

İki Boyutlu Dönüşümlerin Çarpıcısız Gerçeklenmesi

Arda Yurdakul

Boğaziçi Üniversitesi Bilgisayar Mühendisliği Bölümü

a.yurdakul@ieee.org

Abstract

Bu çalışmada sayısal işaret işleme sistemlerinde yaygın olarak kullanılan iki boyutlu doğrusal dönüşümlerin çarpıcı kullanmadan gerçekleştirilmesini sağlayan bir algoritma geliştirilmiştir. Toplayıcı sayısında, orijinal sayıdan yüzde doksanlara varan kazanç elde edilmiştir.

1 Giriş

Sayısal İşaret İşleme (Sİİ) sistemleri günümüzde donanım olarak gerçekleştirilmektedir. Bu tür donanımları otomatik olarak gerçekleştirmemizi sağlayan etkin bilgisayar destekli tasarım programlarının geliştirilmesi ise aktif araştırma konularından biridir.

Sayısal işaret işleme algoritmalarının temelinde dönüşümler bulunmaktadır. Her dönüşümü, giriş işaretinin belirli katsayılarla çarpılıp toplanmasıyla çıkış işareti elde etme olarak yorumlayabiliriz. Bu katsayılar, dönüşümün özelliğine göre gerçel ve/ya hayali olabilmektedir.

Katsayı çarpımları son yıllarda genel çarpıcılar yerine toplayıcı ve/ya çıkarıcılarla gerçekleştirilmektedir. Bu yöntemin geliştirilmesinde ilk sebep genel çarpıcıların toplayıcılarla geliştirilen sistemlere nazaran daha büyük alan kaplamasıydı. Ancak günümüzün ilerleyen çok büyük ölçekli tümleşik devre teknolojisi sayesinde silikon alanı artık sorun olmaktan çıkmıştır. Bugün yüz bin genel çarpıcı içeren tek yongalı çözümler artık doğal karşılanmaktadır [1]. Ayrıca, eskiden işlemsel olarak yavaş tabir edilen çarpıcıların performansını arttırmak için paralel olarak bir çok çarpıcı aynı anda kullanılmaya başlanmıştır. Son olarak, uygulamaya yönelik tümleşik devrelerin üretiminde kaybedilen zaman ve masraf göz önüne alındığında üretilen bir devrenin, yapılacak katsayı değişimlerini yansıtmaya açısından da genel çarpıcılar günümüzde tercih edilebilmektedir. Bütün bu sebeplerden dolayı genel çarpıcılar kullanmadan katsayı çarpım işlemlerini gerçekleştirme yönteminin faydası sorgulanmaya başlanmıştır. Ancak katsayılardaki ortak terimler kullanarak yapılan toplama işlemleriyle gerçekleştirilen katsayı çarpımları tam borulanmış sistemlerde hala yüksek performanslı çözümler üretmektedir. Çarpıcısız sistemlerin bir başka avantajı da çarpıcılı sistemlere göre çok daha az elektriksel güç harcamasıdır, ki bu da pille çalışan taşınabilir sistemler için çok büyük bir avantajdır. Ayrıca yakın gelecekte donanım tasarımında yeni bir boyut olacağı şimdiden belli olan yeniden betimlenebilir yongalar [2] sayesinde katsayı değişikliklerini çarpıcısız sistemlere anında yansıtmak artık problem olmaktan çıkacaktır.

Sayısal işaret işleme kapsamında kullanılan dönüşümlerin bir kısmını çarpıcısız olarak gerçekleştirilmesi için çeşitli tasarım otomasyon algoritmaları geliştirilmiştir. Bunların oldukça önemli bir kısmı sonlu dürtü yanıtı süzgeç tasarımları içindir. Bu algoritmaların bir kısmı bazı özel yapıda olan iki boyutlu sistemlerde de çalışmaktadır. İki veya daha çok boyutlu dönüşümler için geliştirilen algoritmaların biri hariç hepsi, sistem tanımından başlayarak katsayıları ikinin-katsayılarının-ışaretlenmiş-basamaklarını bulmaktadır. Tek boyutlu sistemlerde bu metodun toplayıcı sayısı/performans kriteri açısından çok başarılı olmadığı gözlemlenmiştir. Katsayılar, sistem özelliklerine göre sonsuz kesinlikte belirlendikten sonra belli bir kelime uzunluğuyla sınırlandırıldı takdirde tek boyutlu sistemlerde daha iyi bir toplayıcı sayısı/performans kriteri elde edildiği görülmüştür. İki boyutlu sistemleri bu şekilde çözen algoritma literatürde sadece bir tane bulunmaktadır [3].

Bu çalışmada iki boyutlu dönüşümleri etkin bir şekilde çarpıcısız olarak gerçekleştirecek bir otomasyon algoritması geliştirilmiştir. Bu yöntem, iki boyutlu dönüşüm için kullanılan matristeki katsayılar sonsuz kesinlikte belirlendikten sonra belli bir kelime uzunluğuyla sınırlandırılmasıyla başlar. Metodumuzda kullanılan sınırlandırma, geleneksel iki tabanlı ifade biçimleri (ikinin tamamlayıcısı veya işaretlenmiş basamak biçimleri gibi) yanında herhangi bir tabanla yapılabilecek ifade biçimlerini de kapsamaktadır [4]. Bu şekilde toplayıcı sayısının daha da azaltılması hedeflenmektedir. Algoritmamız iki boyutlu dönüşüm için kullanılan matristeki katsayıların ortak terimlerinin bulunması üzerine kurulmuştur. Çekirdek olarak ortak terimler gerçekleştirildikten sonra matristeki tüm katsayılar ortak terimlerden elde edilmektedir. Her dönüşümde toplama işlemi sonucu oluşturmak için gereklidir. Bu yüzden oluşturulan katsayıların en az toplayıcı kullanacak şekilde gerçekleştirilmesi de önemlidir ve geliştirilen algoritma bunu da gerçekleştirmektedir. Sonuç olarak toplayıcı sayısında, orijinal sayıdan yüzde doksana varan kazanç elde edilmiştir.

İzleyen bölümde iki boyutlu doğrusal dönüşümün algoritmamızı geliştirmemizde yardımcı olan teorik altyapısı anlatılmıştır. 3.Bölüm'de algoritmamız anlatılmaktadır. Son bölüm ise deney sonuçlarının sunulup yorumlanmasına ayrılmıştır.

2 İki Boyutlu Doğrusal Dönüşüm

İki boyutlu doğrusal bir dönüşümü

$$\mathbf{y} = \mathbf{K}_{M \times L} \mathbf{x} \quad (1)$$

şeklinde yazabiliriz. Burada, \mathbf{K} katsayı matrisi, \mathbf{x} giriş işaret vektörü, \mathbf{y} çıkış işaret vektörüdür. Katsayı matrisi \mathbf{K} 'yı aynı boyutlardaki başka matrislerin doğrusal bileşimi olarak yazabiliriz. Bunu, \mathbf{K} matrisinin karışık matris sistemini kullanarak yapabiliriz. Diyelim ki \mathbf{K} matrisi P tane doğrusal olarak bağımsız matris içermektedir. O halde

$$\mathbf{K} = \sum_p^P \alpha_p \mathbf{K}_p \quad (2)$$

diyebiliriz. Buradaki α_p sabiti, \mathbf{K}_p matrisinin bir çarpanıdır. Bu çarpanların ve P 'nin değerleri şu koşullara bağlıdır:

1. **\mathbf{K} 'daki katsayıların ifade edilme biçimi:** Genelde ikili ifade ve/ya türevleri kullanılmaktadır. Bilindiği gibi işaretlenmiş basamak (İB) ve kanonik işaretlenmiş basamak (KİB), ikili ifadenin türevleridir. Son zamanlarda, dörtlü ifade üzerine kurulmuş kanonik işaretlenmiş basamak (KİB-4) veya işaretlenmiş basamak sistemleri (İB-4) de türetilmiştir [4]. Genelde bütün katsayılar için bir tek ifade metodu kullanılır. İfade edilen rakamlardaki sıfırdan farklı basamakların sayısı sistemdeki toplayıcı sayısını doğrudan etkilemektedir. Ayrıca rakamların içinde ve birbirleri arasında bulunabilecek ortak örüntülerin de toplayıcı sayısını azalttığı bilinmektedir.
2. **Çarpanların bulunma metodu:** Toplayıcı sayısını azaltmak temel hedef olduğundan, belli bir sayı sistemiyle ifade edilen katsayılardaki ortak örüntülerin ve onların çarpanlarını bulmak gerekmektedir. Bunun nasıl yapıldığı ise 3.Bölüm'de anlatılmaktadır.

Denklem 2, \mathbf{K}_p matrislerini içermektedir. Bilindiği gibi, matris çarpımlarında bir elemanı elde etmek için bir satır bir sütunla çarpılıp toplanır. Amacımız toplayıcı sayısını azaltmak olduğuna göre elde edilen \mathbf{K}_p matrislerindeki satırların içerdiği ortak terimleri bulmak gerekmektedir. Her \mathbf{K}_p matrisi, satırları kullanılarak şu şekilde yazılabilir:

$$\mathbf{K}_p = \begin{bmatrix} \mathbf{k}_{p1}^\tau \\ \mathbf{k}_{p2}^\tau \\ \vdots \\ \mathbf{k}_{pM}^\tau \end{bmatrix} = \sum_m^M 1_m \mathbf{k}_{p_m}^\tau \quad (3)$$

Burada \mathbf{k}_{p_m} , \mathbf{k}_p vektörünün m 'inci elemanına denk düşer. Algoritmamız, bütün \mathbf{K}_p matrislerindeki bütün satırları karıştıran vektör sistemini oluşturmaktadır. Varsayalım ki algoritmamız R tane böyle vektör oluşturmuş olsun. O halde her $\mathbf{k}_{p_m}^T$, R tane vektörün doğrusal bileşimidir. Böylece, 2.Denklem

$$\mathbf{K} = \sum_p \alpha_p \left[\sum_m 1_m \left(\underbrace{\sum_r \beta_{r,p_m} \mathbf{k}_r^T}_{\mathbf{k}_{p_m}^T} \right) \right] \quad (4)$$

şeklinde yazılabilir. Burada \mathbf{k}_r^T , R adetlik karıştıran vektör sistemindeki r 'inci vektör, β_{r,p_m} ise doğrusal bileşim çarpanıdır. Aynen 2 numaralı denklemde olduğu gibi, çarpanların ve P 'nin değerleri kullanılan metoda bağlıdır. Bizim metodumuz ise 3.Bölüm'de anlatılmaktadır.

Sonuç olarak çözmeyi hedeflediğimiz problem şu forma gelmiştir:

$$\begin{aligned} & \min && \text{toplayıcı sayısı} \\ & \text{öyle ki} && \\ & \mathbf{K} = \sum_p \alpha_p \left[\sum_m 1_m \left(\sum_r \beta_{r,p_m} \mathbf{k}_r^T \right) \right] && \end{aligned} \quad (5)$$

Doğal olarak toplayıcı sayısı P adet α_p 'ye, $P.R$ adet β_{r,p_m} 'ye ve R adet \mathbf{k}_r^T 'ya bağlıdır ve hepsi de değişkendir. Bu problemi, geliştirdiğimiz sezgisel yöntemlerle çözmeye çalışacağız.

3 Toplayıcı Sayısını Azaltma Algoritması

Bu makalede geliştirilen algoritma iki alt bölümden oluşmaktadır. Bunlardan birincisi katsayı matrisini 2.Denklem'deki gibi yazmamızı sağlayacak P adet α_p çarpanlarını ve \mathbf{K}_p matrislerini bulmamız içindir. Bunu yaparken toplayıcı sayısını azaltmak için katsayılardaki ortak terimler seçilmektedir. İkincisi ise birinci bölümde elde edilen \mathbf{K}_p matrislerindeki ortak toplamları bularak 4.Denklem'deki gibi \mathbf{K} matrisini yazabilmeyi hedeflemektedir. Bunun için R adet β_{r,p_m} çarpanları ve \mathbf{k}_r^T satır vektörleri bulunmaktadır. Her iki alt-algoritmanın çalışabilmesi için \mathbf{K} matrisindeki katsayıların belli bir sayı sistemiyle kullanıcı tarafından belirlenen kelime uzunluğu içinde ifade edilmesi gerekmektedir. Şimdi bu alt-algoritmaları inceleyelim:

3.1 Birinci Alt-algoritma: Katsayı matrisindeki elemanların içerdiği ortak örüntülerin bulunması

Daha önceden belirtildiği gibi tek boyutlu sistemler için oldukça verimli sonuçlar üreten algoritmalar halihazırda geliştirilmiştir. Bu tür bir algoritma, bu çalışmada geliştirilen algoritmada ortak örüntüleri bulmak için alt yordam olarak kullanılmaktadır. Bunun için \mathbf{K} 'daki katsayıları içeren tek boyutlu bir $\tilde{\mathbf{k}}$ vektörü oluşturulur ve $\mathbf{y}_d = \tilde{\mathbf{k}}x_d$ sistemi [5] yardımıyla çözülür. [5]'teki algoritmanın sonucunda kökü sahte giriş x_d , yaprakları ise $y_{d_i} = \tilde{k}_i x_d$ olan ikili toplayıcı ağacı oluşur. Yeni geliştirdiğimiz algoritmamız, bu ağacı kullanarak \mathbf{K} 'daki katsayıların ortak örüntüleri şöyle bulmaktadır:

[5] vasıtasıyla elde edilen ağacın $M.L$ tane yaprağı vardır. Bu yaprakları koparıncaya, yeni yaprakları toplayıcılar olan bir ağaç elde edilir. Bu yeni ağacın $S \leq M.L$ tane yaprağı vardır. Her yapraktan köke doğru olan bütün yolların birleşimi, o yaprağa ait alt ağacı oluşturur. Υ , bütün yapraklara ait alt ağaçların oluşturduğu orman olsun. Ormandan oluşturulacak her alt-ormandaki bütün ağaçların kesişimleri yeni alt ağaçlar verir ve bu yeni alt-ağaçlar, \mathbf{K} 'daki katsayıların ortak örüntülerini vermektedir. Ancak amaç, sistemdeki toplayıcı sayısını azaltmak olduğundan şu iki koşulu aynı anda sağlayan alt-orman, Υ^* , ve ona ait alt-ağaç, t^* , idealdir:

1. Υ^* alt-ormanındaki ağaçların sayısının mümkün olduğu kadar çok olması.

2. t^* ağacındaki düğüm (yani toplayıcı) sayısının mümkün olduğu kadar çok olması. Her alt ormandan elde edilen alt-ağaç en az bir düğüm, yani kökü, x_d , içerir.

O halde Υ^* ve t^* 'yi bulmak şu şekilde modellenir:

$$\begin{aligned}
& \max && |\Upsilon^*| |t^*| \\
& \text{s.t.} && \\
& && t^* = \bigcap_{t \in \Upsilon^*} t \quad , \forall \Upsilon^* \subseteq \Upsilon, \\
& && |t^*| \geq 1, \\
& && |T^*| \geq 2, \\
& && |\Upsilon^*|, |t^*| \in \mathbf{Z}^+.
\end{aligned} \tag{6}$$

Bu modelin çözülmesiyle bulunan t^* 'nin yapraklarının her biri 2. Denklem'deki α çarpanlarından birini vermektedir. Varsayalım ki, t^* 'nin $A \leq P$ tane yaprağı olsun. Bilindiği gibi Υ 'deki her ağaç, \mathbf{K} 'daki katsayıları gerçeklemektedir. O halde

$$\mathbf{K} = \sum_a^A \alpha_a \mathbf{K}_a + \mathbf{K}' \tag{7}$$

şeklinde yazılabilir. Burada \mathbf{K}_a 'nın sifıra eşit olan girişleri, \mathbf{K} 'nın Υ^* 'daki ağaçlar tarafından gerçekleştirilemeyen katsayılarına, \mathbf{K}_a 'nın sıfırdan farklı girişleri ise kaydırma ve/ya eksileme işlemlerine karşılık gelmektedir. \mathbf{K}' ise hem $\Upsilon \setminus \Upsilon^*$ 'deki ağaçlarla gerçekleştirilen katsayıları, hem de Υ^* 'da olan ağaçların t^* 'da olmayan kısımlarına karşılık gelen t' alt-ağaçlarıyla gerçekleştirilen katsayıları içermektedir. Υ^* 'da olan her t ağacı, " $t = t^* \cup t'$ öyle ki $t^* \cap t' = \{x_d\}$ " şeklinde yazılabilir.

Buraya kadar anlatılan kısım, \mathbf{K}' matrisinin yeni bir katsayı matrisiymiş gibi ele alınmasıyla tekrarlanır. Yinelenme, t^* ağacı sadece kökten ibaret oluncaya kadar sürer. Yinelenme bittiğinde 2.Denklem'i sağlayan P tane α_p , ki bunlardan sadece sonuncusu 1'e eşittir, ve P tane \mathbf{K}_p elde edilir, ki bunlardan ilk $(P - 1)$ tanesi sadece kaydırma ve/ya eksileme işlemleri içermekte olup sonuncusu ortak terim içermeyen katsayılarından oluşmaktadır.

Bu algoritmanın amacı elde edilen sistemdeki bütün α_p ve \mathbf{K}_p değerlerinin eşsiz olmasıdır. Ancak, bazı durumlarda sayısal değerleri birbirlerine aynı olan birden fazla α_p olabilir. Bu durumda, halihazırda elde edilenden daha iyi bir sonuç, aynı α_p 'ye sahip \mathbf{K}_p matrislerinin toplanmasıyla oluşturulan yeni matrisin üzerinde yukarıda bahsedilen yinelemeli algoritmanın tekrar koşturulmasıyla sağlanır. Bazı durumlarda da giriş değerleri birbirlerine aynı olan birden fazla \mathbf{K}_p matrisi olabilir. Benzer şekilde, halihazırda elde edilenden daha iyi bir sonuç, aynı \mathbf{K}_p 'ye sahip α_p çarpanlarının toplanmasıyla oluşturulan yeni çarpanın üzerinde [5] koşturulmasıyla sağlanır.

3.2 İkinci Alt-algoritma: Katsayı alt-matrislerinin satırlarındaki ortak örüntülerin bulunması

Birinci algoritma ile, 2.Denklem'deki matrislerin doğrusal bileşimiyle ifade edilen sistem oluşturulmaktadır. Bu matrislerdeki satırlar arasındaki ortak örüntülerin bulunması için [5] numaralı kaynakta bahsedilen metot üzerinde ufak değişiklikler yapılarak yeni bir algoritma elde edilmiştir: Bütün satırlardaki sütunlar ikiye ikiye incelenerek satırlar arasındaki benzer ikililer bulunur. Burada benzer ikili diye tanımlanan yapı, [5]'te bahsedilen yapıdan şu şekilde farklıdır: Eğer bir satırdaki bir ikili, başka bir satırda ama aynı kolonlarda bulunan bir ikilinin belli bir sabitle, β , ölçeklendirilmiş şekliyse bu iki ikili bir benzer ikili grubu oluşturur. Her grupta, farklı β değerleriyle ikiden çok ikili olabilir. En çok benzer ikilileri içeren grup ortak benzer ikili olarak seçilir ve içerilen satırlardan silinir. Bu ikili, 4.Denklem'deki k_r^T satır vektörüne denk düşmektedir. Elde edilen ortak benzer ikilinin daha sonraki yinelemelerde kullanılması için bütün matrislere şu şekilde yeni bir kolon eklenir: Ortak benzer ikiliyi daha önceden içeren satırlarda yeni sütunun girişleri, seçilen benzer ikili grubundaki β değerleri,

Table 1: Deney Sonuçları: Toplayıcı Sayıları

<i>Deney</i>	<i>Orijinal</i>	<i>KİB</i> $2n$	<i>KİB-4</i> n	<i>KİB-4</i> $n + 1$	<i>Orijinalden Kazanç</i> <i>(Yüzde olarak)</i>		
					<i>KİB</i> $2n$	<i>KİB-4</i> n	<i>KİB-4</i> $n + 1$
AKD_8	200	86	43	58	57	79	71
AKD_12	264	110	52	64	58	80	75
AKD_16	344	154	94	118	55	73	65
AKD_24	536	211	148	148	61	72	64
ÇE_8	94	55	12	12	41	87	87
ÇE_12	98	81	45	49	18	54	50
ÇE_16	151	112	80	88	26	47	41
ÇE_24	196	164	136	144	16	31	27

içermeyen sütunlarda ise sıfırdır. Bu sütunun girişleri, 4.Denklem'deki β_{r,p_m} çarpanlarına karşılık gelmektedir.

Bu algoritma her satırda sıfırdan farklı tek giriş kalana kadar kendini yineler. Sonuçta, Denklem 5 çözülmüş olur.

4 Deneyler ve Sonuç

Geliştirilen algoritma, Ayrık Kosinüs Dönüşümü (Tablo 1'deki AKD) ve [6] numaralı kaynakta açık tanımı yapılan on dört düğümlü sekiz dalı bir çoklu-evreli sonlu dürtü yanıtı süzgeç (Tablo 1'deki ÇE) üzerinde 8, 12, 16 ve 24 bitlik kelime uzunluğu sınırlandırmasıyla koşturulmuştur. Tablo 1'de de görüldüğü üzere standart KB ile katsayılar ifade edildiğinde toplayıcı sayısında orijinalden yüzde altmışlara varan bir iyileşme yeni geliştirilen algoritma ile elde edilmektedir.

KİB-4 kullanılarak gerçekleştirilen deneylerde kelime uzunluğu olarak 8, 12, 16 ve 24 bit yerine sırasıyla 4, 6, 8 ve 12 bit kullanılmıştır çünkü dört tabanında işlem yapıldığı için iki tabanıyla yapılan kelime uzunluğu sınırlandırılmasının yarısı kullanılmaktadır. Ancak yapılan araştırmalar sonucunda bir katsayı KİB kullanarak $2n$ -bit sınırlandırıldığında elde edilen hata, KİB-4 kullanan ve n -bit ile sınırlandırıldığında elde edilen hatadan daha az, KİB-4 kullanan ve $(n+1)$ -bit ile sınırlandırıldığında elde edilen hatadan daha fazla olduğu görülmüştür. KİB-4 ve n -bit kullanılarak yapılan sınırlandırılmalarda toplayıcı sayısındaki kazancın genelde KİB-4 ve $n+1$ -bit kullanılarak yapılan sınırlandırılmalarından yaklaşık yüzde sekiz daha fazla iyileşme sağladığı görülmüştür.

References

- [1] R. Brodersen, "Keynote: Why we need a custom chip- in- a- day design methodology," *2001 International Conference on Microelectronic Systems Education*, <http://www.mseconference.org/confplan.html>, June 2001.
- [2] P. Schaumont, I. Verbauwhede, K. Keutzer and M. Sarrafzadeh, "A quick safari through the reconfiguration jungle," *Proceedings of DAC2001*, <http://www.dac.com/39th/talkindex.html>, June 2001.
- [3] H. T. Nguyen and A. Chatterjee, "Number-splitting with shift-and-add decomposition for power and hardware optimizations in linear DSP synthesis", *IEEE Transactions on VLSI*, pp 419-423, August 2000.

- [4] J. O. Coleman, "Express coefficients in 13-ary, radix-4 to create computationally efficient multiplierless FIR filters", Proceedings of ECCTD'01, Espoo, Finland, Aug. 28-31, 2001.
- [5] A. Yurdakul ve G. Dündar, "A fast and efficient algorithm for the multiplierless realization of linear DSP transforms," *IEE Proceedings-Circuits, Devices and Systems*, değerlendirilmedi.
- [6] J. O. Coleman, J. J. Alter, and D. P. Scholnik, "FPGA architecture for Gigahertz-sampling wideband IF-to-baseband conversion," in *Proc. Intl Conf. on Signal Processing Applications and Technology (ICSPAT 2000)*, Dallas TX, Oct. 2000.