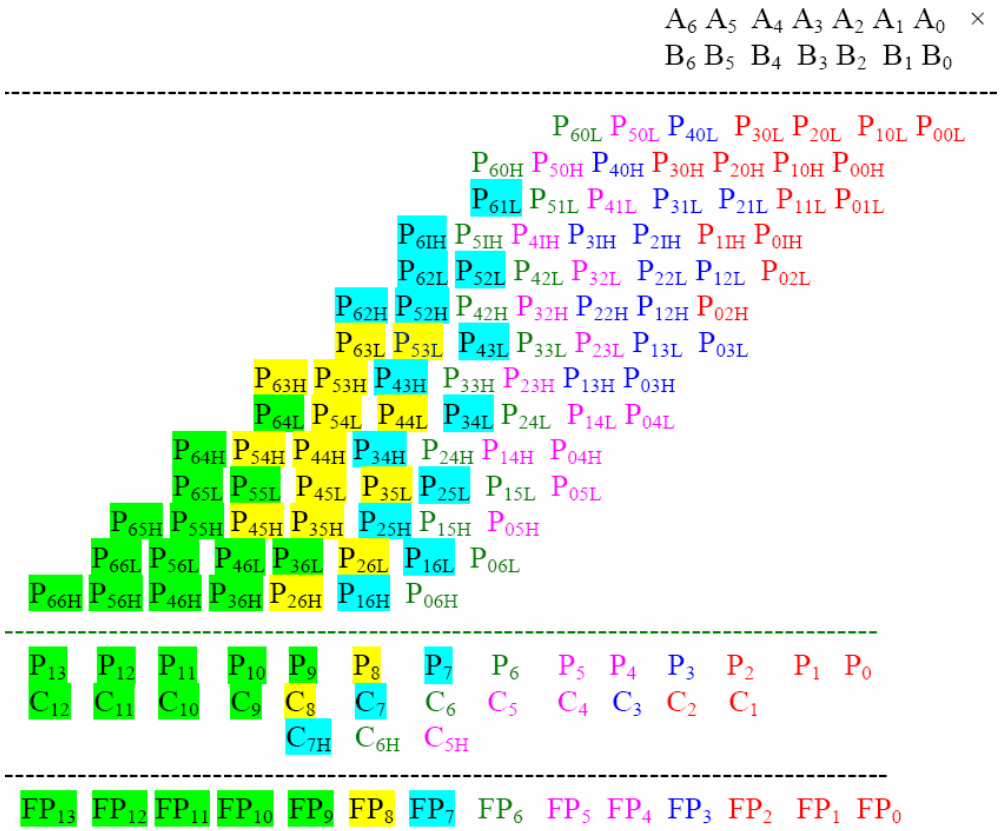
Kenney (Kenney *et al.*, 2004) tarafından yapılan çalışmada, ana hatlarıyla Erle (Erle and Schulte, 2003) tarafından önerilen yapı kullanılmıştır. Erle, toplama işlemi sonucunda elde oluşması durumunda 6 eklenmesini önerirken, Kenney, elde oluşması durumunda 6 eklenmesi işlemini bir sonraki aşamada gerçekleştirmiştir. Bu çalışmada çarpma işlemi hızlandırılmış; ancak donanımın kapladığı alan artmıştır.

Erle başka bir çalışmasında (Erle *et al.*, 2005), çarpan sayının her bir rakamını $[-5, 5]$ aralığında işaretli büyüklük (signed-magnitude) biçiminde rakamlar olacak şekilde yeniden kodlayarak çarpılan sayı ile çarpmıştır. Bu şekilde elde edilen kısmi çarpımlar ardışık olarak toplanmıştır. Erle, rakamları işaretli büyüklük biçiminde yeniden kodlayarak hem kısmi çarpımların elde edilmesini hem de kısmi çarpımların indirgenmesini iyileştirmiştir.

Erle daha önce yapmış olduğu çalışmayı (Erle and Schulte, 2003) kullanarak kayan noktalı sayılar için literatürdeki IEEE (*IEEE 754-2008 - IEEE Standard for Floating-Point Arithmetic, 2008)* standardına uyumlu ilk çalışmayı gerçekleştirmiştir (Erle *et al.*, 2007). Bu çalışmada, kayan noktalı sayılar için standartta belirlenmiş olan veri biçimlerine, aritmetik işlemlere ve yuvarlamalara uygun bir tasarım gerçekleştirilmiştir.

James (James *et al.*, 2008*b*) tarafından yapılan çalışmada, 7 rakamlı iki sayının on tabanında çarpımı gerçekleştirilmiştir. Bu çalışmada, çarpma sonucuna 8 saat darbesinde ulaşılmıştır. Her bir saat darbesinde, çarpan ve çarpılan sayıların rakamlarından 7 tanesi çarpılmıştır. Çarpılan bu rakamlar, birler ve onlar basamağına ayrılarak uygun ağırlıktaki rakamlar toplanmıştır. Bu toplama sonucunda çarpım sonucunun en sağdaki 3 basamağı hesaplanmıştır. Ancak sonraki saat darbelerinde, sağdan sola doğru sırayla $1, 2, 1, 1, 1$ ve 5 basamak hesaplanmıştır (Şekil 3.4). Bu yöntemde, sol taraftaki kalabalık kolonlara doğru hesaplama devam ettikçe her bir saat darbesinde daha fazla sayıda rakamın toplanması gerekmektedir. Bu sebeple 7'den daha fazla rakam için bu yöntem uygun olmayacaktır. Örneğin; 16 rakamlı iki sayının çarpımı için diğer ardışık yöntemlerde, her bir saat darbesinde 2 veya 3 rakam toplanırken, bu yöntemde 16 rakamın toplanması gerekecektir.

James yaptığı başka bir çalışmada (James *et al.*, 2009*b*), Erle'nin önerdiği (Erle and Schulte, 2003) kısmi çarpımların hesaplanması yönteminden yararlanmıştır. Bu çalışmada çarpan sayının iki rakamı paralel olarak çarpılan sayı ile çarpılmış ve (4:2)



Şekil 3.4. Kısmi Çarpımların Hesaplanması ve İndirgenmesi (James et al., 2008b)

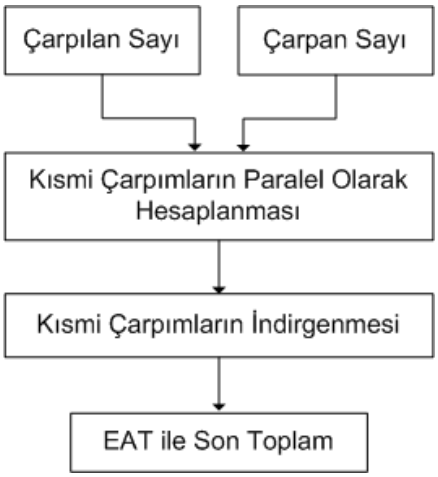
sıkıştırıcı kullanılarak döngüye dahil edilmiştir. James bu çalışmasında 7, 16 ve 34 rakamlı sayıların çarpımı için çarpıcı tasarlamıştır. Bu çalışma, farklı alan programlanabilir kapı dizisi (Field Programmable Gate Array) üreticilerinin farklı aygıtları için sentezlenmiştir. Sentez sonuçlarında üç farklı tasarımın farklı aygıtlarda kapladığı alan ve gecikme başarımları sunulmuştur. James, aynı çalışmayı tekrar etmiş; ancak bu defa sentez sonuçlarında, her tasarımın alt birimlerinin de alan ve gecikme başarımları rapor edilmiştir (James *et al.*, 2009*a*).

On tabanında aritmetik işlemlerin hızlanması ve popülerliğinin artmasıyla beraber, bu işlemlerin kullanım alanı da genişleyecektir. Bu sebeple son kullanıcılar tarafından kullanımı daha elverişli olan FPGA teknolojisi için on tabanında aritmetik işlemlerin donanım tasarımı, yakın dönemde daha fazla sayıda araştırmanın konusu haline gelmektedir. Sutter (Sutter *et al.*, 2009) tarafından FPGA için gerçekleştirilen çalışmada, ardışık ve paralel BCD çarpıcıların tasarımı gerçekleştirilmiştir. Yöntemin ardışık veya paralel olmasına bağlı olarak, FPGA içerisinde yer alan blokların verimli şekilde kullanılmasına dair yaklaşımlar sergilenmiştir. Daha sonra Minchola

bu çalışmayı, kayan noktalı sayılar için kullanmıştır (Minchola and Sutter, 2009). Minchola'nın yapmış olduğu çalışma literatürde yer alan, FPGA ile kayan noktalı sayılar için çarpıcı tasarımı konusunda bir ilk olmuştur.

3.1.2 Paralel Çarpma Yöntemi

Paralel çarpma yönteminde, kısmi çarpımlar birbirinden bağımsız olduğu için paralel olarak hesaplanır. Kısmi çarpımların hesaplanması için ardışık yöntemlerde bahsedilen yöntemler kullanılabilir. Temel fark tüm kısmi çarpımların birbirinden bağımsız olarak hesaplanmasıdır. Kısmi çarpımlar, ağaç yapıları ile veya birden fazla BCD rakam toplayan hızlı toplayıcılar ile indirgenir. Genellikle bu indirgeme sonucu, artık ara biçimde ifade edilen toplam ve elde vektörlerinden oluşur. Bu vektörler, son toplama işleminde elde aktarımı yapılarak toplanır ve çarpım sonucuna ulaşılır (Şekil 3.5). Bu kısımda, literatürde yer alan, paralel çarpma kullanılarak geliştirilen on tabanına göre çarpma yöntemleri irdelenecektir.



Şekil 3.5. Paralel Çarpma Blok Şeması

3.1.2.1 Lang ve Nannarelli Yöntemi

Lang (Lang and Nannarelli, 2006) tarafından önerilen yöntem, literatürde yer alan paralel çarpma yöntemlerinin ilkidir. Bu yöntemde, çarpan sayının rakamları y_i = y_{Hi}+y_{Li} şeklinde yeniden kodlanmıştır. Burada $y_i, y_{Hi} \in \{0, 5, 10\}$ ve $y_{Li} \in \{-2, -1, 0, 1, 2\}$ kümelerinden seçilerek kodlanır. Buna göre, örneğin ilgili rakam 7 olduğunda, ilk kümeden 5 ve ikinci kümeden 2 seçilir. Bu yöntemde kısmi çarpımların hesaplanması

için, çarpılan sayının sadece 2 ve 5 katının saklanmasına ihtiyaç duyulmaktadır. Bu sayede kısmi çarpımların hesaplanması hızlandırılmıştır. Çarpan sayının her bir rakamına göre Çizelge 3.1'de gösterilen şekilde çarpılan sayının katları seçilir.

Çizelge 3.1. Lang Yönteminde Çarpan Sayının Rakamlarına Göre Kat Seçimi

y_i	y_{Hi}	y_{Li}	y_i	y_{Hi}	y_{Li}
0	0	0	**5**	5A	0
1	0	A	**6**	5A	A
2	0	2A	**7**	5A	2A
3	5A	-2A	**8**	10A	-2A
4	5A	-A	**9**	10A	-A

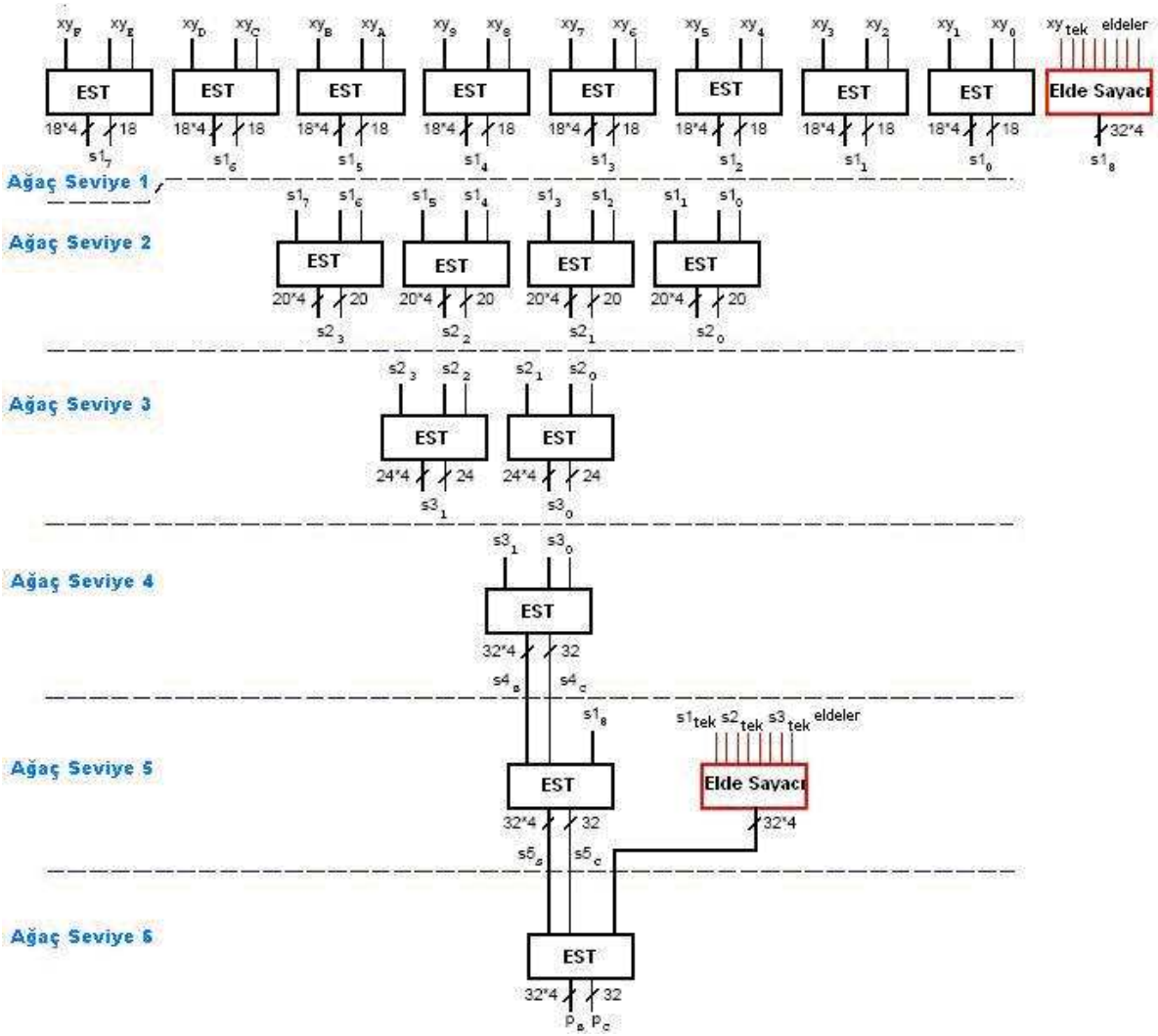
$\{0, 5A, 10A\}$ ve $\{0, A, -A, 2A, -2A\}$ kümelerinden seçilen katlar, EST'ler ile toplanarak kısmi çarpımlar bulunur. On tabanına göre 4 rakamlı iki sayının çarpılması Şekil 3.6'da örnek olarak gösterilmiştir.

$$
\begin{array}{lr}
x: & \\
y: & \\
\hline
xy_0: & 5x \\
 & 0 \\
\hline
xy_1: & 5x \\
 & -x \\
\hline
xy_2: & x \\
 & 0 \\
\hline
xy_3: & 10x \\
 & -2x \\
\end{array}
\qquad
\begin{array}{cccccccc}
 & & & & 1 & 9 & 6 & 3 \quad \times \\
 & & & & 8 & 1 & 4 & 5 \quad = \\
\hline
 & & & & 9 & 8 & 1 & 5 \\
 & & & & 0 & 0 & 0 & 0 \\
\hline
 & & & 9 & 8 & 1 & 5 & \\
 & & - & 1 & 9 & 6 & 3 & \\
\hline
 & & 1 & 9 & 6 & 3 & & \\
 & & 0 & 0 & 0 & 0 & & \\
\hline
1 & 9 & 6 & 3 & 0 & & & \\
- & 3 & 9 & 2 & 6 & & & \\
\hline
1 & 5 & 9 & 8 & 8 & 6 & 3 & 5
\end{array}
$$

Şekil 3.6. Lang Yöntemi İle 4 Rakamlı İki Sayının Çarpılması (Lang and Nannarelli, 2006)

Hesaplanan kısmi çarpımların toplanması için on tabanına göre toplama yapan EST'ler ağaç yapısında kullanılmıştır (Şekil 3.7). Şekil 3.7'de gösterilen ağaç yapısının girişleri, Çizelge 3.1'e göre seçilen bileşenlerin EST ile toplanmasıyla elde edilen toplam ve elde vektörleridir. 16 rakamlı bir çarpıcıda 16 toplam ve 16 elde vektörü oluşur. Toplam vektörlerinin tamamı ve elde vektörlerinin sadece çift indisli olanları 8

EST kullanılarak toplanmaktadır. Geriye kalan 8 tane tek indisli elde vektörü, sayaç kullanılarak toplanmış ve beşinci seviyeden ağaç yapısına dahil edilmiştir. Bu ağaç yapısının çıkışında 32 rakamlı toplam vektörü ve 32 bitlik elde vektörü oluşur. Çarpım sonucuna ulaşmak için bu iki vektörün EAT ile toplanması gerekmektedir. Lang tarafından önerilen bu yöntem, bu tez için referans tasarım olarak alınmış ve donanım tasarım dili kullanılarak gerçeklenmiştir. Tez çalışmalarının olgunlaşması süresince yapılan çalışmalar, bu yöntem baz alınarak gerçekleştirilen tasarımla karşılaştırılmıştır.



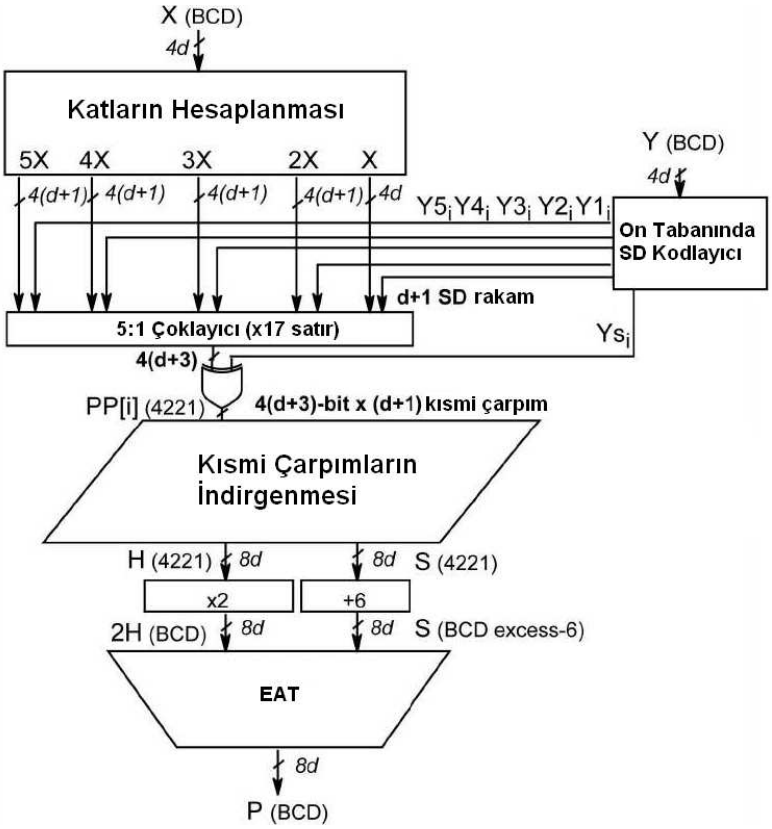
Şekil 3.7. Paralel Çarpma Yönteminde Ağaç Yapısı (Lang and Nannarelli, 2006)

Daha sonraki bir çalışmada Dadda (Dadda and Nannarelli, 2008), Lang tarafından önerilen yöntemde kısmi çarpımların indirgenmesi aşamasında, daha önce yapmış

olduğu çalışmayı (Dadda, 2007) kullanarak, birden fazla BCD rakam toplamını gerçekleştirmiş ve tasarımın gecikme başarımını iyileştirmiştir.

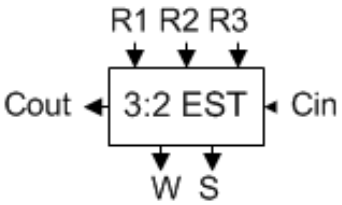
3.1.2.2 Vazquez Yöntemi

Vazquez (Vazquez *et al.*, 2007) tarafından yapılan çalışmada, çarpan ve çarpılan sayıların rakamları, klasik 8421 ağırlıklandırma yerine 4221 ağırlıklandırma kullanılarak yeniden kodlanmış ve kısmi çarpımların hesaplanması ve kısmi çarpımların indirgenmesi aşamaları hızlandırılmıştır (Şekil 3.8). Bu çalışmada kısmi çarpımların indirgenmesi için EST'lerle kurulan ağaç yapısı kullanılmıştır. Bu yöntemde BCD4221 gösterimi kullanımının en önemli avantajı, kısmi toplamların indirgenmesi aşamasında iki tabanında EST'ler kullanılabilmesidir. Ayrıca iki tabanında EST'ler sayesinde aynı tasarım iki tabanında çarpma yapması için de uygun hale gelmektedir. BCD4221 gösterimi ile kurulan yapıda iki tabanında toplayıcı kullanılmasına rağmen, elde hesabında da avantaj sağlamaktadır.



Şekil 3.8. Paralel Çarpma İçin Vazquez Tarafından Önerilen Yapı (Vazquez et al., 2007)

Vazquez tarafından önerilen BCD4221 gösterimi için EST blok şeması Şekil 3.9'da ve bu EST ile örnek bir toplama Şekil 3.10'da verilmiştir. Bu EST'ler kullanılarak 15 rakamın toplanması için kurulan örnek bir ağaç yapısı Şekil 3.11'de verilmiştir. Vazquez sonraki çalışmasında (Vazquez *et al.*, 2010), birden fazla BCD4221 ve BCD5211 biçiminde gösterilen rakamın toplanması için bit sayaçları önermiştir.

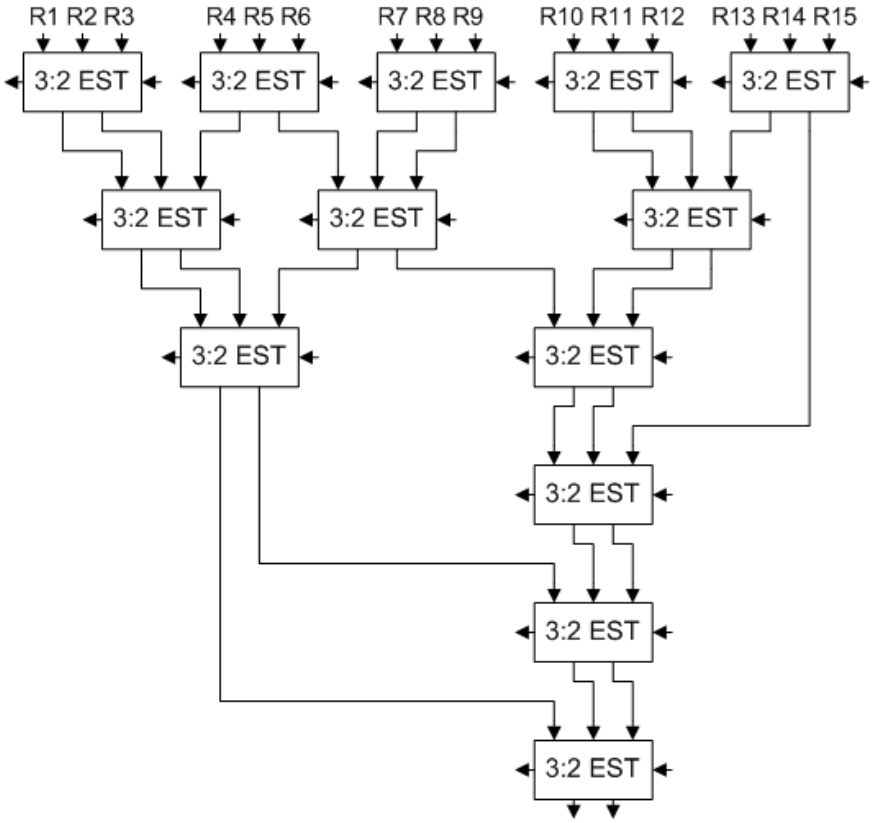


Şekil 3.9. BCD4221 Gösteriminde EST

```
R1=5:    1001
R2=9:    1111
R3=7:    1011
         --------
 S=7:    1101
 C=7:    1011
 H=7:    1100   (5211 gösterimi)
W=2H:    100Cin  (4221 gösterimi)
Cout:    1
-----------------------
T= 10*Cout+S+W
 = 10+7+4 = 21
```

Şekil 3.10. BCD4221 Gösteriminde EST İçin Örnek Toplama

Vazquez tarafından yapılan çalışma bu alanda yapılmış en önemli çalışmalardan biri olmuştur. Daha sonra yapılan çalışmalarda Vazquez'in önerdiği yönteme getirilen iyileştirmeler olmuştur ve bu yöntemi temel alan kayan noktalı çarpıcılar tasarlanmıştır. Örneğin; Hickman bu yöntemi kullanarak kayan noktalı sayılar için IEEE 754-2008 standardına uyumlu çarpıcı tasarlamıştır (Hickmann *et al.*, 2007). Daha sonra Hickmann, Vazquez'in önerdiği ağaç yapısını ve 2 ile çarpma birimini değiştirerek gecikme başarımını iyileştirmiştir (Hickmann *et al.*, 2008). Raafat tarafından gerçekleştirilen (Raafat *et al.*, 2008) ve fikri mülkiyeti Silminds[1] firmasına ait olan, kayan



Şekil 3.11. EST Kullanılarak 15 Rakamın Toplanması

noktalı çarpıcı tasarımında Vazquez tarafından önerilen yöntem kullanılmıştır. Erle, on tabanında kayan noktalı sayılar için çarpıcı tasarımlarında (Erle *et al.*, 2009), ardışık çarpıcı için daha önce yapmış olduğu (Erle and Schulte, 2003) çalışmayı ve paralel çarpıcı için Vazquez'in önerdiği yöntemi kullanmıştır.

3.1.2.3 Diğer Yöntemler

James, ardışık çarpma yöntemi için önermiş olduğu yapıyı (James *et al.*, 2008*b*) değiştirerek paralel çarpma yapmıştır (James *et al.*, 2008*a*). Bu çalışmasında da 7 rakamlı sayıların çarpımı için rakam çarpımlarından faydalanmıştır. Önerdiği paralel yapı içerisinde her bir kolon için gerekli olan, birden fazla BCD rakamın toplanması ve son toplam işlemi hakkında açık bir bilgi mevcut değildir.

Gorgin (Gorgin and Jaberipur, 2009) paralel çarpıcı için yaptığı çalışmada, on tabanında toplama işlemi için önerdiği artık toplayıcıyı kullanmıştır. Bu çalışmada öneri-

[1]http://www.silminds.com

len toplayıcıda eldesiz (carry-free) toplama yönteminden faydalanılmıştır. Önerilen toplayıcının paralel çarpma işleminde kullanılması ile iki farklı paralel çarpıcı tasarlanmıştır. Bu tasarımlardan biri alan diğeri de gecikme başarımı açısından daha avantajlıdır.

Jaberipur (Jaberipur and Kaivani, 2009) on tabanında paralel çarpma işlemini hızlandırmak için Lang (Lang and Nannarelli, 2006) ve Vazquez (Vazquez *et al.*, 2007) tarafından yapılan çalışmalardan faydalanmıştır. Jaberipur bu çalışmada, kısmi çarpımların hesaplanmasında Lang tarafından önerilen yöntemde çarpılan sayının 8 ve 9 katını direkt olarak hesaplayarak negatif katların hesaplanması gerekliliğini ortadan kaldırmıştır. Önerilen yöntemde, 8 ve 9 katları hesaplamak için çarpılan sayının tüm rakamları çarpılmış ve her bir rakam çarpımının birler ve onlar basamağı farklı iki vektörde saklanmıştır. Diğer katların hesaplanması için çarpılan sayının kendisi, iki katı ve beş katı yeterlidir. Hesaplanan bu katlara göre ilgili katın seçimi Çizelge 3.2'de belirtilen şekilde gerçekleştirilmiştir. Bu çizelgede belirtilen sinyaller aktifleştirme sinyalleri olup, hangi katların toplanacağını belirtmektedir. Diğerlerinden farklı olarak 8 ve 9 için verilen sinyaller daha önce hesaplanmış olan birler ve onlar basamağındaki vektörlerin toplanmasını aktifleştirmektedir. Kısmi çarpımların indirgenmesi aşamasında ağaç yapısı kullanılmıştır. Bu yapıda her bir seviyede iki BCD rakam toplanmıştır. Bu toplamlardan ortaya çıkan eldelerin yarısı, ağaç yapısındaki bir sonraki seviyeye aktarılırken geriye kalan eldeler sayaç kullanılarak hesaba daha sonra katılmıştır.

Çizelge 3.2. Jaberipur Yönteminde Çarpan Sayının Rakamlarına Göre Kat Seçimi

Y_j	**Aktif Sinyaller**	Y_j	**Aktif Sinyaller**
0	0 , 0	5	0 , d_5
1	d_1, 0	6	d_1 , d_5
2	0 , d_2	7	d_2 , d_5
3	d_1 , d_2	8	d_8 , d_8
4	d_2 , d_2	9	d_9 , d_9

3.2 Kısmi Çarpımların Hesaplanması

On tabanında çarpma yöntemlerinin ilk aşaması olan kısmi çarpımlarının hesaplanmasında farklı yöntemler mevcuttur. Bölüm 3.1'de bu yöntemlerin bazıları anlatılmıştır. Bu bölümde ise kısmi çarpımların hesaplanması için önerilen yöntemler, avantajları ve dezavantajları ile birlikte incelenecektir. Kısmi çarpımlar, temel olarak iki farklı yöntemle hesaplanabilir:

- Çarpan ve çarpılan sayının rakamlarının çarpılması,
- Çarpan sayının her bir rakamının çarpılan sayı ile çarpılması.

Rakam çarpımına dayalı yöntemlerde, rakam çarpımı çok kısa sürede hesaplandığı için avantaj oluşturmaktadır. Ancak ihtiyaç duyulan rakam çarpımlarının sayısı arttıkça alan dezavantajı ciddi bir sorun haline gelebilmektedir. Özellikle paralel çarpım yöntemlerinde ihtiyaç duyulan rakam çarpımlarının sayısı, sayıların basamak sayısının karesiyle orantılı olarak artmaktadır. Bu sebeple, rakam çarpımına dayalı paralel çarpıcı tasarımlarına duyulan ilgi azdır. Ancak günümüzde yonga üretimlerinin ucuzlaması ve hız parametresinin daha fazla önem kazanması ile bu yöntem, sağlanabilecek herhangi bir hız iyileştirmesi ile alternatif olmaya devam edecektir.

Çarpan sayının her bir rakamının çarpılan sayı ile çarpılması, her bir basamaktan diğer basamaklara elde aktarımı gereksinimini doğurabilir. Bu gereksinim de hesaplamanın uzun sürmesine neden olur. Bu nedenle ilk başlarda sayıların tüm katları hesaplanıp saklanırken, sonraki çalışmalarda, kolay hesaplanabilir katlar saklanmış ve diğer katlar bu kolay katlardan faydalanılarak hesaplanmıştır. Genellikle bu yaklaşımda iki farklı küme oluşturulup çarpan sayının rakamına göre bu iki kümeden alınan birer eleman toplanmıştır. Ayrıca, kısmi çarpımların hesabını veya bir sonraki aşamada indirgenmesini kolaylaştırmak amacıyla, sayılar yeniden kodlanmıştır. Bu yeniden kodlamaların dezavantajı, hem ilk başta hem de en sonda kodlama gereksinimidir. Ama ara basamaklarda elde edilecek avantaj ile bu dezavantaj, önemsiz hale gelebilmektedir.

3.3 Kısmi Çarpımların İndirgenmesi

Kısmi çarpımların indirgenmesi on tabanında toplama işlemidir. Sayısal sistemlerde iki tabanı kullanıldığından, on tabanında işlem yapılması ancak BCD veya benzeri

bir gösterimle mümkün olabilmektedir. Kullanılan gösterime göre iki tabanında toplama yapıldığında elde edilen sonuç istenilen sonucu yansıtmayabilir. Bu durumda düzeltme işlemi gerekmektedir (Örn. BCD için 6 eklemek).

Svoboda (Svoboda, 1969) tarafından önerilen toplayıcıda işaretli sayılar kullanılarak elde aktarımı gereksinimi ortadan kaldırılmıştır.

On tabanında toplama işlemi için önerilen en hızlı yöntem, direkt toplama yöntemidir (Schmookler and Weinberger, 1971). Bu yöntemde, toplanacak rakamların 0 ile 9 arasında olması gerçeğinden yararlanılarak mantıksal kapı seviyesinde hızlı bir devre önerilmiştir. Ancak burada önerilen yöntem özellikle iki rakamın toplanması için uygun iken çok sayıda rakamın toplanması için uygun değildir.

Ohtsuki (Ohtsuki *et al.*, 1987) ve Erle (Erle and Schulte, 2003) tarafından önerilen yapılarda, iki seviyeli (3:2) EST'ler döngü içerisinde kullanılarak kısmi çarpımlar toplanmıştır.

BCD rakamların toplanması sonucunda elde oluşması durumunda 6 eklenerek düzeltme yapılması gerekmektedir. Bu düzeltme yapılmadan bir sonraki aşamaya geçmek hatalara neden olabilir. Ancak düzeltme işlemi zaman kaybına yol açmaktadır. Bu soruna çözüm bulmak için Kenney, spekülatif toplayıcı yapısı önermiştir (Kenney and Schulte, 2004). Önerilen yapıda birden fazla rakam toplamında, ilk iki rakam EST ile toplanırken paralel olarak üçüncü rakama 6 eklenmektedir. Daha sonra EST çıkışındaki eldeye göre üçüncü rakamın kendisi veya 6 eklenmiş hali toplamaya dahil edilmektedir. Böylece düzeltme işlemi için gereken süre ortadan kaldırılmıştır. Ancak, Kenney tarafından önerilen yöntemde kurulan ağaç yapısı, toplanacak rakamların sayısı arttıkça gecikme ve alan başarımları açısından kullanışsız hale gelmektedir.

Lang tarafından önerilen çarpma yönteminde (Lang and Nannarelli, 2006) kolonlar EST kullanılarak indirgenmiştir. Ancak aynı yapıyı koruyan Dadda (Dadda and Nannarelli, 2008), kolon toplamları için daha önce önerdiği yöntemi (Dadda, 2007) uygulamıştır. Kolon toplamı için Dadda, toplanacak rakamların en fazla 9 olabilmesinden faydalanıp toplama işlemini önce iki tabanında yaparak ve daha sonra bu toplamı on tabanına çevirerek gerçekleştirmiştir. Bu sayede çok sayıda rakamın toplanması için uygun bir yapı oluşturmuştur.

Vazquez tarafından önerilen paralel çarpıcıda (Vazquez *et al.*, 2007) *BCD*4221 gösterimini kullanarak kolon indirgenmesinde de fayda sağlamıştır. Bu gösterim sayesinde iki tabanında kullanılan tam toplayıcılar (Full Adder) kullanılarak birden fazla rakam toplanmıştır. Ayrıca Vazquez başka bir çalışmasında (Vazquez *et al.*, 2010) kısmi çarpımların indirgenmesi için sayaç kullanmıştır. Kullanılan sayaçlar *BCD*4221 gösterimi için önerilmiştir ve en fazla 9 bit saymaktadır.

3.4 Elde Aktarımlı Son Toplam

Kısmi çarpımlar, önceki kısımlarda da anlatıldığı gibi genellikle toplam ve elde olarak iki vektöre indirgenirler. Bu artık gösterimden çarpım sonucunun BCD gösterimine EAT kullanılarak ulaşılır. Elde aktarımı en kötü duruma göre tasarlanır. Örneğin en sağdaki basamakta oluşan elde, en soldaki basamağa kadar etki edebilir. Bu sebeple elde aktarımı için hızlı yöntemler tercih edilir.

Lang (Lang and Nannarelli, 2006), toplam ve elde vektörlerinden aktarılacak eldeleri hesaplarken, bu işleme paralel olarak toplam ve elde vektörlerini 0 ve 1 ile toplamıştır. Elde hesabı bittikten sonra elde oluşup oluşmamasına bağlı olarak 0 veya 1 eklenmiş vektörlerden birinden seçim yapmıştır.

Vazquez (Vazquez *et al.*, 2010) ise daha önce yapmış olduğu çalışmada (Vazquez and Antelo, 2006) önerdiği dörtlü ağaç toplayıcı (Quarternary Tree Adder) ile son toplama işlemini gerçekleştirmiştir.

4. RAKAM ÇARPIMINA DAYALI PARALEL ÇARPMA

Bu bölümde Rakam Çarpımına Dayalı Paralel Çarpma (RDPÇ) yöntemi irdelenecektir. Bu yöntem, Vedic matematiğinin kuramlarından (sutra) biri olan Dikey ve Çapraz (Vertically and Cross-wise) Çarpma Algoritmasını temel alarak rakamların çapraz çarpımı ve bu çarpımların uygun şekilde ağırlıklandırılarak toplanması esasına dayanmaktadır. Bu yöntem sinyal işleme uygulamalarında kullanılmak üzere Thapliyal (Thapliyal and Srinivas, 2004) tarafından iki tabanında çarpım yapmak için önerilmiş ve uygulanmıştır. Bu yöntem, bu tez kapsamında kısaca Vedic olarak adlandırılmıştır. Vedic yönteminin BCD sayılar için uyarlanması, anlatıma yardımcı olması amacıyla 4 rakamlı iki BCD sayının çarpımı için Bölüm 4.1'de örneklendirilerek anlatılmıştır. Tez kapsamında nihai hedef, Vedic yönteminin 16 rakamlı iki BCD sayının çarpılması için uyarlanmasıyla elde edilen RDPÇ yöntemi ile donanım tasarımıdır. RDPÇ yöntemi ile ilgili detaylar Bölüm 4.2'de incelenmiştir.

4.1 Vedic Yönteminin 4 Rakam İçin Uyarlanması

Vedic yöntemi çarpan ve çarpılan sayının rakamlarının çapraz çarpımına dayanmaktadır. Bu bölümde Vedic yönteminin 4 rakamlı iki BCD sayının çarpımı için uyarlanması anlatılmaktadır. 4 rakamlı iki BCD sayı aşağıda gösterilen şekilde tanımlanabilir:

$A = a_4a_3a_2a_1$ ve $B = b_4b_3b_2b_1$

Burada a_i ve b_i birer BCD rakamı ifade etmektedir.

Thapliyal (Thapliyal and Srinivas, 2004) tarafından tanımlanan kısmi çarpımlar, A ve B sayıları için Eş. 4.1'de verilen şekilde tanımlanabilir:

$$CP_1 = a_1xb_1$$

$$CP_2 = a_2xb_1 + a_1xb_2$$

$$CP_3 = a_3xb_1 + a_2xb_2 + a_1xb_3$$

$$CP_4 = a_4xb_1 + a_3xb_2 + a_2xb_3 + a_1xb_4$$

$$CP_5 = a_4xb_2 + a_3xb_3 + a_2xb_4$$

$$CP_6 = a_4xb_3 + a_3xb_4$$

$$CP_7 = a_4 x b_4 \quad (4.1)$$

Bu eşitliklerde "*x*" BCD rakam çarpımını ifade etmektedir.
Her bir çapraz rakam çarpımı, $C_{i,j}$, şu şekilde tanımlanabilir:

$$C_{i,j} = a_i x b_j \quad (4.2)$$

Eş. 4.2'de verilen ifade, Eş. 4.1'de yerine koyulduğunda Eş. 4.3 elde edilir:

$$\begin{aligned}
CP_1 &= C_{1,1} \\
CP_2 &= C_{1,2} + C_{2,1} \\
CP_3 &= C_{1,3} + C_{2,2} + C_{3,1} \\
CP_4 &= C_{1,4} + C_{2,3} + C_{3,2} + C_{4,1} \\
CP_5 &= C_{2,4} + C_{3,3} + C_{4,2} \\
CP_6 &= C_{3,4} + C_{4,3} \\
CP_7 &= C_{4,4} \quad (4.3)
\end{aligned}$$

Çarpım sonucu, bu kısmi çarpımların kaydır ve ekle yöntemi ile toplanmasıyla elde edilir (Thapliyal and Srinivas, 2004). Ancak bu tez kapsamında, kısmi çarpımlar direkt olarak hesaplanmamaktadır. Bunun yerine, her bir kısmi çarpımda toplanması gereken çapraz rakam çarpımları 0 ile 81 arasındaki tamsayılardan oluşan kümede yer aldıkları için onlar ve birler basamağında yer alan rakamlar ayrılarak ilgili kolona yerleştirilmektedir. Bu yerleştirme yapıldığında piramit şeklinde bir yapı elde edilir (Bkz. Şekil 4.1). Çapraz rakam çarpımlarının ($C_{i,j}$) onlar ve birler basamağında yer alan rakamlar, sırayla "O" ve "B" indisleri ile ifade edilmektedir. Bu yöntemin sonucunda elde edilen yapı, daha önce ardışık çarpma yöntemi için yapılan bir çalışma ile benzerlik taşımaktadır (James *et al.*, 2008*a*). Bu tez kapsamında ise yapı, paralel çarpma yöntemi için oluşturulmuştur.

Şekil 4.1'de gösterilen şekilde yerleştirme yapıldıktan sonra, her bir kolonda yer alan rakamlar toplanır. Burada her bir rakam toplamı, "*t*" en sağdaki kolondan başlayarak kolon numarasını göstermek üzere, K_t ile ifade edilmektedir. İlk ve son kolonlar

dışındaki tüm kolonlarda birden fazla rakam toplanmaktadır. Bu sebeple her bir K_t toplamı için on tabanında birden fazla BCD rakamın toplanmasını gerçekleştirecek bir tasarıma ihtiyaç duyulmaktadır.

Birden fazla rakamın on tabanında toplanması, beraberinde düzeltme işlemi gerektirdiğinden ve çarpma yöntemlerinin başarımını doğrudan etkilediğinden literatürde bu konuda çeşitli çalışmalar yapılmıştır (Kenney *et al.*, 2004), (Dadda, 2007), (Vazquez *et al.*, 2007). Dadda (Dadda, 2007) tarafından yapılan çalışmada, rakamlar önce iki tabanında toplanmış ve daha sonra BCD gösterimine çevrilmiştir. Toplanacak rakamların sayısı fazla olduğunda Dadda'nın önerdiği yöntem, diğer yöntemlere göre daha kullanışlıdır. Ayrıca Dadda'nın çalışması bu tez kapsamında yapılan çalışmada elde edilen kolon toplamlarının sınırlı olması durumunda da bu sınır değerlerinin kullanılmasına imkan sağlamaktadır. Bu nedenlerle bu tez kapsamında Dadda tarafından önerilen yöntem kullanılmıştır. Yöntem ile ilgili detaylar Bölüm 4.2.2'de irdelenecektir.

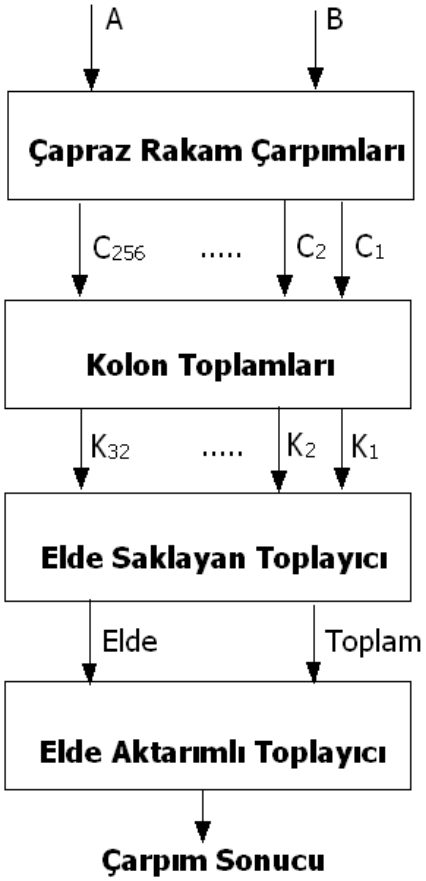
				a_4 b_4	a_3 b_3	a_2 b_2	a_1 b_1
			$C^O_{1,4}$	$C^B_{1,4}$			
			$C^B_{2,4}$	$C^O_{1,3}$			
		$C^O_{2,4}$	$C^O_{2,3}$	$C^B_{2,3}$	$C^B_{1,3}$		
		$C^B_{3,4}$	$C^B_{3,3}$	$C^O_{2,2}$	$C^O_{1,2}$		
	$C^O_{3,4}$	$C^O_{3,3}$	$C^O_{3,2}$	$C^B_{3,2}$	$C^B_{2,2}$	$C^B_{1,2}$	
	$C^B_{4,4}$	$C^B_{4,3}$	$C^B_{4,2}$	$C^O_{3,1}$	$C^O_{2,1}$	$C^O_{1,1}$	
$C^O_{4,4}$	$C^O_{4,3}$	$C^O_{4,2}$	$C^O_{4,1}$	$C^B_{4,1}$	$C^B_{3,1}$	$C^B_{2,1}$	$C^B_{1,1}$
K_8	K_7	K_6	K_5	K_4	K_3	K_2	K_1

Şekil 4.1. Dört Rakam İçin Vedic Algoritması

Elde edilen kolon toplamları (K_t), ağırlıklarına göre konumlandırılarak EST ile toplam ve elde vektörlerine indirgenir. Bu vektörler hızlı bir EAT ile toplanarak çarpım sonucuna ulaşılır.

4.2 Vedic Yönteminin 16 Rakam İçin Uyarlanması

Vedic yönteminin BCD rakamlı sayılar için uyarlanması ve bu uyarlamanın paralel çarpma yönteminde kullanılması ile oluşturulan yöntem, RDPÇ yöntemi olarak adlandırılmaktadır. RDPÇ yöntemi dört ana basamaktan oluşmaktadır. Bu basamaklar, Şekil 4.2'de bloklarla gösterilmiştir.

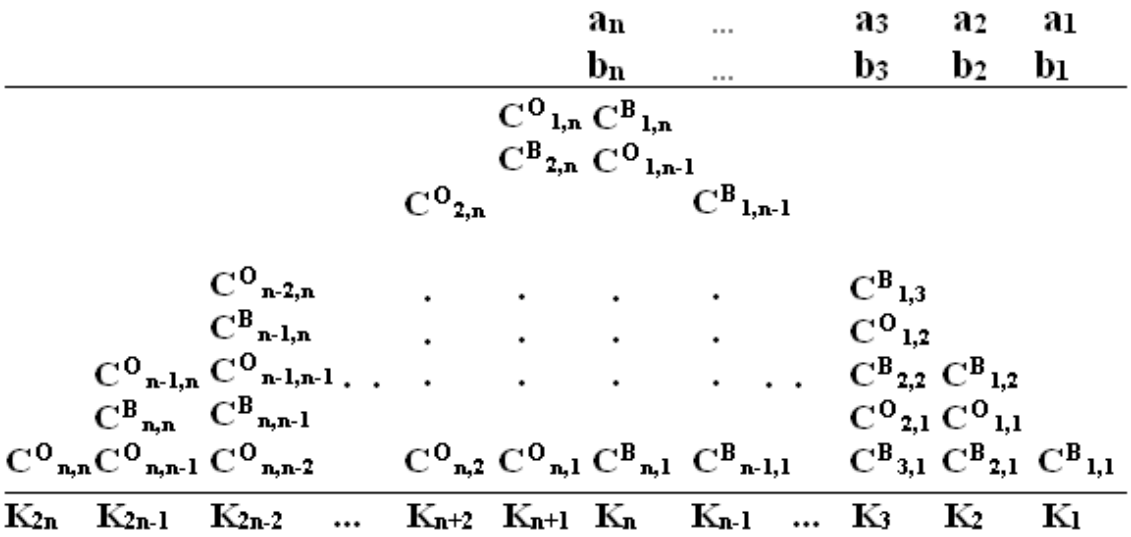


Şekil 4.2. RDPÇ Blok Şeması

Çapraz rakam çarpımları, paralel olarak hesaplanır ve ağırlıklarına göre yerleştirilir. Daha sonra her bir kolonda birden fazla BCD rakam hızlı bir biçimde toplanır. Bu kolon toplamları EST kullanılarak birleştirilir. Son basamakta ise EAT ile toplam yapılır ve çarpım sonucuna ulaşılır.

Bu yöntem, rakamların çapraz çarpımı ve bu çarpımların uygun şekilde ağırlıklandırılarak toplanması esasına dayanmaktadır. İlk aşamada, çarpılacak olan 16 rakamlı iki sayının (A ve B) çapraz rakam çarpımları hesaplanmaktadır. On tabanında

çarpım yapılacağından, rakamlar $0, 1, \ldots, 9$ kümesinin elemanlarından oluşmaktadır. Rakam çarpım sonucu en fazla 81 değerini alabilmektedir ve bu çarpma işlemi tasarlanan mantıksal devre ile gerçekleştirilmektedir. RDPÇ yönteminin alan başarımını en fazla etkileyen birim, rakam çarpım birimidir. Çünkü RDPÇ yönteminde 256 adet paralel çapraz rakam çarpım işlemi vardır ve bu nedenle rakam çarpım biriminden 256 adet kullanılmaktadır. RDPÇ yönteminde çapraz rakam çarpımlarının her birinin birler ve onlar basamağındaki rakamlar ayrılarak piramit yapısında yerleştirilmektedir. *n* rakamlı iki sayının çarpılması işleminde kolonlara yerleştirilen rakamlar Şekil 4.3'te verilmiştir.



Şekil 4.3. n Rakam İçin Vedic Algoritması

Sonraki aşamada, bu yapıda yer alan her bir kolondaki tüm rakamlar toplanmaktadır. On tabanında gösterilen (BCD) birden fazla rakamın iki tabanında toplanması beraberinde düzeltme yapılması sorununu da getirmektedir. Düzeltme yapılmasının sebebi iki tabanında 4 bitle ifade edilebilen rakamların *{0, 1, ..E, F}* kümesinde olabilmesidir. Ancak ikili kodlanmış onlu (BCD) gösterimde *A,B,..F* değerlerinin karşılıkları 6 eklenerek hesaplanmalıdır. RDPÇ yönteminin gecikme başarımını en fazla etkileyen basamaklardan birisi, kolon toplamlarının paralel olarak indirgenmesidir.

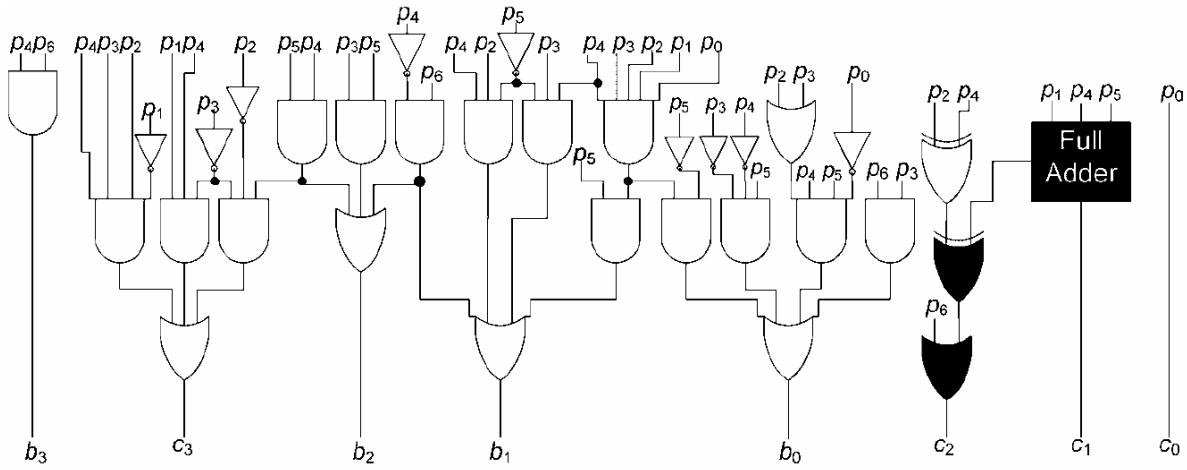
Bir sonraki aşamada EST kullanılmaktadır. Bu toplayıcı ile bir önceki aşamada hesaplanan kolon toplamları toplam ve elde vektörlerine dönüştürülmektedir. Örneğin ardışık iki kolonun toplamından ilkinin 127 ve ikincisinin 189 olduğunu varsaya-

lım. Bu iki kolon toplamının birer basamak kaydırılarak toplanması gerekmektedir. Başka bir ifadeyle, bu iki kolonun toplamı 1890 + 127 olmalıdır. EST'nin çıkışlarını, (*elde*, *toplam*) olarak tanımladığımızda, bu örnekte aynı basamakta yer alan 9 + 2 ve 8 + 1 işlemlerinin sonucu, sırasıyla (1, 1) ve (0, 9) şeklinde ifade edilmektedir.

Son aşamada toplam ve elde vektörlerinden çarpım sonucuna ulaşmak için elde aktarımlı toplama işlemi gerekmektedir. Elde aktarımı için en kötü durumdan yola çıkılarak en sağdaki basamaktan en soldaki basamağa kadar eldenin aktarılması ihtimali göz önünde bulundurularak aktarım yapılmaktadır.

4.2.1 On Tabanında Rakam Çarpımı

On tabanında rakam çarpım birimi için ilk olarak, Jaberipur (Jaberipur and Kaivani, 2007) yöntemi irdelenmiştir. Bu yöntemde, rakamlar önce iki tabanında çarpılmış ve daha sonra on tabanına çevrim işlemi önerilmiştir. Ancak, incelemelerimiz sırasında iki tabanından on tabanına çevrim için önerilen devrenin (Şekil 4.4) hatalı olduğu tespit edilmiştir.



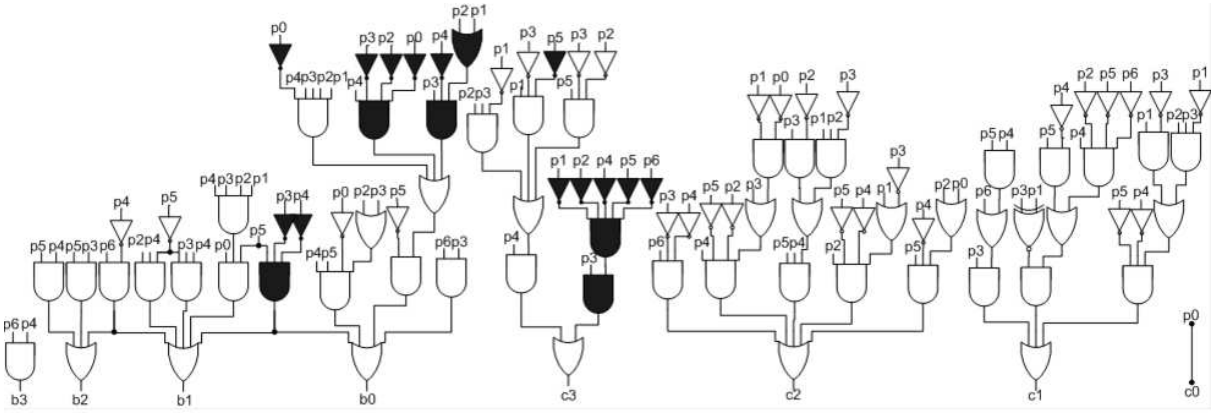
Şekil 4.4. İki Tabanından On Tabanına Çevirici Devresi (Jaberipur and Kaivani, 2007)

Bu devrenin hatalı bitleri Çizelge 4.1'de verilmiştir. Çizelge 4.1'de, çevrilecek olan çarpımın iki ve on altı tabanındaki karşılıkları verilmiştir. Ayrıca çevrim sonucunda elde edilmesi gereken BCD karşılığı verilmiştir. BCD karşılığının bitlerinden hatalı olanlar ve bu bitlerin olması gereken değerleri "Bit Hatası" sütununda belirtilmiştir.

Çizelge 4.1. Jaberipur Tarafından Önerilen Devrenin Hatalı Olduğu Durumlar

p_6	p_5	p_4	p_3	p_2	p_1	p_0	**Hex**	**BCD**	**Bit Hatası**
0	0	0	1	0	1	0	A	10	$b_0 = 1$
0	0	0	1	1	0	0	C	12	$b_0 = 1$
0	0	0	1	1	1	0	E	14	$b_0 = 1$
0	0	0	1	1	1	1	F	15	$b_0 = 1$
0	0	1	0	0	0	0	10	16	$b_0 = 1$
0	0	1	0	0	1	0	12	18	$b_0 = 1$
0	0	1	1	1	1	0	1E	30	$b_0 = 1$
0	1	0	0	0	0	0	20	32	$b_1 = 1$
0	1	0	0	0	1	1	23	35	$b_1 = 1$
0	1	0	0	1	0	0	24	36	$b_1 = 1$
0	0	0	1	0	0	0	8	8	$c_3 = 1$
0	0	0	1	0	0	1	9	9	$c_3 = 1$
0	1	1	0	1	1	0	36	54	$c_3 = 0$

Bu hataların düzeltilmesi için çarpım tablosundan yararlanılarak gerekli ekleme ve düzeltmeler gerçekleştirilmiştir. Gerekli düzenlemeler yapıldıktan sonra elde ettiğimiz çevirici devresi Şekil 4.5'te verilmiştir.



Şekil 4.5. Düzeltilmiş İki Tabanından On Tabanına Çevirici Devresi

Rakam çarpım biriminin, tasarımda kullanılma sayısı fazla olduğundan daha az alan kaplayacak bir yöntem uygulanması uygun görülmüştür. Bu sebeple ikinci yöntem olarak çarpılacak rakamların hangi rakam olduğunu belirlemek için Eş. 4.4'te gösterilen değişkenler tanımlanmıştır. Çarpılacak rakamlar *a* ve *b* olarak tanımlanmış ve bu rakamların İkili Kodlanmış Onlu (BCD) formatında olduğu kabul edilmiştir. Ayrıca her bir a_i ve b_i rakamların bir bitini ifade etmektedir.

$$p_{a9} = a_4.\bar{a}_3.\bar{a}_2.a_1 \quad p_{b9} = b_4.\bar{b}_3.\bar{b}_2.b_1$$

$$p_{a8} = a_4.\bar{a}_3.\bar{a}_2.\bar{a}_1 \quad p_{b8} = b_4.\bar{b}_3.\bar{b}_2.\bar{b}_1$$

$$p_{a7} = \bar{a}_4.a_3.a_2.a_1 \quad p_{b7} = \bar{b}_4.b_3.b_2.b_1$$

$$p_{a6} = \bar{a}_4.a_3.a_2.\bar{a}_1 \quad p_{b6} = \bar{b}_4.b_3.b_2.\bar{b}_1$$

$$p_{a5} = \bar{a}_4.a_3.\bar{a}_2.a_1 \quad p_{b5} = \bar{b}_4.b_3.\bar{b}_2.b_1$$

$$p_{a4} = \bar{a}_4.a_3.\bar{a}_2.\bar{a}_1 \quad p_{b4} = \bar{b}_4.b_3.\bar{b}_2.\bar{b}_1$$

$$p_{a3} = \bar{a}_4.\bar{a}_3.a_2.a_1 \quad p_{b3} = \bar{b}_4.\bar{b}_3.b_2.b_1$$

$$p_{a2} = \bar{a}_4.\bar{a}_3.a_2.\bar{a}_1 \quad p_{b2} = \bar{b}_4.\bar{b}_3.b_2.\bar{b}_1$$

$$p_{a1} = \bar{a}_4.\bar{a}_3.\bar{a}_2.a_1 \quad p_{b1} = \bar{b}_4.\bar{b}_3.\bar{b}_2.b_1 \qquad (4.4)$$

Çarpma sonucu $c = c_7c_6c_5c_4c_3c_2c_1c_0$ olarak tanımlanmış ve çarpma sonucunun bitleri çarpım tablosu yardımı ile tek tek belirlenerek aşağıdaki mantıksal denklemler elde edilmiştir:

$$c_7 = p_{a9}.p_{b9}$$

$$c_6 = (p_{a8} + p_{a9}).p_{b5} + (p_{a7} + p_{a8} + p_{a9}).p_{b6} + (p_{a6} + p_{a7} + p_{a8} + p_{a9}).p_{b7} + (p_{a5} + p_{a6} + p_{a7} + p_{a8} + p_{a9}).p_{b8} + (p_{a5} + p_{a6} + p_{a7} + p_{a8}).p_{b9}$$

$$c_5 = (p_{a7} + p_{a8} + p_{a9}).p_{b3} + (p_{a5} + p_{a6} + p_{a7} + p_{a8} + p_{a9}).p_{b4} + (p_{a4} + p_{a5} + p_{a6} + p_{a7}).p_{b5} + (p_{a4} + p_{a5} + p_{a6}).p_{b6} + (p_{a3} + p_{a4} + p_{a5} + p_{a9}).p_{b7} + (p_{a3} + p_{a4} + p_{a8} + p_{a9}).p_{b8} + (p_{a3} + p_{a4} + p_{a7} + p_{a8}).p_{b9}$$

$$c_4 = (p_{a5} + p_{a6} + p_{a7} + p_{a8} + p_{a9}).p_{b2} + (p_{a4} + p_{a5} + p_{a6}).p_{b3} + (p_{a3} + p_{a4} + p_{a8}$$

$$+ p_{a9}).p_{b4} + (p_{a2} + p_{a3} + p_{a6} + p_{a7}).p_{b5} + (p_{a2} + p_{a3} + p_{a5} + p_{a6} + p_{a9}).p_{b6}$$

$$+ (p_{a2} + p_{a5} + p_{a8}).p_{b7} + (p_{a2} + p_{a4} + p_{a7} + p_{a9}).p_{b8} + (p_{a2} + p_{a4} + p_{a6} + p_{a8}).p_{b9}$$

$$c_3 = (p_{a8} + p_{a9}).p_{b1} + (p_{a4} + p_{a9}).p_{b2} + (p_{a3} + p_{a6}).p_{b3} + (p_{a2} + p_{a7}).p_{b4}$$

$$+ (p_{a3} + p_{a8}).p_{b6} + (p_{a4} + p_{a7}).p_{b7} + (p_{a1} + p_{a6}).p_{b8} + (p_{a1} + p_{a2}).p_{b9}$$

$$c_2 = (p_{a4} + p_{a5} + p_{a6} + p_{a7}).p_{b1} + (p_{a2} + p_{a3} + p_{a7} + p_{a8}).p_{b2} + (p_{a2} + p_{a5} + p_{a8}$$

$$+ p_{a9}).p_{b3} + (p_{a1} + p_{a4} + p_{a6} + p_{a9}).p_{b4} + (p_{a1} + p_{a3} + p_{a5} + p_{a7} + p_{a9}).p_{b5}$$

$$+ (p_{a1} + p_{a4} + p_{a6} + p_{a9}).p_{b6} + (p_{a1} + p_{a2} + p_{a5} + p_{a8}).p_{b7} + (p_{a2} + p_{a3} + p_{a7}$$

$$+ p_{a8}).p_{b8} + (p_{a3} + p_{a4} + p_{a5} + p_{a6}).p_{b9}$$

$$c_1 = (p_{a2} + p_{a3} + p_{a6} + p_{a7}).p_{b1} + (p_{a1} + p_{a3} + p_{a6} + p_{a8}).p_{b2} + (p_{a1} + p_{a2} + p_{a4}$$

$$+ p_{a9}).p_{b3} + (p_{a3} + p_{a4} + p_{a8} + p_{a9}).p_{b4} + (p_{a1} + p_{a2} + p_{a6} + p_{a7}).p_{b6} + (p_{a1}$$

$$+ p_{a6} + p_{a8} + p_{a9}).p_{b7} + (p_{a2} + p_{a4} + p_{a7} + p_{a9}).p_{b8} + (p_{a3} + p_{a4} + p_{a7} + p_{a8}).p_{b9}$$

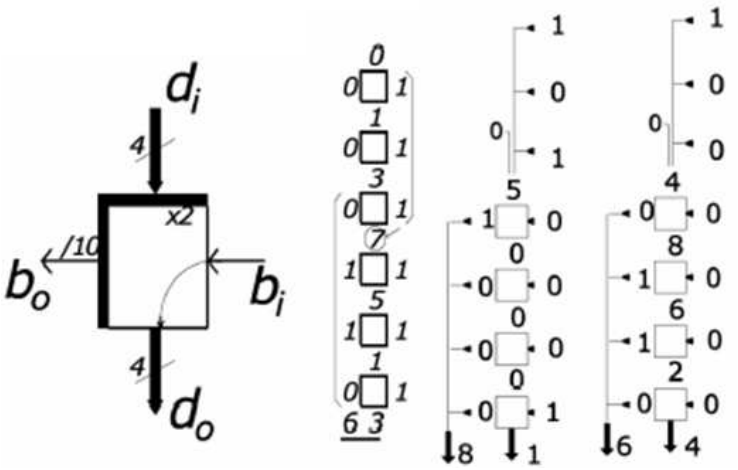
$$c_0 = a_0.b_0 \tag{4.5}$$

Eş. 4.5'te verilen denklemlerin oluşturulma mantığı şöyledir: c_i bitinin hangi rakamlar çarpıldığında 1 olduğu tespit edilir. İlgili rakamın oluşması olarak tanımlanan p_{ai} ve p_{bi}, mantıksal *VE* kapısına sokulur. Her bir bit için olası tüm durumlar mantıksal *VEYA* kapısına sokulur ve sonuca ulaşılır. Örneğin, $a = 8$ ve $b = 6$ ise p_{a8} ve p_{b6} haricindeki tüm bitler sıfır olur. Böylece $p_{a8}.p_{b6} = 1$ olacaktır ve bu ifadeyi barındıran c_6 ve c_3 haricindeki tüm çıkış bitleri sıfır olacaktır. Elde edilmesi gereken değer 01001000 = 48 (BCD) bulunmuş olacaktır. Bu yöntem ile Ueda (Ueda, 1995) tarafından önerilen ve programlanabilir mantıksal dizi (Programmable Logical Array) ile gerçekleştirilen yöntem aynı temele dayanmaktadır.

4.2.2 Kolon Toplamlarının İndirgenmesi

Şekil 4.3'te gösterilen şekilde, aynı ağırlıktaki rakamlar aynı kolonlara yerleştirilir. Her bir kolonda yer alan rakamların toplanması için Dadda (Dadda, 2007) tarafından önerilen yöntem kullanılmaktadır. Bu yöntemde rakamlar iki tabanında toplanıp daha sonra Nicoud (Nicoud, 1971) yöntemi ile iki tabanından on tabanına çevrim işlemi ile sonuca ulaşılmıştır. İki tabanından on tabanına çevrim için Nicoud

(Nicoud, 1971) tarafından önerilen yöntem ile üç farklı sayının çevrimi Şekil 4.6'da örnek olarak verilmiştir.



Şekil 4.6. Nicoud Yöntemi İle İki Tabanından On Tabanına Çevrim (Dadda, 2007)

Şekil 4.6'da verilen kutuda d_i giriş rakamını, b_i giriş bitini, d_o çıkış rakamını ve b_o çıkış bitini göstermektedir. Her kutuya gelen rakam 2 ile çarpılmakta ve giriş biti ile toplanmaktadır. Gelen rakam 5 veya daha büyükse 2 ile çarpma sonucunda elde oluşacaktır; bu elde çıkış biti olarak alınmaktadır. Çıkış rakamı ise $2 * d_i + b_i (mod 10)$ olarak elde edilmektedir. Son çıkan rakam birler basamağını, eldelerden oluşan sayı ise onlar basamağını oluşturur. Eğer eldelerden oluşan sayı 10'dan büyükse aynı işlem elde sayısı için de uygulanır ve sırayla onlar, yüzler, vb. basamağındaki rakamlar bulunmuş olur. Şekil 4.3'ten hareketle her bir kolon toplamı aşağıda verilen eşitlikle ifade edilebilir:

$$K_t = \begin{cases} C^B_{1,1} & , t = 1 \\ \sum_{k=1}^{t} C^B_{k,(t-k+1)} + \sum_{k=1}^{t-1} C^O_{k,(t-k)} & , 2 \le t \le n \\ \sum_{k=t-n+1}^{n} C^B_{k,(t-k+1)} + \sum_{k=t-n}^{n} C^O_{k,(t-k)} & , n+1 \le t \le 2n-1 \\ C^O_{n,n} & , t = 2n \end{cases} \quad (4.6)$$

Şekil 4.3'te gösterilen şekilde her bir kolona yerleştirilen rakamlardan *B* indisli rakamlar en fazla 9, *O* indisli rakamlar en fazla 8 değerini alabilir. Piramidin sağ ya-

rısında yer alan kolonlarda rakamların sayısı ($2t-1$)'dir. Bu kolonlarda yer alan B indisli rakamların sayısı t ve O indisli rakamların sayısı ($t-1$)'dir. Piramidin sol yarısında yer alan kolonlar için $v = 2n+1-t$ değişkeni tanımlarsak, rakam sayısı ($2v-1$) olur. Sol taraftaki kolonlarda B indisli rakamların sayısı $v-1$ ve O indisli rakamların sayısı v'dir. Buna göre, t numaralı kolon toplamının en büyük değeri (K_{tm}):

$$K_{tm} = \begin{cases} 9*t+8*(t-1) & , 2 \leq t \leq n \\ 9*(v-1)+8*v & , n+1 \leq t \leq 2n-1, 2 \leq v \leq n \end{cases} \tag{4.7}$$

olarak bulunur. Ancak burada aynı kolonda yer alan O ve B indisli farklı rakamlar ortak çarpana sahip olabilmektedir (örn. $C^B_{1,3}$ ve $C^O_{1,2}$ rakamlarında a_1 ortak çarpandır). Aynı kolonda yer alan O ve B indisli rakamların birbirinden tamamen bağımsız olmaması, Eş. 4.7'de bulunan değerlerden daha düşük bir üst sınır olabileceği fikrini doğurmuştur. Bu fikir Bölüm 4.2.3'te detaylı bir biçimde irdelenmiştir.

4.2.3 Kolon Toplamlarının Sınırlı Olmasının İncelenmesi

RDPÇ yönteminde, çarpılacak sayıların rakamları a_i ve b_j olarak tanımlandığında, çapraz çarpımları Şekil 4.3'te gösterildiği gibi rakamlarına ayrılarak piramit biçiminde yerleştirilir. İlk ve son kolonlar haricindeki her bir kolonda, onlar basamağını ifade eden "O" indisli ve birler basamağını ifade eden "B" indisli rakamlar yer almaktadır. Kolonlarda yer alan bu rakamlar birbirinden tamamen bağımsız değildir. Farklı rakam çarpım sonuçlarından elde edilen ve aynı kolona yerleştirilen O ve B indisli rakamlar ortak çarpana sahiptirler (örn. $C^B_{1,3}$ ve $C^O_{1,2}$). Bu rakamların birbiriyle ilişkili olması, aynı anda tüm O indisli rakamların 8 ve B indisli rakamların 9 olamayacağı sonucunu doğurur. Bu noktadan hareketle kolon toplamlarının alabileceği en büyük değerin sınırlı olduğu öngörülmüştür. Bu kısımda bu sınırların deneysel yöntemle analizi irdelenecektir.

RDPÇ yöntemi ile n rakamlı iki sayının çarpımında kolon toplamlarının en büyük değerlerinin hesaplanmasında kullanılmak üzere en kalabalık kolon toplamları Bölüm 4.2.3.1'de irdelenecektir. Elde edilen sonuçlar Bölüm 4.2.3.2'de 16 rakam için kullanılacaktır.

4.2.3.1 En Kalabalık Kolonların Toplamı

Piramit yapısında sağ ve sol taraftaki iki en kalabalık kolonun toplamının (EKKT) alabileceği en büyük değerler sırasıyla $EKKTR_n$ ve $EKKTS_n$ olarak adlandırılmıştır. $EKKTR_n$ ve $EKKTS_n$ için Şekil 4.3'ten hareketle aşağıdaki denklemler yazılabilir:

$$EKKTR_n = max(\sum_{k=1}^{n} C^B_{k,(n-k+1)} + \sum_{k=1}^{n-1} C^O_{k,(n-k)}) \tag{4.8}$$

$$EKKTS_n = max(\sum_{k=2}^{n} C^B_{k,(n-k+2)} + \sum_{k=1}^{n} C^O_{k,(n-k+1)}) \tag{4.9}$$

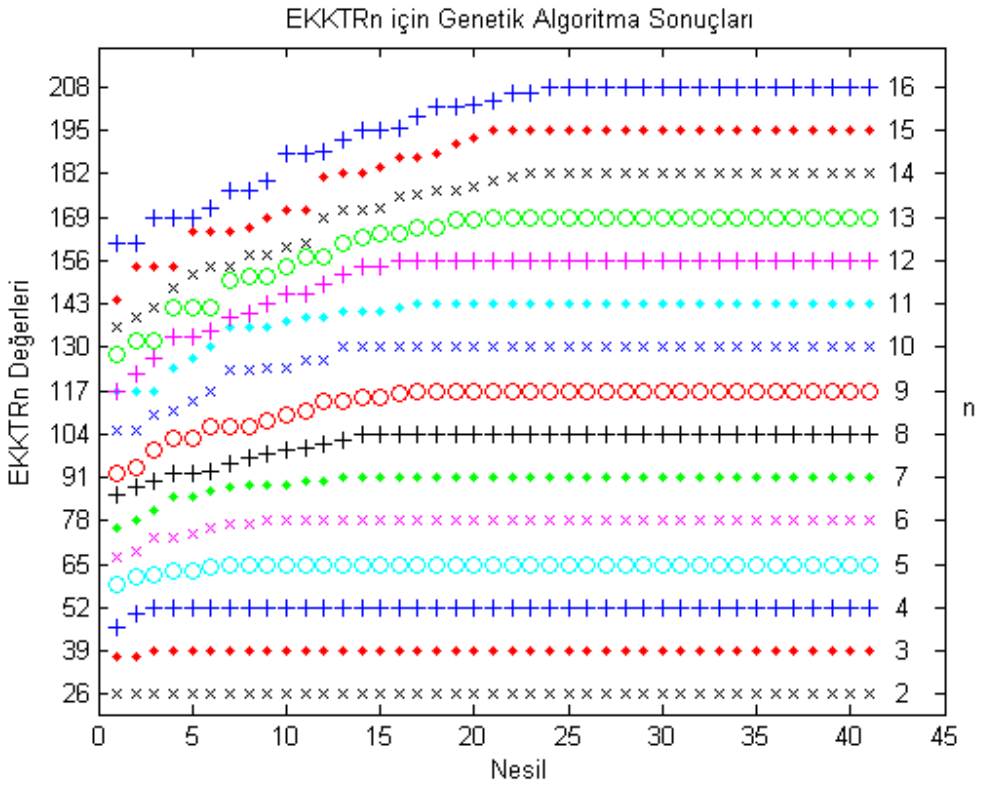
Çapraz rakam çarpımlarının onlar ve birler basamağında yer alan rakamlar (C^O_{ij} ve C^B_{ij}) aşağıdaki şekilde ifade edilebilir:

$$C^B_{i,j} = C_{i,j}(mod\,10), C^B_{i,j} \in 0, 1, \ldots, 9$$

$$C^O_{i,j} = \lfloor C_{i,j}/10 \rfloor, C^O_{i,j} \in 0, 1, \ldots, 8$$

Burada "$mod\,10$" işlemi 10 tabanına göre kalanı ve "$\lfloor\ \rfloor$" işlemi en yakın küçük tamsayı değere yuvarlamayı ifade etmektedir. Bu ifadelerin Eş. 4.8 ve Eş. 4.9'da yerine konulmasıyla bu eşitlikler birer eniyileme problemine dönüşmektedir. Bu sebeple, bu eşitliklerin çözümü için MATLAB Genetic Algorithm (GA) Toolbox (Chipperfield and Fleming, 1995) aracı yardımı ile benzetim yapılmıştır.

Sağ taraftaki en kalabalık kolon toplamının maksimum değerini ($EKKTR_n$) bulmak için GA aracında nüfus 6000 olarak seçilmiş ve diğer parametrelerin ön tanımlı değerleri değiştirilmemiştir. GA aracında, problemin değişkenleri 0-9 aralığındaki tamsayılardan oluştuğu için bireylerin de 0-9 aralığında olması sağlanmıştır. Bunu sağlamak için nüfus işlevinde gerekli değişiklikler yapılmıştır. Ayrıca, mutasyon işlevinin de mutasyon aşamasında tamsayı üretmesi için gerekli değişiklikler yapılmıştır. Eş. 4.8'de verilen eşitliğin negatifi, maliyet işlevi (cost function) olarak alınmış ve GA aracı ile en küçük değeri bulunmuştur. Elde edilen benzetim sonuçları Şekil 4.7'de gösterilmiştir.



Şekil 4.7. Sağ Taraftaki En Kalabalık Kolon Toplamı

GA aracında, deneysel bir yöntem olan genetik algoritma kullandığından, bulunan sonuçlar global minimum veya maksimum noktası olmayabilir (Haupt and Haupt, 2004). Bu nedenle $2, 3, 4, 5$ ve 6 rakamlı sayılar için kapsamlı yineleme (exhaustive iteration) yöntemi ile tüm sayılar çarpılarak problemin bu durumlar için kesin çözümü bulunmuştur.

n rakamlı iki sayının çarpılması işleminde, sağ taraftaki en kalabalık kolonun toplamını maksimum yapan sayı çiftleri ($a_n a_{n-1}...a_2 a_1, b_n b_{n-1}...b_2 b_1$) biçiminde tanımlanmıştır. Kapsamlı yineleme ile elde edilen en büyük değerler ve bu değerlere ulaşılan sayı çiftleri Çizelge 4.2'de verilmiştir.

Kapsamlı yineleme sonuçları ile GA aracı benzetim sonuçlarının birbiriyle tam uyumlu olduğu gözlenmiştir. Elde edilen sonuçlarda bu sayı çiftlerinin hepsinde en soldaki rakamın 1 ve en sağdaki rakamın 9 olduğu gözlenmiştir. Ayrıca $EKKTR_n$ değerlerinin, çarpılan sayıların rakam sayısının 13 katına eşit (13*n) olduğu gözlenmiştir.

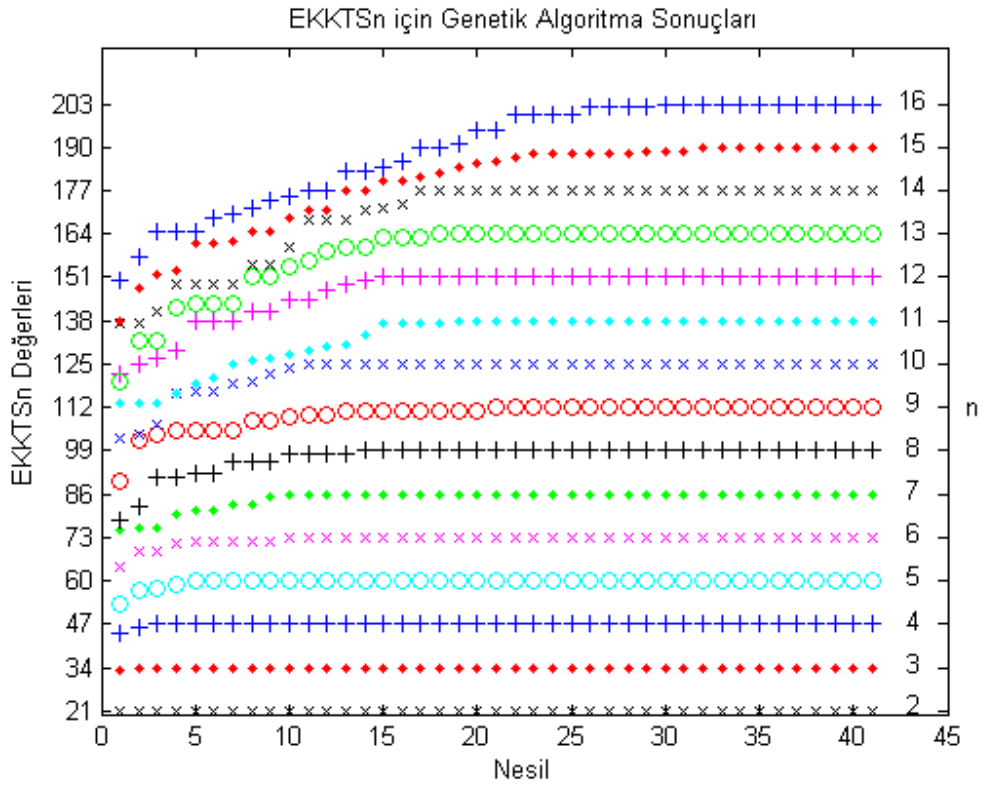
Çizelge 4.2. $EKKTR_n$ ve Karşılık Gelen Sayı Çiftleri İçin Kapsamlı Yineleme Sonuçları

n	$EKKTR_n$	**Sayı Çiftleri**
2	26	(19,19)
3	39	(179,179)
4	52	(1769,1869),(1779,1779), (1879,1769),(1919,1919)
5	65	(17679,17879),(17769,18779),(17779,17779), (17879,17679),(17919,19179),(18769,18769), (18779,17769),(19179,17919)
6	78	(176769,187879),(176779,177879),(176879,176879), (176919,191879),(177679,178779),(177769,187779), (177779,177779),(177879,176779),(177919,191779), (178769,187679),(178779,177679),(179179,179179), (187679,178769),(187769,187769),(187779,177769), (187879,176769),(187919,191769),(191769,187919), (191779,177919),(191879,176919),(191919,191919)

Sol taraftaki en kalabalık kolon toplamının en büyük değerini ($EKKTS_n$) bulmak için, $EKKTR_n$'de izlenen adımlar tekrarlanmıştır. Buna göre, Eş. 4.9'da verilen eşitliğin negatifi, maliyet işlevi olarak alınmış ve GA aracı ile en küçük değeri bulunmuştur. Elde edilen benzetim sonuçları Şekil 4.8'de gösterilmiştir.

$EKKTS_n$ için elde edilen benzetim sonuçlarının doğrulanması amacıyla, kapsamlı yineleme yöntemi ile $2, 3, 4, 5$ ve 6 rakamlı tüm sayılar çarpılmıştır. Elde edilen en büyük değerler ve bu değerlere ulaşılan sayı çiftleri Çizelge 4.3'te verilmiştir. $EKKTS_n$ değerine, 6 rakamlı sayılar için 28 farklı sayı çifti ile ulaşılmıştır; ancak bu çiftler Çizelge 4.3'te gösterilmemiştir. Kapsamlı yineleme sonuçları ile GA aracı benzetim sonuçlarının birbiriyle tam uyumlu olduğu gözlenmiştir. Elde edilen sonuçlara göre bu sayı çiftlerinin hepsinde en sağdaki rakamın 9 olduğu gözlenmiştir.

Kapsamlı yineleme sonuçlarında göre, $n \geq 3$ için $EKKTR_n$ değerlerine ulaşılan sayı çiftleri kümesinin her birinde $(177 \ldots 79, 177 \ldots 79)$ biçiminde çözüm olduğu gözlenmiştir. Bu noktadan hareketle, $7 \leq n \leq 16$ için bu biçimdeki tüm sayılar çarpılarak EKKT değerleri bulunmuştur. Elde edilen sonuçların da GA aracı benzetim sonuçları



Şekil 4.8. Sol Taraftaki En Kalabalık Kolon Toplamı

Çizelge 4.3. *EKKTS*$_n$ ve Karşılık Gelen Sayı Çiftleri İçin Kapsamlı Yineleme Sonuçları

n	*EKKTS*$_n$	**Sayı Çiftleri**
2	21	(79,79)
3	34	(769,869),(779,779),(879,769),(919,919)
4	48	(7679,7879),(7769,8779),(7779,7779),(7879,7679), (7919,9179),(8769,8769),(8779,7769),(9179,7919)
5	60	(76769,87879),(76779,77879),(76879,76879), (76919,91879),(77679,78779),(77769,87779), (77779,77779),(77879,76779),(77919,91779), (78769,87679),(78779,77679),(79179,79179), (87679,78769),(87769,87769),(87779,77769), (87879,76769),(87919,91769),(91769,87919), (91779,77919),(91879,76919),(91919,91919)
6	73	*

ile tam uyumlu olduğu gözlenmiştir.

Sağ ve sol taraftaki en kalabalık kolon toplamları için elde edilen sonuçlar karşılaştırıldığında, $EKKTS_n$ değerini sağlayan sayı çiftleri ile $EKKTR_n$ değerini sağlayan sayı çiftleri arasında ilişki olduğu sonucuna varılmıştır. Bu sonuca göre n rakamlı sayılar için $EKKTR_n$ değerini sağlayan sayı çiftlerinin en soldaki rakamı kaldırıldığında, $n-1$ rakamlı sayılar için $EKKTS_{n-1}$ değerini sağlayan sayı çiftlerine ulaşıldığı gözlenmiştir. Ayrıca $EKKTR_n$ ve $EKKTS_{n-1}$ sayısal değerleri arasında Eş. 4.10'da verilen bağıntının kurulabileceği sonucuna ulaşılmıştır.

$$EKKTS_{n-1} = EKKTR_n - 18 \tag{4.10}$$

4.2.3.2 EKKT Değerlerinin 16 Rakamlı Sayıların Çarpımında Kullanılması

RDPÇ yöntemi ile 16 rakamlı iki sayının çarpımında her bir kolon toplamı K_t ile ifade edildiğinde, bu toplamların en büyük değerlerini bulmak için Bölüm 4.2.3.1'de elde edilen sonuçlardan faydalanılmıştır. Şekil 4.3'te n yerine 16 konulduğunda elde edilen piramit yapısında sağ tarafta yer alan ve kolon numarası t olan kolon toplamının en büyük değerinin (K_{tm}), Bölüm 4.2.3.1'de elde edilen $EKKTR_t$ değerine eşit olduğu öngörülmektedir. Örneğin; n = 16 ve t = 4 için Eş. 4.6 açıldığında:

$$K_4 = C^B_{1,4} + C^B_{2,3} + C^B_{3,2} + C^B_{4,1} + C^O_{1,3} + C^O_{2,2} + C^O_{3,1} \tag{4.11}$$

elde edilir. n = 4 için Eş. 4.8 açıldığında ise:

$$EKKTR_4 = max(C^B_{1,4} + C^B_{2,3} + C^B_{3,2} + C^B_{4,1} + C^O_{1,3} + C^O_{2,2} + C^O_{3,1}) \tag{4.12}$$

elde edilir. Bu iki eşitlikten hareketle, RDPÇ yöntemi ile 16 rakamlı iki sayının çarpılması için oluşturulan piramit yapısında 4 numaralı kolon toplamının maksimum değeri (K_{4m}) ile 4 rakamlı iki sayının çarpımı için oluşturulan piramit yapısının sağ tarafında yer alan en kalabalık kolon toplamının maksimum değeri ($EKKTR_4$) eşittir. Bu eşitlik, piramidin sağ yarısında yer alan tüm kolonlar için genellenebilir:

$$K_{tm} = EKKTR_t \qquad , 2 \leq t \leq 16 \tag{4.13}$$

Piramit yapısının sol tarafında yer alan kolonlar için $v = 33 - t$ değişkeni tanımlandığında K_{tm} değerinin, Bölüm 4.2.3.1'de elde edilen $EKKTS_v$ değerine eşit olduğu öngörülmektedir. Örneğin; $n = 16$ ve $t = 29$ için Eş. 4.6 açıldığında:

$$K_{29} = C^B_{14,16} + C^B_{15,15} + C^B_{16,14} + C^O_{13,16} + C^O_{14,15} + C^O_{15,14} + C^O_{16,13} \tag{4.14}$$

elde edilir. $n = 4$ için Eş. 4.9 açıldığında ise:

$$EKKTS_4 = max(C^B_{2,4} + C^B_{3,3} + C^B_{4,2} + C^O_{1,4} + C^O_{2,3} + C^O_{3,2} + C^O_{4,1}) \tag{4.15}$$

elde edilir. Eş. 4.14'te yer alan ifadenin maksimum değeri ile Eş. 4.15'te yer alan ifade özdeştir. Bu özdeşlik uygun indis değişimi yapılarak daha açık görülebilir. Örneğin; dört rakamlı iki sayının çarpımı için sayı çifti ($x_4x_3x_2x_1, y_4y_3y_2y_1$) yerine ($x_{16}x_{15}x_{14}x_{13}$, $y_{16}y_{15}y_{14}y_{13}$) biçiminde yazılırsa, piramidin sol tarafındaki en kalabalık kolon toplamının maksimum değeri ($EKKTS_4$) ile Eş. 4.14'te verilen ifadenin maksimum değeri (K_{29}) özdeş olacaktır. Bu özdeşlik, piramidin sol tarafında yer alan tüm kolonlar için genellenebilir:

$$K_{tm} = EKKTS_v \qquad , 17 \leq t \leq 31, \qquad v = 33 - t \tag{4.16}$$

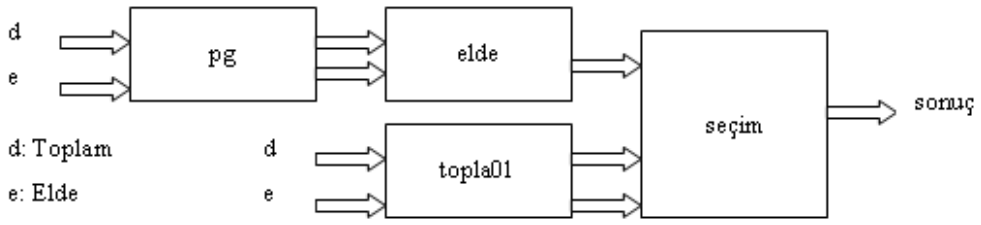
Eş. 4.13 ve Eş. 4.16'dan hareketle genelleme yapılarak şu sonuca varılabilir: Rakam çarpımına dayalı çarpma algoritmasında, çarpılacak iki sayının rakam sayısının n olduğunu varsayalım. Çarpım esnasında her bir kolon toplamının en büyük değerini bulmak için, aynı algoritma ile çarpılacak sayıların rakam sayısının [2,n] aralığında olduğu tüm durumlar ele alınır. Ele alınan bu durumların her biri için sağ ve sol taraftaki en kalabalık iki kolon toplamının en büyük değerleri hesaplanır. Bulunan bu değerlerden hareketle Eş. 4.13 ve Eş. 4.16'da elde edilen bilgiler ışığında, n rakamlı iki sayının çarpılması işleminde her bir kolon toplamının alabileceği en büyük değer bulunmuş olur. Bu genellemenin matematiksel ifadesi Eş. 4.17'de verilmiştir.

$$K_{tm} = \begin{cases} 9 & , t = 1 \\ EKKTR_t & , 2 \leq t \leq n \\ EKKTS_v & , n+1 \leq t \leq 2n-1 \ , v = 2n+1-t \\ 8 & , t = 2n \end{cases} \tag{4.17}$$

16 rakamlı iki BCD sayının RDPÇ yöntemi ile çarpılması için oluşturulan piramit yapısında, önerilen sınır değerleri hesaba katıldığı ve katılmadığı durumlarda, kolon toplamlarının sınır değerleri, bu toplamların iki tabanında ve BCD biçiminde gösterimleri için gerekli bit sayıları Çizelge 4.4'te verilmiştir. Elde edilen sonuçlara göre, 7 kolon toplamı için ihtiyaç duyulan iki tabanında toplayıcının bit sayısı bir azalmıştır. Bu kolonların ilgili alanı Çizelge 4.4'te koyu renkle ifade edilmiştir. Toplayıcının bit sayısının bir azalması, kullanılan toplayıcının hem alan hem de gecikme başarımına katkıda bulunur. Ayrıca 13 kolon toplamı için BCD gösterimde ihtiyaç duyulan bit sayısı bir azalmıştır. BCD gösterimde ihtiyaç duyulan bit sayısının azalması, iki tabanından on tabanında çevrim işleminin daha hızlı olmasını sağlar. 12 kolon toplamı için toplayıcı ve çevirici için ihtiyaç duyulan bit sayılarında değişiklik olmamıştır.

4.2.4 Elde Aktarımlı Toplayıcı ile Son Toplam

Kolon toplamlarının indirgenmesi işleminden sonra elde edilen ara toplamlar, EST kullanılarak toplam ve elde vektörlerine dönüştürülmektedir. Bu aşamadan sonra hızlı bir elde aktarımı yaparak nihai çarpım sonucunun hesaplanması gerekmektedir. EAT birimi Şekil 4.9'da gösterilen biçimde 4 alt birime bölünerek tasarlanmıştır.



Şekil 4.9. Elde Aktarımlı Toplayıcı (EAT)

4.2.4.1 Aktarım ve Üretim Sinyalleri(pg)

Toplam (*d*) ve elde (*e*) vektörleri kullanılarak her bir basamakta aktarım ve üretim sinyalleri hesaplanmaktadır. Elde aktarımı olabilmesi için iki koşuldan birinin gerçekleşmesi gereklidir. İlk koşul, toplanacak rakamın 9 ve eldenin 0 olmasıdır. İkinci koşul, toplanacak rakamın 8 ve eldenin 1 olmasıdır. Elde üretilmesi için toplanacak rakamın 9 ve eldenin 1 olması gerekmektedir. Bu durumlar Eş. 4.18'de gösterilen

Çizelge 4.4. Kolon Toplamlarının Maksimum Değerleri ve Gerekli Bit Sayıları

Kolon No	**Toplanan Rakam Sayısı**	**Sınır Değeri**	**İkili Bit Sayısı**	**BCD Bit Sayısı**	**Önerilen Sınır Değeri**	**İkili Bit Sayısı**	**BCD Bit Sayısı**
1	1	9	4	4	9	4	4
2	3	26	5	6	26	5	6
3	5	43	6	7	39	6	**6**
4	7	60	6	7	52	6	7
5	9	77	7	7	65	7	7
6	11	94	7	8	78	7	**7**
7	13	111	7	9	91	7	**8**
8	15	128	8	9	104	**7**	9
9	17	145	8	9	117	**7**	9
10	19	162	8	9	130	8	9
11	21	179	8	9	143	8	9
12	23	196	8	9	156	8	9
13	25	213	8	10	169	8	**9**
14	27	230	8	10	182	8	**9**
15	29	247	8	10	195	8	**9**
16	31	264	9	10	208	**8**	10
17	31	263	9	10	203	**8**	10
18	29	246	8	10	190	8	**9**
19	27	229	8	10	177	8	**9**
20	25	212	8	10	164	8	**9**
21	23	195	8	9	151	8	9
22	21	178	8	9	138	8	9
23	19	161	8	9	125	**7**	9
24	17	144	8	9	112	**7**	9
25	15	127	7	9	99	7	**8**
26	13	110	7	9	86	7	**8**
27	11	93	7	8	73	7	**7**
28	9	76	7	7	60	**6**	7
29	7	59	6	7	47	6	7
30	5	42	6	7	34	6	**6**
31	3	25	5	6	21	5	6
32	1	8	4	4	8	4	4

biçimde mantıksal kapılarla ifade ebilebilir.

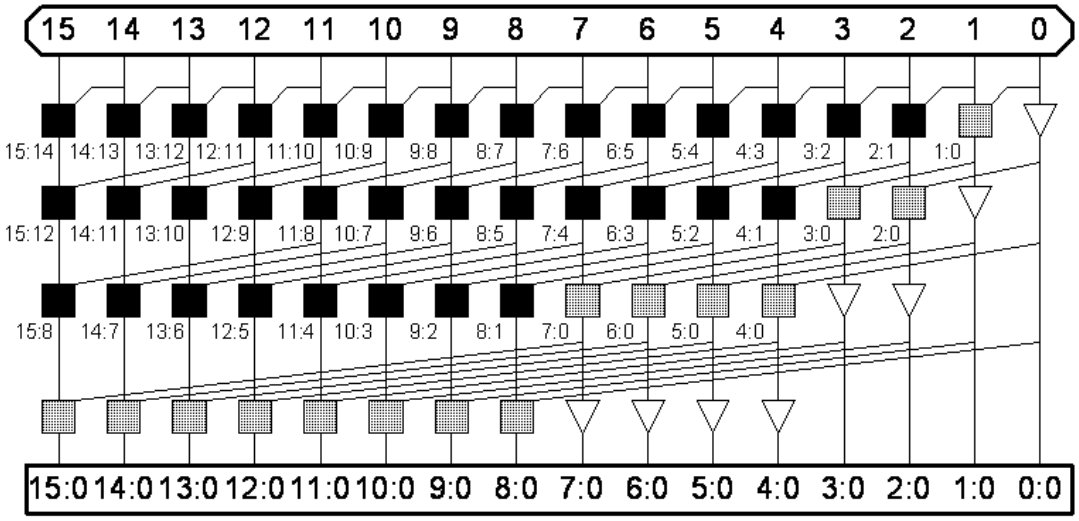
$$p_i = d_{i3}.(d_{i0} \oplus e_i)$$

$$g_i = d_{i3}.(d_{i0}.e_i) \qquad (4.18)$$

Eş. 4.18'de "$\oplus$" sembolü "DIŞARAN VEYA (Exclusive OR)" kapısını, "." ise "VE" kapısını ifade etmektedir. Bu eşitlik için, toplam vektörünün *i*'nci elemanı $d_i = d_{i3}d_{i2}d_{i1}d_{i0}$ ve elde vektörünün *i*'nci biti e_i olarak tanımlanmıştır.

4.2.4.2 Paralel Önek Elde Hesaplama

Şekil 4.9'da verilen "elde" biriminde hızlı bir biçimde elde hesabı yapmak için Kogge-Stone (Kogge and Stone, 1973) tarafından önerilen paralel önek elde (Parallel Prefix Carry) hesabı kullanılmıştır. Şekil 4.10'da 16 (p,g) çifti için Kogge-Stone elde hesabı örnek olarak verilmiştir. RDPÇ yönteminde ise toplam 28 (p,g) çifti vardır.



Şekil 4.10. Kogge-Stone Paralel Önek Elde Hesabı (Kogge and Stone, 1973)

Eş. 4.18'de verilen üretim ve aktarım sinyalleri kullanılarak, Şekil 4.10'da ara basamaklarda ihtiyaç duyulan öbek üretim ve aktarım sinyalleri Eş. 4.19'da verilen eşit-

liklerle hesaplanır.

$$G_{i:j} = G_{i:k} + P_{i:k}.G_{k-1:j}$$

$$P_{i:j} = P_{i:k}.P_{k-1:j}$$

$$G_{i:i} = g_i$$

$$P_{i:i} = p_i \quad (4.19)$$

Eş. 4.19'da hesaplanan öbek üretim sinyallerinden, Eş. 4.20 kullanılarak ilgili elde bulunur.

$$C_i = G_{i:0} \quad (4.20)$$

4.2.4.3 Elde Öncesi Koşulsuz Toplama

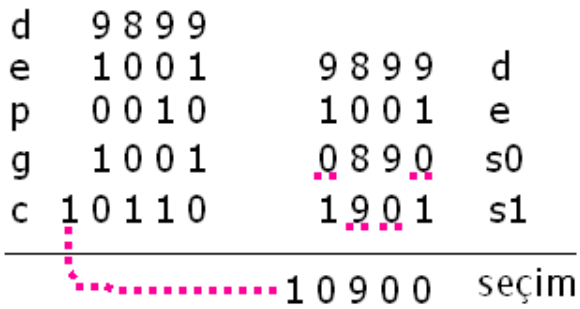
Şekil 4.9'da verilen "topla01" biriminde, elde hesabından bağımsız olarak toplam vektörünün her bir rakamı ve elde vektörünün ilgili biti 0 ve 1 ile toplanarak iki farklı vektör hesaplanmıştır ($s0_i$ ve $s1_i$). Bu vektörlerin i'nci rakamları Eş. 4.21'de verilmiştir.

$$s0_i = d_i + e_i$$

$$s1_i = d_i + e_i + 1 \quad (4.21)$$

4.2.4.4 Eldelere Göre Seçim

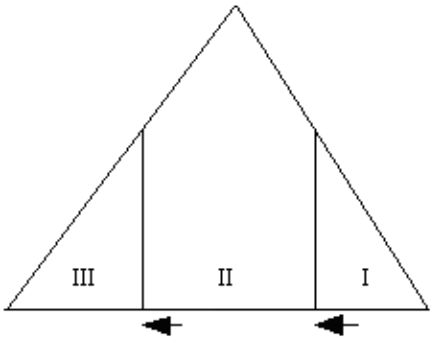
Son aşamada, Eş. 4.20'de hesaplanan eldelere göre, $s0$ ve $s1$ vektörlerinden birinin ilgili rakamı seçilerek sonuca ulaşılmaktadır. Başka bir ifadeyle, ilgili elde 0 ise $s0$ vektöründen veya elde 1 ise $s1$ vektöründen ilgili rakam seçilir. Bu işlem, Şekil 4.9'da verilen "seçim" biriminde gerçekleştirilir. Şekil 4.11'de 4 rakam ve 4 elde kullanılarak elde aktarımlı toplayıcı için örnek verilmiştir. İlk aşamada rakamlara ve eldelere göre p ve g hesaplanmış ve buna paralel olarak $s0$ ve $s1$ vektörleri hesaplanmıştır. Daha sonra p ve g sinyallerinden paralel önek elde hesabı ile c elde vektörü hesaplanmıştır. Bu c vektörünün elemanlarına göre $s0$ ve $s1$ vektörlerinden ilgili rakamlar seçilerek toplam sonucuna ulaşılmıştır. Burada giriş eldesinin sıfır olduğu kabul edilmiştir ve çıkış eldesi olarak da c vektörünün en soldaki biti kullanılmıştır.



Şekil 4.11. Örnek Bir Elde Aktarımlı Toplama

4.3 RDPÇ Yönteminde Piramidin Üç Parçaya Ayrılması

RDPÇ yönteminde elde edilen piramit, Şekil 4.12'de gösterilen şekilde üç parçaya ayrılarak gecikme başarımına katkı sağlanması hedeflenmiştir. İlk kısımda (I), çarpma sonucunun ilk 13 rakamı, ikinci kısımda (II) sonraki 6 rakamı ve son kısımda (III) son 13 rakamı hesaplanmıştır. İlk kısımda oluşan çıkış eldesi ikinci kısıma giriş eldesi olarak aktarılmıştır. Son kısımda, ikinci kısımdan gelecek elde beklenmeksizin biri diğerinden bir fazla olacak şekilde iki farklı sonuç (T ve T+1) hesaplanmıştır. Daha sonra ikinci kısımdan gelen eldeye göre bu iki sonuçtan biri seçilmiştir.



Şekil 4.12. Piramidin Üç Parçaya Ayrılması

Elde edilen sentez sonuçları incelendiğinde, ilk kısımdan gelen eldenin ikinci kısıma aktarılmasının gecikme başarımına katkısı olmadığı gözlenmiştir. Bunun sebebi ilk kısımda oluşan çıkış eldesinin ikinci kısımda başlayan toplama işleminden önce hazır olmasıdır. Ayrıca ikinci kısımdan aktarılan eldenin üçüncü kısımdaki toplama işleminden daha geç oluştuğu sonucuna varılmıştır. Piramidin bu şekilde üç parçaya

ayrılması işleminde en uygun noktanın belirlenmesi kritik öneme sahiptir. Yapılan çalışmaların ilk sonuçlarına göre bu uygun nokta bulunamamıştır. Tasarımın sentez sonuçları detaylı incelendiğinde, kritik gecikme yollarının birbirine çok yakın değerlerde gecikmelere sahip olduğu gözlenmiştir. Bunun sebebi sentez aracı tarafından gerçekleştirilen karmaşık eniyileme süreçlerinin tasarımda meydana getirdiği değişikliklerdir. Bu nedenle piramit birden fazla parçaya ayrılsa bile ayrım noktalarının çok yakınındaki kolonlardan başlayan kritik gecikme yolları, sistemin yeni gecikme başarımını belirlemektedir. Ortaya çıkan bu sonuçlara göre tasarımın gecikme başarımının bu yaklaşımla arttırılması mümkün değildir.

4.4 RDPÇ Yönteminin FPGA İncelemesi

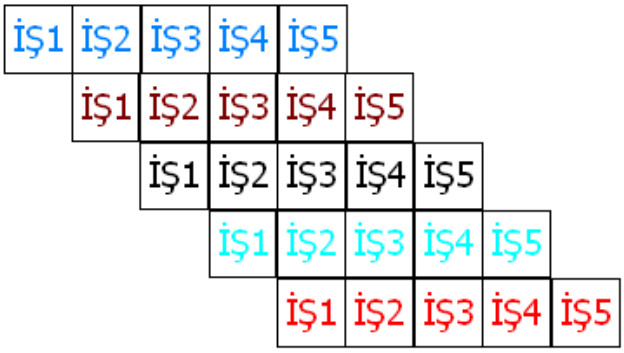
RDPÇ yöntemi Xilinx firmasına ait Virtex4 ailesinden xc4vfx40 modeli FPGA yongası için sentezlenmiştir. Ancak elde edilen sonuçlarda, RDPÇ yönteminin yapısından kaynaklı karmaşık güzergah (route) yapısı nedeniyle gecikme başarımı beklenen düzeyde gerçekleşmemiştir. Bunun nedeni, RDPÇ yönteminde rakam çarpım birimlerinin sayısının fazlalığı ve bu birimler arasındaki güzergahların karmaşıklığıdır. RDPÇ yöntemi için güzergahta kaybedilen zaman, mantıksal devrelerin gecikmesinden daha fazla olmuştur. Ayrıca, donanım tasarımı için yazılan VHDL kodları FPGA yapısında var olan kaynakların etkin bir biçimde kullanılmasına imkan tanımamaktadır. Örneğin; rakam çarpım birimi sadece mantıksal kapılar kullanılarak VHDL ile tasarlanmıştır. Ancak FPGA yongasının içinde ayarlanabilen mantıksal bloklar mevcuttur. Bu blokların içerisinde tam toplayıcı, başvuru çizelgesi ve çoklayıcı gibi bu tasarım için etkin kullanılamayan birimler vardır.

Literatürde son zamanlarda, FPGA teknolojisi için önerilen tasarımlar mevcuttur (Sutter *et al.*, 2009), (Minchola and Sutter, 2009). FPGA tasarımı hem işlevsel test amacıyla hem de son kullanıcıya erişme süresinin kısalığı nedeniyle sıklıkla kullanılmaktadır. Yakın gelecekte on tabanında aritmetik işlem donanımları, FPGA teknolojisine uygun tasarlanarak günlük kullanım alanlarında da yer bulacaktır.

4.5 RDPÇ Yönteminin Ardışık Düzen İncelemesi

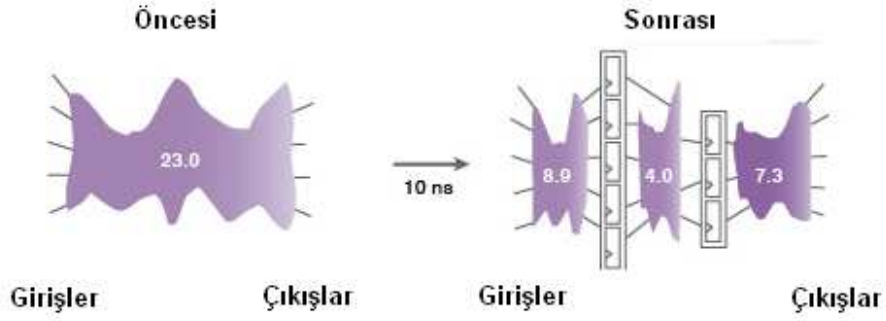
Çıkan iş oranını (throughput) arttırmak amacı ile ardışık düzen (pipeline) incelemesi yapılmıştır. Ardışık düzende tasarım, basamaklar arasında dengeli gecikme başarımı sağlanarak birden fazla basamağa ayrılmalıdır. Bu ayrım yapıldıktan sonra,

herhangi bir basamaktaki iş yapıldıktan sonra aynı kaynak kullanılarak bir sonraki iş başlatılabilir (Şekil 4.13). Ardışık düzende amaç, çıkan iş oranını (throughput) arttırmaktır. RDPÇ yönteminde ara basamaklara ayırma işleminin dengeli ayırma işlemine uygun olmadığı tespit edilmiştir. Ayrıca kullanılan sentez aracının sağlamış olduğu otomatik ayırma işlemi sonuçlarına göre de RDPÇ yöntemi en fazla üç basamağa ayrılabilir.



Şekil 4.13. Ardışık Düzende İş Akışı

Şekil 4.14'te Design Compiler (DC) sentez aracında "retime" seçeneği ile bileşimsel bir devrenin ardışık devreye dönüştürülmesi örnek olarak verilmiştir. Bu örnekte hedef gecikme 10ns olarak belirlenmiş ve sentez sonucunda gecikmesi en fazla olan basamakta gecikme 8,9ns olmuştur. Bu gecikme hedeflenen gecikme kriterini sağlamaktadır.

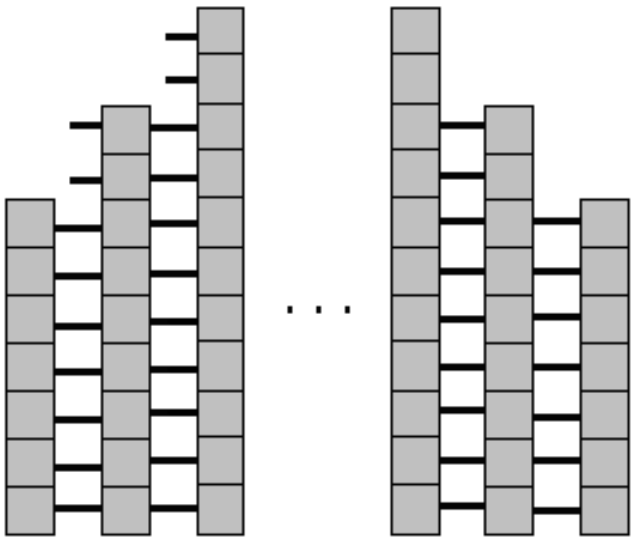


Şekil 4.14. DC "retime" Seçeneği ile Gecikme Dengeleme

4.6 RDPÇ Yönteminde BCD4221 Gösteriminin Kullanılması

Bölüm 2.1.2'de de bahsedilen ve Vazquez tarafından yapılan çalışmalarda (Vazquez *et al.*, 2007) önerilen BCD4221 gösterimi, RDPÇ yönteminde kullanılmıştır. Buna göre rakam çarpım biriminin sonucu BCD4221 gösteriminde olacak şekilde gerekli değişiklikler yapılmış ve kolonlara yerleştirilen her bir rakam BCD4221 gösteriminde ifade edilmiştir.

Vazquez tarafından önerilen bu yöntemin en önemli avantajlarından biri, herhangi bir kolonda 3 rakamın Şekil 3.9'da gösterilen EST ile toplanmasında oluşan eldenin bir sonraki kolondaki EST'ye giriş eldesi olarak aktarılmasıdır. Ancak BCD4221 gösteriminin RDPÇ yöntemine uyarlanmasında bu elde aktarımı sorun olmuştur. Örneğin; piramidin sol tarafındaki bir kolonda oluşan eldelerin sayısı, bir sonraki kolonda kullanılan EST'lerin sayısından fazladır. Şekil 4.15'te bu durumun genel görüntüsü kabaca ortaya konmuştur.

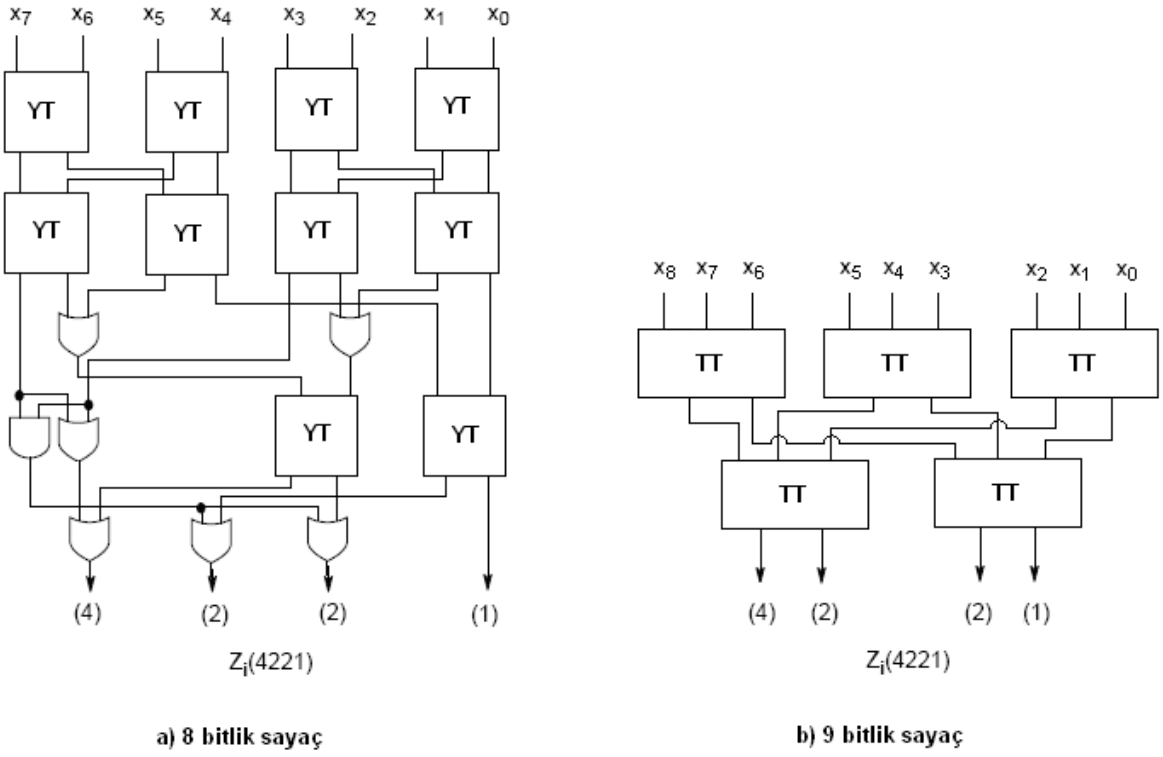


Şekil 4.15. Kolonlar Arası Elde Aktarımı

Şekildeki kutular EST'yi, yatay çizgiler ise aktarılan eldeleri göstermektedir. Oluşan fazlalık eldelerin ayrıca hesaba katılması gerekmektedir ve bu işlem paralel gerçekleştirilen hesaplamaların gecikme başarımını olumsuz etkilemiştir.

4.7 RDPÇ Yönteminde BCD4221 Gösteriminde Sayaç Kullanılması

Kolon toplamları için Vazquez tarafından önerilen (Vazquez, 2009) başka bir yöntem de sayaç kullanılmasıdır. BCD4221 gösterimindeki birden fazla rakamın toplanması için önerilen 8 ve 9 bitlik sayaçlar Şekil 4.16'da verilmiştir. Şekil 4.16a'da gösterilen 8 bitlik sayaçta yarım toplayıcılar (half adder) kullanılmıştır. Şekil 4.16b'de gösterilen 9 bitlik sayaçta ise tam toplayıcılar (full adder) kullanılmıştır.

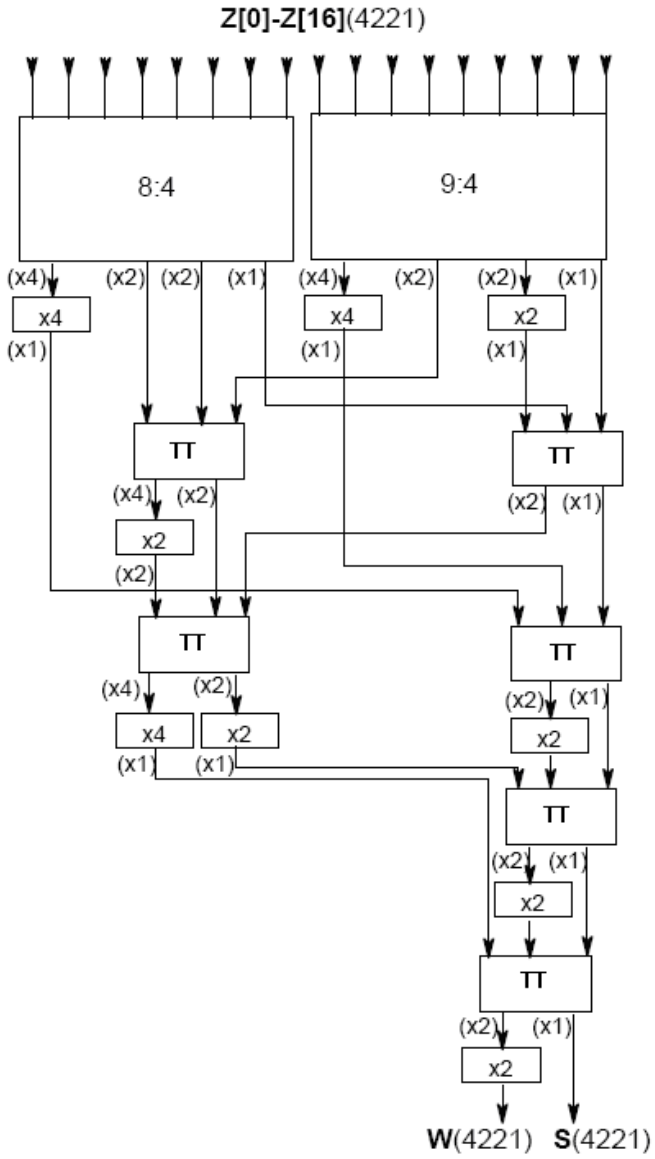


Şekil 4.16. BCD4221 Gösterimi İçin 8 ve 9 Bitlik Sayaçlar

Vazquez önerdiği çarpma yönteminde bir kolonda yer alan 17 rakamın, 8 ve 9 bitlik sayaçlar kullanılarak toplanabileceğini göstermiştir. 17 rakamın, Şekil 4.16'da önerilen 8 ve 9 bitlik sayaçlarla toplanabilmesi için Şekil 4.17'de gösterilen şekilde kullanılması gerekmektedir. Şekil 4.17'de kullanılan "x2" ve "x4" birimleri, BCD4221 gösteriminde sırasıyla 2 ve 4 ile çarpma işlemini gerçekleştirmektedir.

Vazquez'in önerdiği sayaçlar VHDL ile kodlanmış ve sentezlenmiştir. Bu tasarım, tez çalışmasında 17 rakamı toplamak için kullandığımız, önce iki tabanında toplayıp

sonra on tabanına dönüştürme yöntemi ile tasarlanan kolon toplayıcıyla karşılaştırılmıştır. Sayaç kullanılarak gerçekleştirilen tasarımın gecikme başarımı, bizim kullandığımız toplayıcının gecikme başarımından %5 daha iyi sonuç vermiştir. Ancak, sayaç kullanılarak toplanan rakamların BCD4221 gösteriminden BCD8421 gösterimine dönüştürülmesi gerektiğinden, RDPÇ yönteminde kullanılmasının avantaj sağlamayacağı sonucuna varılmıştır.



Şekil 4.17. BCD4221 Gösteriminde Sayaç Kullanılarak 17 Rakamın Toplanması

5. UYGULAMALAR

5.1 Donanım Tasarımı

Tez kapsamında yapılan çalışmaların donanım tasarımı, VHDL donanım tanımlama dili kullanılarak gerçekleştirilmiştir. Bu tez kapsamında 16 rakamlı iki sayının çarpımını yapan BCD çarpıcı tasarlanmıştır.

Tez çalışmasında tasarlanan çarpıcının yanı sıra Lang tarafından önerilen çarpıcı tasarımı da (Lang and Nannarelli, 2006) referans olarak, VHDL ile tasarlanmıştır.

Donanım tasarımı gerçekleştirilen çarpıcıların, Mentor Graphics[1] firmasına ait Modelsim aracı yardımıyla benzetimi gerçekleştirilmiştir. Benzetim aracı yardımıyla, gerçeklenen tasarımların işlevsel denetimi sağlanmıştır.

İşlevsel denetimi başarıyla geçen tasarımlar, Synopsys[2] firmasına ait Design Compiler aracıyla sentezlenmiştir. Sentez aracında, hedef teknoloji kütüphanesi olarak Faraday 90nm SP-RVT kütüphanesi seçilmiş ve normal çalışma koşulları (1V ve 25^{o}C) için sentezlenmiştir.

Tez kapsamında, rakam çarpımına dayalı paralel çarpma yöntemi kullanılarak gerçekleştirilen iki farklı çarpıcı tasarlanmıştır. Bu tasarımlardan ilki, tez çalışmasının bir sonucu olarak ortaya çıkan, kolon toplamlarının en büyük değerinin sınırlı olmasının göz önünde bulundurulmadığı durumda tasarlanmıştır. İkinci tasarım ise, elde edilen sınır değerleri göz önünde bulundurularak gerçekleştirilen tasarımdır.

Paralel çarpıcıların donanım tasarımında, diğer birçok sayısal tasarımda olduğu gibi, alan ve gecikme başarımları öne çıkan unsurlardır. Bu iki başarım parametresi göz önüne alınarak farklı senaryolar oluşturulabilir. Oluşturulan bu senaryolara göre, sentez aracında farklı kısıtlar oluşturulabilir. Çizelge 5.1'de seçilen 4 farklı senaryo için elde edilen sentez sonuçları verilmiştir. İlk senaryoda, gecikme başarımı açısından gevşek (relaxed) tutularak 2,3 ns olarak belirlenmiş ve alan başarımı parametresi açısından irdelenmiştir. Önerilen yeni sınır değerlerinin kullanıldığı tasarımın, diğer tasarımdan %19 daha az alan kapladığı gözlenmiştir. İkinci senaryoda, sınırın hesaba katılmadığı tasarımın ulaşabileceği en iyi gecikme başarımı kısıt olarak

[1] http://www.mentor.com
[2] http://www.synopsys.com

belirlenmiş ve bu durumda sınırların hesaba katıldığı tasarımın alan başarımı incelenmiştir. Yeni tasarımın bu durumda da %20 daha az alan kapladığı gözlenmiştir. Üçüncü senaryoda, iki tasarım da yaklaşık aynı alanı kapladığında, gecikme başarımları açısından karşılaştırılmıştır. Bu durumda yeni tasarımın gecikme başarımının %10 daha iyi olduğu tespit edilmiştir. Son senaryoda ise, her iki tasarımın gecikme başarımı açısından ulaşabileceği en iyi durum incelenmiştir. Bu senaryoda, yeni tasarımın gecikme başarımı %8 iyileşmiştir.

Çizelge 5.1. On Altı Rakamlı BCD Çarpıcı Sentez Sonuçları

	RDPÇ Sınırsız		RDPÇ Sınırlı		İyileştirme	
Kısıt	Alan (NAND2)	Gecikme (ns)	Alan (NAND2)	Gecikme (ns)	Alan (%)	Gecikme (%)
2,3 ns gecikme	81.380	2,3	66.186	2,3	19	-
2,1 ns gecikme	98.135	2,1	78.934	2,1	20	-
Yaklaşık aynı alan	81.380	2,3	82.157	2,07	-	10
En iyi gecikme	98.135	2,1	99.390	1,94	-	8

Literatürde yer alan paralel çarpma yöntemlerinin sonuçları ile tez çalışmasında gerçekleştirilen tasarımların sonuçları Çizelge 5.2'de verilmiştir. Bu çizelgede ilk üç satırdaki tasarımlar, tezde tasarlanan çarpıcıların sentez sonuçlarıdır. Lang (Lang and Nannarelli, 2006) ve Vazquez (Vazquez *et al.*, 2007) tarafından yapılan çalışmaların sentez sonuçları söz konusu çalışmaların raporlarında sunulan değerlerdir. Jaberipur (Jaberipur and Kaivani, 2009) tarafından yapılan çalışmanın sonucu ise ilgili çalışmada Vazquez'in çalışması ile yapılan kıyaslamadan yararlanılarak hesaplanmıştır. Literatürde yer alan çalışmalarda kullanılan sentez aracı ve hedef teknoloji kütüphanesi, bu tez çalışmasında kullanılanlardan farklıdır. Bu nedenle yapılan kıyaslamalar sadece genel fikir edinme amacıyla yapılmaktadır. Ancak sentez aracı ve kullanılan kütüphane farkının en aza indirgenmesi için çaba sarfedilmiştir. Örneğin; Lang tarafından önerilen yöntem, tez çalışması esnasında yeniden tasarlanmış

ve sentezlenmiştir. Ayrıca, Vazquez'in kullandığı kütüphane ile aynı versiyona sahip kütüphane kullanılmıştır.

Tez çalışmasında tasarlanan çarpıcılar için elde edilen sentez sonuçlarından, gecikme başarımı en iyi olanları seçilerek Çizelge 5.2'de mevcut çalışmaların sonuçları ile kıyaslanarak verilmiştir. Bu sonuçlara göre, RDPÇ yöntemi ile önerilen sınırlar göz önünde bulundurularak veya bulundurulmayarak tasarlanan çarpıcıların alan başarımları, mevcut yöntemlerinkine göre daha kötüdür. Bunun nedeni, RDPÇ yönteminde kullanılan rakam çarpım birimlerinin 256 kez kullanılmasıdır. Ancak, gecikme başarımı açısından, özellikle sınırlar göz önünde bulundurulduğunda elde edilen tasarım, mevcut yöntemlerle kıyaslanabilir seviyededir. Yonga üretim maliyetlerinin ucuzlamasıyla, alan açısından dezavantajına rağmen, sağladığı gecikme başarımı ile RDPÇ yönteminin paralel çarpıcılar arasında iyi bir alternatif olduğunu düşünmekteyiz.

Çizelge 5.2. Literatürdeki Çalışmalarla Karşılaştırmalı Sentez Sonuçları

	Alan (NAND2)	**Alan Oranı**	**Gecikme (ns)**	**Gecikme Oranı**
RDPÇ Sınırlı	99.390	1	1,94	1
RDPÇ Sınırsız	98.135	0,99	2,1	1,08
Lang	75.034	0,75	2,47	1,27
Lang (Lang and Nannarelli, 2006)	68.000	0,68	2,65	1,36
Vazquez (Vazquez *et al.*, 2007)	44.500	0,44	2,3	1,18
Jaberipur (Jaberipur and Kaivani, 2009)	79.636	0,8	2,62	1,35

5.2 Hata Tespiti

Tez çalışmasında elde edilen, kolon toplamları için sınır değerleri, hata tespiti için kullanılabilir. Ancak, hatanın tespit edilebilmesi için kolon toplamının sınır değerini aşması gerekmektedir. Bu nedenle, 4 rakamlı iki sayının RDPÇ yöntemi ile çarpılması esnasında, rastgele bitlere hata enjekte edilerek sebep oldukları hatalar-

dan ne kadarlık kısmının tespit edilebileceği belirlenmeye çalışılmıştır. Bu amaçla, MATLAB'da RDPÇ çarpma yöntemi gerçeklenmiş ve üç farklı alanda hata enjekte edilmiştir. İlk olarak çarpılacak sayılardan herhangi birininin, herhangi bir biti değiştirilmiştir. Daha sonra, rakam çarpımlarının rakamlarından herhangi birinin, bir biti değiştirilmiştir. Son olarak da kolonlarda toplanacak rakamlardan herhangi birinin, herhangi bir biti değiştirilmiştir. Enjekte edilen hatanın türü bit çevirmedir. Başka bir deyişle, hata enjekte edilecek konumdaki bitin değeri 0 ise 1, 1 ise 0 yapılmaktadır.

"Telco[3]" denektaşı uygulamasında kullanılan sayılardan ilk 10.000 tanesi alınarak, yukarıda belirtilen noktalarda bir hata enjekte edilmiştir. Elde edilen ortalama hata analizi sonuçları Çizelge 5.3'te gösterilmiştir. Eğer hata, çarpılacak sayılardan birinde meydana gelirse %0,36 rakam çarpımlarında meydana gelirse %1,07 ve kolon toplamlarında meydana gelirse %8,43 oranında tespit edilmiştir. Hatanın bu konumlardan herhangi birinde olması durumunda ise ortalama %3,3 oranında tespit edildiği gözlenmiştir.

Çizelge 5.3. Telco Denektaşı İçin Tek Hata Tespiti

Oluşma Yeri	**Tespit Edilen Hata Sayısı**	**Tespit Oranı (%)**
Çarpılacak Sayı	36	0,36
Rakam Çarpımları	107	1,07
Kolon Toplamları	843	8,43
Herhangi Bir Konumda	330	3,3

Kolon toplamlarının sınır değerleri, hata tespitinde kullanıldığında, birden fazla bitte hata oluşması durumu da belirlenebilmektedir. Hatta hatalı bit sayısı ne kadar fazla ise hata tespit olasılığı da o kadar yüksek olacaktır. Bu özelliği sayesinde, mevcut hata tespit yöntemlerine göre daha avantajlıdır. Telco verileri için, birden fazla bitin hatalı olduğu durumda hata tespit sayıları Çizelge 5.4'te verilmiştir. Burada, her bir hatalı bit sayısı için, uygulama 10 kez çalıştırılmış ve ortalama değerleri elde edilmiştir. Çizelgede belirtilen sayılar, tespit edilen ortalama hata sayısını belirtmektedir. Hatalı bit sayısı arttıkça, tespit edilen hata sayısı da artmaktadır. Bu uygulamada rastgele hata enjeksiyonu için düzgün (uniform) dağılımlı rastgele sayı üreteci

[3]http://speleotrove.com/decimal/telco.html

kullanılmıştır. Bu nedenle hata sayısındaki artış, hatalı bit sayısı ile orantılı olarak artmaktadır.

Çizelge 5.4. Telco İçin Birden Fazla Konumda Gerçekleşen Hatalı Bit İçin Tespit Sayısı

	Hatalı Bit Sayısı							
Yineleme	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	331	648	982	1335	1639	1988	2319	2677
2	336	664	978	1293	1644	1991	2335	2687
3	331	655	981	1304	1645	1977	2340	2674
4	331	655	981	1304	1645	1965	2331	2689
5	329	664	1001	1315	1635	1984	2330	2675
6	337	659	993	1308	1644	1973	2332	2681
7	331	671	992	1310	1654	1988	2328	2689
8	316	644	974	1317	1644	1997	2322	2672
9	334	655	981	1332	1638	2000	2315	2658
10	323	655	978	1294	1660	2003	2337	2673
Ort	330,2	657,5	984,5	1311,5	1645,3	1987,1	2329,3	2678,1
Ort/bit	330,2	328,7	328,1	327,8	329	331,1	332,7	334,7

Dört rakamlı tüm olası sayılar göz önüne alınarak gerçekleştirilen hata tespit sonuçları Çizelge 5.5'te verilmiştir. Çarpılan sayılardan biri [1-9999] aralığında değişirken, diğer sayı bu aralıkta 10 ayrı alt aralığa ayrılarak çarpma işlemi gerçekleştirilmiştir. Burada amaç, çarpılan sayıların bulunduğu aralığın hata tespit sayısına etkisini gözlemektir. Sayılardan birinin aldığı değerler arttıkça hata tespit sayısı da artmaktadır. Bunun sebebi, büyük sayıların çarpımında kolonlarda yer alan rakamlar da büyük değerler aldığından, hatalı toplamın sınır değerini aşma olasılığı artmaktadır. Çizelge 4.2 ve 4.3'te bulunan sayı çiftlerinde gözlenen ortak özellik, bu sayı çiftlerinin rakamlarının genellikle 5'ten büyük olmasıdır. Bu sayı çiftleri kolon toplamlarının en büyük değerlerini aldığı durumları ifade ettiğinden, hata tespit oranında sayılar büyüdükçe bu değerlerin aşılması olasılığı artmaktadır.

Diğer aralıklardan farklı olarak [5001-6000] aralığında hata tespit oranında düşüş gözlenmiştir. Bunun nedeni, çarpılacak sayılardan birinin birler basamağının her za-

man 5 olmasıdır. Herhangi bir rakamın 5 ile çarpılması durumunda birler basamağı 0 veya 5 olabilmektedir. Birler basamağında yer alan bu rakam, kolon toplamlarının sınır değerine ulaşmasını ve hatalı durumlarda da bu sınırı aşmasını engellemektedir.

Dört rakamlı tüm olası sayılar ile gerçekleştirilen hata tespit sonuçları, Telco verileri ile gerçekleştirilen sonuçlardan daha yüksektir. Bunun nedeni, Telco verilerinde yer alan sayıların küçük olmasıdır.

Çizelge 5.5. Dört Rakamlı Tüm Sayılar İçin Hata Tespiti

Sayı Aralığı	**Tespit Edilen Hata Sayısı**	**Tespit Oranı (%)**
[1-1000]	275203	2,75
[1001-2000]	344077	3,44
[2001-3000]	396317	3,96
[3001-4000]	417521	4,17
[4001-5000]	427386	4,27
[5001-6000]	423615	4,23
[6001-7000]	475115	4,75
[7001-8000]	556494	5,56
[8001-9000]	594910	5,95
[9001-9999]	729852	7,30
Toplam	4640490	4,64

6. SONUÇ VE DEĞERLENDİRMELER

Bu tez çalışmasında, günümüzde popüler hale gelen on tabanına göre aritmetik işlemlerin donanımsal tasarımı konusu irdelenmiştir. Aritmetik işlemlerin on tabanında yapılması, özellikle ticari uygulamalarda hassasiyet ve kesinlik kazandırmaktadır. Bu işlemlerin mevcut bilgisayar sistemlerinde yapılması, yazılım kütüphaneleri kullanılarak yapılabilmektedir. Ancak on tabanına göre aritmetik işlemlerin donanım üzerinde yapılması daha hızlı bir çözüm sunmaktadır. Aritmetik işlemler içerisinde sıklıkla kullanılan ve gecikme başarımı önem arz eden işlemlerden birisi çarpmadır. Tez çalışmasında BCD çarpıcıların alan ve gecikme başarımlarının iyileştirilmesine yönelik çalışmalar yürütülmüştür.

Çarpma yöntemleri, ardışık ve paralel olarak ikiye ayrılabilir. Ardışık çarpma yöntemlerinde, işlemin gerçekleşmesi birden fazla saat darbesi sürerken, paralel yöntemlerde genellikle tek saat darbesi sonunda çarpım sonucuna ulaşılır. Paralel yöntemlerinin bu özelliği, beraberinde daha düşük alan başarımını getirir.

Paralel çarpma yöntemlerinin gecikme başarımı, hedef teknolojide ulaşılabilecek saat sıklığı ile sınırlıdır. Ancak mevcut çalışmalarda, bu sınırlardan daha düşük seviyede gecikme başarımlarına ulaşılmıştır. Bu sebeple, gecikme başarımını arttırmak için yapılan çalışmalara olan ilgi devam etmektedir.

Tez çalışmasında, rakam çarpımına dayalı paralel çarpma yönteminin gecikme ve alan başarımlarının iyileştirilmesine çalışılmıştır. Bu çalışmalar esnasında, RDPÇ yöntemine has bir özellik ortaya çıkarılmıştır. Ortaya çıkarılan özelliğe göre, çarpma işlemi esnasında her bir kolonda yer alan rakamların toplamının alabileceği en büyük değer bir eniyileme problemini içermektedir. Bu problemin genetik algoritma ile çözümü aranmış ve bulunan çözümlerden bazıları kapsamlı yineleme ile doğrulanmıştır. Bulunan bu çözümler kolon toplamlarının yeni sınır değerlerini oluşturmaktadır.

Bu yeni sınır değerleri hesaba katılarak çarpıcı tasarımında gerekli değişiklikler yapılmıştır. Yapılan değişikliklerle beraber yeni tasarımın, ilk tasarıma göre hem alan hem de gecikme başarımı açısından daha avantajlı olduğu gözlenmiştir. Bu iyileştirmeler, yeni tasarımda kolon toplamları için kullanılan toplayıcıda, iki tabanına göre toplama işleminde ve/veya iki tabanından BCD gösterimine çevrim işleminde ihtiyaç

duyulan bit sayısının azalmasından kaynaklanmıştır. Bit sayısının azalması ile donanımın kapladığı alan azalmış ve daha düşük bir gecikme başarımına ulaşılmıştır.

RDPÇ yönteminin donanım tasarımı güncel bir sentezleme aracı ve hedef teknoloji kütüphanesi ile sentezlenmiştir. Sentez sonuçların literatürdeki mevcut çalışmalarla kıyaslanması, kesin yargıya varılmasına imkan vermemektedir. Çünkü, mevcut çalışmalarda kullanılan araçların versiyonları ve kullanılan kütüphaneler, bu çalışmalarda belirtilen sonuçları etkilemektedir. Ancak, mevcut diğer yöntemlerle kıyaslama sonuçlarından genel bir değerlendirme yapıldığında, RDPÇ yöntemi bu sonuçlarda alan başarımı açısından kötü olmasına rağmen gecikme başarımı açısından başarılı sayılabilecek sonuçlar vermektedir. RDPÇ yönteminin alan başarımının nispeten düşük olması, yöntemin temelinde yer alan çapraz rakam çarpım biriminin çok sayıda kullanılması gereksiniminden kaynaklanmaktadır.

Tez çalışmasının önemli bir sonucu olarak ortaya çıkan yeni sınır değerleri, ayrıca hata tespit sistemlerinde de kullanılabilir. Kolon toplamlarının sınır değerlerinin üstünde bir değere ulaşması işlem sırasında bir hata oluştuğunu belirtir. Bu sınır değerlerinin hata tespiti için mevcut yöntemlere göre avantajlı olduğu durumlar mevcuttur. Örneğin; hata olup olmadığının tespiti için bir karşılaştırıcı yeterli olacaktır. Ayrıca, bazı hata tespit sistemlerinde sadece sınırlı sayıda bitte meydana gelen hatalar tespit edilebilirken, sınır değerleri kullanıldığında daha fazla bitte hata oluşsa bile bu hata tespit edilebilmektedir. Hatanın tespit edilebilmesi için sınır değerinin aşılması yeterlidir.

Yeni sınır değerlerinin, RDPÇ yöntemini kullanan veya kullanacak diğer çalışmalara da ışık tutacağını düşünmekteyiz. Newton Raphson yöntemi gibi bazı bölme yöntemlerinde, çarpma işleminin sıklıkla kullanıldığı bilinmektedir. Bölme işleminde ihtiyaç duyulan çarpıcının RDPÇ ile tasarlanması, tez çalışmasında elde edilen kazanımların bölme işleminde de kullanılmasını sağlar.

Tez çalışmasında tasarlanan çarpıcı 16 rakamlı iki sayının çarpımını yapmaktadır. Bu tasarım, çift duyarlı kayan noktalı sayıların çarpımında ihtiyaç duyulan tamsayı çarpım birimi olarak kullanılabilir. RDPÇ yönteminin kayan noktalı sayıların çarpımında yuvarlama aşamasında sağlayabileceği avantajlar araştırmaya değer bir noktadır.

RDPÇ yöntemi, FPGA teknolojisi için yeniden kodlanabilir. Ticari FPGA yongalarının içindeki bloklara uygun bir tasarım yapılması, hem işlevsel test hem de son kullanıcıya erişme süresini kısaltma imkanını sunabilir.

RDPÇ yönteminin alan başarımındaki dezavantajı, aynı anda çoklu okuma yapılabilen hafıza birimlerinin tasarımı ile azaltılabilir.

RDPÇ yönteminin gecikme başarımını en fazla etkileyen aşama, kolon toplamlarının indirgenmesi aşamasıdır. Birden fazla BCD rakamın daha hızlı bir şekilde toplanması, çarpıcının gecikme başarımını olumlu etkileyecektir.

KAYNAKLAR DİZİNİ

Anderson, M. J., Tsen, C., Wang, L. K., Compton, K., and Schulte, M. J., 2009, Performance analysis of decimal floating-point libraries and its impact on decimal hardware and software solutions, In: *IEEE International Conference on Computer Design*, pp. 465–471.

Boole, G., 1853, An investigation of the laws of thought, on which are founded the mathematical theories of logic and probabilities, Technical report, Mathematics in Queens College, Cork.

Bradley, J. J., Stoffers, B. L., Jr., T. R. S., and Widen, M. A., 1986, Simplified decimal multiplication by stripping leading zeros, *U.S. Patent No:4615016*.

Busaba, F. Y., Krygowski, C. A., Li, W. H., Schwarz, E. M., and Carlough, S. R., 2001, The IBM z900 decimal arithmetic unit, In: *Proceedings of 35th Asilomar Conference on Signals, Systems, and Computers*, Vol. 2, pp. 1335–1339.

Chipperfield, A. J., and Fleming, P. J., 1995, The MATLAB genetic algorithm toolbox, In: *IEE Colloquium on Applied Control Techniques Using MATLAB, Digest No. 1995/014*.

Cowlishaw, M. F., 2003, Decimal floating-point: Algorism for computers, In: *Proceedings of 16th IEEE Symposium on Computer Arithmetic*, pp. 104–111.

Dadda, L., 2007, Multioperand parallel decimal adder: A mixed binary and BCD approach, *IEEE Transactions on Computers* 56(10), 1320–1328.

Dadda, L., and Nannarelli, A., 2008, A variant of a radix-10 combinational multiplier, In: *ISCAS'08*, pp. 3370–3373.

Dönmez, A., 2002, *Matematik Sözlüğü, Matematiğin Öyküsü ve Serüveni Cilt I*, Toplumsal Dönüşüm Yayınları, İstanbul.

Eckert, Jr. J. P., Weiner, J. R., Welsh, H. F., and Mitchell, H. F., 1951, The UNIVAC system, In: *International Workshop on Managing Requirements Knowledge*, p. 6.

Eisen, L., WardIII, J. W., Tast, H.-W., Mäding, N., Leenstra, J., Mueller, S. M., Jacobi, C., Preiss, J., Schwarz, E. M., and Carlough, S. R., 2007, IBM POWER6 accelerators: VMX and DFU, *IBM Journal of Research and Development* 51(6), 1–21.

Ergin, O., Unsal, O., Vera, X., and Gonzalez, A., 2006, Exploiting narrow values for soft error tolerance, *Computer Architecture Letters* 5(2), 12–15.

Ergin, O., Yalçın, G., Ünsal, O., and Valero, M., 2009, Exploiting the dependency checking logic of the rename stage for soft error detection, In: *Proceedings of Workshop on Design for Reliability*.

Erle, M. A., 2008, Algorithms and Hardware Designs for Decimal Multiplication, PhD thesis, Lehigh University.

Erle, M. A., and Schulte, M. J., 2003, Decimal multiplication via carry-save addition, In: *14th IEEE International Conference on Application-Specific Systems, Architectures, and Processors*, pp. 348–258.

Erle, M. A., Hickmann, B., and Schulte, M. J., 2009, Decimal floating-point multiplication, *IEEE Transactions on Computers* 58(7), 902–916.

Erle, M. A., Schulte, M. J., and Hickmann, B. J., 2007, Decimal floating-point multiplication via carry-save addition, In: *Proceedings of 18th IEEE Symposium on Computer Arithmetic*, pp. 46–55.

Erle, M. A., Schulte, M. J., and Lineberger, J. M., 2002, Potential speedup using decimal floating-point hardware, In: *Proceedings of 36th Asilomar Conference on Signals, Systems, and Computers*, Vol. 2, pp. 1073–1077.

Erle, M. A., Schwarz, E. M., and Schulte, M. J., 2005, Decimal multiplication with efficient partial product generation, In: *Proceedings of 17th IEEE Symposium on Computer Arithmetic*, pp. 21–28.

Goldstine, H. H., and Goldstine, A., 1996, The electronic numerical integrator and computer (ENIAC), *IEEE Annals of the History of Computing* 18(1), 10–16.

Gorgin, S., and Jaberipur, G., 2009, A fully redundant decimal adder and its application in parallel decimal multipliers, *Microelectronics Journal* 40(10), 1471–1481.

Haupt, R. L., and Haupt, S. E., 2004, *Practical Genetic Algorithms*, John Wiley & Sons Inc.

Hickmann, B., Krioukov, A., Schulte, M., and Erle, M., 2007, A parallel IEEE P754 decimal floating-point multiplier, In: *IEEE International Conference on Computer Design*, pp. 296–303.

Hickmann, B., Schulte, M., and Erle, M., 2008, Improved combined binary/decimal fixed-point multipliers, In: *IEEE International Conference on Computer Design*, pp. 87–94.

Hoffman, R. L., and Schardt, T. L., 1975, Packed decimal multiply algorithm, *IBM Technical Disclosure Bulletin* pp. 1562–1563.

Holland, J. H., 1975, *Adaptation in Natural and Artificial Systems*, University of Michigan Press.

IEEE 754-2008 - IEEE Standard for Floating-Point Arithmetic 2008, Technical report, The Institute of Electrical and Electronics Engineer Inc.

Ifrah, G., 1995*a*, *Bir Gölgenin Peşinde, Rakamların Evrensel Tarihi Cilt I*, TÜBİTAK, Kavaklıdere, Ankara.

Ifrah, G., 1995*b*, *Uzak Doğu'dan Maya Ülkesine Bir, İki, Üç..., Rakamların Evrensel Tarihi Cilt IV*, TÜBİTAK, Kavaklıdere, Ankara.

Jaberipur, G., and Kaivani, A., 2007, Binary-coded decimal digit multipliers, *IET Comput. Digit. Tech* 1(4), 377–381.

Jaberipur, G., and Kaivani, A., 2009, Improving the speed of parallel decimal multiplication, *IEEE Transactions on Computers* 58(11), 1539–1552.

James, R. K., Jacob, K. P., and S.Sasi 2009*a*, High performance, low latency double digit decimal multiplier on FPGA, In: *Proceedings of World Congress on Nature and Biologically Inspired Computing*, pp. 1445–1450.

James, R. K., Jacob, K. P., and S.Sasi 2009*b*, Performance analysis of double digit decimal multiplier on various FPGA logic families, In: *Proceedings of 5th Southern Conference on Programmable Logic*, pp. 165–170.

James, R. K., Shahana, T. K., Jacob, K. P., and S.Sasi 2008*a*, Decimal multiplication using compact BCD multiplier, In: *Proceedings of International Conference on Electronic Design*, pp. 1–6.

James, R. K., Shahana, T. K., Jacob, K. P., and S.Sasi 2008*b*, Fixed point decimal multiplication using RPS algorithm, In: *Proceedings of International Symposium on Parallel and Distributed Processing with Applications*, pp. 343–350.

Karaboğa, N., 1994, Sayısal Filtre Katsayılarının Genetik Algoritma Kullanılarak Yuvarlatılması, *Doktora Tezi, Erciyes Üniversitesi*.

Karcı, A., 2002, Genetik Algoritmalarda Iraksama ve Yerel Çözümde Kalma Problemlerinin Giderilmesi, *Doktora Tezi, Fırat Üniversitesi*.

Kenney, R. D., and Schulte, M. J., 2004, Multioperand decimal addition, In: *Proceedings of the IEEE Computer Society Annual Symposium on VLSI Emerging Trends in VLSI Systems Design*, pp. 251–253.

Kenney, R. D., Schulte, M. J., and Erle, M. A., 2004, A high-frequency decimal multiplier, In: *IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 26–29.

Kogge, P.M., and Stone, H.S., 1973, A parallel algorithm for the efficient solution of a general class of recurrence equations, *IEEE Transactions on Computers* 22(8), 786–793.

Lang, T., and Nannarelli, A., 2006, A radix-10 combinational multiplier, In: *Proceedings of 40th Asilomar Conference on Signals, Systems, and Computers*, pp. 313–317.

Larson, R. H., 1973*a*, High speed multiply using four input carry save adder, *IBM Technical Disclosure Bulletin* pp. 2053–2054.

Larson, R. H., 1973*b*, Medium speed multiply, *IBM Technical Disclosure Bulletin* p. 2055.

Mano, M., 2003, *Bilgisayar Sistemleri Mimarisi*, Literatür Yayıncılık, İstanbul.

Mano, M., and Kime, C. R., 1999, *Logic and Computer Design Fundamentals*, Prentice Hall.

Minchola, C., and Sutter, G., 2009, An FPGA IEEE 754-2008 Decimal64 floating-point multiplier, In: *Intl. Conference on Reconfigurable Computing and FPGAs*, pp. 59–64.

Mitchell, M., 1998, *An Introduction to Genetic Algorithms*, The MIT Press, Cambridge, Massachusettes, London.

Nicoud, J.D., 1971, Iterative arrays for radix conversion, *IEEE Transactions on Computers* 20(12), 1479–1489.

Ohtsuki, T., Oshima, Y., Ishikawa, S., Yabe, K., and Fukuta, M., 1987, Apparatus for decimal multiplication, *U.S. Patent No:4677583*.

Raafat, R., Abdel-Majeed, A. M., Samy, R., ElDeeb, T., Farouk, Y., Elkhouly, M., and Fahmy, H. A. H., 2008, A decimal fully parallel and pipelined floating-point multiplier, In: *Asilomar Conference on Signals, Systems and Computers*, pp. 1800–1804.

Schmookler, M., and Weinberger, A., 1971, High speed decimal addition, *IEEE Transactions on Computers* 20(8), 862–866.

Schulte, M. J., Tan, D., and Lemonds, C. E., 2007, Floating-point division algorithms for an x86 microprocessor with a rectangular multiplier, In: *Proceedings of the IEEE International Conference on Computer Design*, pp. 304–310.

Schwarz, E. M., Kapernick, J. S., and Cowlishaw, M. F., 2009, Decimal floating point support on the IBM system z10 processor, *IBM Journal of Research and Development* 53(1), 4:1–4:10.

Shannon, C. E., 1940, A symbolic analysis of relay and switching circuits, Master's thesis, Masachusettes Institute of Technology.

Sutter, G., Todorovich, E., Bioul, G., Vazquez, M., and Deschamps, J-P., 2009, FPGA implementations of BCD multipliers, In: *Intl. Conference on Reconfigurable Computing and FPGAs*, pp. 36–41.

Svoboda, A., 1969, Decimal adder with signed digit arithmetic, *IEEE Transactions on Computers* 18(3), 212–215.

Tanenbaum, A. S., 1996, *Computer Networks*, Prentice Hall.

Thapliyal, H., and Srinivas, M. B., 2004, High speed efficient NxN bit parallel hierarchical overlay multiplier architecture based on ancient Indian Vedic mathematics, In: *Proceedings of World Academy of Science, Engineering and Technology*, Vol. 2, pp. 225–228.

Trimble, G. R., 1986, The IBM 650 magnetic drum calculator, *IEEE Annals of the History of Computing* 8(1), 20–29.

Turgu, C., 2008, Çok Kullanıcılı OFDM İçin Kaynak Tahsisi, Yüksek lisans tezi, Hacettepe Üniversitesi.

Ueda, T., 1995, Decimal multiplying assembly and multiply module, *U.S. Patent No:5379245*.

Vazquez, A., 2009, High-Performance Decimal Floating-Point Units, PhD thesis, Universidade De Santiago De Compostela.

Vazquez, A., and Antelo, E., 2006, Conditional speculative decimal addition, In: *Proceedings of 7th Conference Real Numbers and Computers*, pp. 47–57.

Vazquez, A., Antelo, E., and Montuschi, P., 2007, A new family of high-performance parallel decimal multipliers, In: *Proceedings of 18th IEEE Symposium on Computer Arithmetic*, pp. 195–204.

Vazquez, A., Antelo, E., and Montuschi, P., 2010, Improved design of high performance parallel decimal multipliers, *IEEE Transactions on Computers* 59(5), 679–693.

Wang, L., Tsen, C., Schulte, M.J., and Jhalani, D., 2007, Benchmarks and performance analysis of decimal floating-point applications, In: *International Conference on Computer Design*, pp. 164–170.

Wang, L.K., and Schulte, M. J., 2004, Decimal floating-point division using Newton-Raphson iteration, In: *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors*, pp. 84–95.

Yalçın, G., and Ergin, O., 2007, Using tag-match comparators for detecting soft errors, *Computer Architecture Letters* 6(2), 53–56.

Yilmaz, M., Hower, D. R., Ozev, S., and Sorin, D. J., 2006, Self-checking and self-diagnosing 32-bit microprocessor multiplier, In: *Proceedings of the IEEE International Test Conference*, pp. 1–10.

Yilmaz, M., Meixner, A., Ozev, S., and Sorin, D. J., 2007, Lazy error detection for microprocessor functional units, In: *Proceedings of the IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems*, pp. 361–369.

ÖZGEÇMİŞ

Adı Soyadı		:	KENAN BOZDAŞ
Doğum Yeri		:	VAN
Doğum Yılı		:	15.09.1979
Medeni Hali		:	Evli

Eğitim ve Akademik Durumu

Lise	1993-1996	:	Özel Serhat Fen Lisesi, VAN
Lisans	1996-1999	:	Gazi Üniversitesi Elektrik ve Elektronik Mühendisliği Bölümü, ANKARA
Lisans	1999-2001	:	Hacettepe Üniversitesi Elektrik ve Elektronik Mühendisliği Bölümü, ANKARA
Yüksek Lisans	2001-2004	:	Hacettepe Üniversitesi Elektrik ve Elektronik Mühendisliği Bölümü, ANKARA

Yabancı Dil	:	İngilizce

İş Tecrübesi

2001-2010	:	Hacettepe Üniversitesi Elektrik ve Elektronik Mühendisliği Bölümü, ANKARA Araştırma Görevlisi