SERPİŞTİRİCİNİN KONVOLUSYONEL ÇARPIM KODLARININ PERFORMANSI ÜZERİNDEKİ ETKİLERİ

Orhan Gazi, A. Özgür Yılmaz*
Çankaya Üniversitesi
Elekronik ve Haberleşme Mühendisliği
Balgat, Ankara
o.gazi@ari.cankaya.edu.tr , aoyilmaz@eee.metu.edu.tr

*Orta Doğu Teknik Üniversitesi
Elektrik Elektronik Mühendisliği
Balgat, Ankara

Özet: *Bu makalede serpiştiricinin konvolusyonel carpım kodlarının performansları üzerindeki etkilerini inceliyoruz. Konvolusyonel çarpım kodlarının yapısı paralel çözümleme işlemine daha uygun olduğu için çözümlenme süresi diğer turbo türü kodlara göre önemli ölçüde azaltılabilmektedir. Serpiştiricinin turbo kodlarda olduğu gibi konvolusyonel çarpım kodlarının performansları üzerinde de önemli bir etkisi olmaktadır. Konvolusyonel çarpım kodlarının seri olarak birleştirilmiş konvolusyonel kodlara göre pratik uygulamalardaki avantajlarından bahsedeceğiz.*

1. Giriş

Shannon limitine yaklaşan en başarılı şalışmalardan birisi 1993 yılında Berrou, A. Glavieux and P. Thitimajshima tarafından yapıldı. [1]. Turbo kodlar diğer bir değişle paralel olarak birleştirilmiş konvolusyonel kodlar tanıtıldı. Bu kodlar kapasiteye çok yakın oranlarda çok düş ük (10^{-5} etrafında) bit hata oranları göstermektedir. Ayrıca çözümleme karmaşıklığı da katlanılabilir miktardadır. Sof-in soft-out algoritmalarının kullanımı bu başarıdaki anahtar faktörlerinden biridir. Son on yılda, seri birleşik konvolusyenel kodlar (SCCC) [2], LDPC kodlar [3], ve de çarpım kodları detaylı olarak incelendi. Çarpım kodları üzerindeki çalışmalar Elias'ın önderliğinde başlamıştır [4]. Çarpım kodları paralel çözümlemeler için diğer kodlara göre daha avantajlı durmaktadır. Şu ana kadar oluşturulan çarpım kodlarının yapılarında çogunlukla blok kodlar kullanılmıştır. Bu blok kodlar genelde Hamming, genişletilmiş Hamming], BCH, ve de Reed Solomon [5]-[6] kodlarıdır. Çarpım kodlarının DSP ve de FPGA uygulamarı da son zamanlarda ilgi çeken bir konu olmaktadır. Çarpım kodları genel olarak doğrusal blok kodları kullanılarak oluşturulmuşlardır. Blok kodların örgü yapıları zamanla zamanla değişen bir özelliğe sahiptirler [6]. Bizim bu makalede önereceğimiz çarpım kodları zamandan bağımsız olan konvolusyonel kodlar kullanılarak oluşturulmaktadır. Bu kodların örgü yapıları blok kodlarda olduğu (Hamming, BCH, SPC, Reed Solomon) gibi zaman içerisinde değişim östermezler. Ayrıca, blok kodların örgü yapısındaki durum sayısı kod sözcüğünün ve de veri zunluğunun farkı ile üstsel bir şekilde artmaktadır [6]. Diğer bir değişle C(n,k) doğrusal blok kodu gösteriyor olsun k veri bitlerinin sayısını n de kodlanmış bitlerin sayısını göstersin, bu durumda blok kodun örgü yapısında 2^{n-k} adet durum bulunmaktadır. Oysa konvolusyonel kodlardaki durum sayısını istediğimiz gibi ayarlayabiliriz ve de durum sayısı veri bitlerinin ve de kodlanmış bitlerin uzunluklarından bağımsızdır. Konvolusyonel kodların yukarıda bahsedilen avantajlarından ötürü çarpım kodu oluştururken konvolusyonel kodları kullananağız ve de bu şekilde elde ettiğimiz kodu konvolusyonel çarpım kodu şeklinde adlandıracağız (KÇK). Bu makalede KÇK'nin performansını etkileyen faktörleri inceleyeceğiz. Makalenin içeriği şu şekilde olacak. Bölüm II de önerilen kod yapısı ve de çözümleme algoritması açıklanacak. Bu kodların en küçük uzunlukları Bölüm III'te incelenecek. Numerik sonuçlar Bölüm IV'te verilecek. Bölüm V de avantajlarından bahsediyoruz. Sonuç kısımlarıda Bölüm VI'de belirtilecek.

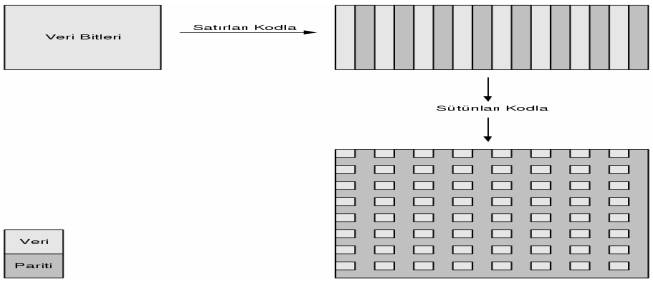
2. KÇK Kodlayıcısı ve Çözücüsü

2.1 KÇK Kodlayıcısı

Çarpım kodlarını oluşturuken doğrusal blok kodlar kullanmak yerine satır ve de sütünları kodlarken konvolusyonel kodlar kullandık. Bu durum Şekil 1 de izah edilmiştir. Kodlama işlemi bir matriksin yardımı ile yapılır. Kodlanacak veri bitleri bir matrikse yerleştirilir. Matriksin her satırı konvolusyonel kod kullanılarak ayrı ayrı kodlanır. Satır olarak kodlanmış veri serpiştiriciden geçirildikten sonra sütün kodlaması yapılır.

2.2 KÇK Çözücüsü

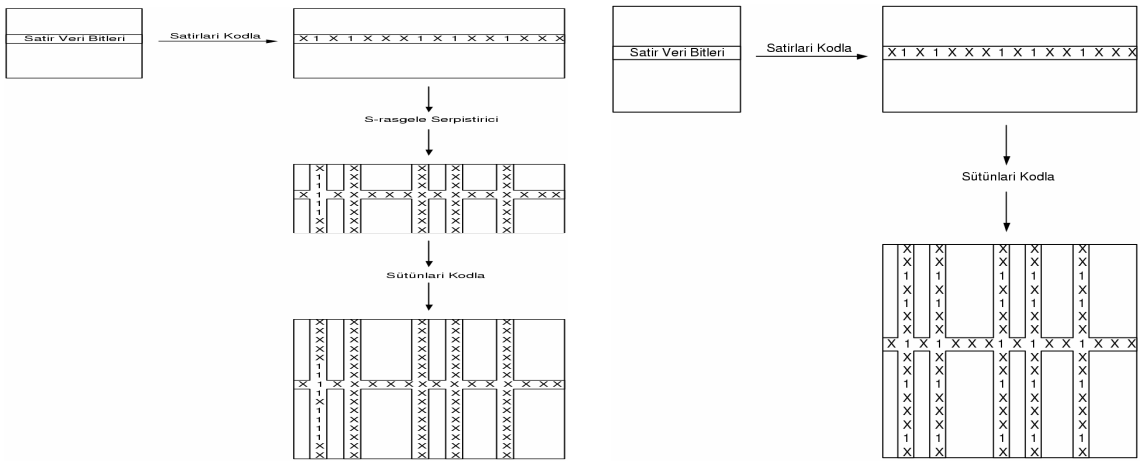
BPSK (Binary Phase Shift Keying) ile module edilen veri beyaz gürültülü kanaldan geçirilir. Çözümleme esnasında log-MAP yumuşak çözümleme algoritması konvolusyonel çarpım kodunu yinelemeli bir şekilde çözmek için kullanıldı [1]. Her sütün ve satır için farklı bit log-MAP çözücüsü kullanıldı.



Şekil 1. KÇK kodlama işlemi

3. KÇK'nin En Küçük Uzaklığı

Bu makalede $d_{bağımsız}$ 'ı parça konvolusyonel kodların bağımsız uzaklıkları olarak kabul ediyoruz. KÇK nin en küçük uzaklığınıda $d_{enKüçük}$ ifadesi ile göstericeğiz. $d_{enKüçük}$ değeri parça konvolusyonel kodların bağımsız uzaklığına ve de kullanılan serpiştiricinin türüne bağlı olarak değişmektedir. Kullanılan serpiştirici satır kodlamasından sonra '1' lerin sayısını değiştirmezse KÇK'nin $d_{enKüçük}$ değeri $d^2_{bağımsız}$ olmaktadır, diğer durumlarda $d^2_{bağımsız}$ değeri garanti edilemez. Bu durum Şekil 2 de tam S-rasgele ve de sadece sütünları S-rasgele karıştırıcılar için izah edilmiştir.



Şekil 2. KÇK'nin en küçük uzaklığının serpiştiriye göre değişimi.

4. Elde Edilen Veriler

Yeterli istatistiksel sonuçlar çıkarmak için simulayonlarda bir milyon veri vektörü kullandık ve de her veri vektörü 1024 bit içermektedir. KÇ K'nin oluşturulması esnasında oranı (1/2) olan sistematik dönüş ümlü konvolusyonel kod (1; 5/7)octal kullanıldı. Çözümlenmesi esnasında 12 yineleme kullanıldı.

4.1. Serpiştirici etkileri

4.1.1. Serpiştirici yok:

Bu durumda satır kodlama işleminden sonra serpiştirici kullanmadık. Örgü sonlandırma bitleri hem satırlara hem de sütünlara eklenmiştir. KÇ K'nin en küçük uzaklığı $d_{enKüçük} = d^2_{bağımsız} = 25$ olmaktadır. Örgü sonlandırma bitlerini eklenmesi $d_{enKüçük} = d^2_{bağımsız}$ eşitliğinin sağlanması için gereklidir, aksi halde $d_{enKüçük}$ değeri $d^2_{bağımsız}$ 'e eşit olmamaktadır. Bu kodun performans grafiği Şekil 3'te gösterilmektedir. Performans grafiğinden görüldüğü üzere en küçük uzaklığının değeri büyük bile olsa, KÇK küçük E_b/N_0 değerleri için iyi sonuçlar vermemektedir. Çok iyi şekilde bilindiği gibi en küçük uzaklığın etkisi yüksek E_b/N_0 değerlerinde daha iyi görülmektedir.

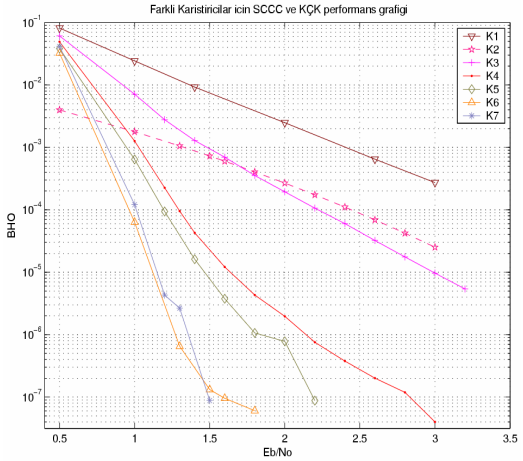
4.1.2. Tam S-rasgele Serpiştirici:

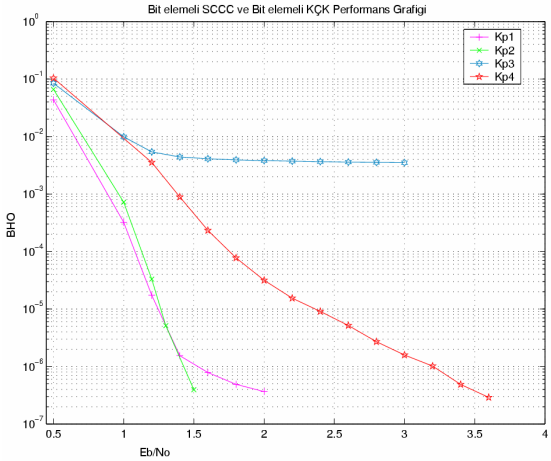
Satır kodlama işleminden sonra S-rasgele serpiştiricisı (S= 18) kullanıldı. Benzer yapılarından ötürü SCCC'u da benzetim yapıp elde edilen sonuçları KÇK ile karşılaştırdık. Her iki kodda küçük kod oranlarda iyi sonuşlar vermektedir. SGO (Sinyal Gürültü Oranı) değerleri bütün senaryolar için normalize edildi. Performans grafikleri Şekil 4'de görülmektedir. Performans eğrisinden de görüleceği üzere, diğer serpiştiricilerle karşılaştırıldığında en iyi performansı bu serpiştiricinin kullanıldığı KÇK vermektedir. Tam S-rasgele serpiştiricinin kullanımından dolayı KÇK'nin en küçük uzaklığı $d^2_{bağımsız}$ değerine eşit olmak zorunda değildir. Teorik olarak yüksek E_b/N_0

değerlerinde bu kodun performansının düşük değerli küçük uzaklığından ötürü diğer serpiştiriciler kullanılarak yapılanlara göre daha kötü olması beklenmektedir.

4.1.3. Sütün S-rasgele Serpiştiricisı:
Serpiştiricisiz duruma göre daha iyi performans elde etmek ve de KÇK'nin $d_{enKüçük} = d^2_{bağımsız}$ eşitliğini korumakiçin S-rasgele serpiştiricisini sadece her sütün için ayrı ayrı uyguladık ve de bu tür serpiştiriciye sütün S-rasgele serpiştiricisi ismini verdik. Bu tür serpiştiricilerin değişik kod oranlarındaki performansları Şekil 3' te gösterilmiştir.

4.2. Bit eleme etkileri:
Bit eleme işleminin sadece satırlara uygulanması KÇ K'nin oranının (2/3)'e yükselmesine sebep olur. Bu durumda tam S-rasgele serpiştiricisi iyi peformans gösterir. Bit eleme işlemi hem satırlara hem de sütünlara uygulanırsa KÇK'nin oranı (4/9)'a yükselmektedir. Bu durumda tam S-rasgele serpiştiricisi kötü performans sağlar. Sütün S-rasgele serpiştiricileri daha iyi performans sağlarlar. Performans grafikleri Şekil 3'te gösterilmiştir.





K1: Serpiştiricisiz KÇK (Oran:1/4)
K2: Teorik sınır (Oran:1/4)
K3: Helikal serpiştiricili KÇK (Oran:1/4)
K4: sütün S-rasgele serpiştiricili KÇK (Oran:1/4)
K5: Helikal ve de sütün S-rasgele serpiştiricili KÇK (Oran:1/4)
K6: Tam S-rasgele serpiştiricili KÇK (Oran:1/4)
K7: S-rasgele serpiştiricili SCCC (Oran:1/4)

Kp1: S-rasgele serpiştiricili SCCC (Oran:2=3)
Kp2: Tam S-rasgele serpiştiricili KÇK (Oran:2=3)
Kp3: Tam S-rasgele serpiştiricili KÇK (Oran:4=9)
Kp4: Sütün S-rasgele serpiştiricili KÇK (Oran:4=9)

Şekil 3. KÇK'nin en değişik serpiştiriciler ve de değişik oranlardaki performansı.

5. Avantajları
KÇK'nin temel avantajı paralel çözümlemeye eleverişli olmasıdır. Her satır ve de her sütün için kullanılacak olan bağımsız bir log-MAP çözücüsü çözümleme süresinin önemli ölçüde düşmesine sebep olacaktır [7].

6. Sonuçlar
Bu makalemizde konvolusyonel kodlar üzerine kurulmuş olan yeni bir tür çarpım kodu üzerinde durduk. Değişik serpiştiricilerin KÇK performansı üzerindeki etkilerini ayrı ayrı inceledik. Eğer iyi bir serpiştirici kullanılmazsa diğer kodlar KÇK den daha iyi performans göstermektedir. KÇK'nin en küçük uzaklığını maksimize etmek için karıştırma metodları önerdik. Yüksek oranlarda performansın korunması için özel serpiştiriciler önerdik.

7. Kaynaklar

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in Proc. ICC'93(Geneva, Switzerland, May 1993), pp. 1064-1070.
[2] S. Benedetto, L. Gaggero, R. Garello, and G. Montorsi, "On the design of binary serially concatenated convolutional codes," in Proc. VIII Communication Theory Mini-Conf. CTMC, Vancouver, BC, Canada, June 1999, pp. 32-36.
[3] R. G. Gallager, "Low Density Parity Check Codes," IRE Trans. Inform. Theory, IT-8:21-28, January 1962.
[4] P. Elias, "Error free decoding," IRE Trans. Inform. Theory, vol. IT-4, pp. 29-37, Sept., 1954.
[5] L. Hanzo, T. H. Liew, B. P. Yeap, Turbo Coding, Turbo Equalisation and Space-Time Coding. Wiley 2002.
[6] Shun Lin, Daniel J. Costello, Jr., Error Control Coding. Prentice Hall, 2004.
[7] Orhan Gazi, A. Özgür Yılmaz, "Konvolusyonel Çarpım Kodlarının Performans Analizi", İTÜSEM 2005, II. iIletişim teknolojileri ulusal sempozyumu, 17-19 Kasım 2005 Ç ukurova ¨ Universitesi, pp. 171-176.