T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ÇEŞİTLİ TOPLAMA VE ÇARPMA-TOPLAMA DEVRELERİNİN SAHADA PROGRAMLANABİLİR KAPI DİZİLERİ ÜZERİNDE GERÇEKLENMESİ VE ANALİZİ

OLCAY KURT

YÜKSEK LİSANS TEZİ

HESAPLAMALI BİLİMLER ANABİLİM DALI

Tez Danışmanı: Yrd. Doç. Dr Uğur ÇİNİ

EDİRNE-2016

T.Ü. Fen Bilimleri Enstitüsü onayı

Prof. Dr. Mustafa ÖZCAN
Fen Bilimleri Enstitüsü Müdürü

Bu tezin Yüksek Lisans tezi olarak gerekli şartları sağladığını onaylarım.

Prof. Dr. Mustafa ÖZCAN
Anabilim Dalı Başkanı

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

Yrd. Doç. Dr. Uğur ÇİNİ
Tez Danışmanı

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

Jüri Üyeleri	İmza
Yrd. Doç. Dr. Uğur ÇİNİ	
Doç. Dr. M. Tolga SAKALLI	
Doç. Dr. Metin YAZGİ	

Tarih: 12/01/2016

T.Ü. FEN BİLİMLERİ ENSTİTÜSÜ
HESAPLAMALI BİLİMLER YÜKSEK LİSANS PROGRAMI
DOĞRULUK BEYANI

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

12/01/2016

Olcay KURT



Yüksek Lisans Tezi

Çeşitli Toplama ve Çarpma-Toplama Mimarilerinin Sahada Programlanabilir Kapı Dizileri Üzerinde Gerçeklenmesi ve Analizi

T.Ü. Fen Bilimleri Enstitüsü

Hesaplamalı Bilimler Anabilim Dalı

ÖZET

Bu tez çalışmasında sahada programlanabilir kapı dizileri (Field Programmable Gate Arrays - FPGA) yapıları üzerinde çeşitli toplama ve çarpma-toplama devrelerinin iyileştirilmesi amaçlanmıştır. Çalışmanın ilk kısmında çeşitli toplama devreleri FPGA'lar üzerinde gerçeklenmiştir. Pek çok FPGA ailesinde Fast-Carry Logic mimarisinden dolayı ripple-carry toplayıcı devresi yüksek hızda çalışmaktadır. Çalışmamızda, carry-skip, carry-lookahead, carry-select toplayıcı devreleri farklı bit genişlikleri kullanılarak gerçekleştirilmiştir. Devrelerin hız performansı, güç-gecikme çarpanı değerleri ve kullanılan lojik eleman sayıları karşılaştırılmıştır. Tezin ikinci kısmında ise çarpma-toplama devreleri FPGA üzerinde gerçekleştirilmiştir. Çeşitli alternatif aritmetik tekniklerinden faydalanılarak 16×16 bit ve 32×32 bit çarpma-toplama devresi tasarlanmıştır. Tasarlanan yapı iki farklı şekilde gerçekleştirilmiştir. İlki (6, 3) sayıcı ve çift-elde saklama yedekli mimari yapısı kullanılarak; diğeri çift elde-saklama yedekli mimari yapısı ile çok girişli toplayıcı yöntemi kullanılarak sentezlenmiştir. Önerilen yapılar sıradan çarpma-toplama devrelerine göre çok daha iyi performans sağlamaktadır. Önerilen çarpma-toplama devrelerinin işaret işleme, matris çarpımı gibi matematiksel işlemlerde önemli performans artışı sağlayacağı düşünülmektedir.

Yıl : 2016

Sayfa Sayısı : 51

Anahtar Kelimeler : Çeşitli toplama devreleri, FPGA, Booth algoritması, yedekli sayı sistemleri, çift-elde saklama mimarisi, multi-operand toplayıcı, elde-saklamalı aritmetik

Master's Thesis
Analysis and Design of Various Addition and Multiply-Add Architectures Using Field Programmable Gate Arrays
Trakya University Institute of Natural Sciences
Computational Sciences

ABSTRACT

In this thesis, optimization of various adders structures and multiply-add units using Field Programmable Gate Arrays (FPGAs) is presented. In the first part of the work, various adder architectures are synthesized using a variety of FPGA structures. Many of the FPGA families employ fast carry logic architecture which provides very high performance for the ripple-carry adder structures. In this thesis, carry-skip, carry-lookahead and carry-select adder structures are configured in various bit-lengths for higher performance. The speed performance, power-delay product and logic element usage is compared. In the second part of the thesis, various multiply-accumulate structures are implemented on FPGAs. 16×16 and 32×32 bit multiply-accumulate units are implemented using various of arithmetic techniques. Two main techniques are used for the arithmetic unit implementation. In the first structure, (6, 3) counters and carry double save redundant architecture is implemented. In the second structure redundant carry-save structure is implemented together with multi-operand carry-propagate structure. Both of the structures provide superior performance compared to conventional multiply-add structures. The proposed multiply-add structures are suitable for high performance signal processing and various mathematical implementations which require matrix multiplication operations.

Year : 2016
Number of Pages : 51
Keywords : Various adder circuits, FPGA, Booth encoding, redundant number systems, double carry-save architecture, multi-operand adder, carry-save arithmetic

TEŞEKKÜRLER

Yüksek lisans çalışmamın her safhasında bana destek olan tez danışmanım Yrd. Doç. Dr. Uğur ÇİNİ'ye, değerli vakitlerini ayıran ve tez jürimde olan Sayın Doç. Dr. M. Tolga SAKALLI'ya ve Sayın Doç. Dr. Metin YAZGI'ya, bu süre zarfında dostluğunu ve desteğini esirgemeyen Sayın Dr. Gamze GÜLTEKİN'e, çalışmamın her aşamasında yardımlarını esirgemeyen bölüm hocalarıma ve arkadaşlarıma, bugünlere gelmemi sağlayan ve hep yanımda olan aileme:

Sonsuz teşekkürler…

İÇİNDEKİLER

ÖZET i
ABSTRACT ii
TEŞEKKÜRLER iii
İÇİNDEKİLER iv
SEMBOLLER ve KISALTMALAR vi
ŞEKİLLER DİZİNİ vii
TABLOLAR DİZİNİ viii
BÖLÜM 1 1
GİRİŞ 1
BÖLÜM 2 5
TOPLAMA DEVRELERİ 5
2.1. Ripple-Carry Adder 5
2.2. Elde Üretimi ve Yayılımı 6
2.3. Carry-Skip Adder 7
2.4. Carry-Lookahead Adder 7
2.5. Carry-Select Adder 9
BÖLÜM 3 10
TOPLAMA DEVRELERİNİN ANALİZİ 10
BÖLÜM 4 15
YEDEKLİ SAYI SİSTEMLERİ 15
4.1. Genelleştirilmiş Yedekli Sayı Sistemleri 17

4.2.Olağan İşaretli-Sayı Sistemi Gösterimi 18
4.3. İkili İşaretli-Sayı Mimarisi 21
4.4. Elde-Saklama Aritmetiği 22
BÖLÜM 5 26
ÇARPMA DEVRELERİ 26
5.1. Çarpma-Toplama Devresi 27
5.2. Booth Algoritması 28
5.3. (6, 3) Sayıcı ve Altı Binary Sayının Eşzamanlı Olarak Toplanması 29
5.4. Pipeline Tekniği 30
5.5. Çift Elde Saklamalı Mimari 31
BÖLÜM 6 32
ÖNERİLEN ÇARPMA-TOPLAMA DEVRELERİ 32
BÖLÜM 7 40
ÇARPMA-TOPLAMA DEVRESİ DENEYLERİ 40
BÖLÜM 8 42
SONUÇLAR 42
KAYNAKLAR 44
ÖZGEÇMİŞ 50
TEZ ÇALIŞMASI SIRASINDA GERÇEKLEŞTİRİLEN BİLİMSEL FAALİYETLER 51

SEMBOLLER ve KISALTMALAR

ALU **:** Aritmetic Logic Unit
BSCB **:** Binary Stored or Carry Borrow
BSD **:** Binary Signed-Digit
CLA **:** Carry Lookahead Adder
CSKA **:** Carry Skip Adder
CSLA **:** Carry Select Adder
FA **:** Full Adder
FIR **:** Finite Impulse Response
FPGA **:** Field Programmable Gate Array
LAB **:** Logic Array Block
LSB **:** Least Significant Bit
LUT **:** Look-Up Table
MAC **:** Multiply-Accumulate Unit
MSB **:** Most Significant Bit
OSD **:** Ordinary Signed Digit
RCA **:** Ripple Carry Adder
SB **:** Stored-Borrow
SCB **:** Stored Carry or Borrow
SD **:** Signed Digit
VLSI **:** Very Large Scale Integration

ŞEKİLLER DİZİNİ

Şekil 1.1. Temel FPGA Mimarisi 2
Şekil 2.1. 4-bit Ripple Carry Adder 6
Şekil 2.2. 16-bit Carry Skip Adder 7
Şekil 2.3. 16-bit Carry Lookahead Adder 8
Şekil 2.4. 16-bit Carry Select Adder 9
Şekil 3.1. Stratix III f_{max} değerleri 10
Şekil 3.2. Cyclone III f_{max} değerleri 11
Şekil 3.3. Stratix III powerxdelay product değerleri 11
Şekil 3.4. Cyclone III powerxdelay product değerleri 12
Şekil 3.5. Stratix III'de farklı bit boyutları için RCA ile CSLA'nın karşılaştırılması 13
Şekil 3.6. Cyclone III'de farklı bit boyutları için RCA ile CSLA'nın karşılaştırılması 14
Şekil 4.1. İki işaretli sayının paralel toplanması 17
Şekil 4.2. (a) Tam toplayıcı gösterimi; (b) Tam toplayıcı kullanarak toplama işlemi 23
Şekil 4.3. Elde-saklama yapısı: (a) notmal ikili girişler; (b) yedekli girişler 24
Şekil 5.1. Genel çarpma işlemi 26
Şekil 5.2. Genel MAC ünitesi 28
Şekil 5.3. İşaretli çarpma işlemi için Radix-4 Booth kodlanan ara toplamlar 29
Şekil 5.4. (a) (6, 3) sayıcı tek bit yapısı; (b) (6, 3) sayıcı kullanılarak birçok terimin toplanması 30
Şekil 6.1. 40 bite genişletilerek yapılan Booth algoritması 34
Şekil 6.2. (6, 3) sayıcı kullanılarak önerilen 16×16 bit yedekli MAC ünitesi 35
Şekil 6.3. Bilinen yedekli CSA yapısıyla tasalanılan yedekli MAC ünitesi 35
Şekil 6.4. Çeşitli pipeline kombinasyonlarıyla tasarlanılan düzenli MAC üniteler 36
Şekil 6.5. (6, 3) sayıcı kullanılarak önerilen 32×32 bit MAC mimari 36
Şekil 6.6. Multi-operand toplayıcı kullanılarak önerilen yedekli 16×16 bit MAC ünite 37
Şekil 6.7. Multi-operand toplayıcı kullanılarak önerilen yedekli 32×32 bit MAC ünite 39

TABLOLAR DİZİNİ

Tablo 3.1. Stratix III için kullanılan lojik elemanların sayısı 12
Tablo 3.2. Cyclone III için kullanılan lojik elemanların sayısı 12
Tablo 3.3. Değişken bit gruplama yapılarak karşılaştırılması 13
Tablo 4.1. İkili SD sayılarının toplanması için değiştirilmiş kurallar 21
Tablo 5.1. Radix-4 değiştirilen Booth kodlama değerleri 29
Tablo 6.1. Çeşitli indirgeme operatör metrikleri 33
Tablo 7.1. 16×16 bit çeşitli MAC ünitelerinin kıyaslanması 41
Tablo 7.2. 32×32 bit çeşitli MAC ünitelerinin kıyaslanması 41

BÖLÜM 1

GİRİŞ

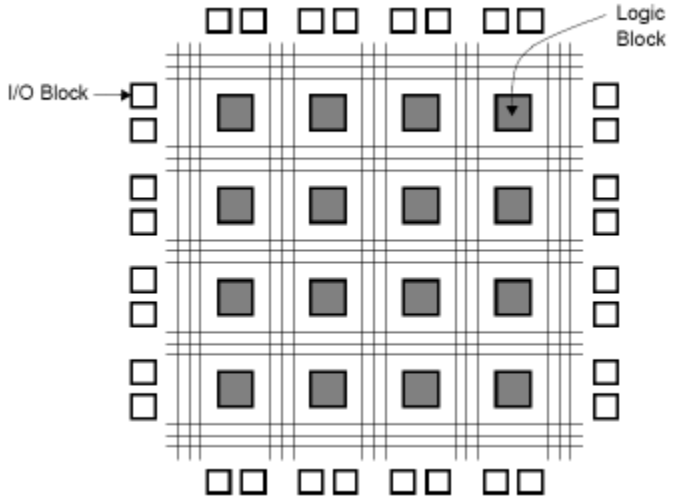
Sahada Programlanabilir Kapı Dizi (FPGA) cihazları günümüzde çok popüler olan elektronik cihazlardır ve ileri blok yapıları mevcut sayısal tasarım piyasasında geliştirilmiştir. Yüksek performansa sahip aritmetik devreler için birçok mimari son yıllarda sanayide yaygın olarak kullanılan sahada programlanabilir kapı dizilerinde etkili bir şekilde uygulanabilir.

Sayısal devreleri uygulamak için "Field Programmable Gate Array" (FPGA)'ların kullanımı her yıl artmaktadır. FPGA düşük hacim üretimi ve düşük maliyet açısından avantajlıdır. FPGA'lar genelleştirilmiş operatörlerin etkili uygulamalarında büyük öneme sahiptir. Toplama ve çarpma devreleri; aritmetik lojik ünitelerinde (ALU), filtreleme işlemlerinde ve işaret işlemeye kadar pek çok uygulamada kullanılır. Bu nedenle toplama ve çarpma devrelerinin sayısal devre tasarımında oldukça önemli bir yeri vardır [1].

Bir FPGA'nın temel yapısı Şekil 1.1'de gösterilmiştir. Bir FPGA'da herbir mantık blok (logic block), mantıksal modüllerden oluşur. Mantıksal modüller, Look-up Table (LUTs) içeren en küçük fonksiyonel bloklardır. Mantıksal modüller, sıralı ve kombinasyonel mantık işlevleri için kaydedilen çıkışlar ve kaydedilmeyen çıkışlar ile desteklenir.

LUT'lar, bloklar etrafında ulaşılabilir arabağlantılar yoluyla diğer lojik bloklarına bağlanır. FPGA'nın model ve üreticilerine bağlı olarak çeşitli lojik blok yapıları vardır. FPGA'yı programlayabilmek için kullanılan donanım dilinden biri olan Verilog HDL (Hardware Description Language) tez çalışmamızda tercih edilmiştir. Verilog HDL sayısal sistemlerin modellenmesi ve simülasyonu için kullanılır [2, 3].

Bir FPGA, arabağlantı matrisi (interconnection matrix) ile look-up tablolarını (LUT – Look-up Table) bağlayarak herhangi bir kombinasyonel mantık işlevini uygular. Mantık blokları düzenli bir yapıda FPGA bünyesinde düzenlenir. Herbir blok genellikle LUT, flip-flop ve diğer mantık bloklarını bağlayan kablolardan oluşur. Çarpıcı DSP bloklar gibi özel aritmetik devreler ve elde zincirleri (carry chain) olası uygulamalarda performans ve lojik yoğunluğu geliştirmek için modern FPGA'larda gömülü olarak bulunmaktadır.



Şekil 1.1 Temel FPGA mimarisi

Herhangi bir devrede yüksek performans sağlamanın anahtarı devrenin kritik yolunu (critical path) optimize ederek saat hızını arttırmaktır. Aritmetik devreler bu kritik yol için genellikle elde zincirlerinden geçer. Veriyolu hesaplamalarını desteklemek amacıyla pek çok FPGA, elde hesaplamalarını uygulamak için özel olarak optimize edilen özel kaynaklar içerir. Bu kaynaklar yonga (chip) alanında nispeten önemsiz bir artışla devre performansını arttırır. Elde zinciri bir hücreden diğer bir hücreye yaklaşık bir hesaplamayla bilginin geçmesine olanak sağlayan mantıksal bir yapıdır. Elde zincirleri FPGA yapılarında hızlı elde-yayılımını kolaylaştırmak için genellikle toplama devrelerinde kullanılır. Hızlı elde zinciri bitişik aritmetik hücrelerini bağlar, bu sayede önemli bir şekilde gecikme azalır.

Toplama devreleri pek çok aritmetik işlemde en çok kullanılan bloklardır ve toplama devrelerinin optimizasyonu birçok uygulama için daha yüksek performans sağlar. Günümüzde, FPGA'ların çoğunda toplama devrelerinin performansını arttırmak için hızlı elde zincir (fast carry chain) blokları bulunur. Bu yapı, ripple-carry toplayıcı kullanılarak yüksek performanslı toplama işleminin sentezini sağlar. Bu yüzden, en temel toplama devresi ripple-carry devresi genellikle hiçbir optimizasyon olmadan çok yüksek performanslarda çalışır [4-6], bu durum genellikle VLSI tasarım stillerinde en düşük performansa sahip ripple-carry toplayıcı devresinin zıttıdır. Hızlı elde mantığı FPGA mimarisinde aritmetik fonksiyonların verimliliğini arttırmak amacıyla yaygın şekilde kullanılmaktadır.

Tez çalışmamızın ilk bölümünde çeşitli toplama devrelerinin FPGA üzerinde performansları, kullandıkları lojik eleman sayıları ve güç-gecikme çarpanı (power-delay product – PDP) değerleri test edilmiştir. Çalışmamızda, Altera™ Quartus II yazılımı kullanılmıştır. Quartus II derleyicisi otomatik olarak tasarım süreci sırasında hızlı elde mantığını uygulayarak toplama devrelerini sentezler. En temel toplama devresi ripple-carry FPGA içerisinde bulunan bu özellik sayesinde oldukça hızlı çalışmaktadır.

Çalışmamızda, herbir toplama devresi öncelikle dört bit bloklanmıştır ve carry-skip, carry-lookahead, carry-select toplama devrelerinin performans analizi ölçülmüştür. Bu devrelerin hız performansına ek olarak, powerxdelay product ve kullanılan lojik eleman sayıları incelenmiştir. Test edilmek için Altera™ Stratix III ve Cyclone III mimarileri seçilmiştir. Ripple-carry toplayıcı çeşitli bit genişlikleri için Stratix III ve Cyclone III üzerinde en hızlı performans göstermesine rağmen, carry-select toplayıcı yüksek bit genişlikleri için daha yüksek performans sağlamıştır.

Tezin ikinci bölümünde ise çarpma-toplama devreleri (MAC – Multiply-Accumulate Unit) ele alınmıştır. Burada daha hızlı çarpma işlemi yapabilmek için Booth algoritmasından, yedekli sayı sistemlerinden, pipeline tekniğinden, çarpma işlemi sırasında oluşan kısmi ürünlerin azaltılması için gerekli olan yapılar anlatılacaktır. Bilinen FPGA'lar geleneksel olarak 4-giriş LUT elemanları kullanılarak inşa edilir ve sentezlenen lojik devre bir FPGA'daki 4-girişli hafıza blokları içerisine eşleştirilir. Son

yıllarda, Altera™ Stratix aileleri ve Xilinx™ Virtex aileleri gibi yüksek performanslı FPGA'lar 6-girişli LUT elemanları kullanarak inşa edilmektedir [6, 7]. Daha karmaşık yapı blokları daha az lojik derinlik ile eşleneceği için, daha yüksek girişli LUT yapıları daha hızlı lojik devrelerin sentezine olanak sağlamaktadır. Bu bölümde 16×16 bit ve 32×32 bitlik MAC üniteleri tasarlanmıştır.

Çarpma-toplama (multiply-accumulate MAC) üniteleri matris çarpma işlemleri, konvolüsyon, filtre işlemleri gibi matematiksel işlemlerde yaygın şekilde kullanılırlar. Bu yüzden MAC üniteleri, sayısal sinyal işlemciler ve genel işlemci mimarileri için önemli elemanlardır [1]. FPGA gibi yeniden yapılandırılabilir sistemler sinyal işleme uygulamalarında yaygın olarak kullanılmaktadır. Çalışmada yedekli sayı sistemleri kullanılmıştır. Yedekli sayı sistemleri giriş işaretinin bit genişliğinden bağımsız olarak toplama işlemleri için sabit gecikmeye sahip olmasından dolayı çarpma-toplama ünitesi için avantaj sağlamıştır [8-10]. MAC ünitesinde oluşan kısmi ürünlerin indirgenmesi için iki yaklaşım modeli kullanılmıştır. Birinci yaklaşım (6, 3) sayıcılar (counter) kullanarak kısmi ürünlerin indirgenmesi, ikinci yaklaşım ise multi-operand toplayıcı kullanarak kısmi ürünlerin indirgenmesi işlemidir. Her iki yöntemde çift elde saklama mimarisi kullanılmıştır. Tasarlanan 16×16 bit ve 32×32 bit yedekli sayı sistemi kullanılarak yapılan MAC üniteleri; Altera™ firmasının Stratix III FPGA ailesinde sentezlenmiş olup, çeşitli yazılımsal çarpıcı, donanımsal çarpıcı, pipeline edilmiş çarpıcı gibi yapıları kullanan çeşitli MAC ünitelerle performansları karşılaştırılmıştır.

Tezin akışı şu şekildedir: Bölüm 2, çeşitli toplama devreleri hakkında bilgi vermektedir. Bölüm 3, çeşitli toplama devrelerinin FPGA üzerindeki uygulama sonuçlarını vermektedir. Bölüm 4, yedekli sayı sistemleri hakkında bilgi vermektedir. Bölüm 5 çarpma devreleri, Booth Algoritması, çift elde-saklama tekniği, pipeline tekniği gibi çeşitli performans arttırmayı sağlayan yapılar hakkında bilgi vermektedir. Bölüm 6 önerilen yöntemi (yedekli çarpma-toplama ünitesi) sunmaktadır, bu bölümde (6, 3) sayıcı yapısı ve multi-operand toplayıcı yöntemleri ile önerilen MAC mimari kullanılması hakkında bilgi vermektedir. Bölüm 7 önerilen MAC mimarilerinin FPGA üzerinde uygulama sonuçlarını vermektedir. Bölüm 8 ise tez çalışması sonucunda elde edilen sonuçlar hakkında bilgi vermektedir.

BÖLÜM 2

TOPLAMA DEVRELERİ

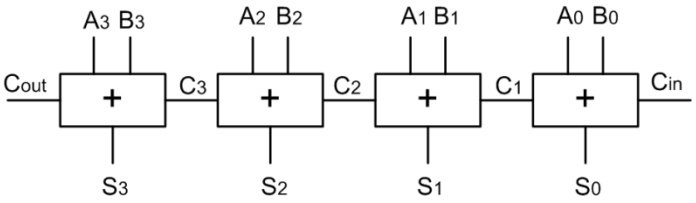
İki sayının toplamı neredeyse tüm aritmetik ünitelerde en çok kullanılan işlemdir. Toplama devreleri, toplama ve çıkartma işlemlerinin yanı sıra, çarpma ve bölme gibi daha karmaşık işlemlerde de kullanılır. Toplama devreleri birçok elektronik uygulamasında da kullanılmaktadır ve aritmetik ünitelerde hız belirleyici unsurdur. İki tabanındaki iki sayının toplamı, sayısal işaret işlemede, tümleşik devrelerde ve mikroişlemcilerde en temel aritmetik işlem olmaktadır. Yüksek hızlı paralel toplayıcılar için çeşitli toplama algoritmaları literatürde bulunmaktadır. Bu toplama algoritmaları içerisinde carry-skip, carry-lookahead, carry-select toplama algoritmalarının FPGA üzerinde performas ve kullandıkları lojik eleman sayıları test edilmiştir.

Toplama devrelerinde hız optimizasyonu için en temel faktör elde oluşumudur. Oluşan elde mümkün olduğu kadar kısa sürede çıkışa ulaşmalıdır. Bu süre devrenin gecikmesi olarak adlandırılır. Devrenin gecikmesini azaltmak için ödenen bedel ise daha çok lojik kapı kullanmak ve sonuç olarak devrenin kapsadığı alanın artması olacaktır. Bu bölümde kullanılan çeşitli toplama algoritmalarının çalışma prensipleri açıklanacaktır. Test edilen mimariler, ripple-carry (FPGA içerisindeki hızlı elde mantığı ile), carry-skip, carry-lookahead ve carry-select toplayıcı mimarileridir [11-20].

2.1. Ripple-Carry Adder (RCA)

n-bitlik bir toplayıcı, *n* tane bir bitlik toplayıcıyı kaskad bağlanarak yapılabilir. Bu tür toplayıcılara “Elde Zincirli Toplayıcı” ismi verilmektedir. Şekil 2.1’de, 4-bitlik bir ripple-carry toplama devresinin blok diyagramı verilmiştir. Bu devrede, her katın

elde çıkış (C_{out}) sinyali bir sonraki katın elde giriş (C_{in}) sinyalidir, bu nedenle toplamanın sonucu olan S sinyali, elde sinyallerinin oluşumuna bağlı olacaktır. Toplamanın sonucunun doğru bir şekilde oluşması için her katın elde sinyalinin oluşturulması gerekir, dolayısıyla bu da daha fazla gecikme olacağı anlamına gelmektedir. Ripple-carry adder, çok basit bir toplama devresidir ve bit uzunluğu fazla olan sayılar için kolayca tasarlanabilir, fakat bit sayısı arttıkça devrenin gecikmesi de orantılı olarak artmaktadır.



Şekil 2.1. 4-bit Ripple Carry Adder

Ripple-carry toplayıcı diğer toplayıcıların temel bir yapı bloğudur. Genel olarak diğer toplayıcılarla kıyaslandığına en kötü durum gecikmesi veren toplayıcıdır, ancak FPGA'lar söz konusu olduğunda hızlı elde mantığı (fast-carry logic) sayesinde diğer toplama devrelerine göre ripple-carry ile yüksek performans sağlamak mümkündür.

2.2. Elde Üretimi ve Yayılımı

Elde üretimi, toplama devrelerinde elde oluşumunu hızlandırmak için en çok kullanılan yöntemdir. Bu yöntemde ana fikir bir önceki kattan doğru eldenin gelmesini beklemeden elde sinyallerini giriş bitlerinden üretmektir. Elde yayılma sinyali (P_i) ve elde üretme sinyali (G_i), (2.2.1) ve (2.2.2) numaralı denklemlerde belirtildiği gibi tanımlanmaktadır:

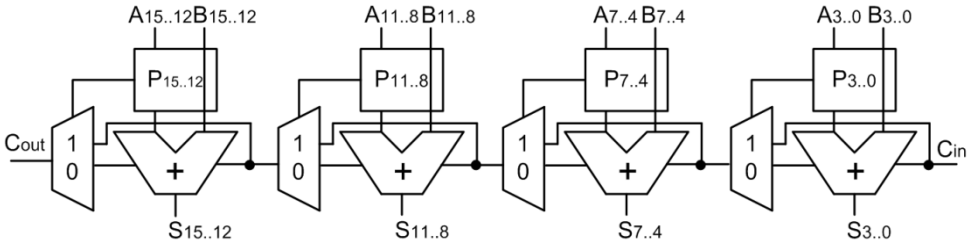
$$P_i = A_i \oplus B_i \tag{2.2.1}$$

$$G_i = A_i . B_i \tag{2.2.2}$$

Görüldüğü gibi yayılma ve üretme sinyallerinin her ikisi de sadece giriş bitlerine bağlıdırlar.

2.3. Carry-Skip Adder (CSKA)

Carry-skip adder, Şekil 2.2'de belirtildiği gibi devreye uygulanan carry biti birinci toplayıcı devresine uygulanmakta ve aynı zamanda multiplexer'a uygulanmaktadır. Carry-skip adder her bir elde zinciri için grup propagate sinyallerini hesaplar ve uzun elde dalgalanmasının üzerinden atlaması için bunu kullanarak kritik yolu (critical path) kısaltır. Böylece zaman gecikmesi azalır, devrenin çalışma frekansı artar.



Şekil 2.2. 16-bit Carry-Skip Adder

Şekil 2.2'deki dikdörtgen ile temsil edilen devre yapısı, propagate sinyalini hesaplar ve 4-bit grubun propagate sinyali için 4-bit girişli bir AND kapısı içerir. Çıkıştaki multiplexer ise grup propagate sinyali tam geldiğinde ya da ripple elde çıkışı olmazsa grup elde girişini seçer.

2.4. Carry-Lookahead Adder (CLA)

Carry-lookahead toplama devresi, carry-skip toplayıcı devresine benzemektedir. Burada farklı olarak yayılım (propagate) sinyalin yanı sıra üretim (genarate) sinyali de hesaplanır. Şekil 2.3'te carry-lookahead toplama devresi belirtilmiştir.

Carry-lookahead toplayıcının işleyişini daha iyi anlayabilmek için bir tam toplayıcı ünitesini göz önüne alırsak, tam toplayıcı devresinin toplamı (S) denklem (2.4.1) ve carry çıkışı (C_{out}) denklem (2.4.2)'deki gibi ifade edilebilir;

$$S = C_{in} \oplus (A \oplus B) \quad (2.4.1)$$

$$C_{out} = A.B + C_{in}.(A \oplus B) \quad (2.4.2)$$

P_i ve G_i sinyalleri kullanılarak tam toplayıcıdan oluşan her katın sonuç ve elde çıkışını yazarsak;

$$S_i = P_i \oplus C_i \quad (2.4.3)$$

$$C_{i+1} = G_i + P_i . C_i \quad (2.4.4)$$

elde edilir.

Bu denklemleri 4-bitlik toplayıcıya uygularsak;

$$C_1 = G_0 + P_0 C_0 \quad (2.4.5)$$

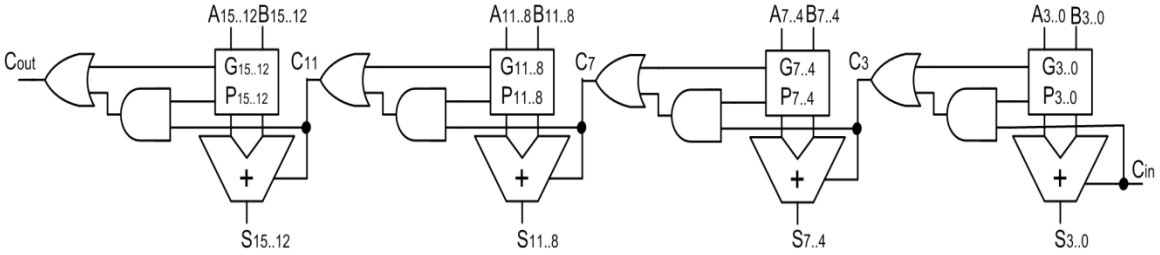
$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0 \quad (2.4.6)$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0 \quad (2.4.7)$$

$$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0 \quad (2.4.8)$$

Bu ifadelerden C_2, C_3 ve C_4'ün bir önceki katın elde çıkışından bağımsız oldukları anlaşılmaktadır. C_0 bilindiği anda C_2, C_3 ve C_4 de eş zamanlı olarak oluşmaktadır. Bu denklemlerin genelleştirilmiş ifadesi ise (2.4.9) numaralı denklemde verilmektedir.

$$C_{i+1} = G_i + P_i . G_{i-1} + P_i . P_{i-1} . G_{i-2} + \ldots P_i . P_{i-1} \ldots . P_2 . P_1 . G_0 + P_i . P_{i-1} \ldots . P_i . P_0 . C_0 \quad (2.4.9)$$

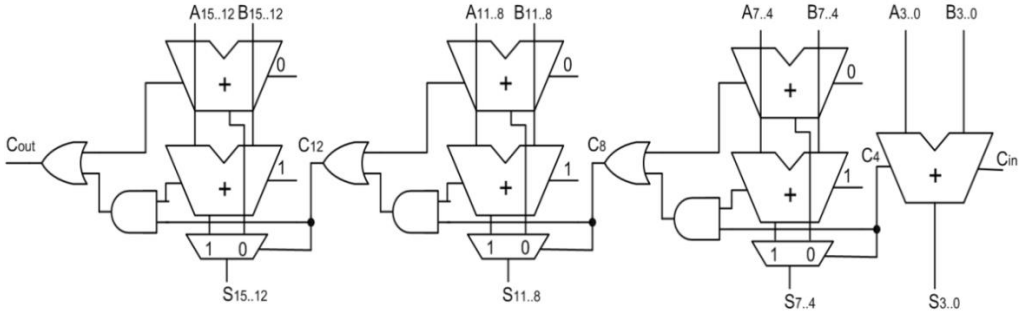


Şekil 2.3. 16-bit Carry-Lookahead Adder

2.5. Carry-Select Adder (CSLA)

Şekil 2.2 ve Şekil 2.3'te carry-skip adder ve carry-lookahead adder'ın kritik yolu (critical path) her 4-bit grup bitini hesaplamayı ve elde girişine (carryin) dayalı her bir gruptaki herbir bit için toplamı hesaplamayı gerektirir.

Carry-select yapısı için ise kritik yolu (critical path) hızlandırmada standart, bir mantık tasarım tekniği her iki muhtemel giriş için çıkışları önceden hesaplamaktır ve böylece iki çıkış arasında toplamı seçmek için bir multiplexer kullanılır.



Şekil 2.4. 16-bit Carry-Select Adder

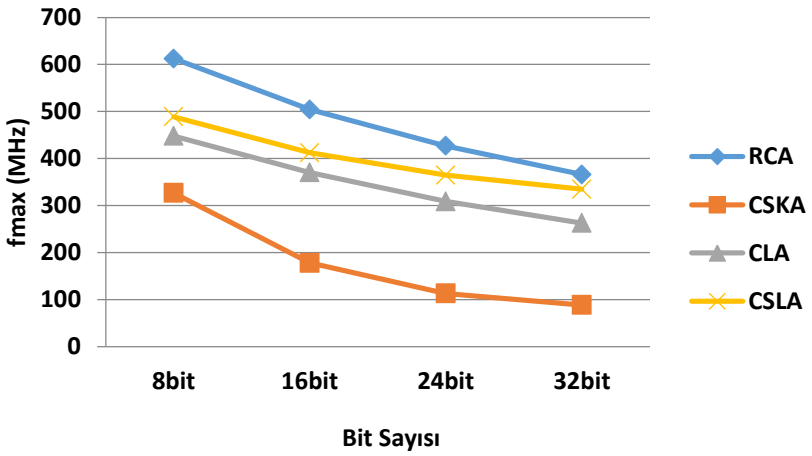
Şekil 2.4'de görüldüğü üzere, bir toplayıcı carryin'nin 0 olduğunu varsayarak toplamı hesaplar, diğer toplayıcı carryin'nin 1 olduğunu varsayarak toplamı hesaplar. Burada gerçek carry uygun toplamı seçen multiplexer'ı tetikler.

BÖLÜM 3

TOPLAMA DEVRELERİNİN ANALİZİ

Bir önceki bölümde bahsedilen toplama devreleri, Altera™ Stratix III ve Cyclone III FPGA mimarilerinde test edilmiştir. Çalışmada, öncelikle alternatif toplama mimarileri 4'er bitlik gruplamalarla gerçekleştirilmiştir. Şekil 3.1'de Stratix III'e ait ve Şekil 3.2'de Cyclone III'e ait farklı bit genişlikleri için hız performans grafikleri verilmiştir.

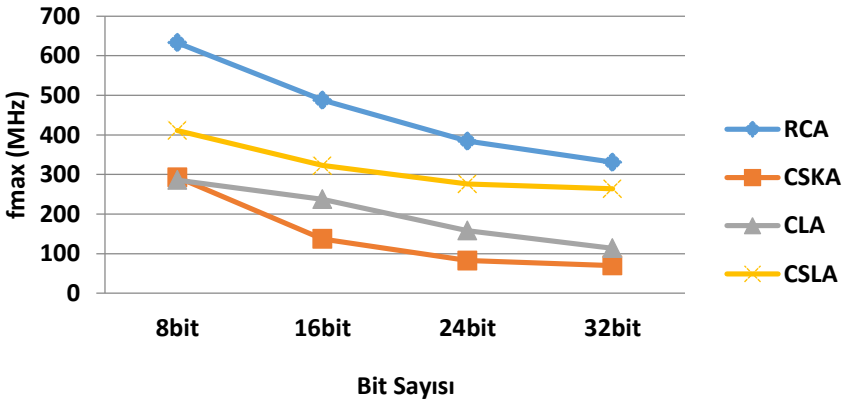
Şekil 3.1'de görüldüğü gibi, ripple-carry adder Stratix III mimarisi için hız performans grafiği verilmektedir. Şekilde de görüldüğü gibi, Stratix III mimarisinde en hızlı çalışan toplayıcı fast carry logic içeren RCA devresidir. Bunu, carry-select adder ve diğer toplama devreleri izlemektedir.



Şekil 3.1. Stratix III f_{max} değerleri

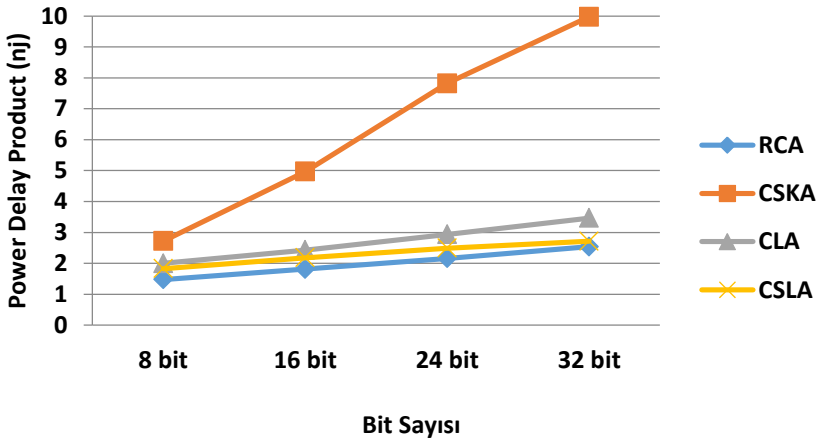
Şekil 3.2'de ise Cyclone III mimarisi için hız performans grafikleri görülmektedir. Şekilde de görüldüğü gibi, Cyclone III mimarisinde en hızlı çalışan

toplayıcı fast carry logic içeren RCA devresidir. Bunu, carry-select adder ve diğer toplama devreleri izlemektedir.

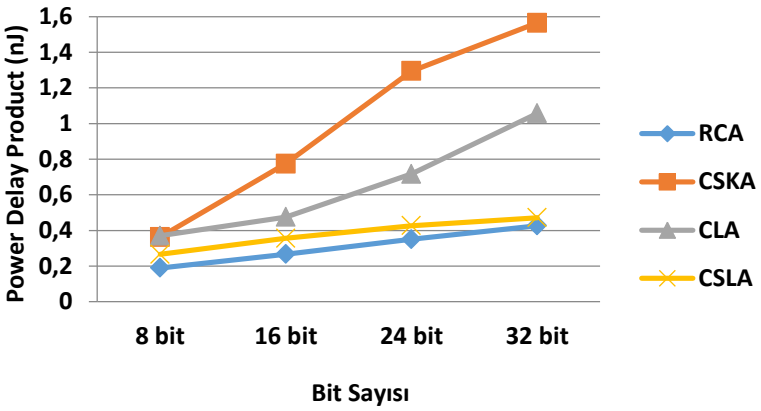


Şekil 3.2. Cyclone III f_{max} değerleri

Şekil 3.3'de görüldüğü gibi, Stratix III mimarisinin güç-gecikme çarpanı (powerxdelay product – PDP) değerleri gösterilmiştir. Şekil 3.4'de ise Cyclone III mimarisinin güç-gecikme çarpanı (powerxdelay product – PDP) değerleri gösterilmiştir.



Şekil 3.3. Stratix III powerxdelay product değerleri



Şekil 3.4. Cyclone III powerxdelay product değerleri

Sttarix III ve Cyclone III mimarileri için kullanılan toplam lojik eleman sayıları Tablo 3.1'de ve Tablo 3.2'de gösterilmiştir. En fazla lojik eleman kullanan devre carry-select adder toplayıcı devresidir. En az lojik eleman kullanan devre ise tahmin edileceği üzere riple-carry adder devresidir. Buna karşılık performans/alan açısından her zaman fast-carry logic içeren ripple-carry toplayıcı her zaman en iyi sonucu vermektedir.

Tablo3.1**.** Stratix III için kullanılan lojik elemanların sayısı

	8-bit	16-bit	24-bit	32-bit
RCA	9	17	25	33
CSKA	14	28	42	56
CLA	14	28	42	56
CSLA	20	49	82	107

Tablo 3.2. Cyclone III için kullanılan lojik elemanların sayısı

	8-bit	16-bit	24-bit	32-bit
RCA	26	50	74	98
CSKA	31	62	94	128
CLA	27	60	88	113
CSLA	33	69	101	132

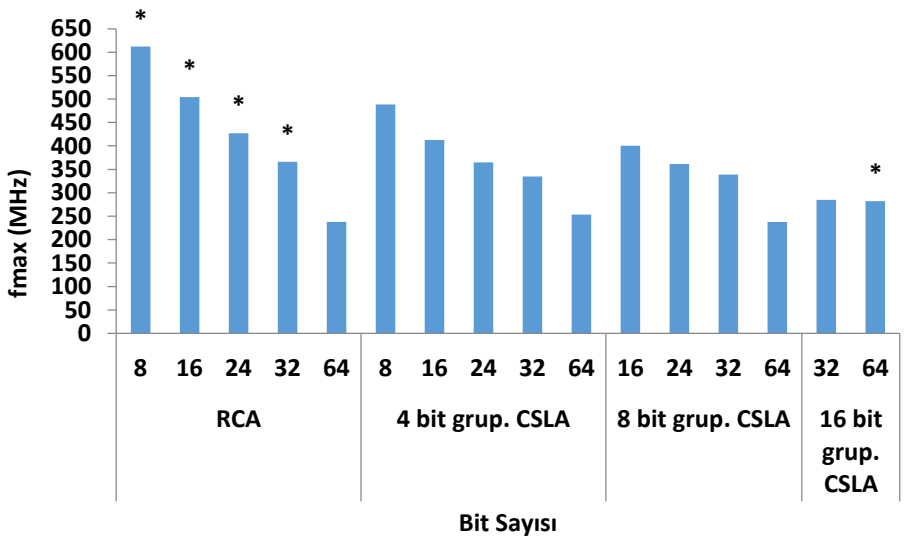
Ancak bu sonuçlar, alternatif toplama devrelerinin 4'er bitlik gruplandırılması sonucu elde edilen değerlerdir. Sonuçta, alternatif toplama devreleri içerisinde

FPGA'lar için en iyi alternatif yine carry-select adder olarak görülmektedir. Bundan sonraki deneylerde, farklı bit gruplamaları yapılarak carry-select toplayıcıların iyileştirilmesi ve RCA devrelerinden hızlı çalışabilecek yapıların Stratix III ve Cyclone III mimarisi için de elde edilmesi hedeflenmiştir. Her iki mimaride de hızlı-elde mantığından (fast carry logic) kaynaklanan durumdan dolayı RCA devrelerinin hızını geçmek adına farklı sayıda bitleri gruplandırarak deneyler tekrarlanmıştır.

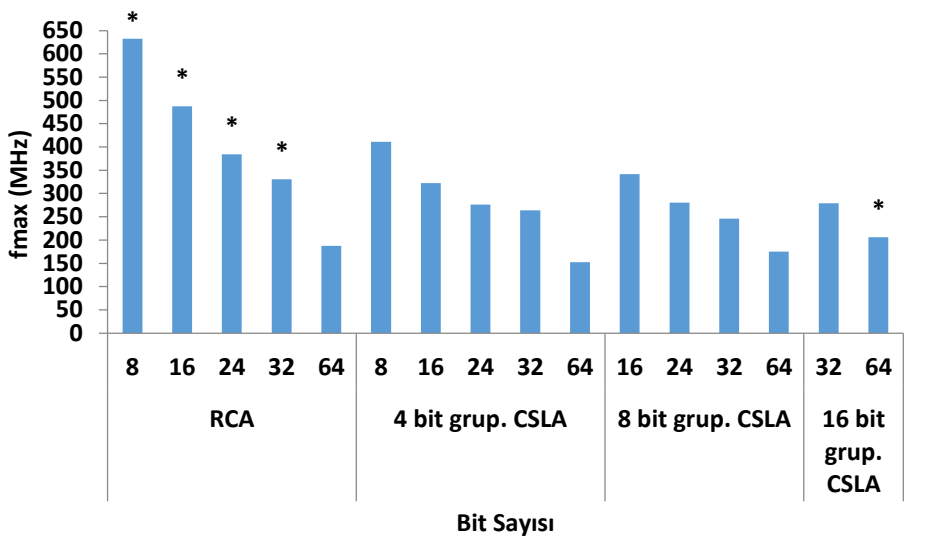
Ayrıca sabit bit blokları gruplamasının yanı sıra değişken bit gruplamada yapılmıştır. Sonuçlar yaklaşık olarak eşittir ve Tablo 3.3'de gösterilmiştir.

Tablo 3.3. Değişken bit gruplama yapılarak karşılaştırılması

Bit Gruplama	Stratix III	Cyclone III
	Hız (MHz)	Hız (MHz)
16-16-16-16	281,85	206,1
32-32	263,5	206,7
24-20-20	279,96	211,91
28-20-16	285,39	203,46



Şekil 3.5. Stratix III'de farklı bit boyutları için RCA ile CSLA'nın karşılaştırılması



Şekil 3.6. Cyclone III'de farklı bit boyutları için RCA ile CSLA'nın karşılaştırılması

Şekil 3.5 ve Şekil 3.6'da görüldüğü gibi, Stratix III ve Cyclone III mimarisinde çeşitli bit genişliklerinde (8-bit ve 16-bit) bit gruplandırma yapılarak sonuçlar gösterilmiştir. Burada bit sayısı arttıkça carry-select toplayıcı devresinin hızlı elde mantık özelliğine rağmen ripple-carry toplayıcı devresini geçtiği gözlemlenmiştir. 8 bit gruplama yapıldığında ripple-caryy toplayıcı hala geçilememektedir ve en nihayetinde 16-bit gruplamada 64-bit'te carry-select toplayıcı ripple-carry toplayıcıyı geçmektedir.

BÖLÜM 4

YEDEKLİ SAYI SİSTEMLERİ

Yedekli sayı sistemlerinde (redundant number systems) bir sayı alternatif yollarla temsil edilebilir. Yedeklilik bir sayıyı temsil ederken fazla bilgi verir. Ayrıca, yedeklilik toplama işlemlerinde elde yayılım zincirlerini kırmak için gereklidir ve toplayıcılar, toplama ağaçları, çarpıcılar gibi aritmetik devreleri tasarlarken çok kullanışlıdır.

İşaretli-sayı (signed-digit), aritmetik ve elde-saklama (carry-save) aritmetik yedekli sayı sistemlerinin en iyi bilinen örnekleridir. İşaretli-sayı aritmetiği ilk kez Avizienis tarafından ortaya çıkarılmıştır [21]. İşaretli-sayı sistemleri tamamen paralel toplamayı uygulamak amacıyla tasarlanmıştır. Avizienis elde yayılımını ortadan kaldıran bir aritmetik sistem önermiştir. Elde yayılımı sadece iki giriş sayısının herbir sayıyı sonuç toplamının bir fonksiyonunu hesaplayarak ortadan kaldırır. Bu sayı temsilinde yedekliliği ortaya çıkarmayı olası yapar. Kullanılan sayı toplamının tam ara temsili $x_i + y_i$ seçilir, böylece son toplama sonucu elde sinyallerini gerektirmeden ve üretmeden kullanarak üretilebilir [8-10].

Bilinen sayı sistemlerinde, sayılar $x_i \in \{0, 1,\ldots, r\text{-}1\}$ olarak gösterilir. Burada r radix değeridir, x_i sayının i. sayısıdır. Signed-digit sayı sisteminde x_i, $x_i \in \{(\overline{r-1}),(\overline{r-2}), \ldots,\bar{1}, 0, 1, \ldots(r - 1)\}$ değer aralığına sahiptir. Bu tasarımda negatif sayılar sayıların üzerinde eksi işaret ile gösterilir, örneğin $\overline{(r-1)}$, - (r - 1)'e eşittir ve $\bar{1}$, -1'e eşittir. Herbir sayı pozitif ve negatif değerlere sahip olabilir. Bu yüzden sistem signed-digit (SD) gösterimi olarak isimlendirilir [21-24].

İşaretli-sayı gösterimi sinyal gösterimlerinde fazlalığı verir. Örnek olarak -1, $0\bar{1}$ veya $\bar{2}1$ olarak temsil edilebilir. Radix 10'da ise (r = 10), iki-digit sayı X, $\overline{99} \leq X \leq 99$

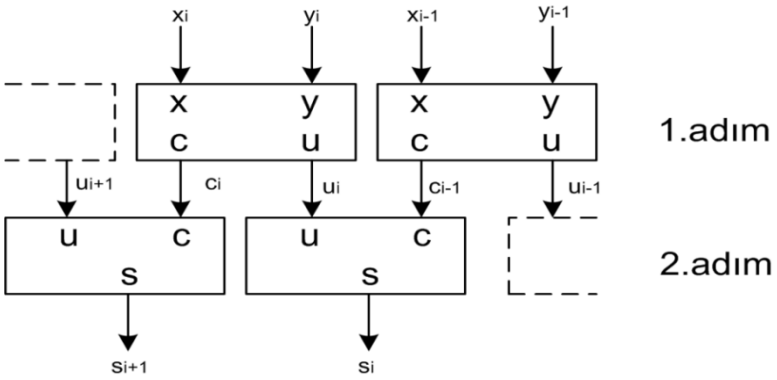
değer aralığına sahiptir. X iki-digit sayı olduğu için, herbir digit $\bar{9}$'dan 9'a kadar 19 farklı gösterime sahiptir. Burada X, $19^2 = 361$ gösterime sahiptir. X sayısı -99'dan 99'a kadar 199 tane değer içerir. Bu yüzden, bu sayı sistemindeki X, 361 – 199 = 162 yedekli sayıya sahiptir, bu durum X'in (162/199) yüzde 81 yedekli değere sahip olması anlamına gelir.

İşaretli-sayı sisteminin cebirsel ifadesi Denklem 4.1'de verilmiştir.

$$Z = \sum_{i=1}^{m} z_i r^i \tag{4.1}$$

Bu gösterimde, r radix olarak adlandırılan pozitif tamsayı değeridir ve herbir sayı z_i olarak belirtilmiştir. Radix r'li yedekli gösterimde, herbir sayı r değerlerinden daha fazlasını üstlenir. Bilinen sayı gösterimlerinde sayılar $z_i \in \{ 0, 1, \ldots(r - 1)\}$ tam olarak r değerlerini üstlenir. Radix ve dijitlerin sayısı olan z_i cebirsel değer $Z = 0$ için tek bir gösterim koşulu karşılanmalıdır. Cebirsel ifade Z sıfırsa ve onun işaretli-sayı gösteriminde tüm sayıların değeri $z_i = 0$ ise, o zaman $Z = 0$ koşulunu ispatlamak kolaydır. Cebirsel değer Z'nin işareti en önemli sıfır olmayan sayıların işareti tarafından belirlendiği açıktır. Benzer şekilde, Z'nin toplamaya göre tersi olan –Z'nin işaretli-sayı gösterimi Z'nin her sıfır olmayan z_i sayısının işaretini değiştirerek elde edilir.

Şekil 4.1 işaretli-sayı aritmetik sistemindeki toplam paralel toplama yaklaşımını göstermektedir. Şekilde, u_i ara toplam olarak isimlendirilmiştir. İki sayının x_i ve y_i'nin toplanması iki koşul sağlanmasıyla tamamen paraleldir. İlk olarak, ara toplam sayısı olan u_i toplanacak sayıların x_i ve y_i'nin sadece bir fonksiyonudur [8-10]. İkincisi, sonraki pozisyon c_i için elde sayısı x_i ve y_i'nin toplanacak sayıların sadece bir fonksiyonudur. Toplam paralel çıkarma işlemi x_i - y_i'nin toplamaya göre tersi ve x_i'nin paralel toplama işlemi olarak $x_i – y_i = x_i + (-y_i)$ gerçekleştirilir.



Şekil 4.1 İki işaretli-sayının paralel toplanması

4.1. Genelleştirilmiş Yedekli Sayı Sistemleri

Genelleştirilmiş işaretli-sayı sistemi { $-a$, $-a+1$, …, 0, …, $\beta-1$, β } rakam kümelerinin durumlarıyla ilgili bir sistemdir, burada $a \geq 0$, $\beta \geq 0$ ve yedekli sayı sistemleri için [23];

$$a + \beta + 1 > r \tag{4.1.1}$$

İfade (4.1.1)'de, r sayı göstergesi radix'tir. Bu durum, yedekli olmayan sayı sistemlerinde $a + \beta + 1 = r$ ile sonuçlanan durum için geçerlidir. $a = 0$ ve $\beta = r - 1$ ile bilinen radix-r sistemi genel r'nin tümleyen sayıları içindir ve 2'ye tümleyen sistem en bilinen olanıdır. Genelleştirilmiş bir işaretli-sayı sisteminin yedeklilik göstergesi denklemdeki gibi tanımlanır.

$$\rho = a + \beta + 1 - r \tag{4.1.2}$$

Bir sayı Denklem 4.1.3' deki gibi gösterilir.

$$Z = \sum_{i=1}^{m} z_i r^i \tag{4.1.3}$$

İfade (4.1.3)'te, r sayı sisteminin radix değeridir ve $z_i \in \{ -a, -a+1, \ldots, 0, \ldots, \beta-1, \beta \}$ sayı setlerini kapsar, burada $a \geq 0$, $\beta \geq 0$'dır. Parhami genelleştirilmiş işaretli-sayı sistemlerini bazı özel durumlarına göre sınıflandırmıştır [23].

1. İkili saklanan-elde (binary stored-carry (BSC)) : $r = 2$, $a = 0$, $\beta = 2$
2. Radix- r saklanan elde (Radix- r stored carry (SC)) : $a = 0$, $\beta = r$
3. İkili saklanan-ödünç veya ikili işaretli sayı (Binary stored-borrow or binary signed digit (BSB veya BSD)): $r = 2$, $a = \beta = 1$
4. Radix-r saklanan-borç (Radix-$r$stored-borrow (SB)) : $a = 1$, $\beta = r - 1$
5. İkili saklanan elde veya borç (Binary stored carry-or-borrow (BSCB)) : $r = 2$, $a = 1$, $\beta = 2$
6. Radix-r saklanan elde veya borç (Radix-r stored-carry-or-borrow (SCB)): $a = 1$, $\beta = r$
7. Minimum yedekli simetrik işaretli-sayı (Minimally redundant symmetric signed-digit) : $r \geq 4$, $2a = 2\beta = r$
8. Olağan işaretli-sayı (Ordinary signed-digit (OSD)) : $r \geq 3$, $1/2\ r < a = \beta < r$
    a. Minimum yedekli sayı : $a = \beta = \left\lfloor \frac{1}{2} r \right\rfloor + 1$
    b. Maksimum yedekli sayı : $a = \beta = r - 1$

İşaretli-sayı sistemleri $r>2$ için Avizienis tarafından ilk kez ortaya konulmuştur [21]. Ayrıca, Parhami [23] ikili durum ve diğer alternatif durumları içeren yedekli sayı sistemini açıklayan genelleştirilmiş işaretli-sayı sistemi gösterimini önermiştir. Avizienis'in işaretli-sayı sistemi Parhami'nin çalışmasında olağan işaretli-sayı sistemi (ordinary signed-digit number system (OSD)) olarak adlandırılır. Elde-saklama (carry-save) aritmetik aynı zamanda yedekli sayı sistemlerinden biridir ve işaretli-sayı sistemleriyle benzer özelliklere sahiptir.

4.2. Olağan İşaretli-Sayı Sistemi Gösterimi (Ordinary Signed-Digit Number Representation)

Bir sayı sistemine birkaç yedekli sayı eklemek aritmetik işlemlerin bazılarında faydalıdır; diğer bir taraftan, ihtiyaç duyulduğundan daha fazla kullanıldığı zamanda masraflıdır. İşaretli-sayı gösteriminde yedekliliğin miktarı Denklem 4.2.1'de belirtildiği gibidir.

$$Z_i \in \{\overline{a},\ \overline{a-1},\ \ldots \overline{1},\ 0,\ 1, \ldots, a\text{-}1\ , a\} \text{ ile } \left\lceil \frac{r-1}{2} \right\rceil \leq a \leq r-1 \tag{4.2.1}$$

Burada z_i bir sayının i. basamağıdır. Bir sayı Denklem 4.1'in basamak setlerini kullanarak 4.2.1'deki gibi gösterilebilir. Yukarıda anlatılan genelleştirilmiş yedekli sayı sistemlerinin tanımına göre, $a = \beta = \alpha$ olağan işaretli-sayı gösteriminde karşılık gelir. Basamak setlerinde en az r farklı basamaklarda radix r sistemindeki bir sayıyı temsil etmek için gereklidir. Ayrıca, herbir basamak z_i bu gösterimdeki herbir basamak için $\overline{\alpha} \leq z_i \leq \alpha$ aralığında olmalıdır. Bu gösterimde herbir basamağın $2\alpha + 1$ alternatifi vardır. Bu yüzden, $2\alpha + 1 \geq r$ eşitsizliği sağlanmalıdır ve eşitsizliğin alt sınırı [8, 21]'deki gibidir. Örnek olarak, radix-8 gösteriminde α'nın değeri (4.2.1)'e göre $4 \leq \alpha \leq 7$ arasındadır.

Yukarıda anlatıldığı gibi, işaretli-sayı gösteriminin (SD) en önemli avantajı toplama ve çıkarmada elde yayılım zincirlerini (carry propagation chains) ortadan kaldırmasıdır. Bu iki adımda [21]'deki gibi toplamayı gerçekleştirir.

Adım 1: ara toplam (u_i) ve elde basamağını (c_i) hesaplamak

$$u_i = x_i + y_i - rc_i \tag{4.2.2}$$

Burada,

$$c_i = \begin{cases} 1\ eğer\ (xi + yi) \geq \alpha \\ \overline{1}\ eğer\ (xi + yi) \leq \alpha \\ 0\ eğer\ |xi + yi| < \alpha \end{cases} \tag{4.2.3}$$

Adım 2: son toplamı hesaplamak

$$s_i = u_i + c_{i-1} \tag{4.2.4}$$

Örneğin, $r = 10$ ve $\alpha = 6$ olsun. Adım 1 şu şekilde sonuçlanacaktır.

$$U_i = x_i + y_i - 10c_i$$

c_i, adım 2'deki gibi hesaplanacaktır.

$$c_i = \begin{cases} 1 \; eğer \; (xi + yi) \geq 6 \\ \bar{1} \; eğer \; (xi + yi) \leq \bar{6} \\ 0 \; eğer \; |xi + yi| < 6 \end{cases}$$

4536 ve 1466 iki sayı bilinen şekilde toplanacak ise,

$$\begin{array}{r} 4\ 5\ 3\ 6 \\ +\ 1\ 4\ 6\ 6 \\ \hline 6\ 0\ 0\ 2 \end{array}$$

İlk basamaktan en son olana kadar elde yayılır. Belirtilen sayıları SD gösteriminde toplarsak;

$$\begin{array}{lr} 4\,5\,3\,6 & \\ +1\,4\,6\,6 & \\ \hline 0\,1\,1\,1 & c_i \\ +5\,\bar{1}\,\bar{1}\,2 & u_i \\ 6\ 0\,0\,2 & s_i \end{array}$$

Örnekte de görüldüğü gibi SD toplama şemasında hiçbir elde yayılımı görülmez.

SD toplama tasarımını uygularken, yeni eldenin üretilmemesini garanti etmek için, α'nın aralığı doğru bir şekilde seçilmelidir. Bu değer aşağıda belirtildiği gibi olmalıdır.

$$\left\lceil \frac{r+1}{2} \right\rceil \leq a \leq r-1 \tag{4.2.5}$$

4.3. İkili İşaretli-Sayı Mimarisi (Binary Signed-Digit Architecture)

İkili işaretli-sayı toplama mimarisinde, son toplam (s_i) ve ara toplam (u_i) hesaplanırken yeni elde üretileceği garanti değildir ve elde oluşumu son toplamın hesaplanmasında yeni elde meydana gelmemesini garanti etmek için Tablo 4.1'e göre tekrar ele alınmıştır [13]. Tablo 4.1'de belirtildiği gibi, ara toplam ve elde oluşumu *i*. basamak $x_i y_i$ ve önceki basamak $x_{i-1} y_{i-1}$'e göre yapılır.

Tablo 4.1. İkili SD sayılarının toplanması için değiştirilmiş kurallar

$x_i y_i$	00	01	01	$0\bar{1}$	$0\bar{1}$	11	$\bar{1}\bar{1}$
$x_{i-1} y_{i-1}$	Kullanıl-maz	İkiside $\bar{1}$ değil	En az biri $\bar{1}$	İkiside $\bar{1}$ değil	En az biri $\bar{1}$	Kullanıl-maz	kullanıl-maz
c_i	0	1	0	0	$\bar{1}$	1	$\bar{1}$
u_i	0	$\bar{1}$	1	$\bar{1}$	1	0	0

Radix-2 için, buradaki aritmetik ikili işaretli-sayı (binary-signed-digit (SD)) veya ikili depolanan-borç (binary stored-borrow) olarak adlandırılır ve kurallar olağan SD sistemlerden dallanmıştır. Tablo 4.1'de gösterilen ikili SD sayılarını toplamak için değiştirilmiş kurallar tek değildir. Çeşitli ikili SD toplama şemaları üretilebilir [25, 26]. SD ikili aritmetik uygulamaları [27-31] görülebilir.

İkili SD toplama şeması aşağıdaki örnek kullanılarak analiz edilebilir:

$$
\begin{array}{rl}
0\,\bar{1}\,1\,\bar{1}\,1\,1 & \\
+\,1\,0\,0\,\bar{1}\,0\,1 & \\
\hline
0\,0\,0\,\bar{1}\,1\,1 & c_i \\
1\,\bar{1}\,1\,0\,\bar{1}\,0 & u_i \\
\hline
1\,\bar{1}\,0\,1\,0\,0 & s_i
\end{array}
$$

Burada iki tane ikili işaretli-sayı, ikili SD toplama kuralları baz alınarak toplanmıştır. Ara toplam ve elde basamakları Tablo 4.1'e göre üretilmiştir. Son toplama

ara sonuçları toplayarak hesaplanmıştır. Bu şekilde tabloyu kullanarak, en son toplama işleminde yeni elde meydana gelmemiştir.

Yedekli sistemler ile ilgili bir diğer önemli konu format dönüşümüdür. SD sistemler için, yedekli ikiliden normal ikili sayıya dönüşüm önemsizdir. Herbir sayı pozitif ve negatif basamakla gösterildiği için, pozitif basamaklardan negatif basamakların çıkarılması normal ikili dönüşümün sonucudur. Bu uygulamaların format dönüşümü için hız geliştirici gibi diğer algoritmik yaklaşımlar vardır. Dönüşüm işleminin hızını geliştirmek için, diğer algoritmik yaklaşımlar [28, 29]'da önerilmiştir. İşaretsiz normal ikili sayıdan SD ikili sayıya dönüşüm de önemsizdir, öyle ki, normal ikili sayıların sıfır olmayan tüm bitleri SD sayıların pozitif basamaklarına eşittir ve negatif sayıların tümü sıfıra eşittir. İkinin tümleyeni ikili sayı için, işaret biti SD sayıların negatif basamaklarına karşılık gelene eşittir ve diğer tüm pozitif basamaklar normal ikili sayıların sıfır olmayan bitlerine eşittir.

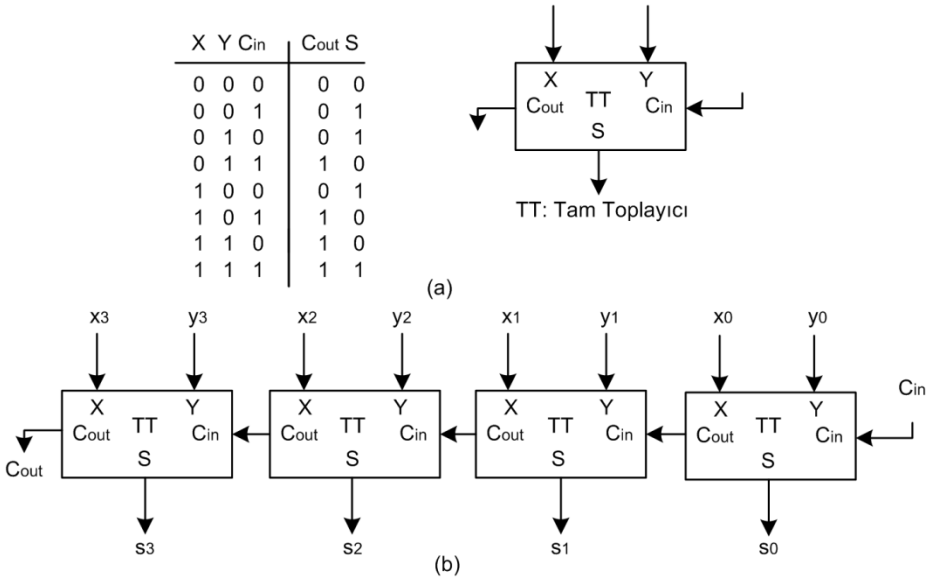
4.4. Elde-Saklama Aritmetiği (Carry-Save Arithmetic)

Elde-saklama toplama kavramı kırk yıldan daha fazla süredir kullanılmaktadır. Wallace görünüşte (pseudo) toplayıcı olarak elde-saklama aritmetiğinin kullanımı için klasik tam toplayıcı (full-adder) mimarisini adlandırmıştır, elde-saklama toplayıcı üç giriş operandı ikiye indirger [32-34]. Aynı zamanda Wallace [32] çarpma devrelerinin kısmi ürünlerin indirgenmesi için paralel indirgeme ağacını tasarlamıştır. Elde-saklama indirgemesi çoğunlukla çarpmaların kısmi ürünlerini indirgemek için kullanılan sayısız aritmetik uygulamalarında kullanılır. Elde-saklama aritmetik ayrıca yedekli ikili sistem olarak da adlandırılabilir [23, 35].

Elde-saklama aritmetiği tanımlanmadan önce, en temel toplama tasarımı olan tam toplayıcıdan burada bahsedilmelidir. Bir tam toplayıcı üç giriş operandı x, y, z'ye ve iki çıkışa s ve c'ye sahiptir. İşlem denklem (4.4.1) de tanımlandığı gibidir.

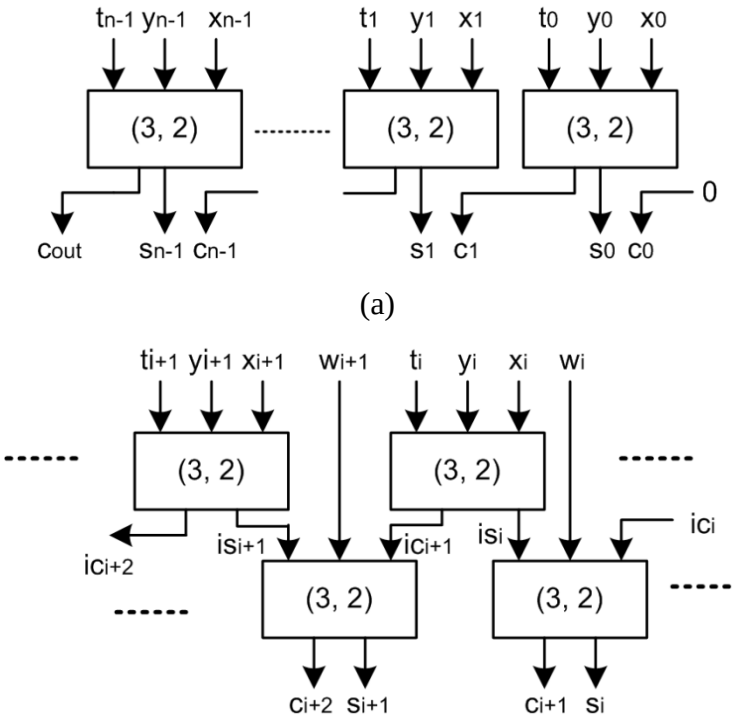
$$x + y + c_{in} = 2c_{out} + s \tag{4.4.1}$$

Burada denklem (4.4.1) de, "+" operatörü aritmetik toplamadır ve herbir *x*, *y*, *z*, *c* ve *s* ∈ {0, 1} değerlerine sahiptir. Bir tam toplayıcının tek bit hücresi ve çoklu bit tam toplayıcı kullanarak toplama işlemleri Şekil 4.2' de sırasıyla gösterilmiştir.



Şekil 4.2. (a) Tam toplayıcı gösterimi (b) tam toplayıcıları kullanarak toplama işlemi

Sayıcı (counter) devreleri ikili sayıların çoklu-operand toplama işlemleri için kullanılır. En basit sayıcı devresi bir tam toplayıcı olan (3, 2) sayıcıdır. Tek farklılık uygulamadadır, yani tüm *x*, *y* ve c_{in} girişleri benzer fonksiyonelliğe sahiptir ve Şekil 4.3'de gösterilen durumun aksine elde yayılım hareketi yoktur. Bir (3, 2) sayıcı sıfır olmayan girişlerin sayısını sayar ve iki dijit ikili bir sayıya sonucu kodlar. Sıfır olmayan giriş basamaklarının sayısı en fazla 3'tür. Bu yüzden Şekil 4.2'de gösterildiği gibi sonucu ifade etmek için en az 2 çıkışa ihtiyaç duyulur. Devrenin gecikmesi operandların bit genişliklerinden bağımsız toplam olarak tek hücre (3, 2) sayıcı gecikmesidir (tek FA gecikmesi).



Şekil 4.3 elde-saklama yapısı: (a) normal ikili girişler (b) yedekli girişler

Bir elde-saklama sisteminde, herbir basamak toplam (sum) ve elde (carry) olmak üzere iki sayının toplamı şeklinde gösterilir. Fakat son ikili sonuç, yani bu toplam ve eldenin toplanması gerekmediği sürece hesaplanmaz. İki sayının *S* ve *C*'den oluşan (3, 2) sayıcının çıkışı elde-saklama sayısına bir örnektir. Elde-saklama gösteriminde, herbir basamak şu şekilde temsil edilir:

$$z_i = \{s_i, c_i\} \tag{4.4.1}$$

burada, toplam *Z* ile gösterilir ve $Z = \{S, C\}$ şeklinde temsil edilir.

Elde-saklama aritmetik bir basamağın $z_i \in \{0, 1, 2\}$ olarak tanımlanan ve *Z*'nin herbir basamağının $z_i = s_i + c_i$ eşit olduğu bir yedekli sistemdir. Ayrıca elde-saklama aritmetik SD sisteme benzer elde yayılımından bağımsız toplama işlemi sağlar. Elde-saklama formatında iki keyfi sayı Şekil 4.3.b'de gösterilen (4, 2) indirgeme şemasını kullanarak toplanabilir. (4, 2) indirgeme iki aşama (3, 2) sayıcı indirgeme şeması tarafından gerçeklenir. Ayrıca, VLSI tasarım için daha sıkı (4, 2) indirgeme şemaları

daha hızlı performanslar mevcuttur [16]. Elde-saklama aritmetik pek çok elde-bağımsız aritmetik uygulamalarda SD sistemlere göre avantajlıdır. Ayrıca belirtilmelidir ki, elde-saklama aritmetik sistem [10]'da daha az yedekliliği sağlayan daha yüksek radix elde-saklama aritmetiği uygulamak olasıdır.

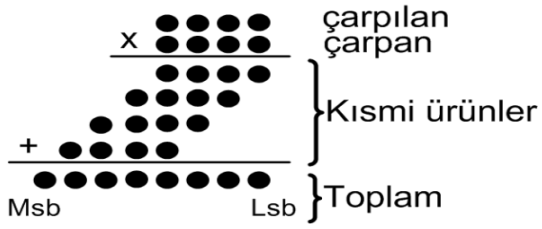
Normal ikili sayıdan elde-saklama dönüşümü ya da tersi şu şekildedir:

1. Normal bir ikili sayı X (4.4.1)'de belirtildiği gibi $S = X$ ve $C = 0$ olarak gösterilebilir.
2. İki normal ikili sayının toplamı X ve Y hiçbir ek donanım uygulamaksızın $S = X$ ve $C = Y$ şeklinde tek bir elde-saklama sayısı olarak gösterilebilir.
3. Üç normal ikili sayı Şekil 4.3.a'da gösterildiği gibi elde-saklama formatına dönüştürülebilir (elde saklama formatta çıkış ile üç normal ikili sayının toplanması).
4. Bir elde-saklama formatlı sayı iki operandlı bir toplayıcı kullanarak normal ikili sayıya dönüştürülebilir, örneğin ripple-carry toplayıcı, burada sonuç s_i ve c_i elemanlarını toplamaktır, yani sonuç = $S + C$ şeklindedir.

BÖLÜM 5

ÇARPMA DEVRELERİ

Çarpma devresi günümüzde sayısal işaret işleme, mikroişlemciler gibi birçok uygulamada kullanılmaktadır. Çarpma devresi oluşturmanın en temel yöntemi ikilik tabandaki sayının geleneksel yöntemle çarpımından yola çıkmaktır. Bu yöntemde alt alta toplamalar ve kaydırmalar mevcuttur.



Şekil 5.1. Genel çarpma işlemi

Burada çarpma işlemini şu iki basamaktan oluşur:

1. Kısmi çarpımların elde edilmesi
2. Kısmi çarpımların kaydırılarak toplanması

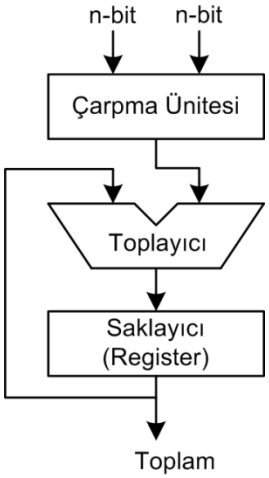
İkilik tabanda çarpma işleminin mantıksal VE işlemine denk olduğu yapılan işlemlerde görülmektedir. Dolayısıyla kısmi çarpımların bulunması işlemi, çarpılanı sıra ile çarpanın bitleri ile mantıksal VE işleminden geçirmektir. Daha sonra kısmi çarpımların sütunları birbirleri ile toplanacaktır ve gerekli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

Bu bölümde işaretli çarpma-toplama (MAC) devresi incelenmiştir ve daha hızlı sonuç almak için gerekli teknikler kullanılmıştır. Burada kısmi ürünleri yarıya indiren Booth algoritması ve oluşan kısmi ürünleri daha hızlı toplayabilmek için yedekli sayı sisteminde bahsedilen sayıcı ve compresser'lar, Quartus II yazılımına ait sentez araçlarından multi-operand toplayıcı, pipeline tekniği kullanılarak daha hızlı çarpma-toplama devresi amaçlanmıştır.

5.1. Çarpma-Toplama Devresi (MAC Unit)

Çarpma-toplama (multiply-accumulate MAC) üniteleri matris çarpma işlemleri, konvolüsyon, filtre işlemleri gibi matematiksel işlemlerde yaygın şekilde kullanırlar. Bu yüzden MAC üniteleri sayısal sinyal işlemciler ve genel işlemci mimarileri için önemli elemanlardır [36-40]. Şekil 5.2'de genel MAC ünitesi gösterilmiştir. Sahada programlanabilir kapı dizileri FPGA gibi yeniden yapılandırılabilir sistemler sinyal işleme uygulamalarında yaygın olarak kullanılmaktadır.

Tez çalışmasında düşük gecikmeli bir MAC ünitesi yüksek performanslı 6-giriş LUT tabanlı FPGA'lar için tasarlanmıştır. Bilinen FPGA'lar geleneksel olarak 4-giriş LUT elemanları kullanılarak inşa edilir ve sentezlenen lojik devre bir FPGA'daki 4-girişli hafıza blokları içerisine eşleştirilir. Son yıllarda, Altera™ Stratix aileleri ve Xilinx™ Virtex aileleri gibi yüksek performanslı FPGA'lar 6-girişli LUT elemanları kullanarak inşa edilmektedir. Daha karmaşık yapı blokları daha az lojik derinlik ile eşleneceği için, daha yüksek girişli LUT yapıları daha hızlı lojik devrelerin sentezine olanak sağlamaktadır.



Şekil 5.2. Genel MAC ünitesi

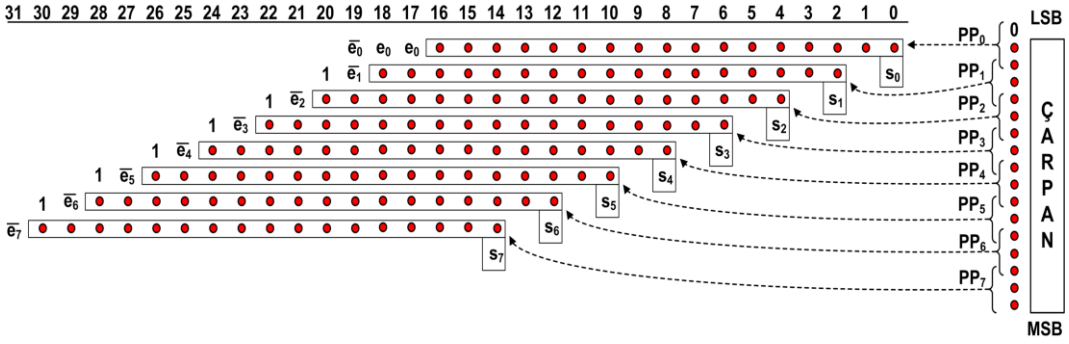
5.2. Booth Algoritması

Çarpma devrelerinde oluşan ara toplamları (partial product) azaltmak amacıyla kullanılan bir algoritmadır. Radix 2^r çarpıcılar çarpıcıların r bitlerine dayanan N/r ara toplam oluşturur [12, 41-44]. Daha az ara toplamın oluşması daha hızlı toplama işlemini sağlar. Örneğin, radix-4 çarpıcılar N/2 ara toplam oluşturur. Her bir ara toplam X'in bit çiftlerine dayanan 0, M, 2M, 3M'dir. 2M'yi hesaplamak basit bir kaydırma işlemidir. Fakat ara toplam üretimine başlamadan önce, 3M = M + 2M'den oluştuğu için ara bir toplama işlemi gerektirir, bu da sistemi yavaşlatacağı için istenen bir işlem değildir. Bunun yerine değiştirilmiş Booth kodlaması negatif ara ürünleri kullanarak (3M = 4M - M ya da 2M = 4M - 2M) 3M işlemini oluşturmaksızın daha yüksek radix paralel işleme olanak sağlar [45]. Fakat radix-4 çarpıcı dizilerindeki 4M ağırlığı dört kez taşıyan dizinin gelecek sırasındaki M'ye eşittir. Bu yüzden, ara toplamlar önceki çiftten en önemli bit ile birlikte bit çiftlerini alarak seçilir. Tablo 5.1 ara toplamların nasıl seçildiğini göstermektedir. Negatif ara toplamlar çarpılan sayının ikiye tümleyeni alınarak üretilir.

Tablo 5.1. Radix-4 değiştirilen Booth kodlama değerleri

Girişler			Ara ürünler
X_{2i+1}	X_{2i}	X_{2i-1}	PP_i
0	0	0	0
0	0	1	M
0	1	0	M
0	1	1	2M
1	0	0	-2M
1	0	1	-M
1	1	0	-M
1	1	1	-0(=0)

Değiştirilen Booth kodlamasında, toplanacak ara toplamların sayısı (n+2) +1 dir. Şekil 5.3'de değiştirilen Booth kodlama verilmiştir.



Şekil 5.3. işaretli çarpma işlemi için Radix-4 Booth kodlanan ara toplamlar

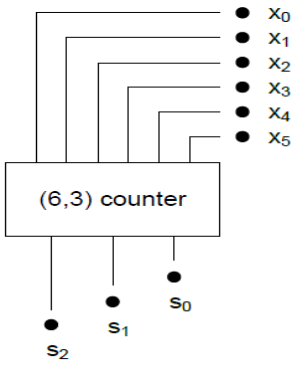
5.3. (6,3) Sayıcı ve Altı Binary Sayının Eşzamanlı Olarak Toplanması

Çoğunlukla kullanılan sayıcı devreleri (3,2), (7,3) ve (15,4) sayıcılardır, burada ilk terim girişlerin sayısını için kullanılır ve ikinci terim çıkışların sayısını temsil eder. (6,3) sayıcılarda, altı binary giriş üç bit çıkışa karşılık gelmektedir [46].

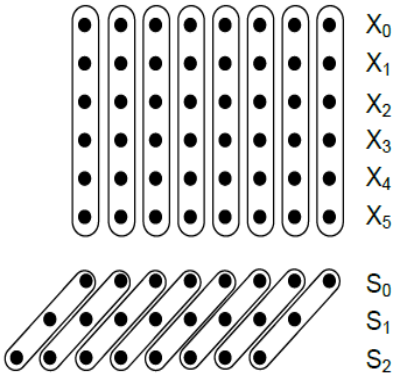
Geleneksel carry-save yapılarında, carry-save sonucu iki digit setleri tarafından tutulur, $Z = S + C$. Burada, fazlalık toplam Z (5.3.1) ve (5.3.2) ifadeleriyle tanımlanır.

$$S_a = S_0$$
$$S_b = S_1 << 1$$
$$S_c = S_2 << 2 \qquad (5.3.1)$$
$$Z_i = \{S_{ai}, S_{bi}, S_{ci}\}$$
$$Z = S_a + S_b + S_c \qquad (5.3.2)$$

Burada "<< a " operatörü a tarafından sola kaydırmayı tanımlar. Bu ifade ile fazla toplamın herbir dijiti üç bit set $Z_i = \{S_{ai}, S_{bi}, S_{ci}\}$'nin birleşiminden oluşur. Sonuç olarak Z_i'nin herbir biti 0 ($S_{ai} + S_{bi} + S_{ci} = 0$) ile 3 ($S_{ai} + S_{bi} + S_{ci} = 3$) arasında değerlere sahip olabilir. Bu yapıda, herbir dijit üç bit tarafından temsil edilir. Geleneksel carry-save sisteminde herbir dijit iki bit tarafından temsil edilir. (6,3) sayıcıyı kullanan tek bit ve çoklu-bit toplama Şekil 5.4.a ve Şekil 5.4.b'de belirtilmiştir



Şekil 5.4.a (6,3) sayıcı tek bit yapısı



Şekil 5.4.b (6,3) sayıcılar kullanılarak birçok terimin toplamı

Çalışmamızda 16×16 bit ve 32×32 bit MAC ünitelerinde oluşan kısmi ürünleri (6, 3) sayıcılar kullanılarak hızlı performans sağlanmıştır.

5.4. Pipeline Tekniği

Aritmetik işlemleri hızlandırmak için kullanılan yöntemlerden birisi de pipeline tekniğidir. Girişte uygulanan sinyallerin çıkışa daha hızlı bir şekilde varması için

varolan gecikmenin azaltılması gerekir. Pipeline tekniği giriş çıkış arasını belli sayıda böler. Bu sayıların herbiri arasına bir register ilave edilir, böylece devrelerin yüksek hızda çalışması sağlanır [47, 48].

5.5. Çift Elde Saklamalı Mimari (Double Carry Save Adder)

Elde-saklamalı aritmetikte, her bir sayı iki sayının birleşmesi olarak temsil edilir ($Z = \{S, C\}$). Çift elde-saklama aritmetikte ise şu şekilde kullanılır.

$$Z^a = S_0 \tag{5.5.1}$$

$$Z^b = S_1 << 1 \tag{5.5.2}$$

$$Z^c = S_2 << 2 \tag{5.5.3}$$

$$Z = (Z^a, Z^b, Z^c) \tag{5.5.4}$$

"<< k" operatörü k-bit sola kaydırmayı temsil eder. Yukarıda belirtildiği gibi herbir çıkış bir bit sırasıyla kaydırılır. Yedekli sayı olan Z üç adet ikili sayının bir birleşimidir (Z^a, Z^b, Z^c). Her ne zaman normal ikili dönüşüm gerekirse, $S = Z^a + Z^b + Z^c$ şeklinde üç operand toplama işlemi gerekir.

Çift elde-saklamalı gösterimde herbir sayı üç adet radix-2 sayısının bir birleşimidir ve herbir yedekli sayı Z_i'nin sayı seti {0, 1, 2, 3} değerlerine sahiptir. Burada herbir sayı 0 ile 3 arasındadır ($z^a_i + z^b_i + z^c_i = 0$ - $z^a_i + z^b_i + z^c_i = 3$). Z_i'nin herbir dijiti üç bit tarafından gösterilir [49, 50].

BÖLÜM 6

ÖNERİLEN ÇARPMA-TOPLAMA DEVRELERİ

Tez çalışmamızda yedekli sayı sistemini ve çift elde saklamalı mimarisini kullanan farklı yedekli MAC üniteleri tasarlanmıştır. MAC üniteleri içerisinde çarpma işleminden üretilen kısmi ürünler Booth algoritması ile yarıya indirgenmiştir. Bu aşamadan sonra şu iki yöntemle kısmi ürünler azaltılmıştır:

1. (6, 3) Sayıcı kullanılarak oluşan kısmi ürünlerin indirgenmesi,
2. Multi-operand toplayıcı kullanılarak kısmi ürünlerin indirgenmesi.

6.1. (6,3) Sayıcı Kullanarak İndirgeme

Burada 16×16 bit ve 32×32 bit yedekli MAC ünitesi çıkışında oluşan kısmi ürünleri azaltmak için (6, 3) sayıcılar kullanılmıştır. Çok kullanılan radix-4 değiştirilmiş Booth kodlamayı [12, 41-44] kullanarak kısmi ürün üretilmesinden sonra, çarpma ve toplama işlemleri (6, 3) sayıcı ağaçları altında birleştirilmiştir. Sayıcılar bir önceki bölümde anlatıldığı gibi genellikle çok sayıda toplanacak sayının indirgenmesinde kullanılmaktadır [46]. Şekil 5.4.a ve 5.4.b'de tek-bit ve çoklu-bit (6, 3) sayıcı şeması sırasıyla gösterilmektedir. Şekil 5.4.a'da, altı giriş operandı 0 ve 6 arasında çıkış değerlerine sahip olabilecek şekilde 3-bitle çıkışta kodlanır (yani $(000)_2$ - $(110)_2$). Şekil 5.4.b'de, toplanılacak sekiz-bit altı tane binary giriş operandı (6, 3) diziler ile temsil edilir ve toplanılacak giriş operandlarının sayısı altıdan üçe azaltılır. (6, 3) sayıcıların kullanımı, 6-giriş LUT tabanlı FPGA'lar için avantajlıdır.

Genellikle, elde saklama indirgeme ağacı elde-ilerleme (carry-propagate) şemalarından daha fazla alan gerektirdiği için FPGA sistemlerde kısmi ürün

indirgemesinde tercih edilmez [39]. Ayrıca, hızlı elde zincirleri (fast carry chain) elde-ilerleme tasarımından daha popüler yapan etkili performans sağlar. Fakat elde-yayma tasarımı bit genişliğinde doğrusal bağımlılığa sahiptir ve bit genişliği arttıkça, elde-ilerleme tasarımı yetersiz kalır. Elde-saklayıcı tasarım, elde-yayma tasarımlarıyla kıyaslandığında alan gereksinimini ikiye katlar.

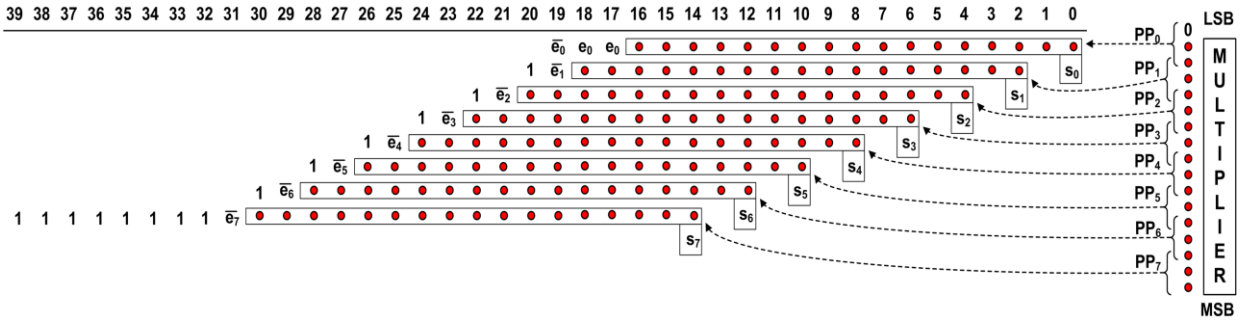
Alan gereksinimi asıl sorun değilse, (4, 2) compressor'ler, binary signed-digit toplama ağaçları, sayıcı ağaçları gibi yedekli indirgeme ağaçları FPGA çarpmada kullanılabilir. Tablo 6.1 çeşitli indirgeme operatörlerinin performans şemalarını ve alan gereksinimlerini gösterir. Çeşitli indirgeme operatörleri arasında, (6, 3) sayıcılar Tablo 6.1'de gösterildiği gibi 6-giriş LUT tabanlı FPGA'lar için en iyi performansı vermektedir. Tablo 6.1'e göre, (4, 2) compressor'lar ve (6, 3) sayıcılar aynı miktarda kaynak gerektirmektedir, fakat (6, 3) sayıcılar çok daha hızlıdır. Binary signed-digit tasarımı daha fazla alan gerektirir ve işlemin gecikmesi (4, 2) compressor ve (6, 3) sayıcı dizileri ile karşılaştırıldığında daha fazladır. Kıyaslanan indirgeme operatörleri arasında, (6, 3) sayıcı dizisi, (4, 2) indirgeme dizisi ve binary signed-digit operatörlerinin hepsi giriş operandlarını yarıya indirir.

Tablo 6.1. Çeşitli indirgeme operatör metrikleri

Bit genişlikleri	Binary signed-digit		(4,2) compressor		(6, 3) counter	
	Gecikme (ns)	Alan (LUT)	Gecikme (ns)	Alan (LUT)	Gecikme (ns)	Alan (LUT)
16-bit	2,77	80	1,84	48	1,15	48
24-bit	3,45	120	1,91	72	1,13	72
32-bit	3,27	160	1,95	96	1,15	96
64-bit	3,19	320	1,93	192	1,15	192

Bir MAC ünitesi bir çarpma yolundan, bir kaydedilen toplama çıkışından oluşur [36, 40]. Önerilen mimarinin çarpma işlemi için, standart Booth kodlama şeması 40-bit çıkışa işaret genişlemesi ile kullanılmıştır. Şekil 6.1'de 40 bite genişletilmiş 16×16 çarpma işlemi için kullanılan değiştirilmiş Booth algoritması gösterilmiştir. Birçok çarpma-toplama işlemlerinden sonra toplama çıkışı taşmamasına rağmen, işaret genişlemesi yapılmıştır. İşaret genişlemesi ile radix-4 değiştirilmiş Booth kodlama

Şekil 6.1'de gösterilmiştir. Burada, herbir hat (line) üretilen kısmi ürünleri temsil eder. Kısmi ürünler Tablo 4.1'de gösterilen kurallara göre üretilir. Burada, herbir kısmi ürün çarpma bitlerine göre üretilir (x_{2i+1}, x_{2i},x_{2i-1}).



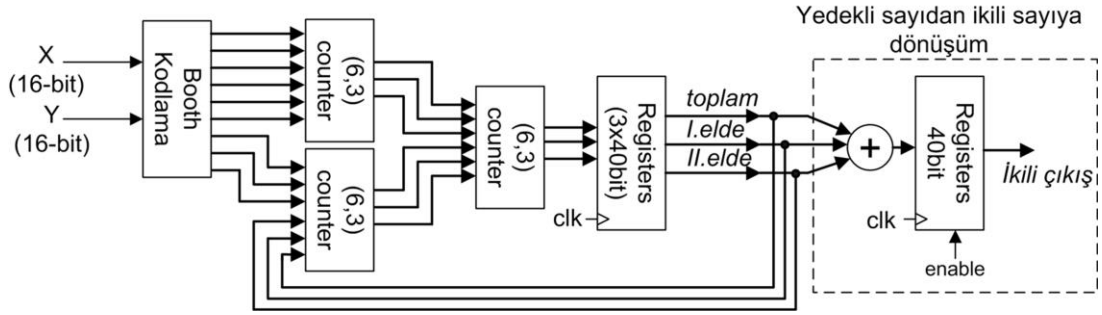
Şekil 6.1. 40 bite genişletilerek yapılan Booth algoritması

Herbir hattaki e_i ilgili kısmi ürünün işaret bitine eşittir. Tablo 4.1'e göre üretilen Y'nin işareti tersine çevrilmiş ise, herbir hattaki s_i 1, Y'nin işareti pozitif ise s_i 0'dır. Booth kodlama aşamasının son kısmi ürünü (Booth kodlama çıkışındaki 8. hat) birçok çarpma-toplama işlemlerinden sonra çıkışın taşmaması amacıyla 16x16 = 32 bitten 40 bite kadar çarpma çıkışını genişletmek için 1'ler ile doldurulmuştur. Booth kodlama şemasının detayları [12, 41-44]'de açıklanmıştır.

Booth kodlama kısmi ürünlerin sayısını yarıya indirgemesine (yani 8 kısmi ürüne) rağmen, bir diğer operand olarak en son hattaki s_7 birlikte toplanılacaktır. Sonuç olarak, Booth kodlamanın çıkışında toplanacak 9 operand olacaktır. Herbir toplama operandları için boş hatlar kolay donanım uygulaması için 0'lar ile doldurulmuştur, yani kısmi ürün indirgeme şeması için 9 satırda 40 bit vardır.

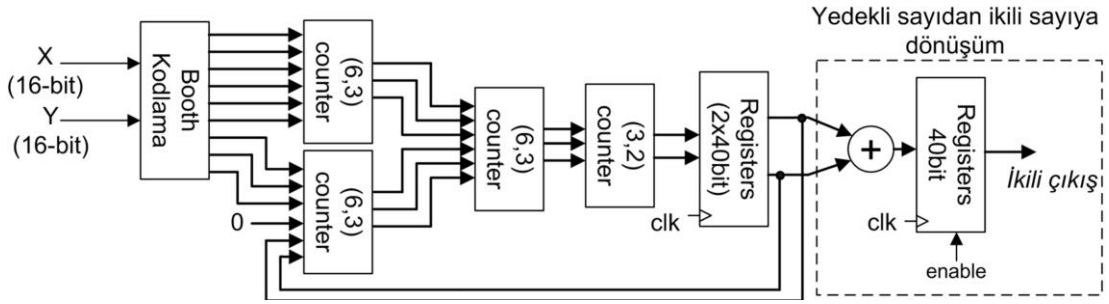
Önerilen MAC ünitesi Şekil 6.2'de gösterilmiştir. Burada, kalın hatlar 40 bit operandları temsil eder. MAC ünitesinin çıkışı (6, 3) sayıcılardan dolayı çift elde saklamalı formatta temsil edilir. Toplama işlemi için elde edilen çıkış , üç bit çıkış (6, 3) sayıcılar ile geri beslenmiştir. Önerilen mimariyi kullanarak, herbir çarpma-toplama işlemi için kritik yol bir Booth kodlama ve iki (6, 3) sayıcı aşamasından oluşur. Hem Booth kodlama hem de (6, 3) sayıcı aşamaları tek LUT lojik derinlik maliyeti ile uygulandığı için, toplam sistemin kritik yolu sadece 3 LUT gecikmedir. Sonuç olarak,

minimum register gecikmesiyle yüksek çıkış elde edilmiştir. Şekil 6.2'de gösterildiği gibi, yedekli çarpma-toplama işlemi açık bir pipeline aşamasına sahip değildir. Hatta işlemde pipeline olmadan, yapı bilinen çarpma-toplama yapılarının aksine yüksek hızda çalışır. Yedekli sayıdan normal ikili dönüşüm (redundant to binary conversion) için tekrar üç girişli bir toplama devresi gerekmektedir. Yedekli sayıdan ikili sayıya dönüşüm aşaması Şekil 6.2'de noktalı kutu içerisinde gösterilmiştir. Ayrıca, Şekil 6.3 bilinen yedekli elde-saklamalı yapı ile 16×16 bit MAC ünitesi tasarlanılmıştır. Bilinen elde-saklamalı çıkışa sahip yedekli MAC ünitesiyle önerilen çift-elde saklamalı çıkışa sahip yedekli MAC ünitesi karşılaştırılmıştır.



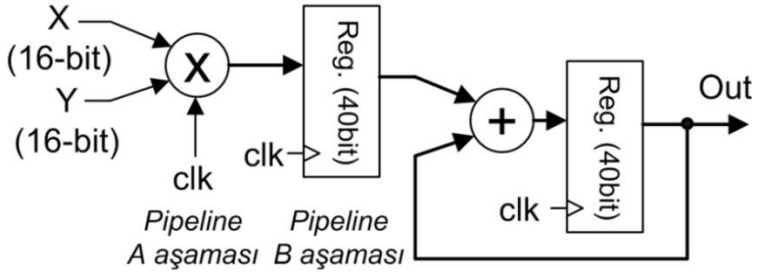
Şekil 6.2. (6, 3) sayıcı kullanılarak önerilen 16×16 bit yedekli MAC ünitesi

Normal ikili dönüşüm, genelde dijital sinyal işleme uygulamalarının çoğunda (her çarpma toplama işlemi için) gerekmez. Örnek olarak, 100-tap FIR filtre için, yedekli sayıdan ikili sayıya dönüşüm her 100 çarpma-toplama işlemlerinden sonra gereklidir. Bu koşul matris çarpma, genel konvolüsyon işlemleri gibi diğer dijital sinyal uygulamaları için geçerlidir.



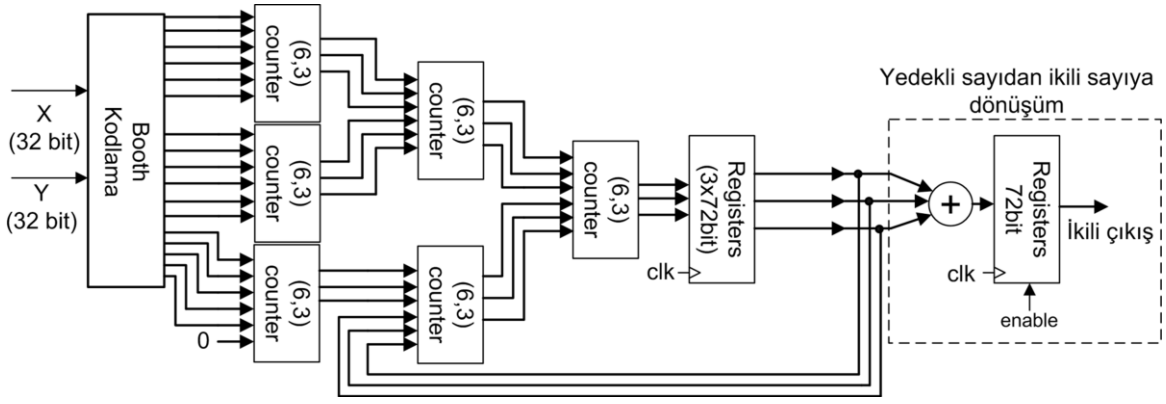
Şekil 6.3. Bilinen yedekli CSA yapısıyla tasarlanılan MAC ünitesi

Performans ve kaynak gereksinimi kıyaslaması için bilinen çarpma-toplama üniteleri donanım ve yazılım çarpıcı pipeline aşamalı çarpıcı gibi MAC üniteleri kullanılarak yapılmıştır. Şekil 6.4'de önerilen yapının kıyaslanması için yapılan çeşitli pipeline aşamaları ile genel bir pipeline edilmiş MAC ünitesi görülmektedir. İlk aşama pipeline olarak temsil edilen çarpma işleminde görülmektedir. İkinci aşama B aşaması, çarpma ve toplama işlemi arasındaki pipeline aşamasıdır. Hızlı elde zincirleri 40-bit toplama için etkili bir şekilde hızlı işlem sağladığı için, toplama işlemi için bit seviyesinde pipeline aşamasına gerek yoktur. Verilen pipeline aşamalarına göre, performans değerleri ölçülüp Tablo 5.3'te gösterilmiştir. Kaynak kullanımı uyarlanabilir LUT birimleri (ALUT) ve register sayısı (Reg.) tarafından belirlenmiştir.



Şekil 6.4. Çeşitli pipeline kombinasyonlarıyla düzenli MAC üniteleri

32×32 bit çarpma devresi içinde aynı senaryo kullanılarak hızlı bir MAC ünitesi tasarlanmıştır. Bu çarpma devresinde ise, elde oluşmasını engellemek için çıkışlar 72 bite genişletilmiştir.



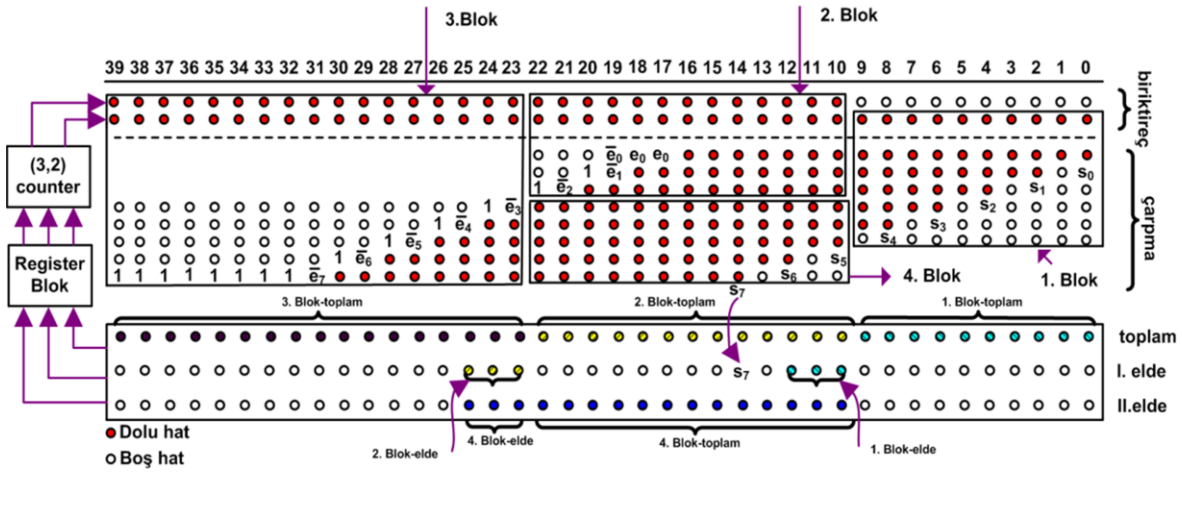
Şekil 6.5. (6, 3) sayıcı kullanılarak önerilen 32×32 bit yedekli MAC mimari

Booth kodlamadan oluşan kısmi ürün sayısı son işaret biti ile 17’ dir. 16x16 bit ünitesinde olduğu gibi çıkışlar (6, 3) sayıcılar ile indirgenmiştir. Şekil 6.4’te önerilen mimari belirtilmiştir. Şekil 6.4’de verilen çeşitli pipeline kombinasyonları da 32×32 bit yedekli MAC ünitesi için kullanılmıştır.

6.2. Multi-operand Toplayıcı ile Kısmi Ürünleri Azaltmak

Multi-operand toplayıcı ile çok sayıda operand paralel şekilde toplanır [51]. Quartus II programında multi-operand toplayıcı sekmesini seçerek ilgili operandlar kolaylıkla toplanır. 16×16 bit ve 32×32 bit çarpma işlemlerinden sonra, oluşan kısmi ürünleri toplamak üzere multi-operand toplayıcı ile toplama işlemleri hızlı şekilde gerçekleştirilmiştir.

16×16 bit çarpma-toplama işlemi için öncelikle Booth algoritması kullanılmış, dolayısıyla 9 tane kısmi ürün oluşmuştur ve çıkışta herhangi bir elde oluşmaması için çıkışa 8 adet bir biti kullanılarak işaret genişlemesi yapılmıştır, dolayısıyla Şekil 6.1’de olduğu gibi çıkış 40 bittir. Bu oluşan kısmi ürünleri indirgemek için belli sayıda multi-operand toplayıcıyı kullanarak hızlı bir 16×16 bit yedekli MAC ünitesi tasarlanmıştır. Şekil 6.6’de bu senaryo gösterilmiştir.



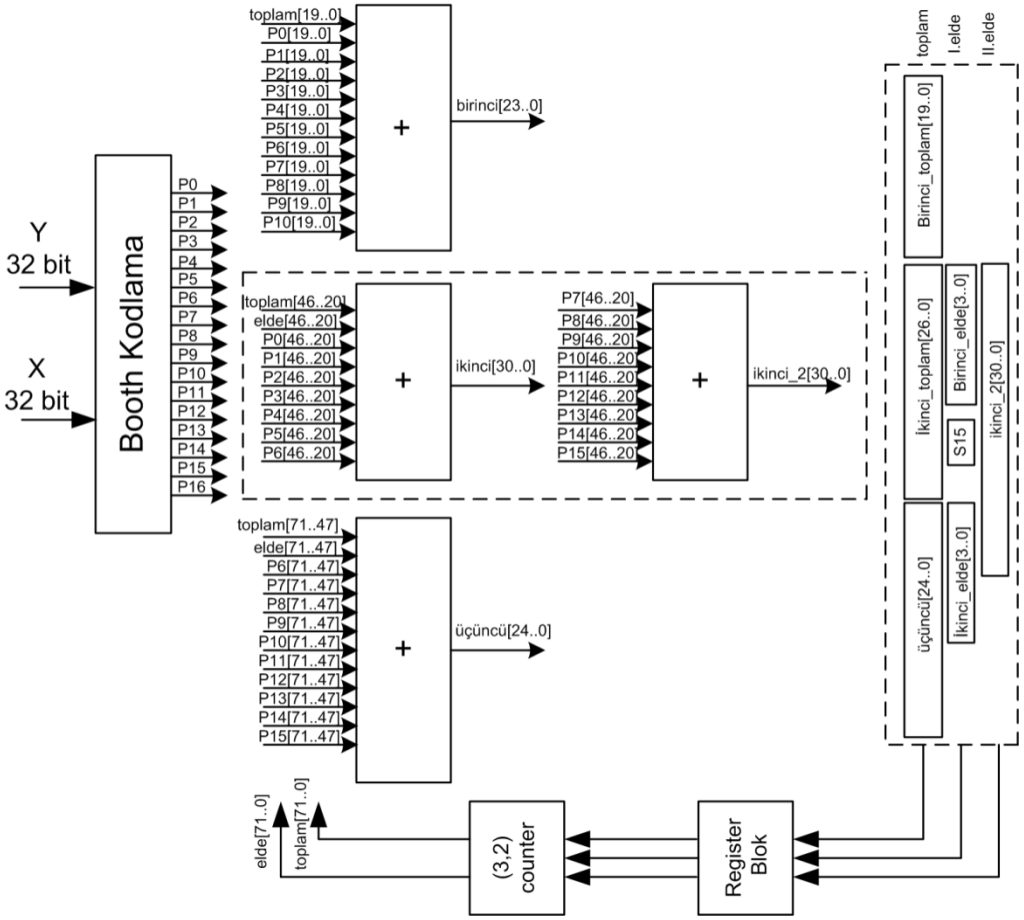
Şekil 6.6. Multi-operand toplayıcı kullanılarak önerilen yedekli 16×16 bit MAC ünite

16×16 bit MAC ünitesinde Booth algoritması çıkışında 9 tane kısmi ürün oluşmaktadır ve MAC ünitesinden de toplam ve elde olmak üzere 2 tane daha kısmi ürün gelecektir, dolayısıyla 11 tane kısmi ürün multi-operand toplayıcıyla toplanacaktır. Fakat S_7 biti Şekil 6.6'da belirtildiği gibi I.elde bloğunda 14. bitine taşınarak multi-operand toplayıcıya ilave edilmemiştir, dolayısıyla toplam 10 adet 40 bitlik uzunluğunda operand vardır.

Burada öncelikle 40 bit uzunluğundaki bit genişliği üç kısma ayrılmıştır, ilk blok 10 bit (9-0 bit arası), ikinci blok 13 bit (22-10 bit arası), üçüncü blok ise 17 bit (39-23 bit arası) genişliklerine ayrılmıştır. Çünkü multi-operand toplayıcı belirli bit sayısına kadar olan bit genişliklerinde hızlı şekilde toplamaktadır, bu özellikten dolayı tek seferde ya da iki kısma ayırarak multi-operand toplayıcı ile toplanmamıştır. Bir de Şekil 6.6'ya dikkat edilirse ikinci kısımda çok sayıda operand olduğundan bu sistemi yavaşlatmaktadır, bunun önüne geçebilmek için ikinci kısım ortadan ikiye olacak şekilde bir bloğa daha ayrılmıştır.

İlk blokta elde oluşmayacağından dolayı elde kısmının ilk 10 biti multi-operand toplayıcıya dahil edilmemiştir. İlk blokta toplanan sonuca göre toplam kısmı, toplam bloğunun ilk 10 bitine, elde kısmı ise I.elde olarak adlandırılan bloğun 12-10 bit arasına yerleştirilmiştir. İkinci blokta ise aynı şekilde toplam kısmı, toplam bloğunun 22-10 bitleri arasındadır, elde kısmı ise I.elde bloğunun 25-23 bitleri arasındadır. Üçüncü blokta ise toplam kısmı doğrudan toplam bloğunda 39-23 bitleri arasındadır, bu blokta işaret genişlemesi yapıldığından dolayı elde oluşmaz dolayısıyla elde ihmal edilmiştir. Dördüncü blok ise toplam kısmı ve elde kısmı doğrudan II.elde bloğunda 25-10 bitleri arasındadır. Bu belirtilen toplam, I.elde ve II.elde blokları register bloklarından geçirildikten sonra (3, 2) sayıcıdan geçirilerek elde ve toplam kısmını oluşturulmuştur. Bu da sistemi geri besleyerek accumulate (biriktireç) kısmını oluşturmuştur. En nihayetinde bu oluşan üç blok üçlü bir paralel-toplayıcı kullanılarak yedekli sayıdan ikili sayıya dönüşüm yapılmıştır. Böylece 16×16 bit çift-elde saklamalı yedekli MAC ünitesi tasarlanmıştır. Aynı işlemler 32×32 bitlik MAC ünitesi içinde uygulanmıştır. Burada ise Booth algoritmasından elde oluşmaması için çıkış 72 bite genişletilmiştir.

Yukarıda belirtilen senaryo uygulanarak MAC ünitesi tasarlanmıştır. Şekil 6.7'de ilgili MAC ünite belirtilmiştir.



Şekil 6.7. Multi-operand toplayıcı kullanılarak önerilen 32×32 bit yedekli MAC ünite

.

6.3. Diğer Çarpıcılar

MAC ünitesindeki çarpıcılar lojik eleman kullanarak sentezlenmişse, yapı yazılımsal çarpıcı (soft multiplier) olarak isimlendirilir [52, 53] ve çarpıcı fiziksel bağlantılı çarpma ünitelerini kullanarak sentezlenmişse, yapı sıkı çarpıcı (hard multiplier) olarak isimlendirilir [54, 55]. Hem yazılımsal çarpıcı hem de donanımsal çarpıcı programın Megawizard kısmı kullanılarak tasarlanmıştır.

BÖLÜM 7

ÇARPMA-TOPLAMA DEVRESİ DENEYLERİ

Bir önceki bölümde bahsedilen çarpma-toplama devreleri, 16×16 ve 32×32 bit yedekli MAC ünite tasarlanmıştır. Oluşan kısmi ürünler iki şekilde kullanılarak indirgenmiştir. Bunlardan birincisi (6, 3) sayıcılar kullanılarak işlemleri azaltmak, bir diğeri ise olan multi-operand toplayıcı kullanmaktır. Burada kısmi ürünler çeşitli bit genişliklerine ayrılarak multi-operand toplayıcı ile toplanmıştır. Tasarlanılan devreler çeşitli şekilde pipeline kullanılarak devrelerin hız performasları incelenmiştir. Tablo 7.1'de gösterildiği gibi 16×16 bit, Tablo 7.2'de de 32×32 bit MAC üniteleri yazılımsal ve donanımsal çarpma devreleriyle kıyaslanarak hız performansları, kullanılan kaynak miktarları belirtilmiştir. Devreler Altera™ Stratix III serisinden EP3SL70F780C4 kiti seçilerek test edilmiştir.

Pipeline edilmemiş çarpıcılar için, önerilen yapılar Tablo 7.1 ve Tablo 7.2'de gösterildiği gibi çok daha hızlıdır. Donanım çarpıcılar kullanılırsa, yüksek performansa ulaşılır. Fakat önerilen yapılar daha hızlıdır. İlk önerilen (6,3) sayıcı yapısını kullanarak yapılan indirgeme işleminde pipeline yoktur; fakat yedekli çıkıştan ikili sayıya dönüşüm için tek bir fazla saat gecikmesi gerekebilir, böylece saat gecikmesi Tablo 7.1 ve Tablo 7.2'de (1+1) olarak temsil edilmiştir. İkinci önerilen yaklaşımda ise multi-operand toplayıcı kullanılarak yapılan indirgeme işlemlerinde ise sadece multi-operand toplayıcılarda bir clock kullanılmıştır. Karşılaştırılan devreler yüzeysel pipeline aşamaları kullanılarak yapılmıştır ve önerilen şema pipeline edilmiş çarpma-toplama ünitelerinden daha hızlıdır.

Tablo 7.1. 16×16 bit çeşitli MAC ünitelerin kıyaslanması

Yapı	Kaynak Kullanımı	Performans (MHz)	Saat Gecikmesi
Yazılımsal Çarpıcı (pipeline'sız)	258 ALUT + 72 Reg.	124	1
Yazılımsal Çarpıcı 1-seviye pipeline: (B =1)	259 ALUT + 104 Reg.	160	2
Yazılımsal Çarpıcı 2-seviye pipeline: (A =1; B =1)	263 ALUT + 190 Reg.	210	3
Donanımsal Çarpıcı (pipeline'sız)	1 DSP Blok 40 ALUT + 40 Reg.	141	1
Donanımsal Çarpıcı 1-seviye pipeline: (B =1)	1 DSP Blok 40 ALUT + 80 Reg.	261	2
CSA yapısıyla tasarlanılan yedekli MAC unite (Şekil 6.3)	493ALUT + 150 Reg.	265	1+1
Önerilen (6, 3) sayıcı ile tasarlanılan yedekli MAC ünite (Şekil 6.2.)	418 ALUT + 178 Reg.	**286**	1+1
Önerilen Multi-operand toplayıcı kullanılarak tasarlanılan yedekli MAC ünitesi (Şekil 6.6)	336 ALUT + 279 Reg.	**330**	2+1

Tablo 7.2. 32×32 bit çeşitli MAC ünitelerinin kıyaslanması

Yapı	Kaynak Kullanımı	Performans (MHz)	Saat Gecikmesi
Yazılımsal Çarpıcı (pipeline'sız)	897 ALUT + 136 Reg.	89,77	1
Yazılımsal Çarpıcı 1-seviye pipeline: (B =1)	898 ALUT + 200 Reg.	109,77	2
Yazılımsal Çarpıcı 2-seviye pipeline: (A =1; B =1)	918 ALUT + 633 Reg.	169,2	3
Donanımsal Çarpıcı (pipeline'sız)	4 DSP Blok +72 ALUT + 72 Reg.	104,79	1
Donanımsal Çarpıcı 1-seviye pipeline: (B =1)	4 DSP Blok +72 ALUT + 72 Reg.	185,53	2
Önerilen (6, 3) sayıcı ile tasarlanan yedekli MAC Ünite (Şekil 6.5.)	1427 ALUT + 334 Reg.	**212,27**	1+1
Önerilen Multi-operand toplayıcı kullanılarak tasarlanılan yedekli MAC ünitesi (Şekil 6.7)	1099 ALUT + 628 Reg.	**216,92**	2+1

BÖLÜM 8

SONUÇLAR

Yapılan çalışmada, FPGA üzerinde çeşitli toplama devreleri ve çarpma-toplama devrelerinin hız performansları incelenmiştir. Bunun yanında kullanılan lojik elemanlar, güç-hız çarpan (PDP) değerleri incelenmiştir.

Toplama devrelerinde sonuçlar genel tümleşik devre uygulamalarından farklıdır. FPGA ailelerinin çoğunda standart mimari olan hızlı elde zinciri, küçük bit genişliklerinden daha ılımlı bit genişliklerine kadar tüm gruplamalarda ripple-carry toplama devresinde hızlı performans sağlamaktadır. Performans analizleri Stratix III ve Cyclone III mimarilerinde incelenmiştir. Performans/alan metriği açısından hızlı elde zinciri içeren ripple-carry toplayıcı her zaman en iyi sonucu vermektedir. Daha yüksek bit gruplanarak yapılan toplama devreleri için carry-select toplama devresi daha iyi performans sağlamaktadır. Uygulanan mimarilerde bit genişlikleri arttırıldıkça, toplama devrelerinin hız performansıda artar. Çalışma sonucunda FPGA'lar için ripple-carry toplayıcısına en iyi alternatif carry-select toplayıcı devresi olarak görülmüştür. Her iki mimaride farklı sayıda bit gruplandırılması sonucunda 16 bit gruplandırma yapılarak 64 bitte carry-select toplayıcı en yüksek hızı sağlamıştır. Ancak diğer bit genişlikleri için hızlı-elde mantığı ile ripple-carry en yüksek performansı vermekte olup farklı toplama devrelerine ihtiyaç görülmemektedir.

Çarpma-toplama devresinde ise 16×16bit ve 32×32 bit çarpma-toplama devreleri çeşitli yöntemlerle FPGA üzerinde hız performansları, kullandıkları kaynak miktarları incelenmiştir ve alternatif bir yedekli sayı tabanı kullanılarak yüksek performanslı çarpma-toplama (MAC) ünitesi gerçeklenmiştir. Önerilen MAC ünitesinde kısmi ürünlerin indirgenmesi için iki yaklaşım kullanılarak tasarlanmıştır. İlk önerilen yapı (6,

3) sayıcı kullanılarak oluşan kısmi ürünlerin indirgenmesi, ikinci yaklaşım ise multi-operand toplayıcıları parçalı olarak kullanarak oluşan kısmi ürünlerin indirgenmesidir. İlk önerilen yapıda elde ilerlemesinden (carry-propagation) bağımsız olduğundan, avantajlıdır. Bu yapı simetrik (6, 3) sayıcı ağacı ve değiştirilmiş Booth kodlaması ile gerçeklenmiştir ve sistemin çıkışı çift-elde saklamalı formattadır. Tasarlanan MAC ünitesi düşük saat gecikmesiyle yüksek çıkışa sahiptir. Elde yayılımından bağımsız olduğu için yapı oldukça avantajlıdır. Yedekli aritmetik sistemlerin hepsi elde yayılımdan bağımsız olduğundan sistem elde-saklamalı çıkış kodlamalı klasik elde-bağımsız diğer MAC üniteleriyle kıyaslanmıştır. Önerilen yapı pipeline olmadan tek bir aşamada hızlı çarpma-toplama işlemini sağlayan en düşük lojik derinlik sağlar. Sistem özellikle 6-giriş LUT tabanlı FPGA yapıları için uygundur. Önerilen yapı düşük clock gecikmeli yüksek performans gerektiren uygulamalar için avantajlıdır.

İkinci önerilen yapıda ise Booth algoritması kullanılarak yarıya indirgenen kısmi ürünler belli bit sayılarına ayrılarak multi-operand toplayıcı ile toplanmıştır. Multi-operand toplayıcı ripple-carry toplayıcılardan oluşan toplama dizileridir (adder array), elde-yayılımı (carry-propagate) paralel şekilde ilerler, böylece paralel toplama yapılır. Elde edilen çıkış çift-elde saklamalı formattadır. Önerilen yedekli MAC ünitesi sadece tek saat gecikmesiyle diğer MAC ünitelerinde çok daha iyi performans vermektedir.

Önerilen yapılar pipeline edilen MAC üniteleriyle kıyaslandığında tüm yapılar arasında makul register gereksinimi ile makul LUT elemanlarına sahiptir. Önerilen MAC üniteleri ile en az saat gecikmesi ile yüksek performanslı sistemler gerçeklenebilmektedir. Önerilen çarpma-toplama devrelerinin işaret işleme, matris çarpımı gibi matematiksel işlemlerde önemli performans artışı sağlayacağı düşünülmektedir.

KAYNAKLAR

[1] K. K. Barhi, ***VLSI Digital Signal Processing Systems Design and Implementation***, John Wiley & Sons, (1999).

[2] J. E Stine, ***Digital Computer Arithmetic Datapath Design***, Kluwer Academic Publishers Boston/Dordrecht/London, 180 p., (2012).

[3] Z. Salcic, ***Digital Systems Design And Prototyping Using Field Programmable Logic and Hardware Description Languages***, Kluwer Academic Publishers Boston/Dordrecht/London, (2000).

[4] Altera Corporation, Advanced Synthesis Cookbook, (2011).

[5] Altera Corporation, Cyclone III Device Handbook, (2010).

[6] Altera Corporation, Stratix III Device Handbook, (2011).

[7] Xilinx, Inc. :***Virtex-6 FPGAConfigurable Logic*** **Block User Guide**, UG364 (v1.2) 3, (2012).

[8] A. F., Gonzalez, P., Mazumder, ***Redundant Arithmetic Algorithms and Implementations***, Integration, the VLSI journal, 30, 13-53, (2000).

[9] G. Jaberipur, B. Parhami, ***Constant-Time Addition With Hybrid-Redundant Numbers: Theory and Implementations***, Integratıon, the VLSI Journal 41, 49-64, (2008).

[10] M. D. Ercegovac, T. Lang, ***Digital Arithmetic***, Elsevier, (2004).

[11] B. Parhami, ***Computer Arithmetic Algorithms and Hardware Designs***, v2: Presentation Material, (2001).

[12] N. Weste, D. Harris, ***CMOS VLSI Design: A Circuits and Systems Perspective***, Addison-Wesley, 4th ed., (2010).

[13] I. Koren, ***Computer Arithmetic Algorithms,*** Prentice Hall Inc., Canada, (2002).

[14] R. Zimmermann, ***Binary Adder Architectures for Cell-Based VLSI and Their Synthesis***, Swiss Federal Institute of Technology Zurich, PhD Thesis, (1997).

[15] C. Nagendra, ***Power, Delay and Area Tradeoffs in CMOS Arithmetic Modules***, Doctoral Dissertation, The Pennsylvania State University, 105p, USA, (1996).

[16] S. Xing, W. H. Yu, ***FPGA Adders: Performance Evaluation and Optimal Design***, Design & Test of Computers, v. 15: 1, p: 24-29, (1998).

[17] V. Kantabutra, ***A Recursive Carry-Lookahead/Carryy-Select Hybrid Adder,*** IEEE Transactions on Computers, Vol: 12, 1495-1499, (1993).

[18] S. Majerski, ***On Determination of Optimal Distributions of Carry Skips in Adders,*** IEEE Transactions on Electronic Computers, Vol: 1, 45-58, (1967).

[19] A. Guyot, B. Hochet, J. M. Muller, ***A Way to Build Efficient Carry-Skip Adders,*** IEEE Transactions on Computers, Vol: 10, 1144-1152, (1987).

[20] P. K. Chan, M. D. F. Schlag, C. D. Thomborson, V. G. Oklobdzija, ***Delay Optimization of Carry-Skip Adders and Block Carry-Look Ahead Adders Using Multidimensional Dynamic Programming***, IEEE Transactions on Computers, Vol. 4i: 8, 920-930, (1992).

[21] A. Avizienis, ***Signed-Digit Number Representations for Fast Parallel Arithmetic***, IRE Transactions on Electronic Computers, 3: 389-400, (1961).

[22] G. C. Cardarilli, S. Pontarelli, M. Re, A. Salsano, ***On the Use of Signed Digit Arithmetic for the New 6-Inputs LUT Based FPGAs***, IEEE Electronics Circuits and Systems, 15th International Conference, 602-605, (2008).

[23] B. Parhami, ***Generalized Signed-Digit Number Systems: A Unifying Framework for Redundant Number Representations***, IEEE Transactions on Computers, Vol. 39:1, 89-98, (1990).

[24] P. Kornerup, J-M. Muller, ***Leading Guard Digits in Finite Precision Redundant Representations***, IEEE Transactions on Computers, Vol. 55: 5, 541-548, (2006).

[25] M. A. Thornton, ***Signed Binary Addition Circuitry with Inherent Even Parity Outputs***, IEEE Transactions on Computers, Vol. 46: 7, 811-816, (1997).

[26] N. Takagi, H. Yasuura, S. Yajima, ***High-Speed VLSI Multiplication Algorithm with A Redundant Binary Addition Tree,*** IEEE Transactions on Computers, Vol. 100: 9, 789-796, (1985).

[27] S. Kunınobu, T. Nısıdyama, H. Edamatsu, T. Tanıguchi, N. Takagi, ***Design Of High Speed MOS. Multiplier and Divider Using Redundant Binary Representation***, Computer Arithmetic (ARITH), 8th Symposium, pp: 80-86, (1987).

[28] S. M. Yen, C. S. Laih, C-H. Chen, J. Y. Lee, ***An Efficient Redundant-Binary Number to Binary Number Converter,*** IEEE Journal of Solid-State Circuits, Vol 27: 1, 109-112, (1992).

[29] H. R. Srinivas, K. K. Parhi, ***A Fast VLSI Adder Architecture,*** IEEE Journal of Solid-State Circuits, Vol 21: 5, 761-767, (1992).

[30] Y. Harata, Y. Nakamura, H. Nagase, M. Takigawa, N. Takagi, ***A High-Speed Multiplier Using A Redundant Binary Adder Tree,*** IEEE Journal of Solid-State Circuits, Vol. Sc-22: 1, 28-34, (1987).

[31] H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara, K. Mashiko, ***An 8.8-ns 54 x 54-Bit Multiplier with High Speed Redundant Binary Architecture***, IEEE Journal of Solid-State Circuits, Vol. 31: 6, 773-783, (1996).

[32] C. S. Wallace, ***A Suggestion for a Fast Multiplier***, IEEE Transactions on Electronic Computers, 14-17, (1964).

[33] M. Ortiz, F. Quiles, J. Hormigo, F. J. Jaime, J. Villalba, E. L. Zapata, ***Efficient Implementation of Carry-Save Adders in FPGAs***, 20th International Conference on Application-Specific Systems, Architectures and Processors, (2009).

[34] H. Parandeh-Afshar, P. Brisk, P. Ienne, ***Improving Synthesis of Compressor Trees on FPGAs via Integer Linear Programming,*** Proceedings of the Conference on Design, Automation and Test in Europe, pp: 1256-1261, (2008).

[35] T. G. Noll, ***Carry-Save Arithmetic for High-speed Digital Signal Processing,*** Corporate Research and Development Siemens, Journal of VLSI Signal Processing, pp: 121-140, (1991).

[36] W. Kamp, A. Bainbridge-Smith, ***Multiply Accumulate Unit Optimised for Fast Dot-Product Evaluation***, Field-Programmable Technology, International Conference, pp: 349-352, (2007).

[37] X. Huang, W. J. Liu, B. W. Wei, ***A High-Performance CMOS Redundant Binary Multiplication-and-Accumulation (MAC) Unit,*** Transactions on Circuits and Systems-I: Fundamental Theory and Applications, Vol. 41:1, pp:33-39, (1994).

[38] D. Lee, C. Ryu, J. Park, K. Kwon, W. Choi, ***Design and Implementation of 16-Bit Fixed Point Digital Signal Processor***, International SoC Design Conference, Vol. 2, (2008).

[39] C. D. Moreno, F. J. Quiles, M. A. Ortiz, M. Brox, J. Hormigo, J. Villalba, E. L. Zapata, ***Efficient Mapping on FPGA of Convolution Computation based on Combined CSA-***

CPA Accumulator, Electronics Circuits and Systems, 16th IEEE International Conference, 2009.

[40] F. Elguibaly, ***A Fast Parallel Multiplier-Accumulator Using the Modified Booth Algorithm***, Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 47, No. 9, 902-908, (2000).

[41] O. Salomon, J.-M. Green, H. Klar, ***General Algorithms for a Simplified Addition of 2's Complement Numbers***, IEEE Journal of Solid-State Circuits, Vol. 30:7, pp: 839-844, (1995).

[42] S. K. Hsu, S. K. Mathew, M. A. Anders, B. R. Zeydel, V. G. Oklobdzija, R. K. Krishnamurthy, S. Y. Borkar, ***A 110 GOPS/W 16-bit Multiplier and Reconfigurable PLA Loop in 90-nm CMOS*** ,IEEE Journal Of Solid-State Circuits, Vol. 41:1, p: 256-264, (2006).

[43] B. Jose, D. Radhakrishnan, ***Fast Redundant Binary Partial Product Generators for Booth Multiplication***, Circuits and Systems, 50th Midwest Symposium, (2007).

[44] B. A. Jose, D. Radhakrishnan***, Redundant Binary Partial Productgenerators For Compact Accumulation in Booth Multipliers,*** Microelectronics Journal 40: 11, 1606–1612, (2009).

[45] G. W. Bewick,, ***Fast Multiplication: Algorithms and Implementation***, Diss. Stanford University, Doktora Tezi, (1994).

[46] U. Çini, ***Alternative Arithmetic Structures Using Redundant Numbers and Multi-Valued Circuit Techniques***, Graduate Program in Electrical and Electronics Engineering, Boğaziçi University, İstanbul, Doktora Tezi, (2010).

[47] P. Mokrian, ***A Reconfigurable Digital Multiplier Architecture***, Applied Science at the University of Windsor, A Thesis Submitted to the Faculty of Graduate Studies and Research, 230p., (2003).

[48] A. Tajalli, Y. Leblebici,, ***Extreme Low-Power Mixed Signal IC Design, Subthreshold Source-Coupled Circuit***, Springer Science, (2010).

[49] W. Kamp, B.-S. Andrew, M. Hayes, ***Efficient Implementation of Fast Redundant Number Adders for Long Word-lengths in FPGAs***, Field-Programmable Technology (FPT), International Conference, pp. 239-246, (2009).

[50] U. Çini, M. Aktan, A. Morgül, ***An Alternative Carry-Save Arithmetic for New Generation Field Programmable Logic Arrays,*** Turkish Journal of Electrical Engineering and Computer Sciences, p:1-13, (2013).

[51] T. Matsunaga, S. Kimura, Y. Matsunaga, ***Multi-Operand Adder Synthesis on FPGAs using Generalized Parallel Counters***, Proceedings of the 2010 Asia and South Pacific Design Automation Conference. IEEE Press, 2010.

[52] H. P. Afshar, P. Ienne, ***Measuring and Reducing The Performance Gap Between Embedded and Soft Multıplıers on FPGAS,*** 21st International Conference on Field Programmable Logic and Applications, 225-231, (2011).

[53] M. Kumm, S. Abbas, P. Zipf, ***An Efficient Softcore Multiplier Architecture for Xilinx FPGAs***, 22nd Symposium on Computer Arithmetic, 18-25, (2015).

[54] P. Jamieson, J. Rose, ***Mapping Multiplexers onto Hard Multipliers in FPGAs***, IEEE-NEWCAS Conference, The 3rd International, p: 323-326, (2005).

[55] E. G. Walters, ***Partial-Product Generation and Addition for Multiplication in FPGAs With 6-Input LUTs***, (2014).

ÖZGEÇMİŞ

1990 yılında Çorlu'da doğdum. Lise eğitimini Tekirdağ Teknik Lisesi'ni okul birincisi olarak tamamladım. Sonrasında 2008 yılında Kocaeli Üniversitesi Teknik Eğitim Fakültesi Elektronik Öğretmenliği bölümüne girerek 2012 yılında mezun oldum. 2014 yılından itibaren Trakya Üniversitesi Fen Bilimleri Enstitüsü Hesaplamalı Bilimler Anabilim Dalı Yüksek Lisans programına kayıtlı olup Ocak 2016'da tez çalışmasını başarı ile tamamladım. Bilgisayar aritmetiği ve FPGA sistemleri üzerinde araştırmalarıma devam etmeyi hedeflemekteyim.

TEZ ÇALIŞMASI SIRASINDA GERÇEKLEŞTİRİLEN BİLİMSEL FAALİYETLER

[1] U. Cini, O. Kurt, ***A MAC Unit with Double Carry-Save Scheme Suitable for 6-Input LUT Based Reconfigurable Systems***, ICECS 2015, Int. Conf.on Electronics, Circuits,and Systems, Cairo, Egypt, (2015).

[2] U. Cini, O. Kurt, ***A High Performance Multiply-Accumulate Unit Double Carry-Save Scheme for 6-Input LUT Based Reconfigurable Systems***, ELECO-2015, 9th Int. Conf. on Electrical and Electonics Engineering, Bursa, Turkey (2015).

[3] O. Kurt, U. Çini, ***6-Giriş LUT Tabanlı Kapı Dizileri için Elde-Saklayıcılı Çarpma-Toplama Ünitesi Tasarımı***, EEMKON-2015, Elektrik-Elektronik Mühendisliği Kongresi, İstanbul (2015).

[4] O. Kurt, U. Çini, ***Çeşitli Toplama Devrelerinin FPGA Üzerinde Performans Analizi***, Eleco-2014, Elektrik-Elektronik-Bilgisayar ve Biyomedikal Mühendisliği Sempozyumu, Bursa (2014).