KLASİK ÇARPMA ALGORİTMALARININ DONANIMSAL SİMÜLASYONLARI VE PERFORMANS DEĞERLENDİRİMİ

R. Selami Özbey[1] ve Ahmet Sertbaş[2]

[1]TUBİTAK/UEKAE (**U**lusal **E**lektronik ve **K**riptoloji **A**raştırma **E**nstitüsü)
[2]İstanbul Üniversitesi, Bilgisayar Mühendisliği Bölümü
34320, Avcılar-İstanbul
[1]e-posta: selami@uekae.tubitak.gov.tr [2]e-posta: asertbas@istanbul.edu.tr

Anahtar sözcükler: Çarpma algoritmaları , Aritmetik devreler, VHDL simülasyonu

ÖZET

Bu çalışmada, temel aritmetik işlem olarak pek çok kullanım alanı bulunan çarpma işlemi için geliştirilmiş klasik algoritmalar incelenmiştir. Bu algoritmalara dayanarak geliştirilen aritmetik devrelerin tasarımları yapılmıştır. Bu amaçla, VHDL Donanım Tanımlama Dili kullanılarak, Sıralı Topla/Ötele, Booth yöntemi, Baugh-Wooley ve Dizin klasik çarpma yöntemlerine dayanan çarpma devrelerinin simülasyonları elde edilmiştir. Ele alınan aritmetik çarpma devrelerinin analitik gecikme ve alan değerleri analitik yolla hesaplanarak, devrelerin VHDL simülasyonları ile uygunlukları gösterilmiştir. Ayrıca, bu çalışmada, incelenen aritmetik devrelerin performansını belirlemek için, seçilen güç tüketimi kriterine (alanxgecikme) göre, karşılaştırmalı olarak yöntemlerin performansları değerlendirilmiştir.

1. GİRİŞ

Çarpma işlemi, aritmetik işlemlerde oldukça sık kullanılan, sinyal işleme ve bilimsel uygulamalarda önemli rol oynayan, toplama ve öteleme işlemlerine dayanan bir temel aritmetik işlemdir. Tipik bir bilimsel programdaki komutların yaklaşık %9'u çarpma işlemi gerektirir. Bir işlemcide, çarpma işlemi, 2-8 saat çevrimi kadar sürede yapılabilmektedir. Bu nedenle, bir işlemci performansını belirleyen en kritik faktörlerden biri olarak çarpma algoritmalarının iyileştirilmesi ve hızlı/etkin donanımsal gerçeklemeleri konusu, bilgisayar aritmetikçilerinin yoğun ilgisini çekmektedir.

Son yıllarda, ardarda geliştirilen çoğu hızlı çarpma devrelerinin temelinde klasik çarpma algoritmaları yatmaktadır. Klasik bir çarpma işlemi, $\boldsymbol{a}$ çarpılan, $\boldsymbol{x}$ çarpan ve $\boldsymbol{p}$ kısmi çarpan terimlerini göstermek üzere, aşağıdaki gibi tanımlanabilir:

$$\boldsymbol{P}^{(j+1)} = (\boldsymbol{p}^{(j)} + \boldsymbol{x}_j \, \boldsymbol{a} \, 2^k) \, 2^{-1} \qquad \boldsymbol{p}^{(0)}=0, \; \boldsymbol{p}^{(k)}=\boldsymbol{p} \qquad (1)$$

$\leftarrow$ toplama $\rightarrow$
$\leftarrow$ öteleme $\rightarrow$

Bu işlem, Bölüm (2.1)' de tanımlanan Sıralı Topla/Ötele yöntemi olarak bilinen klasik en eski yöntemdir[1]. Çarpma işlemini hızlandırmak için çarpan sayısı Booth kodlamasına göre düzenlenerek, toplama sayısı minimize edilip daha etkin çarpma devreleri elde edilmiştir[2-3]. Daha sonra işaretli sayıların çarpımında son derece etkin olan Baugh-Wooley yöntemi geliştirilmiştir[4]. VLSI tasarımı için elverişli ve düzenli yapısı ile kolay gerçeklenebilir ancak daha yavaş bir metod olan Dizin Çarpma, Baugh-Wooley yönteminin değiştirilmiş haline dayanan bir başka klasik çarpma algoritması olarak ortaya konulmuştur[5].

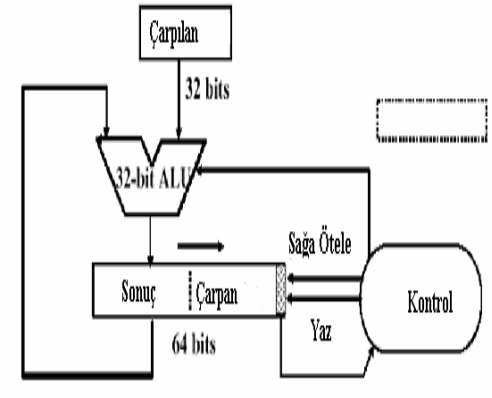
Bu çalışmada, VHDL donanım tanımlama dili kullanılarak, klasik ötele/topla temel çarpma algoritması ve çoğu nümerik işlemcinin çarpma işleminde çok sık kullandığı Sıralı Topla/Ötele, Booth, Baugh-Wooley ve Dizin çarpma algoritmalarının temel donanımsal simülasyonları yapılarak, performans analizleri elde edilmiştir. Algoritmaların performansları; kullanım yeri ve amacı, gerçekleme teknolojileri ve entegre devre olarak gerçeklenmeleri halinde harcayacakları güç tüketimine göre farklılık göstermektedir. Burada, performans kriteri olarak, son yıllarda gittikçe önem kazanan donanımsal (VLSI) devrelerinin sarfettikleri güç miktarı seçilmiştir[6].

2. ÇARPMA YÖNTEMLERİ

Bu bölümde, çalışmada incelenen 4 farklı temel klasik çarpma algoritması olan Sıralı Topla/Ötele, Booth, Baugh-Wooley ve Dizin çarpma yöntemleri temel mantıkları ve donanımsal devreleri verilmiştir.

2.1. Sıralı Topla / Ötele Çarpma

Sıralı Ötele/Topla algoritmasının temeli, çarpan sayısının LSB bitinin değerine göre çarpılan sayının toplanarak kısmi toplam olarak değerlendirilmesine dayanmaktadır. Herbir saat çevriminde çarpan bir bit sağa kaydırılır ve bu bitin değeri test edilir. Bu bitin değeri '0' ise sadece kaydırma işlemi gerçeklenirken, bitin değeri '1' ise çarpılan sayının değeri kısmi toplama eklenir ve bu kısmi toplam değeri bir bit sağa kaydırılır. Çarpan sayının tüm bitleri test edildikten sonra 2n uzunluğunda sonuç elde edilir. Burada,. gecikme değeri en fazla n çevrim zamanı kadar olur. Şekil 1' de, 32 bitlik çarpma işlemi için topla/ötele algoritmasının genel mimari yapısı verilmiştir.



Şekil 1. Sıralı Ötele/Topla Çarpma Devresi

2.2. Booth Çarpma

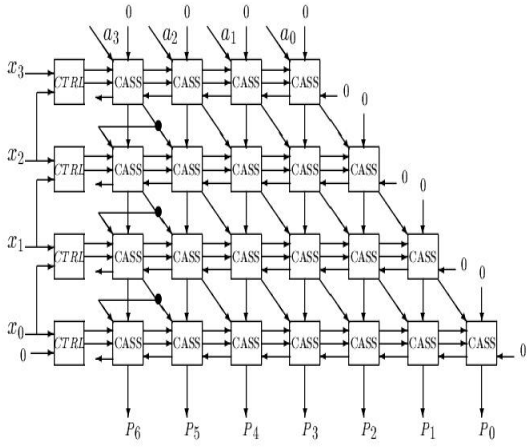
Booth çarpma algoritması işaretli sayıların çarpımında çok etkili olan bir algoritmadır.Booth algoritmasının temeli, belirli bir aralıkta sunulan çarpılan sayıyı daha yüksek tabanlı bir sayıya çevrilip basamak sayısının azaltılmasına dayanmaktadır. Bu durumda, k bitlik bir sayı k/2 basamak olarak 4'lük tabanda bir sayı, k/3 basamak olarak 8'lik tabanda bir sayı gibi yorumlanabilmektedir. Böylece yüksek tabanlı çarpma işlemi yaparak herbir çevrimde birden fazla çarpan ile işlem yapmak avantajı sağlanır.
Klasik olarak, 2'li tabanda yapılan Booth Kodlama (Recoding) işlemi modern aritmetik devrelerde direk olarak uygulanmadığından, Tablo 1'de 4 tabanlı Booth Kodlama işlemi verilmiştir. Booth Kodlama işlemi ile kodlanan çarpan sayı aşağıda formülüze edilmiştir.

$$Z_i = -2x_{i+1} + x_i + x_{i-1} \quad (2)$$

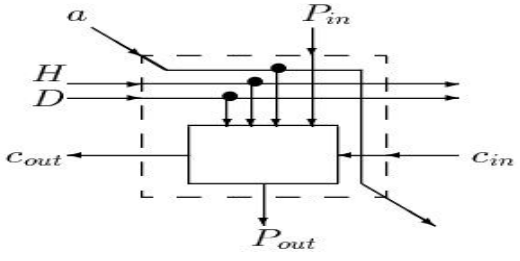
(Örnek olarak, çarpan değerinden 01 01 11 0 Booth kodlama ile 01 01 11 00 değeri türetilir-burada başlangıç biti X_{-1}=0 alınmıştır-).

Tablo 1. BOOTH-4 Kodlama İşlemi

X_{i+1}	X	X_{i-1}	$Z_{i/2}$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	2
1	0	0	-2
1	0	1	-1
1	1	0	-1
1	1	1	0



Şekil 2 BOOTH algoritmasının temel yapısı.



Şekil 3 CASS hücresinin mantıksal gösterimi.

Şekil 2'te, bu algoritmanın donanımsal tasarımı, Şekil 3' te ise tek bir hücre bloğu görülmektedir.

Burada: $P_{out} = P_{in} \oplus (a \,.\, H) \oplus (c_{in} \,.\, H)$

$$c_{out} = (P_{in} \oplus D) \,.\, (a + c_{in}) + a \,.\, c_{in}$$

$H=0 \Rightarrow P_{out} = P_{in}$; $H=1 \Rightarrow P_{out} = P_{in} \oplus a \oplus C_{in}$

$D=0 \Rightarrow C_{out} = P_{in}(C_{in} \oplus a) + (aC_{in})$

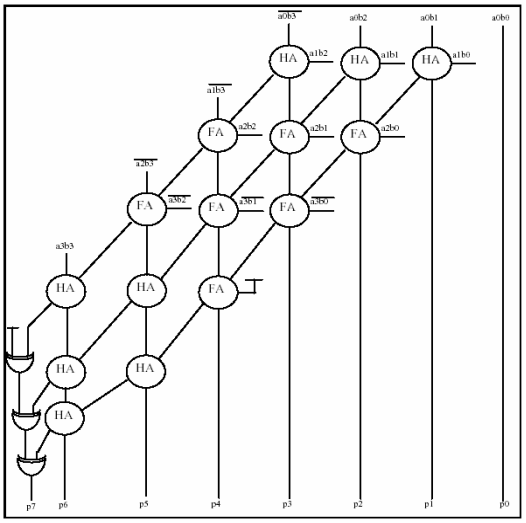
$D=1 \Rightarrow C_{out} = P_{in}(a \oplus C_{in}) + (aC_{in})$

2.3. Baugh - Wooley Çarpma

Şekil 4'te, simülasyonu yapılan 4x4 bitlik Baugh-Wooley çarpma algoritması kullanılarak yapılan işlem, Şekil 5'te ise işlemi gerçekleyen donanımsal devre görülmektedir.

					x_4	x_3	x_2	x_1	x_0
					y_4	y_3	y_2	y_1	y_0
					W_{40},	W_{30}	W_{20}	W_{10}	W_{00}
				W_{41},	W_{31}	W_{21}	W_{11}	W_{01}	
			W_{42},	W_{32}	W_{22}	W_{12}	W_{02}		
		W_{43},	W_{33}	W_{23}	W_{13}	W_{03}			
	W_{44}	$W_{3'4}$	$W_{2'4}$	$W_{1'4}$	$W_{0'4}$				
1	x'_4				x_4				
	y'_4				y_4				
Z_9	Z_8	Z_7	Z_6	Z_5	Z_4	Z_3	Z_2	Z_1	Z_0

Şekil 4. Baugh-Wooley çarpma işlemi.

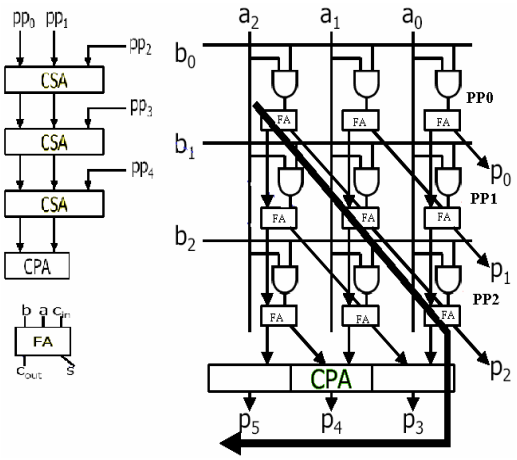


Şekil 5. 4x4 bit Baugh-Wooley Çarpma devresi

2.4. Dizin Çarpma

Dizin çarpma algoritmaları düzenli yapılarından dolayı en iyi bilinen çarpma şeklidir. Bir bitlik çarpan değerinin çarpılan sayı ile çarpımından elde edilen kısmi toplam terimleri bit sırasına göre kaydırılır ve toplanır. Bu toplama işlemi bilinen elde iletim toplayıcıları ile yapılabilir. Bu işlem için, çarpan uzunluğuna bağlı olarak, n-1 tane toplayıcı gerekir.

Bu algoritmanın çok yavaş olmasına karşın tercih edilmesinin sebebi düzenli yapısı ve kullanılan bir hücreden diğer bir hücreye giderken diyagonel, yatay ve dikey olarak en kısa yolun kullanılabilir olmasıdır. Bu özellik, VLSI tasarımda çok büyük bir kolaylık sağlamaktadır.
nxn bitlik bir dizi çarpma algoritmasında gecikmenin hesaplanabilmesi için bulunması gereken kritik yol Şekil 6'da görülmektedir. Tam toplayıcılardan kaynaklanan gecikme değeri en üstte sağdan sola doğru, ortada diyagonel olarak ve en altta sağdan sola doğru sıralanan mantıksal kapıların sayısına göre hesaplanır.



Şekil 6. nxn bitlik Dizin Çarpma devresi

3. VHDL SİMÜLASYONLARI

Çarpma işlemi, toplama ve öteleme işlemleri ile gerçekleştirilebilirken, bu işlemler içinde en fazla zamanın toplama işlemi için harcandığı bilinmektedir. Bu yüzden, bir çarpma işleminin performansı, büyük oranda, yapılan toplama işlemlerinin performansına bağlı olmaktadır. En kötü durum için, n bitlik iki sayının çarpımında, n tane toplama işlemine gereksinim duyulmaktadır.

Bu çalışmada incelenen çarpma algoritmalarının 4 bitlik sayılarla VHDL (MAX+plus) [7]

kullanılarak elde edilen simülasyonları Şekil 7-10 da verilen grafiklerle gösterilmiştir. Bu simülasyonlar ile aritmetik çarpma devrelerinin fonksiyonel olarak doğrulanması ve en uzun yoldaki gecikme süreleri (Tablo 2) belirlenmiştir.

Tablo 2. Çarpma Algoritmaları İşlem Zamanları

	Sıralı Topla /Ötele	Booth Yöntemi	Baugh-Wooley	Dizin Çarpma
T (ns)	20,6	17	10	9,9

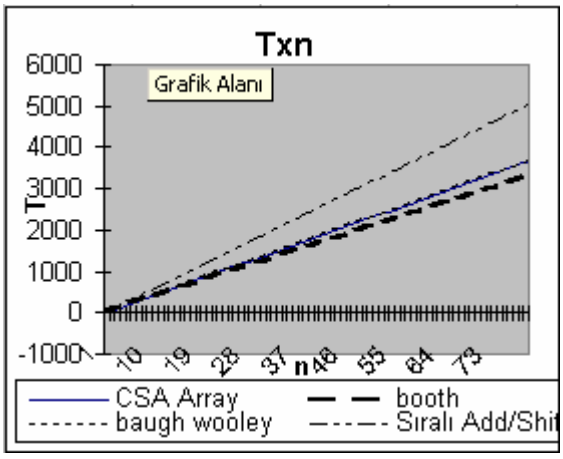
4. PERFORMANS ANALİZLERİ

Bit sayısı *n*, işlem gecikme süresi (T), işlem devre (çip) alanı (A) ve harcanan güç ölçüsü (AxT) olmak üzere Tablo 3'te, incelenen tüm çarpma işlem devrelerine ilişkin analitik alan ve gecikme değerleri ve Şekil 11-13 de ise, elde edilen değerler aracılığıyla çizdirilen performans grafikleri görülmektedir. Burada, T süresi hesaplanırken, algoritmaların iterasyon sayıları, oteleme sayıları, yeniden kodlama süreleri de göz önünde bulundurulmuştur.

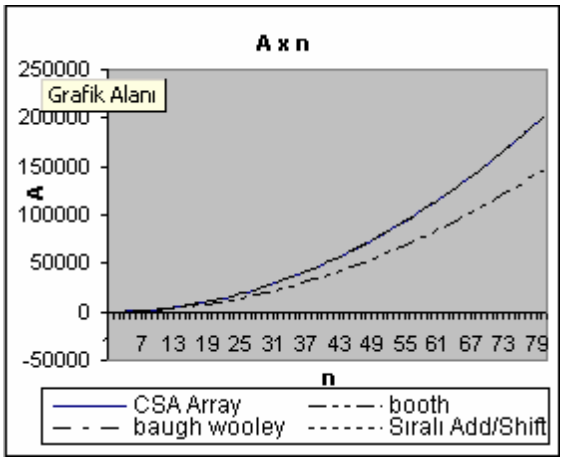
Tablo3. A ve T Analitik İfadeleri

Yöntem	**A** Alan	**T** Gecikme
Sıralı Topla/Ötele	64n – 72.4	$32n^2 – 36n$
Booth Yöntemi	41.5n – 1.9	$22n^2 + 5n – 58$
Baugh-Wooley	46.4n – 14.8	$32n^2 - 32n + 88$
Dizin Çarpma	46.4n – 54.8	$32n^2 – 36n$

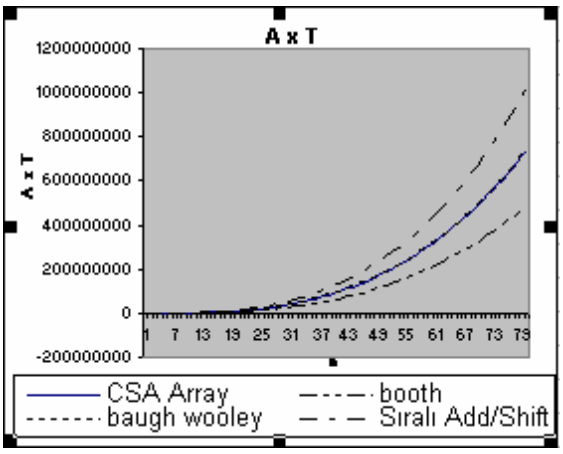
Şekil 11 te görülen grafiklerinden anlaşılacağı gibi, çarpma algoritmalarının en hızlısı ***Booth*** , en yavaşı ise Sıralı Topla/Ötele çarpma yöntemidir. Şekil 12 de, tam tersine, Sıralı Ötele/Topla algoritması en küçük, Booth yöntemi en büyük alan ihtiyacı göstermektedir. ***Baugh-Wooley*** ve **Dizin Çarpma** (Array) algoritmalarının ise orta gecikme ve daha çok alana ihtiyaç gösterdiği bu grafiklerden belirlenmişitr. Şekil 13' te, temel performans kriteri olarak kullanılan AxT grafiği üzerinde, minimal güç tüketen çarpma devresinin Booth yöntemi ile gerçeklenen devreye ait olduğu gösterilmiştir.



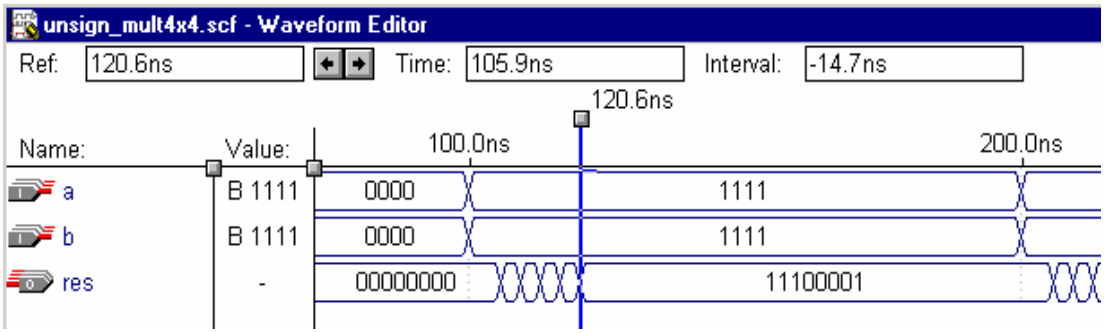
Şekil 11. Çarpma Algoritmalarının Txn grafiği



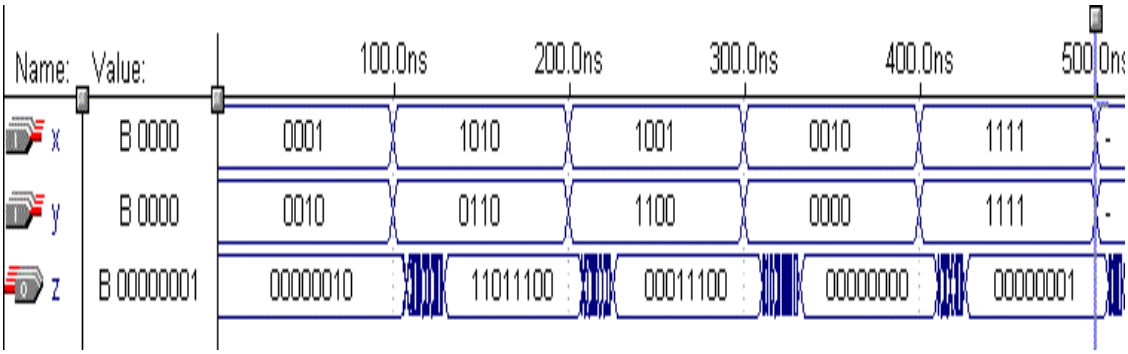
Şekil 12. Çarpma Algoritmalarının Axn Grafiği



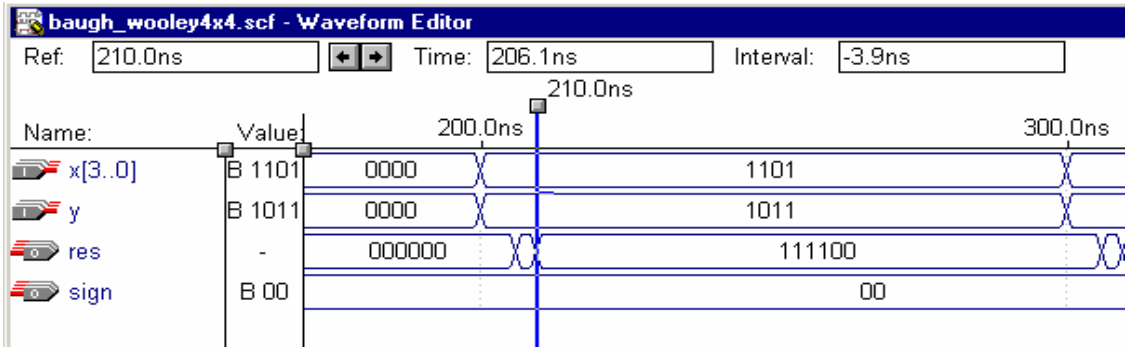
Şekil 13. Çarpma Algoritmalarının ATxn Grafiği



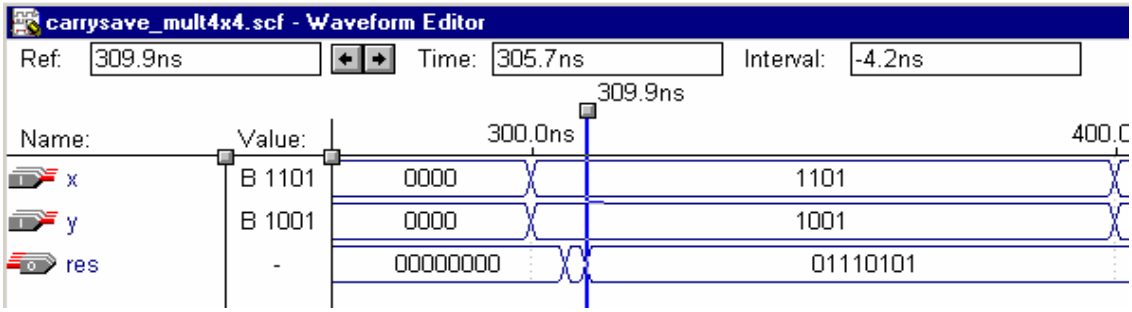
Şekil 7. Sıralı Add/Shift Çarpma Devresi VHDL simülasyonu.



Şekil 8. BOOTH Çarpma Devresi VHDL simülasyonu.



Şekil 9. Baugh-Wooley Çarpma Devresi VHDL simülasyonu



Şekil .10. Dizin Çarpma Devresi VHDL simülasyonu

5. SONUÇLAR

Bu çalışmada, günümüzde hızlı nümerik işlemciler için geliştirilen aritmetik çarpma devrelerin pek çoğunun temel aldığı klasik çarpma algoritmaları incelenmiştir. Bilgisayar donanım tasarımlarında etkin olarak kullanılmakta olan VHDL yardımıyla incelenen çarpma donanımlarının simülasyonları elde edilmiştir. Ayrıca, devrelerin çip olarak kapsadığı alanın ölçütü olarak kullanılan birim kapı sayılarının ve en uzun gecikme değerlerinin analitik ifadeleri hesaplanarak grafiksel olarak çizdirilmiştir. Çarpma devrelerin gecikme süreleri, giriş operandı bit uzunluğuna (n) üstel olarak bağlı olduğu görülmüştür. Modern tasarımlarda gittikçe önem kazanan devrenin minimal güç tüketimi kriterinin ölçütü olarak seçilen A*T değeri performans incelemelerinde kullanılan temel değerlendirme faktörüdür. Bu yüzden, tüm çarpma yöntemleri için A*T grafikleri verilmiştir.

Elde edilen sonuçlara göre, Booth çarpma yöntemi, en hızlı, yani gecikme süresi en düşük bir algoritma olduğu belirlenmiştir. Özellikle, yüksek bit değerlerinde Booth çarpma devresinin gecikmesi, diğer yöntemlerle tasarlanan çarpma devrelerine göre daha hızlı küçüldüğü elde edilen grafiklerle ortaya çıkmıştır. Buna karşılık, en yavaş çarpma devresi ise, Sıralı Topla/Ötele yöntemi ile tasarlanan devre olduğu tespit edilmiştir. Toplam performans kriterine (A*T) göre, Booth yöntemi en iyi, Sıralı Topla-Ötele yöntemi en kötü, Baugh-Wooley ve Dizin Çarpma yöntemi ise orta seviyede performans gösterdiği elde edilen bir sonuçtur.

Burada incelen çarpma yöntemleri literatürde ayrıntılı olarak incelenmiş olmalarına rağmen, karşılaştırmalı performans değerlendirimlerine ilişkin çalışma son derece kısıtlıdır. Bu açıdan, makalenin bilgisayar aritmetik algoritma ve tasarımları ile ilgilenen araştırmacılara yararlı olacağı ümit edilmektedir. Bir önceki yayınımızda[8], hızlı toplama devreleri için benzeri incelemeler yapılmıştı. Geleceğe dönük bir çalışma olarak, aynı incelemenin bölme, karekök, üstel ve logaritmik fonksiyonların hesaplanmaları için de yapılması önerilebilir.

6. KAYNAKLAR

[1] Shaw R.F., 'Arithmetic Operations in a Binary Computer', Rev. Scientific Instruments, Vol. 21, pp. 687-693, 1950.

[2] Booth A.D., 'A Signed Binary Multiplication Technique', Quarterly J. Mechanics and Applied Mathematics, Vol.4, pp. 236-240, June 1951.

[3] Rubinfield L.P., 'A proof of the Modified Booth's Algorithm for Multiplication', IEEE Trans. Computers, Vol.25, No.10, pp. 1014-1015, 1975.

[4] Baugh W.R., and Wooley B.A., 'A Two's Complement Parallel Array Multiplication Algorithm', IEEE Trans. Computers, Vol.22, pp. 1045-1047, 1973.

[5] Ciminiera L. And Montushi P., Carry-save Multiplication Schemes Without Final Addition', IEEE Trans. Computers, Vol.45, No.9, pp. 1050-1055, 1996.

[6] Zimmerman, R., 'Binary Adder Architectures forCell-Based VLSI and their Synthesis Acknow-ledgments', Thesis(PhD).,Swiss Federal Inst. Of Tech., Zurich, 1997.

[7] ALTERA Corperation, MAX Plus II Data Sheets [online], http://www.altera.com.

[8] Sertbaş A., and Özbey R.S., 'A Performance Analysis Of Classified adder Architectures and the VHDL Simulations', İstanbul Üniversitesi Elektrik-Elektronik Dergisi Cilt 4, Sayı 1, Shf. 1 Shf.1025, 1030, 2004.