YÜKSEK TABANLI SRT BÖLME ALGORİTMALARININ KARŞILAŞTIRMALI PERFORMANS ANALİZİ

Doğal Acar Ahmet Sertbas

Computer Engineering Department
Engineering Faculty, Istanbul University
34320, Avcılar, İstanbul
asertbas@istanbul.edu.tr

Özet: Bu çalışmada, hızlı hesaplama için geliştirilen yüksek tabanlı SRT bölme yöntemleri incelenmiştir. Bu amaçla, hızlı arama tablosu kullanan farklı tabanlı SRT bölme algoritmalarının (SRT-2, SRT-4 ve SRT-8) avantaj/dezavantajlı taraflarını karşılaştırmalı olarak görmek için, C dilinde bir simülasyon programı geliştirilmiştir. Farklı tabanlı bölme algoritmalarının kullandıkları toplam bellek alanı (A), toplama işlemi ve gerekli adım sayısı yardımıyla toplam işlem süreleri (T) hesaplanarak performans analizleri yapılmıştır.

Anahtar Kelimeler: SRT bölme tekniği, yüksek tabanlı bölme, performans analizi

1. GİRİŞ

Son yıllarda, bilgisayar uygulamalarının hesaplama karmaşıklıkları gittikçe artmaktadır. Bölme işlemi, bilgisayar aritmetik işlemleri içinde, hesaplama karmaşıklığı en fazla olan ve en uzun gecikme süreli bir işlemdir. Bu yüzden, hızlı bölme algoritmalarının geliştirilmesi, bilgisayar aritmetiğinde önem arzeden bir çalışma alanıdır. Bölme için, karasel (quadratik) ve dogrusal yakınsama algoritmaları temelinde birçok algoritma geliştirilmiştir. En önemli yakınsama algoritması ilk grup için Newton-Raphson Yöntemi [1,2] iken, diğer grup için, SRT[3] yöntemi olduğu bilinmektedir. SRT yöntemi, her adımda bir bölüm bitinin belirlenmesine dayanır. Bu tür SRT algoritmalarının hızı bölüm bitinin seçimi, yapılan toplama işlemi sayısı ve adım sayısına bağlıdır. Bölme işlemini hızlandırmak için, taban (β) artırılarak toplam adım sayısı azaltılabilir. Yüksek tabanlı SRT algoritmalarında, $\beta=2^m$ seçilerek, adım sayısı n/m'e indirilebilmektedir. Öte yandan, tabanın (β) arttırılması, bölüm biti seçim işlemini daha karmaşıklaştırdığından, adım sayısının azaltılması avantajını sınırlandırır.

Bölüm bitinin seçim işlemini kolaylaştırmak için birçok yöntem geliştirilmiştir[4-6]. Bunlardan en basiti *bölüm biti seçim tablosu* olarak adlandırılan look-up table (arama tablosu) yöntemidir. Bu yöntemde her adımda kısmı kalan ile tablo değeri karşılaştırılarak sonuca ulaşılır. Fakat taban arttıkça arama tablosunun boyutuda artmakta bu da maliyeti yükseltmektedir.

Bu çalışmada, hızlı arama tablosu kullanan farklı tabanlı SRT bölme algoritmalarının[7-13] (SRT-2, SRT-4 ve SRT-8) avantaj/dezavantajlı taraflarını karşılaştırmalı olarak görmek için, C dilinde bir simülasyon programı geliştirilmiştir. Farklı tabanlı bölme algoritmalarının kullandıkları toplam bellek alanı, toplama işlemi ve gerekli adım sayısı yardımıyla toplam işlem süreleri hesaplanarak performans analizleri yapılmıştır. Bu sonuçlardan yararlanarak, bölme algoritmalarının donanımsal gerçeklenmeleri durumunda, devrelerin maliyetleri ve hızları tahmin edilebilmiştir.

2. SRT BÖLME ALGORİTMASI

Taban *β,* bir önceki kalan r_{i-1}, bölüm biti q_i ve bölen D olmak üzere, genel olarak bir bölme işleminde, (r_i) kalana ilişkin rekürsif denklem (1)'deki gibi verilebilir:

$$r_i = \beta \cdot r_{i-1} - q_i \cdot D \qquad (1)$$

SRT algoritmalarında, bölüm dijiti (q_i) aşağıda gösterilen bir dijit kümesinden değerler alabilir.

$q_i \in \{-\alpha.....-1,0,1,..... \alpha;\ \}$

Burada α, $\lceil(\beta-1)/2\rceil \le \alpha \le (\beta-1)$ aralığında tanımlıdır.

Yüksek tabanlı bölme algoritmalarında, β=4 ve β=8, ardarda gelen iki bölüm bitine karşı düşen iki komşu bölge arasında bir örtüşme bölgesi mevcuttur. Bu durumda, örtüşme bölgesini birbirinden ayıran kısmi kalan ve bölen değerlerinin seçimi önem kazanır. Ardışık bölüm bitlerinin ayrışımı, K=α/(β-1) oranı ile tanımlanan karşılaştırma sabitinin belirlenmesine bağlıdır. Klasik yüksek tabanlı SRT algoritmaları, bölüm bitinin belirlenmesi ve kısmi kalanın güncellenmesi adımlarını içermektedir. Bölüm biti, şu anki kısmi kalan (P_i) ile arama tablosundan seçilir ve daha sonra kısmi kalan bölüm biti ile güncellenir.

$P_{i-1}=\beta r_{i-1}$ önceki kısmi kalanı temsil ettiğini kabul edelim. Bu durumda, P_{max} ve P_{min} aşağıdaki gibi verilebilir:

$$P_{max} = (k+q)D \,, \qquad P_{min} = (-k+q)D \qquad (2)$$

Örtüşme bölgesini birbirinden ayıran, seçilen kısmi kalan ve bölen değerlerinin bölme işlemi sırasında karşılaştırma sabitleri olarak kullanılabilmesi için, bir c sabit değeri mevcut olmalıdır.

$$(-k+j+1)\, D_{max} \le c \le (k+j)\, D_{min} \qquad (3)$$

(3) şartı sağlanırsa, q, D'den bağımsız, sadece kısmi kalan P'ye bağımlı olacaktır. Bu eşitsizlik sağlanmazsa, $[D_{min},D_{max})$ aralığı bir çok küçük parçacığa bölünmelidir.

- **SRT-2 Bölme Algoritması**

Klasik SRT yöntemi olarak ta bilinen SRT-2 bölme algoritması, taban 2 olduğu için, kalan denklemi aşağıdaki gibi olur:

$$r_i = 2r_{i-1} - q_i.D \qquad (4)$$

Burada, q_i bölüm biti seçme kuralı (3) teki gibi tanımlıdır:

$q_i = 0$; $|2r_{i-1}| < 1/2$ ise (5)

$q_i = 1$; $|2r_{i-1}| \ge 1/2$ & r_{i-1} ve D aynı işaretli

$q_i = \bar{1}$; $|2r_{i-1}| \ge -1/2$ & r_{i-1} ve D farklı işaretli

- **SRT-4 Bölme Algoritması**

SRT-4 bölme algoritması, taban 4 olduğu için, kalan denklemi aşağıdaki gibi verilebilir:

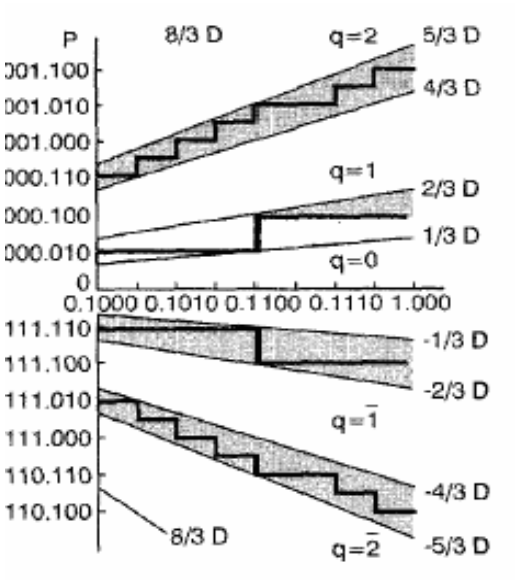
$$r_i = 4r_{i-1} - q_i.D \qquad (6)$$

Algoritmada, bölüm bitlerini seçmek için kullanılan arama tablosu, α=2 için, Şekil 1'de verilmiştir. Uygulamamızda α =2 olarak seçildiğinden q bölümünün olası dijitleri {-2, -1, 0, 1, 2} kümesinin elemanları olabilir.

- **SRT-8 Bölme Algoritması**

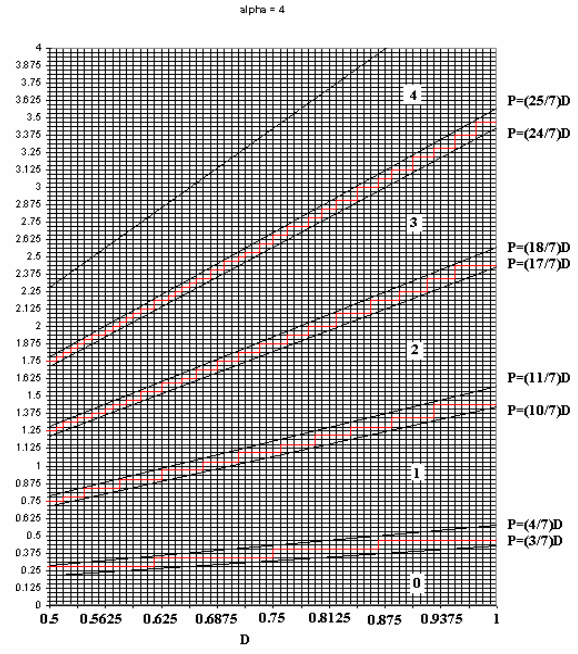
SRT-8 bölme algoritması, taban 8 olduğu için, kalan denklemi aşağıdaki gibi verilebilir:

$$r_i = 8r_{i-1} - q_i.D \qquad (7)$$

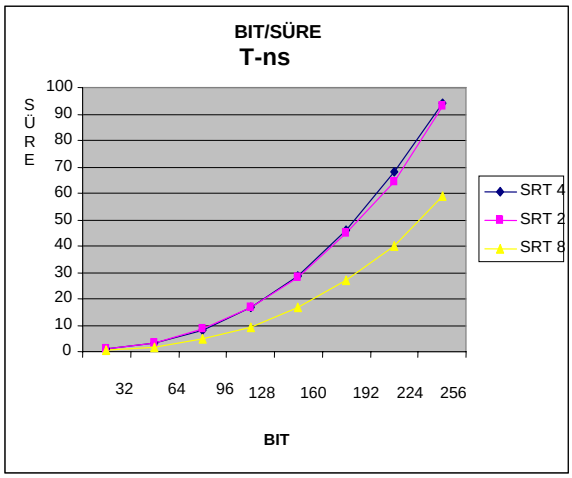


Şekil 1. SRT-4, α =2 için çizilen P-D grafiği [13]

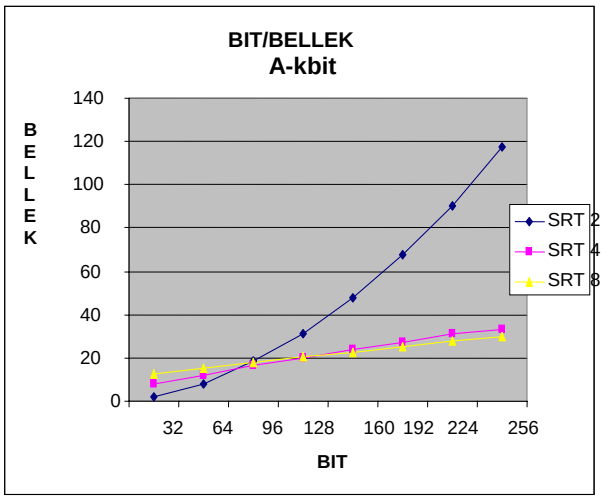
Algoritmada, bölüm bitlerini seçmek için kullanılan arama tablosu, α=4 değerleri için, kıyaslama değerleri Şekil 2'de görülmektedir. Bu çalışmada, (α =4) seçildiğinden, q bölümünün olası dijitleri {-4, -3, -2, -1, 0, 1, 2, 3, 4} kümesinin elemanları olabilir. Şekil 2'de sadece pozitif α değerleri gösterilmiştir, negatif α değerleri pozitif için çizdirilen diyagramın D'ye (yatay eksen) göre simetriğidir.



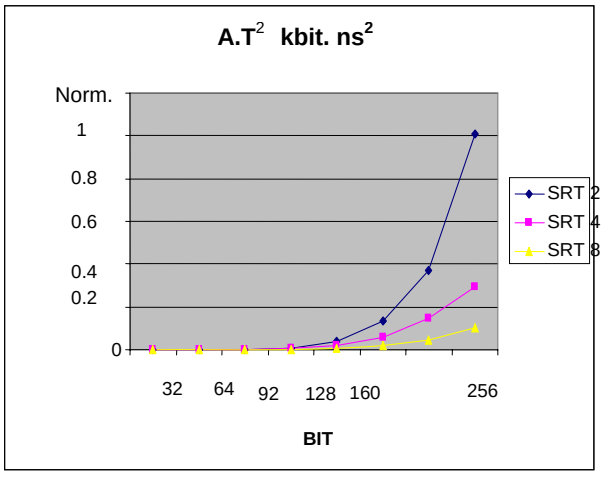
Şekil 2. SRT-8, α =4 için çizilen P-D grafiği



(a)



(b)



(c)

Şekil 3. SRT algoritmalarının performans grafikleri
(a) İşlem süresi – Bit uzunluğu değişimi (T-n)
(b) Bellek Alanı - Bit uzunluğu değişimi (A-n)
(c) A*T^2- Bit uzunluğu grafiği (performans ölçütü)

3. PERFORMANS ANALİZİ

Bu bölümde, SRT-2, SRT-4 ve SRT-8 algoritmalarının performanslarını karşılaştırmak için, tüm bölme algoritmalarının gerçeklenmesi amacıyla, C# (CSharp) ortamında bir uygulama geliştirilerek, işlemler sırasında algoritmaların kullandıkları adım sayısı, toplama ve öteleme işlemleri sayısı ve toplam bellek kullanım miktarları hesaplanmıştır. Program, bölünen ve bölen değerleri onluk sayı düzeninde almakta, sayıları ikilik (64 bitlik) sisteme çevirerek işlemleri yapmakta ve bölüm sonucu ile kalanı hesaplamaktadır.

(Şekil 3a)'da , 32 bit aralıklarla, maksimum 256 bit uzunluklu bölme işlemleri için, algoritma adım süreleri-T; (Şekil 3b)'de, kullanılan bellek uzunlukları-A; (Şekil 3c)'de ise, çoğu donanımsal gerçeklemelerde kullanılan performans kriteri olarak belirlenen AxT^2 değerleri hesaplattırılmıştır.

Yazılımsal simülasyonla elde edilen T adım süresi, donanımsal devrenin hızını belirlerken, kullanılan bellek alanı, A ise çip gerçekleme alanı ile orantılı olmaktadır. Burada yapılan analiz, devre gerçeklemeleri için de performans açısından bir değerlendirme sağlayacaktır.

4. SONUÇLAR

Bu makalede, yüksek hızlı bölme işlemleri için kullanılan SRT yönteminin seçilen tabana göre performansları analiz edilerek, karşılaştırmalı sonuçlar elde edilmiştir. Analiz sonucu olarak, bölme işlemleri için kullanılan bellek miktarlarının algoritma adım sayısıyla ters orantılı olduğu görülmüştür. SRT-2 algoritmasında, arama tablosu kullanılmadığı için, daha düşük miktarda bellek kullanılmakta, fakat adım sayısı çok daha fazla olmaktadır. Bununla birlikte, SRT-4 ve SRT-8 algoritmaları çok daha düşük sayıda algoritmik adım kullanmakta, ancak, arama tablosu kullanılması sebebiyle daha yüksek miktarda bellek alanına ihtiyaç duymaktadır. Diğer önemli bir sonuç ise, SRT-2 algoritmasında kullanılan toplama ve öteleme miktarı yüksek oranla öteleme lehineyken, SRT-4 ve SRT-8 algoritmalarında toplama miktarı beklenenden daha fazla çıkmaktadır, aynı zamanda öteleme sayısı da büyük çarpım değerleri yüzünden daha fazla sayıdadır.

Sonuç olarak, SRT-8 seçilen performans ölçütüne göre, SRT-2 ve SRT-4 algoritmalarından daha iyi performanslı olduğu görülmüştür. Benzeri bir çalışma, incelenen tüm SRT bölme tekniklerinin donanımsal gerçeklemelerini VHDL simülasyon aracı ile sentezlemek ve performans analizlerini elde etmek olacaktır.

KAYNAKLAR

[1] M. D. Ercegovac and T. Lang. Division and Square Root: Digit-Recurrence Algorithms and Implementations. Kluwer Academic Publishers, 1994.

[2] I. Koren. , 'Computer arithmetic algorithms' A.K. Peters Ltd., ISBN 1-56881-160-8., 2002.

[3] J. E. Robertson., A new class of digital division methods. IRE Transactions on Electronic Computers, EC-7(3):88--92, September 1958.

[4] G. S. Taylor. Radix 16 SRT dividers with overlapped quotient selection stages. In Proceedings of the 7th IEEE Symposium on Computer Arithmetic, pages 64--71, June 1985.

[5] M.D Ercegovac and T. Lang: 'A division algorithm with prediction of quotient digits'. Proceedings of 7th IEEE symposium on *Computer arithmetic,* Urbana, *IL,* pp. 64-71, June 1985.

[6] P. Montuschi. and L. Ciminiera: 'Design of a radix-4 division unit with simple selection table', *IEEE ~rans.:* C-41, (12), pp. 1606-1611, 1992.

[7] M.D. Ercegovac and T. Lang.: 'Simple radix-4 division with operands scaling', *IEEE Trans.,* 1990, C-39,'pp. 1204-1207

[8] T. Carter and J. Robertson. Radix-16 signed-digit division. IEEE Transactions on Computers, 39(12):1243--1433, December 1990.

[9] S. Oberman and M. Flynn. An analysis of division algorithms and implementations. Technical Report No. CSL-TR-95-675, Computer Systems Laboratory, Stanford Uni-versity, July 1995.

[10] H. Srinivas and K. Parhi. A fast radix-4 division algorithm and its architecture. IEEE Transactions on Computers, 44(6):826--831, June 1995.

[11] T.-H. Pan, H.-S. Kay, Y.Chun, and C.L.Wey: 'High-radix SRT division with speculation of quotient digits'. Proceedings of intemational conference on *Computer design* (ICCD), Austin, TX, pp. 479-482, October 1995

[12] C.L.Wey, and C.-P.Wang: 'Design of a fast radix-4 SRT divider and its VLSI implementation' , *IEE Proc., Comput. Digital Tech.,* 1999, 146, (4), pp. 205-210

[13] C.L.Wey, 'Design of fast high-radix SRT dividers and their VLSI implementations', IEE Proc. Comput. Dig. Tech, Vol. 147, No. 4, July, 2000.