İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

DVB-T MODELLEMESİ VE SİMÜLASYONU

YÜKSEK LİSANS TEZİ
Müh. Ahmet Koray YALÇIN
(504971058)

Tezin Enstitüye Verildiği Tarih : 11 Haziran 2001
Tezin Savunulduğu Tarih : 3 Temmuz 2001

Tez Danışmanı : Prof.Dr. Osman PALAMUTÇUOĞULLARI
Diğer Jüri Üyeleri Doç.Dr. Melih PAZARCI
Doç.Dr. Osman Nuri UÇAN (İ.Ü.)

HAZİRAN 2001

ÖNSÖZ

Yüksek lisans tez çalışmamı hazırlamamda bana yol gösteren değerli danışmanım Prof. Dr. Osman Palamutçuoğlu'na, yardımlarından ötürü Selçuk Paker ve Bülent Yağcı'ya, eğitimim boyunca benden maddi ve manevi desteği esirgemeyen aileme teşekkür ederim.

Haziran, 2001 Ahmet Koray Yalçın

İÇİNDEKİLER

KISALTMALAR

COFDM	: Dik Frekans Bölmeli Çoğullama (Coded Orthogonal Frequency Multiplexing)
DVB	: Digital Video Broadcasting
DVB-T	: Sayısal Karasal Yayın Standardı (Digital Terrestrial Video Broadcasting)
SAG	: Simgeler Arası Girişim
TAG	: Taşıyıcılar Arası Girişim
QASK	: Dik Genlik Kaydırmalı Anahtarlama (Quadrature Amplitude Shift Keying)
HFD	: Hızlı Fourier Dönüşümü
AFD	: Ayrık Fourier Dönüşümü
SQAM	: Staggered QAM
FEC	: İleri Yönde Hata Düzeltme (Forward Error Correction)
KDB	: Kanal Durum Bilgisi
MSE	: Ortalama Karesel Hata (Mean Square Error)
MAC	: Multiplexed Analog Component
HDTV	: Yüksek Tanımlı Televizyon (High Definition Television)
EU-95	: Eureka 95 Projesi
ETSI	: Avrupa Haberleşme Standartları Enstitüsü (European Telecommunications Standards Institue)
İGİÇ	: İlk Giren İlk Çıkar (Firs In First Out - FIFO)
QPSK	: Dik Evre Kaydırmalı Anahtarlama (Quadrature Phase Shift Keying)
QAM	: Dik Genlik Modülasyonu (Quadrature Amplitude Modulation)
PRBS	: Pseudo Rastgele İkili Dizi(Psuedo Random Binary Sequence)
TPS	: İletim Parametre İşaretleşmesi (Transmission Parameter Signalling)

TABLO LİSTESİ

ŞEKİL LİSTESİ

SEMBOL LİSTESİ

α	: İşaret uzayı oranı
Δ	: Koruma aralığı uzunluğu
B(e)	: İç bit dizileyicisine giren vektör
$b_{e,w}$	: İç bit dizileyicisi vektörünün w numaralı biti
$c_{m,l,k}$	: l. OFDM simgesinin k. taşıyıcısında yer alan karmaşık işaret
g(x)	: Reed-Solomon kod üreteç polinomu
H(q)	: İç simge dizileyicisi permütasyon işlevi
He(w)	: İç bit dizileyicisi permütasyon işlevi
I	: Dış konvolüsyonel dizileyicisinin dizilenim derinliği
I0,I1,I2,I3,I4,I5	: İç dizileyiciler
T_S	: COFDM simge uzunluğu
T_F	: Çerçeve uzunluğu
T_U	: Etkin simge uzunluğu (koruma aralığı olmadan)
Y	: İç simge dizileyici çıkış vektörü
Y'	: İç simge dizileyici ara vektörü
y_q	: İç simge dizileyici çıkışındaki q numaralı bit
y'_q	: İç simge dizileyici ara vektöründeki q numaralı bit
z	: Karmaşık modülasyon simgesi

DVB-T MODELLEMESİ VE SİMÜLASYONU

ÖZET

Avrupa'da, gelişmiş ve yüksek tanımlı televizyon için 1980'lerin başından beri devam eden çalışmalar, teknolojinin gelişmesiyle de paralellik göstererek bir çok aşamadan geçmiş ve 1990'ların başından itibaren, çok sayıda kurum ve kuruluşun içinde yer aldığı bir projeyle (DVB), tamamıyla sayısal televizyon dizgesi geliştirme doğrultusunda ilerlemiştir.

DVB Projesi'nde, kanal kodlaması ve televizyon işaretlerinin çoğullanması konusunda bir takım öneriler üretilmiş, uydu, kablo ve karasal iletim ortamları için televizyon yayın dizgeleri tasarlanmıştır.

Uydu ve kablo üzerinden sayısal televizyon yayını için standartlar kısa bir sürede hazırlanmış ve uygulamaya geçmiştir. Fakat oldukça zorlu ve işareti bozucu iletim kanalına sahip olan sayısal karasal yayının uygulamaya geçmesi o kadar kolay ve de hızlı olmamıştır.

Mikroelektronik ve işaret işleme teknolojisinde, DVB Projesi'nin başlamasına yol açan hızlı gelişmeler, uzun bir süredir bilinen fakat uygulaması karmaşık ve maliyetli olan dik frekans bölmeli çoğullama yöntemini (OFDM) , çoklu yol yayılmasına karşı sahip olduğu özellikler nedeniyle, sayısal karasal televizyon yayınında kullanmak için cazip hale getirmiştir.

Çalışmada DVB tarafından karasal yayın için geliştirilen DVB-T dizgesi ve bu dizgenin temelini oluşturan dik frekans bölmeli çoğullama yöntemi (OFDM) ele alınmış, sahip olduğu özellikler incelenmiş, kodlamayla birlikte, bu özelliklerin sayısal televizyon yayını için kullanılabilirliği araştırılmıştır.

Dizgenin daha iyi anlaşılması ve anlatılması ve AWGN kanallardaki başarımının ortaya koyulması amacıyla benzetime başvurulmuştur. Benzetim için Matlab kullanılmış ve Simulink'ten faydalanılmıştır. Benzetim sonucunda elde edilen "Eb/No – Bit hata oranı" eğrisinden yararlanarak, DVB'nin önerdiği "quasi-error-free" iletim (MPEG-2 ayrıştırıcı girişindeki işaretin bit hata oranının 10^{-11} - 10^{-10} aralığında bulunması şartı) için gerekli olan minimum Eb/No değeri bulunmuştur.

Elde edilen bu değeri ile DVB tarafından oluşturulmuş şartnamede yer alan benzetim sonuçları arasında %3'lük bir farkın olduğu görülmüş ve tasarlanan DVB-T benzetiminin doğru bir şekilde çalıştığı sonucuna varılmıştır.

MODELLING AND SIMULATION OF DVB-T

SUMMARY

The European development efforts for enhanced and high-definition television which has been carried out since early 1980s, went through a number of stages in parallel with the technology . Finally, the aim changed as to develope a complete digital television system based on a unified approach in the beginnig of 1990s and a project called DVB started with the gathering of many companies and institues in Europe.

Some suggestions about channel coding and the multiplexing of television signals was made and television broadcast systems for satellite, cable and terrestrial media was designed in the project.

The standards for digital satellite and cable television systems were preapared and applied in a short time by DVB but the application of digital terrestrial system, having a very challenging transmission channel structure, had not been so easy and fast.

Orthogonal Frequency Division Multiplexing (OFDM), which is available for a long period of time, became very attractive in parallel with the technolgy, for digital terrestrial television system due to the powerful features (such as immunity to the multipath fading) which it has.

In this study, DVB-T system, developed for digital terrestrial broadcasting and OFDM which is the heart of DVB-T, are examined. The features of OFDM and the feasibility of these features in digital terrestrial broadcasting are investigated.

The system is simulated in order to fully understand its structure and to obtain the performance in AWGN (additive white noise) channels using Matlab. The minimum Eb/No value for quasi-error-free transmission is obtained using the "Eb/No – Bit error rate" figure. The value obtained by the simulation is compared with the simulation results in DVB-T specification and 3% deviation is detected between the values.

As a result we can say that the designed simulator can be used for the simulation of the DVB-T in AWGN channels.

1. GİRİŞ

Sayısal teknoloji, temel bant verimliliği, esneklik ve RF başarımı gibi konularda büyük üstünlüklere sahiptir. Bu nedenden ötürü, televizyon işaretlerinin sayısal olarak iletilmesi yayıncılar için oldukça cazip bir seçenek haline gelmiştir.

Özellikle televizyon yayını açısından, sayısal işaretler daha sağlam ve spektrum kullanımı daha verimlidir. Analog yayının iletildiği bant genişliği içerisinde birden fazla program yayınlanabilmekte ve daha iyi görüntü kalitesine sahip olmaktadır. Ayrıca sayısal işaretlerin işlenmesi, analog işaret işlemeye kıyasla, daha kolaydır.

Sayısal televizyon yayıncılığına geçiş kolay ve hızlı bir şekilde gerçekleşmemiştir. Temelinde, 1980'lerin başında, gelişmiş ve yüksek tanımlı televizyon yayıncılığı için MAC(Multiplexed Analog Component) donanım geliştirme çalışmaları yatmaktadır. Bu çalışmalar, bölüm 3'te de anlatıldığı gibi, bir çok aşamadan geçerek, 1990'ların başından itibaren Avrupa kökenli DVB ve Amerika Birleşik Devletleri kökenli ATSC(Advanced Television System Committee) grupları tarafından organize edilen girişimlerle, değişik iletim ortamları için, tamamıyla sayısal yayın dizgesi standardları oluşturma gayretine dönüşmüştür.

Uzun yıllardır kullanılmakta olan frekans bölmeli çoğullama yöntemi, başlarda, telgrafta olduğu gibi, birden fazla düşük hızlı işaretin nispeten geniş bir banttan, her bir işaretin farklı taşıyıcı frekansı ile iletilmesi suretiyle kullanılırken, zaman içinde gelişerek, bölüm 2'de açıklandığı gibi yüksek hızlı verilerin verimli bir şekilde taşınmasında kullanılır hale gelmiştir.

Özel bir çok taşıyıcılı modülasyon yöntemi olan COFDM ile DVB'nin yolları 1990'ların ortasında, işaret işleme teknolojisinde yaşanan büyük gelişmeler sayesinde kesişti ve ortaya karasal ortamlarda, sayısal televizyon yayınlarının verimli bir şekilde iletimi için kullanılabilecek bir dizge çıktı.

Bu çalışmada amaçlanan, DVB-T sayısal karasal yayın dizgesini, tüm bileşenleriyle (özellikle, dizgenin kalbini teşkil eden COFDM ile) birlikte ayrıntılı bir şekilde incelemek ve dizgenin temel bant benzetimini gerçekleştirip AWGN kanallardaki başarımını ortaya koymaktır.

2. KODLU DİK FREKANS BÖLMELİ ÇOĞULLAMA (COFDM)

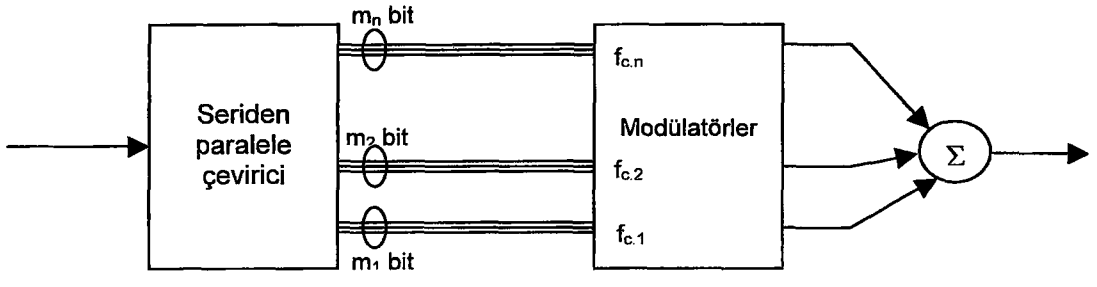
Tasarımında yenilikçi bir yaklaşım izlenen ve çoklu yol yayılmasına karşı başarımı oldukça yüksek, özel bir çok taşıyıcılı modülasyon yöntemi olan COFDM, sayısal karasal yayın işaretlerinin kodlanması ve iletimi için kullanılabilecek en iyi seçenek olarak karşımıza çıkmaktadır.

2.1 Çok Taşıyıcılı Modülasyon :

Alışılagelmiş bir seri veri dizgesinde, simgeler, her bir simgenin sahip olduğu frekans spektrumu tüm bandı kaplayacak şekilde, art arda iletilmektedir.

Paralel veri iletim dizgeleri, seri iletim dizgelerinde karşılaşılan pek çok sorunu ortadan kaldıran özelliklere sahiptir. Böyle bir dizgede, çok sayıda ardışık bit dizisi eş zamanlı iletilmektedir. Bunlara çok taşıyıcılı modülasyon dizgeleri de denilebilir.

Çok taşıyıcılı modülasyon, bir çeşit frekans bölmeli çoğullamadır. Çok sayıda ardışık bit dizisi, paralel bir şekilde eş zamanlı iletilir. Bu sayede, tek taşıyıcılı dizgelerde yüksek hızda iletilecek olan veri, çok sayıda parçaya bölünerek daha düşük hızlarda birden fazla frekans kanalı üzerinden iletilmektedir. Çalışma ilkesi, şekil 2.1'de gösterilmiştir.



Şekil 2.1 Temel çok taşıyıcılı verici

Şekil 2.1'den de görülebileceği gibi, toplam işaret frekans bandı n adet, birbiriyle örtüşmeyen, alt-kanala ayrılır. Her alt-kanal farklı bir simgeyle modüle edilir ve sonra, n adet alt-kanal, frekans uzayında çoğullanır.

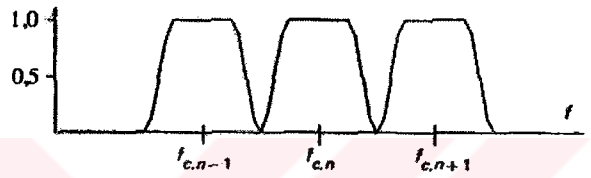
$Mf_s\ bit/sn$ frekansına sahip giriş verisi, f_s frekansında M bitlik bloklar halinde gruplandırılırlar. Girişteki M bit, N_C adet taşıyıcıyı modüle etmek için kullanılır. Taşıyıcılar Δf frekans aralığıyla spektrumda yer almaktadırlar.

$$f_{c,n} = n\Delta f \quad ;\ n = n_1\text{'den} \quad n_2\text{'ye} \tag{2.1}$$

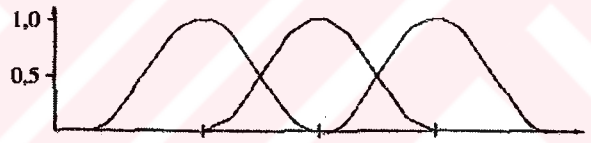
$$M = \sum_{n=n_1}^{n_2} m_n \tag{2.2}$$

burada $N_C = n_2 - n_1 + 1$'dir.

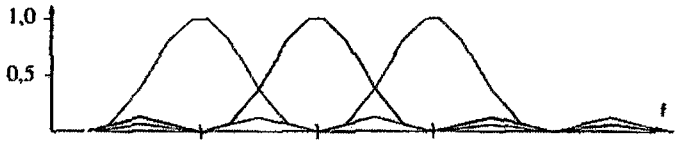
Modüle edilen taşıyıcılar iletim için toplanırlar. Taşıyıcılar, demodülasyondan önce, alıcı tarafından tek tek alınabilir olmalıdırlar.



Şekil 2.2 Frekans bölmeli çoğullama süzgeçlemesi



Şekil 2.3 SQAM spektrumu



Şekil 2.4 QASK spektrumu

Klasik bir çok taşıyıcılı dizgede, alt-kanalları birbirinden ayırmak için çok sayıda süzgeç kullanılmaktadır. Bitişik alt-kanallar arasında, alıcı tarafında süzgeçler kullanılarak rahatlıkla alınabilmeleri için, yeterli koruma aralığı bırakılmaktadır. Böyle bir dizge kullanılarak iletilen güç spektrumu şekil 2.2'de gösterilmiştir.

Çok taşıyıcılı dizgede yer alan alt-kanalların birbirleriyle örtüşmesine izin verilerek bant genişliği daha verimli kullanılabilir.

Alt-kanallar, alıcı tarafında ayrılabilmeleri (seperation) için diklik koşulunu sağlamalıdırlar. Bu ise alt-kanalları $f_U = 1/T_U$ aralıklarla yerleştirerek gerçekleştirilebilir. Burada T_U simge periyodunu temsil etmektedir. Böyle yapmakla taşıyıcılar dik bir işaret kümesi (orthogonal set) oluşturmuş olur :

Temel bantta, $k.$ taşıyıcı şu şekilde yazılabilir :

$$\psi_k(t) = e^{jk\omega_u t} \tag{2.3}$$

Burada $\omega_u = 2\pi / T_u$ olmaktadır. Taşıyıcının sağladığı diklik koşulu ise şu şekildedir:

$$\begin{aligned}\int_a^b \psi_p(t)\psi_q^*(t)dt &= \int_a^b e^{j[2\pi(p-q)t/T_U]}dt \\ &= \frac{e^{j[2\pi(p-q)b/T_U]} - e^{j[2\pi(p-q)/T_U]}}{j2\pi(p-q)/T_U} \\ &= \frac{e^{j[2\pi(p-q)b/T_U]}\left[1 - e^{j[2\pi(p-q)(a-b)/T_U]}\right]}{j2\pi(p-q)/T_U} \\ &= \begin{cases} 0 & P \neq q \quad ise \\ T_U & p = q \quad ise \end{cases}\end{aligned} \tag{2.4}$$

Bu eşitlikte $(b-a) = T_U$'dur.

Bu sayede, ilgilendiğimiz taşıyıcının frekansıyla aynı frekansa sahip bir işareti söz konusu taşıyıcıyla çarpıp sonucun periyot boyunca integralini aldığımızda, söz konusu taşıyıcı dışındaki (frekansları ω_U 'nun tam katlarına eşit) taşıyıcıların hepsi bizi 0 sonucuna ulaştıracaktır. Böylelikle tüm taşıyıcıların tek tek rahatlıkla alınabilmesi sağlanmış ve çok sayıda süzgeç kullanma gerekliliği ortadan kalkmış olur.

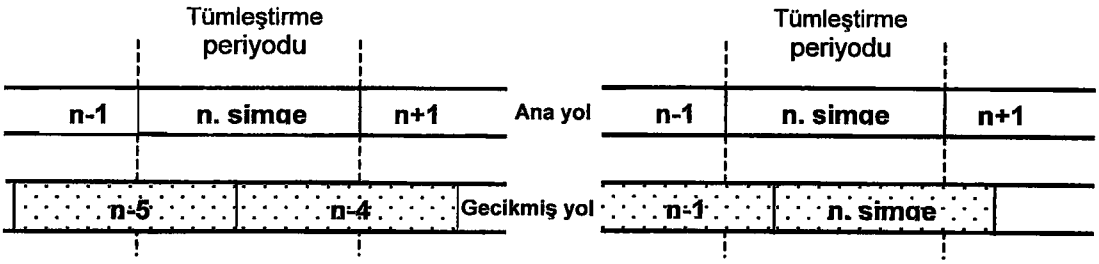
Böyle bir dizgeye örnek olarak SQAM gösterilebilir. Alt-kanallar birbirleriyle kesim frekanslarında örtüşmektedir. Verinin yarım simge periyodu kadar ötelenmesiyle diklik koşulu yerine getirilmektedir. İletim spektrumu şekil 2.3'te gösterilmiştir.

Uygulamada, alınan işaret örneklenmiş şekilde işlenmektedir. Bu durumda tümleştirme işlemi toplama, demodülasyon işlemi ise Ayrık Fourier Dönüşümü halini alır. Bu durumda alt-kanalların spektrumu, sınırlı bantlı olmayan $\sin c(f)$ fonksiyonu şeklinde olacaktır. Bu şekilde frekans bölmeli çoğullama işlemi bant geçiren süzgeçlemeyle değil de temel bant işaret işlemeyle gerçekleştirilmiş olur. Bu yöntem kullanılarak, alıcı ve verici HFD(Hızlı Fourier Dönüşümü) yöntemleri kullanılarak gerçekleştirilebilir. HFD işlemlerini hızlı ve de kolay bir şekilde yapan tümleşik devrelerin varlığı, dizgenin oldukça basit ve de ucuz bir şekilde gerçekleştirilmesine yol açar. QASK (Quadrature Amplitude Shift Keying) işaretleri bu şekilde gerçekleştirilebilir. QASK ile elde edilen işaretlerin sahip olduğu iletim spektrumu şekil 2.4'te gösterilmiştir.

2.2 COFDM'in Temel İlkeleri

COFDM, taşıyıcı aralıklarının her birinin diğerlerine dik olmasını sağlayacak şekilde seçildiği bir çeşit çok taşıyıcılı modülasyon olarak tanımlanabilir.

COFDM yöntemi, karasal yayında da söz konusu olan güçlü bozulmalardan etkilenen işaretlerin, alıcı tarafından düzgün bir şekilde alınmasında oldukça başarılıdır. COFDM, bölüm 2.1'de de anlatıldığı gibi, iletilecek bilginin çok sayıda taşıyıcıya bölünmesi ilkesine dayanmaktadır. Bu şekilde, modüle edilmiş simge periyodu, yansıma gecikme yayılmasından (echo delay spread) çok daha büyük tutulmuş olur.



(a) Uzun gecikme (b) Kısa gecikme

Şekil 2.5 İşaret gecikmesiyle SAG'in oluşması

Sayısal bilgiyle modüle edilmiş bir taşıyıcının, alıcı tarafından, iletim kanalının sahip olduğu bozulmadan dolayı, iki ayrı yol üzerinden alındığını varsayalım. Ana yol (alıcı ve verici arasındaki en kısa yol) üzerinden alınan işaretle gecikmiş işaret arasındaki gecikme bir simge periyodundan fazlaysa gecikmiş işaret, önceki simge veya simgelere ait bilgi taşıdığından, tamamıyla girişim gibi davranır. Buradan, sadece

düşük seviyelerdeki gecikmelerle başedilebileceği anlaşılmaktadır. Bu durum şekil 2.5a'da gösterilmiştir.

Gecikme bir simge periyodundan az olduğunda ise gecikmiş yol üzerinden alınan işaretin bir bölümü, önceki simgeye ait bilgi taşıdığından, girişim gibi davranır. Bu durum şekil 2.5b'de gösterilmiştir.

Bu incelemeden, makul seviyelerdeki gecikmelerle baş edebilmek için simge frekansının düşük tutulması gerektiği sonucu ortaya çıkmaktadır. Bu şekilde toplam gecikme yayılması, simge periyodunun çok küçük bir kısmına eşit olur. Bu da bizi yüksek hızlı verinin düşük simge frekansıyla iletilebilmesi için bu verinin çok sayıda düşük hızlı paralel veri dizisine bölünüp, her veri dizisinin kendi taşıyıcısıyla iletilmesi fikrine götürmektedir.

Simge periyodunu ya da taşıyıcı sayısını, bozulmayı önemsiz hale getirecek şekilde arttırmayı sınırlayıcı etkenler olarak, taşıyıcı kararlılığı, Doppler kayması, FFT boyutu ve gecikme sayılabilir.

Paralel yaklaşım, toplam işaretleşme aralığını yayarak dizgenin, gecikme yayılmasına (delay spread) karşı olan duyarlılığını azaltmasının yanı sıra frekans seçici sönümü birçok simge üzerine yaymaktadır. Bu şekilde, Rayleigh sönümünden kaynaklanan, bitişik simgelerin bloklar halinde bozulması durumu ortadan kalkmaktadır. Bunun yerine çok sayıda simge hafif şekilde bozulur ve büyük bir çoğunluğunun, FEC kullanmadan bile, düzgün bir şekilde alınabilmesi sağlanır.

Tüm iletim bandı pekçok dar alt-kanala bölündüğünden, her bir alt-kanalın düz frekans yanıtına sahip olduğu varsayılabilir. Her bir alt-kanal, orijinal işaret bant genişliğinin çok küçük bir parçasını oluşturduğundan, seri bir dizgeye kıyasla, eşitleme(equalization), bölüm 2.6 ve bölüm 2.7'de gösterildiği gibi daha kolay ve de daha hızlı yapılabilmektedir.

2.3 COFDM İşaretinin Matematiksel Gösterimi

Matematiksel olarak, tek taşıyıcılı dizgelerde, taşıyıcı şu şekilde ifade edilebilir :

$$s_C(t) = A_C(t)e^{j[\omega_C t + \Phi_C(t)]} \tag{2.5}$$

Gerçek işaret, $s_C(t)$'nin gerçel kısmıdır. İşaretin genliği, $A_C(t)$ ve evresi $\Phi_C(t)$, simgeden simgeye değişebilir.

Çok sayıda taşıyıcı içeren bir COFDM işareti ise şu şekilde gösterilebilir :

$$s_s(t) = \frac{1}{N}\sum_{n=0}^{N-1} A_n(t) e^{j[\omega_n t + \Phi_n(t)]} \tag{2.6}$$

Burada $\omega_n = \omega_0 + n\Delta\omega$ olmaktadır.

İşaret, $1/T$ örnekleme frekansı ile örneklenirse sonuç olarak :

$$s_s(kT) = \frac{1}{N}\sum_{n=0}^{N-1} A_n e^{j[(\omega_0 + n\Delta\omega)kT + \Phi_n]} \tag{2.7}$$

eşitliği elde edilir. Burada $NT = T_U$ olmaktadır.

(2.7) numaralı eşitlikte, genelliği bozmadan, $\omega_0 = 0$ yaparak :

$$s_s(kT) = \frac{1}{N}\sum_{n=0}^{N-1} A_n e^{j\Phi_n} e^{j(n\Delta\omega)kT} \tag{2.8}$$

eşitliğini elde ederiz.

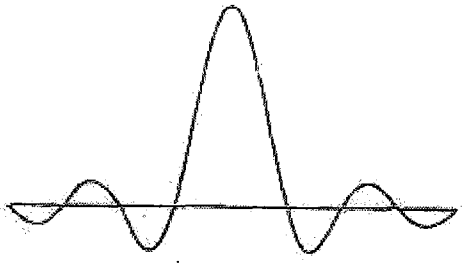
(2.8) numaralı eşitlik, ters HFD'nün genel şekline (2.9) benzemektedir :

$$g(kT) = \frac{1}{N}\sum_{n=0}^{N-1} G\left(\frac{n}{NT}\right) e^{j2\pi nk/N} \tag{2.9}$$

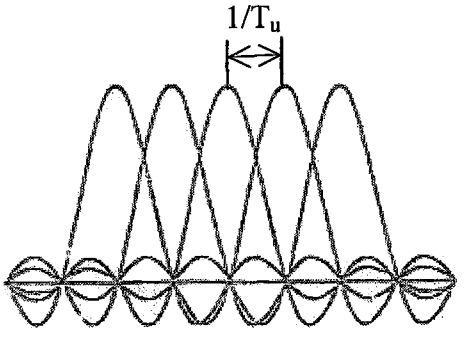
(2.8) numaralı eşitlikte yer alan $a_n = A_n e^{j\Phi_n}$, işaretin ayrık frekans uzayındaki değerini, $s(kT)$ ise zaman uzayındaki değerini temsil etmektedir.

Eğer $\Delta f = \frac{1}{NT} = \frac{1}{T_U}$ şartı sağlanırsa (2.8) ve (2.9) numaralı eşitlikler birbirlerinin eş değeri olur. Bu şart, daha önce de açıklandığı gibi diklik için gerekli olan bir şarttır. Buradan da görülmektedir ki, işaretler arasındaki dikliği sağlamakla COFDM işaretini, HFD kullanarak ifade edebilmekteyiz.

Bu şartları sağlayan $\sin(x)/x$ işlevi, COFDM modemde taşıyıcı olarak kullanılmaktadır.

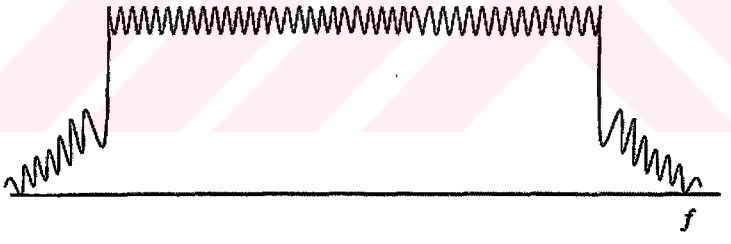


Şekil 2.6 COFDM alt-kanal spektrumu



Şekil 2.7 COFDM'de spektrum örtüşmesi

Şekil 2.6'da bir COFDM alt-kanalının, şekil 2.7'de ise alt-kanalların ne şekilde üst üste bindiği gösterilmiştir. COFDM işaretinin spektrumu ise şekil 2.8'de verilmiştir.



Şekil 2.8 COFDM işaretinin spektrumu

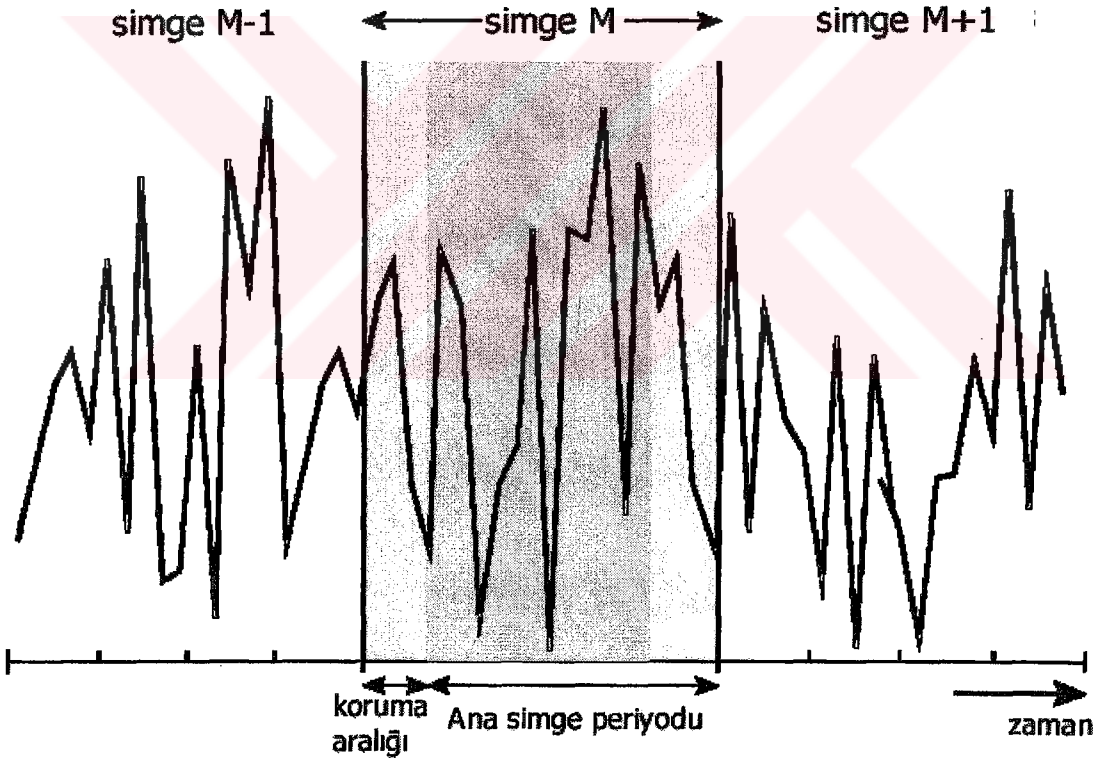
2.4 Koruma Aralığı

Gecikme yayılması bir simge periyodundan küçük bile olsa, her bir simgenin sonuyla bir sonraki simgenin başı arasında, düşük miktarda simgeler arası girişim meydana geldiğini gördük. Her COFDM simgesinin başına, işaretin sonundan alınan, belli uzunlukta, bir parça koyularak bu önlenebilir. Bu eklentiye koruma aralığı adı verilmektedir. Bu şekilde oluşturulan yeni simgenin periyodu, alıcının gelen işaretleri değerlendirdiği tümleştirme periyodundan (T_U) daha büyük hale getirilmektedir.

Gecikme süresi koruma aralığı süresinden kısa olduğu sürece, bütün işaret bileşenleri alıcı tarafından başarıyla alınır ve işaretler arasındaki diklik korunur.

Bu durumda bile, alıcıya gecikerek ulaşan işaret bileşenleri, ana yol üzerinden alınan işareti yapıcı veya yıkıcı olarak etkilemektedir.Simgeler arası girişim ortadan kalkmış, onun yerini doğrusal bozulma almıştır. Bu durmda, basit bir şekilde, alınan işaretin, iletilen işaretin kanalın frekans yanıtıyla çarpılması sonucu elde edilebileceği düşünülebilir (Matematiksel olarak, işarete koruma aralığının eklenmesi, işaretin, kanalın darbe yanıtıyla konvolüsyonunu dairesel konvolüsyona dönüştürmüştür. Bu nedenle, alınan işaret, iletilen işarete ve kanal frekans yanıtına ait AFD frekans katsayılarının çarpılması ile kolaylıkla ifade edilebilir.). Ancak gecikme, koruma aralığı süresinden uzun olduğu takdirde simgeler arası girişim (SAG) gerçekleşir.

Koruma aralığının bir başka olumlu tarafı da, sağladığı toleransla, alıcı eş zamanlamasının daha kolay yapılabilmesini sağlamasıdır.



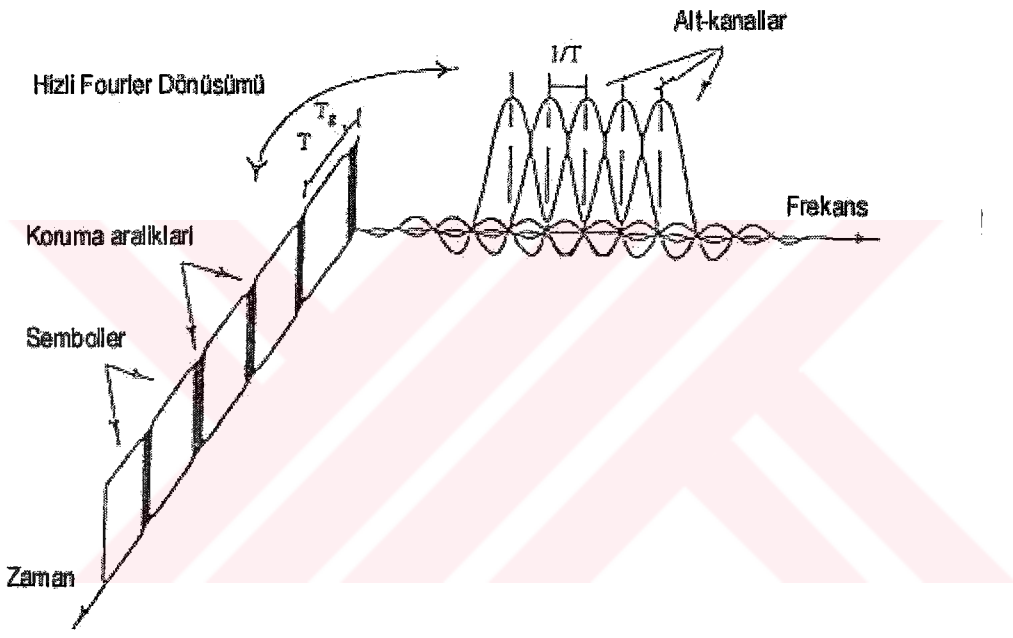
Şekil 2.9 Koruma aralığı için bir örnek

Koruma aralığı süresinin bilginin taşındığı simge süresine (T_U) oranı şartlara bağlı olarak belirlenebilir. Bu oran çok büyük tutulmamalıdır. Aksi takdirde, veri taşıma kapasitesi ve frekans veriminin düşmesi kaçınılmazdır. Ayrıca veri taşıyıcı sayısı sabit tutulduğu takdirde daha uzun koruma aralığına sahip işaretleri iletmek için

daha fazla güce ihtiyaç duyulur. Uygulamada oran, 1/4'ün katları olacak şekilde seçilmektedir.

SAG'in ve taşıyıcılar arası girişim (TAG)'in meydana gelmesinde rol oynayan diğer etkenler arasında, lokal osilatörde oluşan hatalar, alıcının örnekleme frekansında oluşan hatalar, lokal osilatörde oluşan evre gürültüleri sayılabilir. Koruma aralığı kullanılarak SAG'den kurtulabilmek olası olduğu halde TAG varlığını sürdürür.

Her bir COFDM simgesi zaman uzamında, her bir simgede yer alan taşıyıcılar ise frekans uzamında gösterilebilir. Şekil 2.10'da koruma aralığı kullanılarak oluşturulmuş COFDM işaretin zaman ve frekans uzamındaki spektrumu gösterilmiştir.



Şekil 2.10 COFDM işaretinin zaman ve frekans uzamında gösterimi

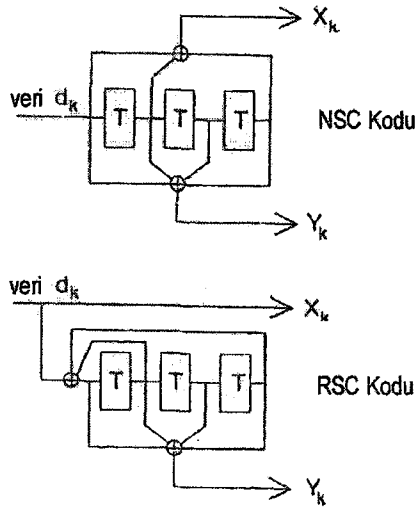
2.5 Kodlama

Çok taşıyıcılı modülasyon yönteminin kullanıldığı, hemen hemen tüm uygulamalarda, kodlama olmadan tatmin edici bir başarım elde edilememektedir. Sönümlemeye maruz kalan kablosuz dizgelerde, kabul edilebilir bir hata olasılığına ulaşabilmek için oldukça yüksek işaret-gürültü oranları gerekmektedir. Buna ek olarak, diğer kablosuz kanallardan kaynaklanan girişim oldukça şiddetli olabilmektedir. Bu gibi durumlarda, mümkün olan en yüksek bit iletim hızlarına ulaşabilmek için kodlama gerekmektedir.

Kuvvetli yansımalar söz konusuyken iletilen bazı taşıyıcılar, yansıyan işaretlerin yaptığı yıkıcı etki sonucunda çok fazla sönümlenirken diğer taşıyıcılar ise yapıcı etki sonucunda kuvvetlenir, alıcı girişindeki ortalama işaret gücü artar. Spektrumun belli kısımlarının oldukça fazla sönüme uğramış olmasına rağmen bu güç artışından, sağlam bir kanal kodlama yöntemi kullanılarak, faydalanılabilir.

Kodlama ile, frekans ve zaman uzayında gerçekleştirilen serpiştirme işlemleriyle beraber, farklı taşıyıcılarda iletilen bitler arasında bir ilinti kurulmaktadır: Serpiştirme işleminden sonra bitler, iletim dizisi içinde dağıtılmış olur. Bu şekilde, aynı simgeye ait bitlerin tümünün, hata patlamaları olarak ortaya çıkan kanal bozulmalarından büyük derecede etkilenmesi önlenmiş olur. Kodlama sayesinde de, alıcı tarafında sorunsuz bir şekilde alınmış taşıyıcı bitleriyle, diğer sorunlu taşıyıcı bitleri arasında bir ilinti kurulmuş olur ve bu ilinti sayesinde, sönüme uğramış taşıyıcılarda yer alan bilgi, alıcı tarafından yeniden oluşturulabilir. Bu yüzden kodlamayı, sönümü tüm işaret bandına yayan bir araç olarak da görebiliriz.

Sayısal iletim dizgelerinde, konvolüsyonel kanal kodlaması oldukça yaygın olarak kullanılmaktadır. Bunun en önemli nedeni, dikkate değer bilgi kaybına yol açmadan, gerçek zamanda kod çözme işleminin, Viterbi algoritması sayesinde, gerçekleştirilebilmesidir. Aynı Viterbi kod çözücü blok, delme olarak adlandırılabilecek bir işlem (puncturing) ile çeşitli kod oranlarının elde edilebilmesine olanak tanımaktadır. Uygulamada iki tip konvolüsyonel kodlamadan söz edilebilir : Dizgesel olmayan konvolüsyonel kodlama ve yinelemeli dizgesel konvolüsyonel kodlama. Şekil 2.11'de, aynı parametrelerle (kod belleği $v = 3$, üreteç polinomu : 15,17 ve oran $R = 1/2$) oluşturulmuş her iki tip kod görülebilir.



Şekil 2.11 Dizgesel olmayan konvolüsyonel (NSC) ve yinelemeli dizgesel konvolüsyonel (RSC) kodlar

Eğer oran (R) sabit tutulursa, konvolüsyonel kodların hata düzeltme yetenekleri, kod belleği (v) ile doğru orantılı olarak değişir. Kod çözücünün karmaşıklığı ise $v2^v$ şeklinde, üstel olarak artar.

Konvolüsyonel ve blok kodları birleştirerek art arda kullanmak oldukça etkili bir yöntemdir. Bu şekilde yüksek kodlama kazançlarına, makul kod çözücü karmaşıklığı ile beraber, ulaşılabilir. Dış kodlamada(Reed-Solomon) blok kodlar kullanılmaktadır. Vericide, dış kodlamadan hemen sonra uygulanan (alıcıda, kod çözme işleminde bu sıra tam tersidir) konvolüsyonel iç kodlama, özellikle yumuşak kararlı kod çözme uygulandığında, hata olasılığının düşürülmesinde oldukça etkilidir. Konvolüsyonel kodlama bloğunda bir hata yapıldığında (örneğin Viterbi algoritması yanlış bir yol seçtiğinde) bu, geniş bir hata patlaması şeklinde ortaya çıkar. Bu durumda dış kodlama bloğu devreye girmekte ve hatanın düzeltilmesinde etkili olmaktadır.

COFDM'de kodlama, hem frekans hem de zaman uzamında gerçekleştirilerek frekans ve zaman seçici sönümlemeye karşı olan başarım arttırılmaktadır. Bu amaçla, bir önceki paragrafta bahsedilen her iki tip kodlamadan sonra frekans ve zaman uzamında serpiştirme(interleaving) işlemi uygulanır.

2.6 Taşıyıcı İyileştirmesi ve Frekans Uzayında Eşitleme

Koruma aralığı kullanılarak, koruma aralığı süresinden daha kısa olan yansımaların işaretin dikliğini bozması engellenmiş olur. Ama, alıcı tarafında eş zamanlamada meydana gelen sorunlar (örnekleme frekansı ve evresinin kayması), verici ve alıcı tarafında oluşan evre gürültüsü, kanalın yapısı yüzünden meydana gelen evre kaymaları, donanımdan dolayı doğrusallığın bozulması gibi nedenlerden dolayı COFDM simgesinde yer alan taşıyıcıların konumları, iletildikleri gibi kalmayabilir. Bundan dolayı da taşıyıcılar arası diklik bozulabilir.

Daha önce de açıklandığı gibi, her bir alt-kanal dar bir bant genişliğine sahip olacaktır. Bu yüzden her bir alt-kanal üzerindeki sönüm, düz sayılabilir (frekans seçici olmayan sönüm). Bu durumda, $n.$ taşıyıcıda iletilen simgede, iletim kanalından geçerken sadece genlik değişikliği ve evre dönmesi meydana gelir. Bu yüzden, alıcı tarafında alınan işaretlerin $H_n a_n$ şeklinde gösterilmesi yanlış olmaz. Burada a_n , verici tarafından iletilen modülasyon simgelerini ve H_n, $n.$ frekans

indisindeki kanal frekans yanıtını temsil etmektedir. $H_n a_n$ alıcı tarafında AFD kullanılarak elde edilebilir. $s'(t)$ alınan işaret olmak üzere :

$$H_n a_n = \sum_{k=0}^{M-1} s'(kT) e^{-2j\pi nk/N} \qquad (2.10)$$

olarak elde edilebilir.

Eş zamanlı demodülasyon yapabilmek için kanal frekans yanıtı H_n, n'nin alacağı her değer için bulunmalıdır. Bunu gerçekleştirmenin bir yolu, iletilen işaretin içine, zaman ve frekans uzayında, düzenli bir şekilde referans bilgisi taşıyan pilot taşıyıcılar yerleştirmektir. İletilen pilot taşıyıcılar, konumları alıcı tarafından da bilindiğinden, işaret eş zamanlaması gerçekleşir gerçekleşmez, alınabilir.

Frekans seçmeli kanalda iletim söz konusu ise, pilot taşıyıcıları ve veri taşıyıcıları arasında yüksek ilinti sağlanmalıdır. Bu da, daha önce de ifade edildiği gibi, kodlamayla mümkün olmaktadır.

Pilot taşıyıcılar, sabit ve de alıcı tarafından bilinen bir genlik değeriyle iletilirler. Alınan her pilot işaretinin sahip olduğu genlik değeri, beklenen değerle karşılaştırılarak, söz konusu taşıyıcı için, o anki kanal yanıtı elde edilebilir. Düzeltilecek veri taşıyıcıları pilot taşıyıcıların arasında yer almaktadır. Pilot taşıyıcılar için elde edilen kanal yanıtı değerlerinin ara değerlerini almak suretiyle H_n, n'nin alacağı her değer için tahmin edilebilir.

Tahmin işleminden sonra, alınan veri taşıyıcıları ($H_n a_n$), ara değer bulma işlemi (enterpolasyon) sonucu elde edilen değerlere (H_n) bölünür ve ilgilenilen taşıyıcı değerine (a_n) ulaşılır. Bu yöntem kullanılarak, kanal kestirimi ve eşitlemesi, sorunsuz ve de hızlı bir şekilde yapılabilir.

Kanal kestiriminde, aynı COFDM simgesi içerisinde yer alan pilot taşıyıcılardan faydalanarak frekans, birbirini takip eden COFDM simgelerinde yer alan pilot taşıyıcılardan yararlanarak zaman uzamında ara değer bulma işlemi yapılmaktadır.

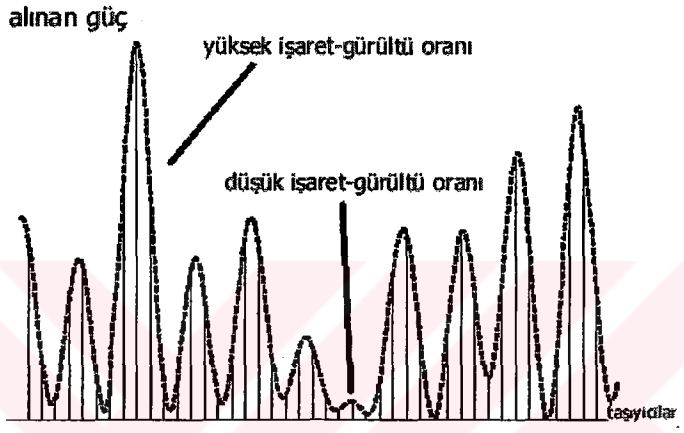
Pilot taşıyıcılarının sayısı arttıkça, dizgenin kanal bozulmalarına, sönümlemelere karşı olan direnci artar fakat bu durumda da mevcut bant verimli kullanılmamış olur. Bu iki etken arasındaki denge, tasarımda dikkat edilmesi gereken noktalardan biridir.

Bu konuyla ilgili daha ayrıntılı bilgi, bölüm 2.7'de verilmektedir.

2.7 Kanal Durum Bilgisi (KDB)

Verinin tek taşıyıcı kullanılarak iletilmesi durumunda tüm veri simgeleri, hemen hemen aynı gürültü gücünden etkilenecektir. Alıcı tarafında, yumuşak kararlı (soft decision) kod çözme yönteminin kullanılması durumunda, sadece, bu gürültü nedeniyle simgeden simgeye oluşan rastgele değişimlerin dikkate alınması yeterli olacaktır.

Verinin çok taşıyıcılı bir dizge kullanılarak iletilmesi durumunda ise her bir taşıyıcının sahip olduğu işaret-gürültü oranı, frekans seçici sönümlemeden dolayı, farklı olacaktır. Bu durum şekil 2.12'de gösterilmiştir.



Şekil 2.12 Kanal yanıtı

Bu yüzden, yumuşak karar vermede, simgeden simgeye olan değişimlerin yanı sıra yüksek işaret-gürültü oranına sahip olan taşıyıcılar tarafından taşınan verinin düşük işaret-gürültü oranına sahip taşıyıcılar tarafından taşınan veriye göre daha doğru bir şekilde alınabileceği etkeni de dikkate alınmalıdır. Bu fazladan bilgi, genellikle kanal durum bilgisi (Channel State Information - CSI) (KDB) olarak adlandırılır. Kanal durum bilgisinin iç kod çözücüde kullanılması bizleri daha iyi bir dizge başarımına götürecektir.

2.7.1 KDB'nin elde edilmesi

$k.$ taşıyıcının değerini bildiğimizi varsayalım (pilot taşıyıcı). Alıcıda, HFD bloğu çıkışını (p'_k) şu şekilde ifade edebiliriz (koruma aralığının eklenmesiyle doğrusal konvolüsyon işlemi dairesel konvolüsyon halini almış ve işlemler HFD katsayılarının çarpımıyla yapılabilir hale gelmiştir):

$$p'_k = p_k h_k + n_k \tag{2.11}$$

Bu eşitlikte, p_k, k numaralı pilot taşıyıcının sahip olduğu referans değer, h_k, kanal yanıtı, n_k ise gürültü bileşenidir. Eşitlenmiş e_k çıkışı şu şekilde kestirilebilir :

$$e_k = (p_k h_k + n_k)/h'_k \tag{2.12}$$

Bu eşitlikte, h'_k, kanal yanıtı h_k'nın kestirilmiş değeridir.

e_k'nın MSE (ortalama karesel hata – mean square error) değeri şu şekilde kestirilebilir :

$$MSE(e_k) = \left\langle |p_k - e_k|^2 \right\rangle = \left\langle \left| p_k \left(1 - \frac{h_k}{h'_k}\right) - \frac{n_k}{h'_k} \right|^2 \right\rangle \tag{2.13}$$

Burada $\langle \cdot \rangle$ zaman ortalaması işlevini temsil etmektedir.

Eğer kanal yanıtı doğru bir şekilde kestirilebilirse yani h'_k yaklaşık olarak h_k'ya eşit olursa aşağıdaki eşitlik elde edilebilir :

$$MSE(e_k) \cong \left\langle \frac{n_k^2}{h'^2_k} \right\rangle = \frac{gürültü \quad gücü}{kanal \quad yanıtı \quad gücü} \tag{2.14}$$

Görüldüğü gibi, e_k'nın MSE değeri işaret-gürültü oranının tersi olmaktadır.

Sonraki bölümlerde ele alınacak olan DVB-T dizgesinde olduğu gibi, pilot işaret sadece gerçel kısma sahip olursa, n_k'nın sanal ve gerçel kısımlarının birbirleriden bağımsız olma koşulu da sağlanırsa, (2.13) numaralı eşitlik (2.15) numaralı eşitlik haline gelebilir.

$$MSE(e_k) \cong \left\langle \left(img[e_k]^2 \right) \right\rangle \tag{2.15}$$

Burada $img[e_k]$, e_k'nın sanal kısmını temsil etmektedir. Eşitlikten p_k'ya artık gerek duyulmadığı görülmektedir.

Sonuç olarak, (2.15) numaralı eşitlik kullanılarak pilot taşıyıcı konumlarındaki MSE hesaplanır, sonra bu değerin tersi alınır ve sonuç normalize edilir ve en sonunda, taşıyıcı konumlarında hesaplanan bu değerler enterpole edilir ve diğer veri taşıyıcılarına ait KDB elde edilir.

Hesaplanan KDB, her bir etkin taşıyıcının kanal tarafından nasıl etkilendiğini gösterir. Geleneksel Viterbi kod çözücüde, tüm dal uzunlukları (metrics) eşit ağırlıklı olarak hesaplanmaktadır yani alınan işaret bitleri ile kodlayıcı kafes diyagramı arasındaki Öklid uzaklıkları hesaplanarak elde edilmektedir. Fakat COFDM'de her etkin taşıyıcı, çoklu yol (multipath) sönümlemeden dolayı diğerlerinden farklı bir şekilde bozulmaktadır.

Weon-cheol Lee, Hyung-Mo Park ve Jong-Seok Park tarafından önerilen bir yöntemde, yumuşak kararlı kod çözme için gelen işaretlerin Öklid uzaklıkları hesaplanır[10]. Eğer alınan taşıyıcıların KDB değeri, belli bir eşik değerinin altında ise dal uzunluğu (branch metric) KDB değeri ile çarpılır. Bu şekilde KDB, dal uzunluklarının belirlenmesinde ağırlaştırma çarpanı olarak kullanılır. Alınan taşıyıcıların KDB değerlerinin eşik değerin üstünde kalması durumunda ise ağırlaştırma işlemi gerçekleştirilmez. Bu yöntemle, Viterbi kod çözücünün başarımı arttırılmış olur.

3. DVB (DIGITAL VIDEO BROADCASTING) PROJESİ

Avrupa'da gelişmiş ve yüksek tanımlı (high-definition) televizyonun kökleri, çoğullanmış analog bileşen (multiplexed analog component - MAC) donanımının geliştirilmesiyle ilgili çalışmaların başladığı, 1980'lerin başına kadar uzanmaktadır. Başlangıçta doğrudan uydu yayın servisi (direct brodacast satellite) olarak tasarlanmış ve daha sonra birçok aşamadan (A,B,C,E olarak adlandırılan) geçmiştir. MAC geliştirilerek, D ve D2 şekilleriyle olgunluğa erişmiştir.

Bu aşamadan sonra, D-MAC ve D2-MAC dizgeleriyle uyumlu olacak Avrupa HDTV dizgesi için standartlar oluşturmak amacıyla Eureka programı, EU-95, başlatıldı. Bu program için kullanılan yaygın ad HD-MAC'ti.

HD-MAC analog ve sayısal kısımlara sahip, melez bir dizgeydi. Görüntü işareti analog olarak, ses kanalları, teletekst ve koşullu erişim verisi sayısal olarak taşınmaktaydı.

EU-95 ve D-MAC/D2-MAC dizgelerinin geliştirilmesi için çok büyük bir çaba harcandığı halde, pazarın durumu ve sayısal işaret işlemede gerçekleşen hızlı gelişmeler Euraka programı, yerini, tamamıyla sayısal yöntemler üzerine kurulu bir televizyon dizgesi geliştirmek amacıyla başlatılan DVB projesine bıraktı.

DVB, sayısal yayıncılığı dünya çapında standart hale getirmek için çalışan, çok sayıda firma ve kurumun ortaklaşa kurduğu Avrupa kökenli bir girişimdir. 1993 yılının eylül ayında faaileye geçen bu girişim, 35'i aşkın ülkeden 300 civarında üyeye sahiptir.

Kurulduğundan bugüne çok sayıda şartname hazırlamış ve hazırlanan bu şartnameler, Avrupa Haberleşme Standartları Enstitüsü (ETSI – European Telecommunications Standards Institute) gibi kurumlar tarafından standart haline getirilmiştir.

DVB projesinde, EU-95 projesinde kazanılmış olan araştırma sonuçları ve bilgi birikimi kullanılmış, bunun yanında, analog tabanlı donanım kullanımından tamamıyla vazgeçilmiştir.

Değişik ortamlarda sayısal görüntü işareti iletimi için hazırlanan DVB standartlarının temel ilkesi "birlikte çalışabilirlik"tir. Dizgeler, mümkün olduğunca, ortak bloklar kullanılarak tasarlanmakta ve dizgeler arası uyum arttırılmaktadır.

Tüm şartnamelerde, ses ve görüntü işaretlerinin kodlanması için MPEG-2 seçilmiştir. İşaretlerin kodlamasıyla birlikte, çoğullama, modülasyon, kanal kodlayıcı, şifreleme, veri iletimi, etkileşimli servisler gibi pek çok nokta standartlarla belirlenmiştir.

4. DVB-T – SAYISAL KARASAL YAYIN

DVB-T için en karışık DVB iletim dizgesi denilebilir. Dizgenin anahtar özelliği, sayısal işaretlerin iletiminde COFDM'in kullanılmasıdır. Standart oluşturulurken, ilgili şartname'de özellikle iletimin ne şekilde olacağı ve iletilen işaretin nasıl oluşturulması gerektiği üzerinde durulmuş, iletilen servis sayısı ve servislerin kalitesi konusunda bir sınırlama, ölçü getirilmemiştir [1].

4.1 Karasal Kanalın Sahip Olduğu Karakteristikler

Sayısal televizyon için bir modülasyon yöntemi seçiminde ilgili kanal karakteristikleri büyük bir öneme sahiptir. DVB-S (EN 300 421) ve DVB-C (EN 300 429) için, ilgili ortamlarda en iyi başarımı sağlayan ve alıcı karmaşıklığını azaltan modülasyon yöntemleri (DVB-S için tek taşıyıcılı QPSK modülasyonu, DVB-C için tek taşıyıcılı QAM modülasyonu) belirlenmiştir.

Karasal sayısal yayın için modülasyon dizgesi belirlenmesi aşamasında, karasal kanalın sahip olduğu, uydu ve kablo kanallarından oldukça farklı ve çetin olan ve aşağıda listelenen, özel karakteristiklerin dikkate alınması gerekmekteydi :

- Yeryüzü yapısından ve binalardan kaynaklanan şiddetli çoklu yol (multipath) etkisinden dolayı kanal yapısı oldukça bozulmakta ve işaret yansımaları meydana gelmektedir.
- Dizge, mevcut VHF ve UHF bandında hizmet verecek sayısal televizyon servisleri için tasarlandığından mevcut analog PAL/SECAM/NTSC yayınlarından kaynaklanan yüksek seviyeli kanallar arası girişim sorunlarının meydana gelmesi olasılığı yüksektir.

Bahsedilen bu bozucu etkenlerin etkisini azaltmak amacıyla, dizgenin geliştirilmesinde, bir çok taşıyıcılı modülasyon yöntemi olan ve yukarıda sayılan şartlara karşı oldukça dirençli olan COFDM seçilmiştir. Güçlü bir iç kodlama ve etkili bir kanal yanıtı tahmin etme yöntemiyle beraber COFDM, 0 dB'lik yansımalara bile dayanabilmektedir.

Uydu ve kablo temel dizgeleriyle olan uyumu arttırmak amacıyla, DVB-T standartında önerilen dış kodlama ve serpiştirme DVB-S ve DVB-C'dekiyle, iç kodlama ise DVB-S'tekiyle aynı tutulmuştur.

DVB-T şartnamesi 1995 yılının aralık ayında tamamlanmış ve 1996 yılının nisan ayında Avrupa Haberleşme Standartları Enstitüsü tarafından onaylanmıştır.

4.2 DVB-T Temel Dizge Yapısı

Temel bant televizyon işaretlerini, karasal kanal karakteristiklerine uygun hale getirmede kullanılabilecek, DVB tarafından önerilen dizgenin blok diyagramı şekil 4.1' de verilmiştir.

Bu dizgede gerçekleştirilen işlevler şu şekilde sıralanabilir :

- İletilen çoğullanmış verinin uyumlaştırılması ve enerji dağılımı
- Dış kodlama
- Dış serpiştirme (Outer interleaving)
- İç kodlama
- İç serpiştirme (Inner interleaving)
- Eşleştirme ve modülasyon
- Dik Frekans Bölmeli Çoğullama (COFDM) iletimi

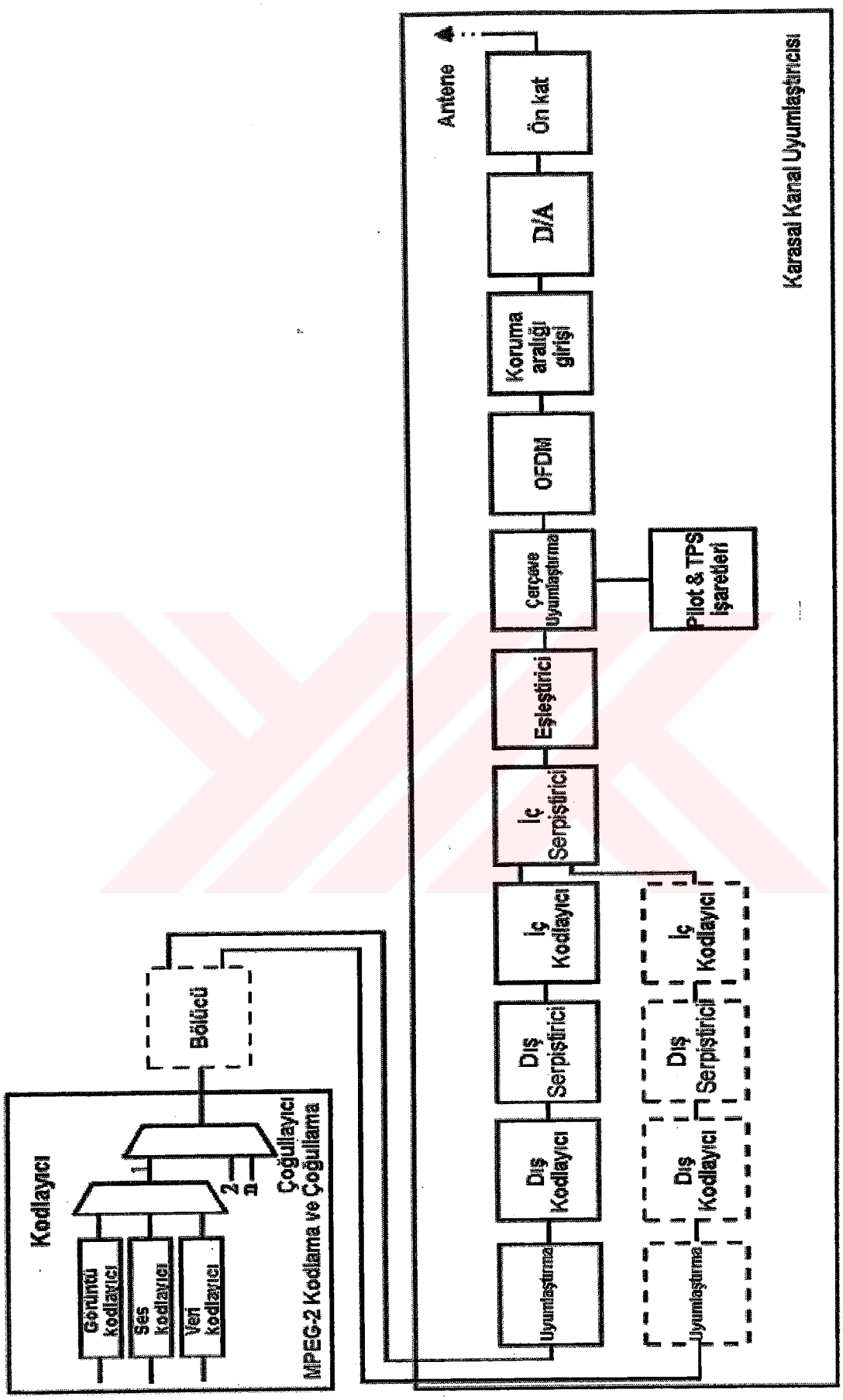
Dizge, COFDM iletiminde kullanılacak olan taşıyıcı sayısı kullanıcı (operatör) tarafından seçilebilecek şekilde geliştirilmiştir. Bu amaçla, sayısal karasal yayın için iki iletim modu önerilmiştir :

- **2K modu :** Tek vericinin yer aldığı uygulamalarda ve sınırlı verici gücünün söz konusu olduğu küçük tek frekanslı şebekelerde kullanmak için uygundur. İletimde 1705 adet taşıyıcı kullanılmaktadır.
- **8K modu :** İletimde 6817 adet taşıyıcı kullanılmaktadır. Taşıyıcı aralığı küçük olduğu için $224\,\mu sn$'ye kadar varan koruma aralıklarını desteklemektedir. Bu özelliğinden ötürü, tek vericinin yer aldığı uygulamalarda, küçük ve özellikle büyük tek frekanslı şebekelerde kullanmak için uygundur.

Dizge, farklı modülasyon şekillerinin (QPSK, 16-QAM, 64-QAM) ve iç kod oranlarının kullanılmasına izin vermektedir. Bu özellik sayesinde, frekans planlamasında esneklik sağlanmış ve değişken kanal şartlarında mümkün olabilecek en fazla bit iletim hızına ulaşılması hedeflenmiştir.

Dizge ayrıca, iki seviyeli hiyerarşik kanal kodlama ve modülasyonuna izin vermektedir. Hiyerarşik iletim durumunda şekil 4.1'de gösterilen dizge blok diyagramında, kesikli çizgi ile çizilmiş bloklar da dizgeye dahil edilmektedir.

Hiyerarşik kodlama ve modülasyon söz konusu iken, önerilen dizgede yer alan bölücü, gelen iletim bit dizisini (transport stream) öncelikli olan ve öncelikli olmayan olmak üzere iki bağımsız MPEG iletim bit dizisine böler. Bu şekilde, aynı içeriğe fakat farklı iletim kodlama parametrelerine sahip bit dizileri aynı anda iletilir ve duruma göre alıcı tarafında hangi bit dizisinin çözüleceğine karar verilir. Alıcı yapısını basit tutmak için hiyerarşik kaynak kodlaması kullanımı önerilmemiştir. Alıcı karmaşıklığına getirilecek tek eklenti dış ters-serpiştirici, iç kod çözücü ve dış kod çözücü bloklarıdır. Bu yöntemin, alıcı tarafındaki, dezavantajı, bit dizileri arasında geçiş sırasında kodlanmış görüntü ve ses verisinin çözülmesindeki sürekliliğinin bozulması ve bir duraksamaya gereksinim duyulmasıdır. Hiyerarşik kodlama, 16-QAM ve 64-QAM kullanılması durumunda uygulanabilir.

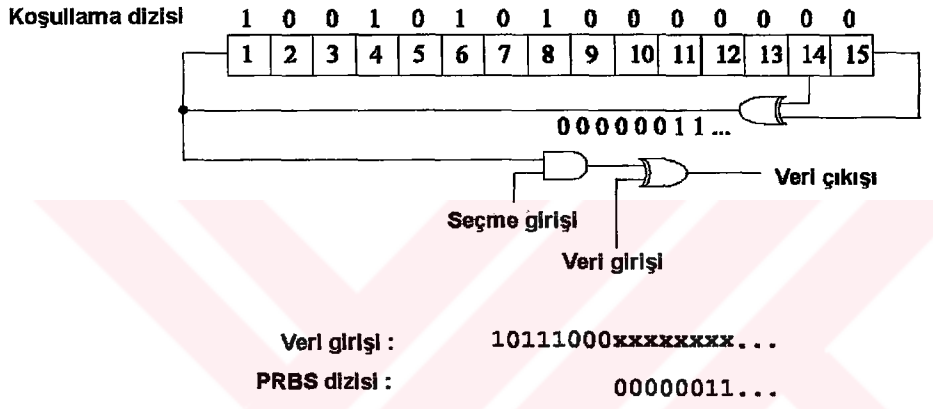


Şekil 4.1 DVB-T temel dizge yapısı

4.2.1 Kanal kodlama ve modülasyon

4.2.1.1 Veri uyumlaştırma ve enerji dağılımı

Bit dizisi, MPEG-2 çoğullayıcının çıkışından itibaren sabit uzunlukta paketler halinde düzenlenmektedir. İletilen MPEG-2 çoğullanmış veri paketinin toplam uzunluğu 188 sekizlidir. Buna 1 sekizlik eş zamanlama verisi de (0x47) dahildir. Yeterli sayıda ikili geçişin (0'dan 1'e ve 1'den 0'a) sağlanabilmesi, dolayısıyla taşıyıcı frekansı etrafında güç yoğunluğu konsantrasyonunun önlenmesi amacıyla girişe gelen çoğullanmış MPEG-2 verisine ait bitlerin yerleri, şekil 4.2'de önerilen yöntemle karıştırılmaktadır.



Şekil 4.2 Karıştırıcı–çözücü şematik diyagramı

PRBS (Pseudo Random Binary Sequence) üreteç polinomu şu şekildedir :

$$1 + x^{14} + x^{15} \tag{4.1}$$

Her sekiz iletim paketinde bir, PRBS kütüklerine 100101010000000 dizisi yüklenmektedir. Birleştirici için koşullama işareti üretmek amacıyla, sekiz iletim paketinden oluşan grubun ilk paketinin eş zamanlama sekizlisi, 0x47'den 0xB8'e evrilir. İletilen çoğullanmış veriye uygulanan bu işleme uyumlaştırma adı verilmektedir.

PRBS üretecinin çıkışındaki ilk bit, evrilmiş MPEG-2 eş zamanlama sekizlisini izleyen sekizliye ait ilk bitle işleme sokulur. Gruptaki diğer yedi iletim paketinin eş zamanlama sekizlileri boyunca, dizgedeki diğer eş zamanlama işlevlerine yardım etmek amacıyla, PRBS üretilmesine devam edilir fakat üretecin çıkışı etkisizleştirildiğinden, söz konusu eş zamanlama sekizlileri rastgeleleştirilmeden

(unrandomized) kalır. Bu şekilde, bir PRBS periyodunda 1503 ((187 + 188*7) – ilk sekizli yok) adet sekizli yer almaktadır.

4.2.2 Dış kodlama ve dış serpiştirme

Dış kodlama ve serpiştirme giriş paketlerine uygulanmaktadır.

Giriş uyumlaştırma bloğundan geçmiş olan, 188 sekizlikten oluşan her pakete, orijinal sistematik Reed-Solomon RS(255,239,t=8) kodundan türetilen, Reed-Solomon RS(204,188,t=8) kısaltılmış kodu uygulanır. Bu kodlama, aynı zamanda, evrilmiş veya evrilmemiş, tüm eş zamanlama sekizlilerine de uygulanır.

Reed-Solomon kodu, 204 sekizlik uzunluğa, 188 sekizlik boyuta sahiptir ve oluşturulan 204 sekizlinin içindeki rastgele 8 hatalı sekizlinin düzeltilmesini sağlar. Kod üreteç polinomu şu şekilde ifade edilebilir :

$$g(x) = (x + \lambda^0)(x + \lambda^1)(x + \lambda^2)\ldots(x + \lambda^{15}) \qquad (4.2)$$

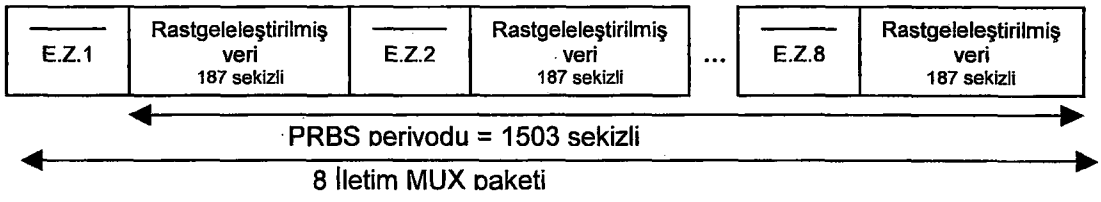
Burada $\lambda = (02)_{16}$ olmaktadır

Alan (field) üreteç polinomu ise aşağıdaki gibi ifade edilebilir :

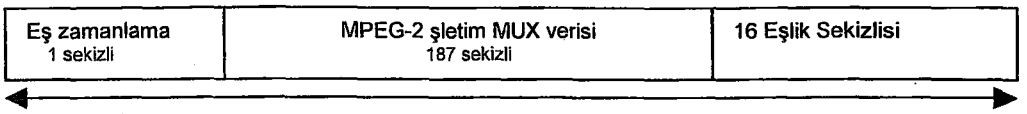
$$p(x) = x^8 + x^4 + x^3 + x^2 + 1 \qquad (4.3)$$

Kısaltılmış Reed-Solomon kodu, RS(255,239,t=8) kodlayıcısının girişine gelen sekizlilerden hemen önce, hepsi sıfır olan 51 adet sekizli eklenerek oluşturulabilir. Kodlama işleminden sonra fazladan eklenen sekizliler silinerek 204 sekizliden oluşan RS kod kelimesi elde edilebilir.

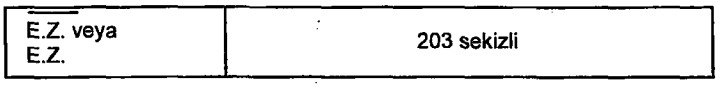
Şekil 4.3 MPEG-2 iletim paketi



Şekil 4.4 Rastgeleleştirilmiş iletim paketleri



Şekil 4.5 Reed-Solomon RS(204,188,8) hata korumalı paketler



Şekil 4.6 Dış serpiştirme işleminden sonraki veri yapısı, serpiştirme derinliği I=12 sekizli

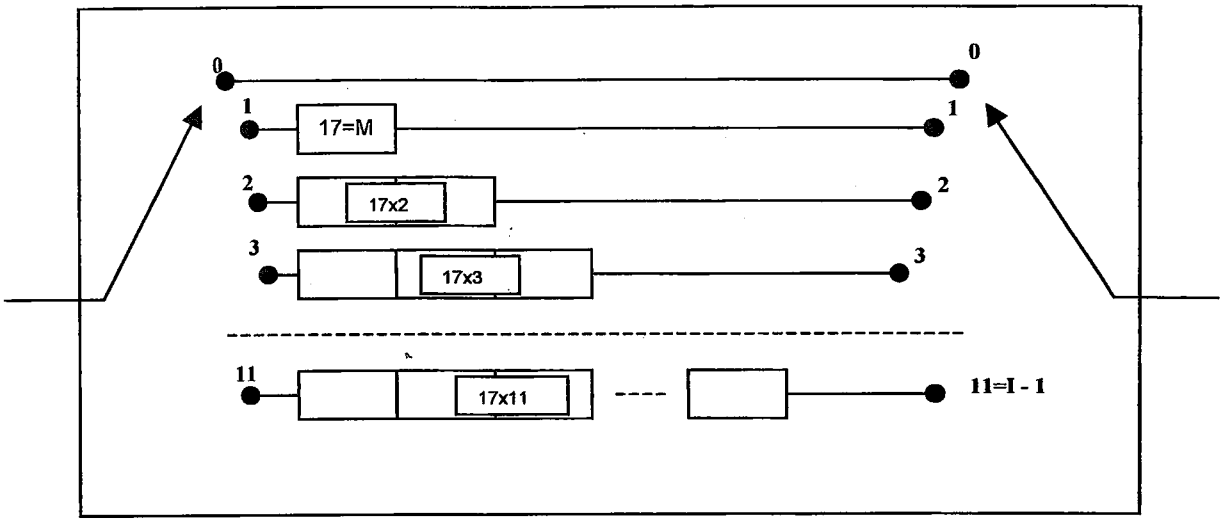
Bu aşamadan sonra hata korumalı paketlere, $I = 12$'lik bir derinliğe sahip konvolüsyonel serpiştirme işlemi uygulanır. Bu şekilde, bitişik olarak iletilen simgeler birbirinden ayrılmış olur. Hata patlaması halinde hatanın farklı simgeler üzerine yayılması sağlanıp dizgenin başarımının arttırılması sağlanmıştır. Elde edilen veri yapısı şekil 4.6'da gösterilmiştir.

Yapısı şekil 4.7'de gösterilen serpiştirici, $I = 12$ daldan oluşacak şekilde tasarlanabilir. Bu dallar giriş bit dizisine bir anahtar vasıtasıyla bağlı olmaktadır. Blok girişine gelen sekizliler, her defasında başka bir dala doğru yönlendirilmektedir. j numaralı dal, jxM hücre derinliğine sahip İGİÇ (ilk giren ilk çıkar – FIFO) kaydırmalı kütük olarak tasarlanabilir (M = 17 = N/I, N=204). Kaydırmalı kütüklerin (shift register) her bir hücresi 1 sekizli içermeli ve serpiştirici bloğunda yer alan giriş ve çıkış anahtarları eş zamanlı hareket etmelidir.

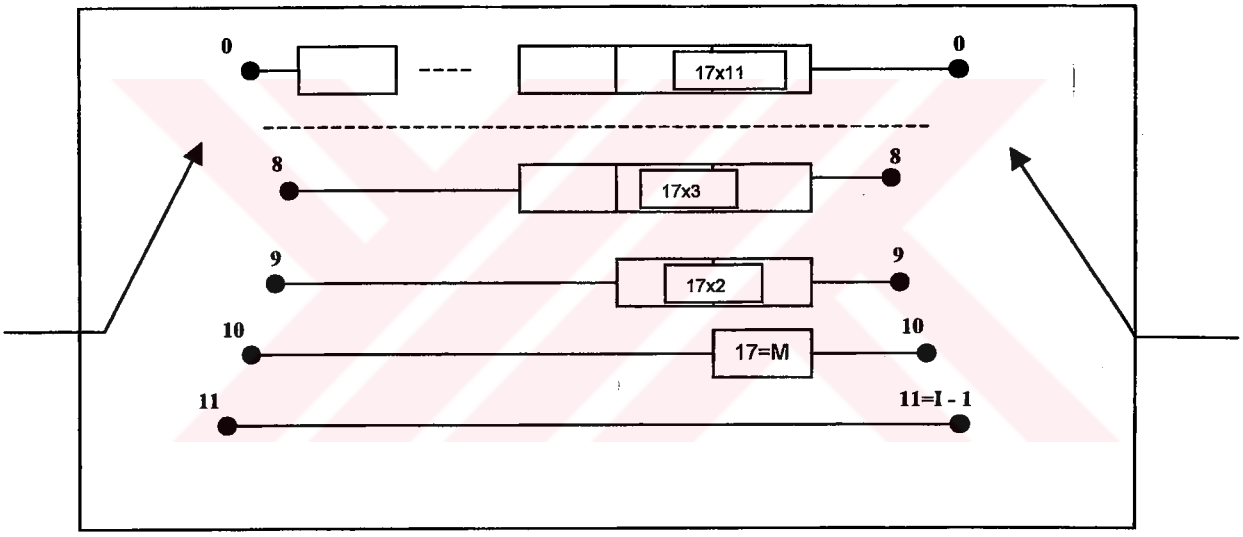
Eş zamanlama amacıyla, $E.Z.$ ve $\overline{E.Z.}$ sekizlileri daima 0 numaralı dala yönlendirilmelidir (0 numaralı dal 0 gecikmeyi işaret etmektedir).

Yapısı şekil 4.8'de gösterilen ters-serpiştiricinin (deinterleaver) çalışma ilkesi serpiştiricininkiyle aynıdır. Tek farkı dal indislerinin tersine çevrilmiş olmasıdır, yani 0 numaralı dal en uzun gecikmeye sahip olmaktadır. Ters-serpiştirici eş zamanlaması, alınan ilk eş zamanlama sekizlisinin 0 numaralı dala yönlendirilmesiyle sağlanır.

Bu şekilde, işaretin uzun süreler boyunca bozulması durumunda, bu etkiyi azaltan, hatayı rastgeleleştiren, zaman serpiştirme işlemi gerçekleştirilmiş olur.



Şekil 4.7 Dış serpiştirici yapısı

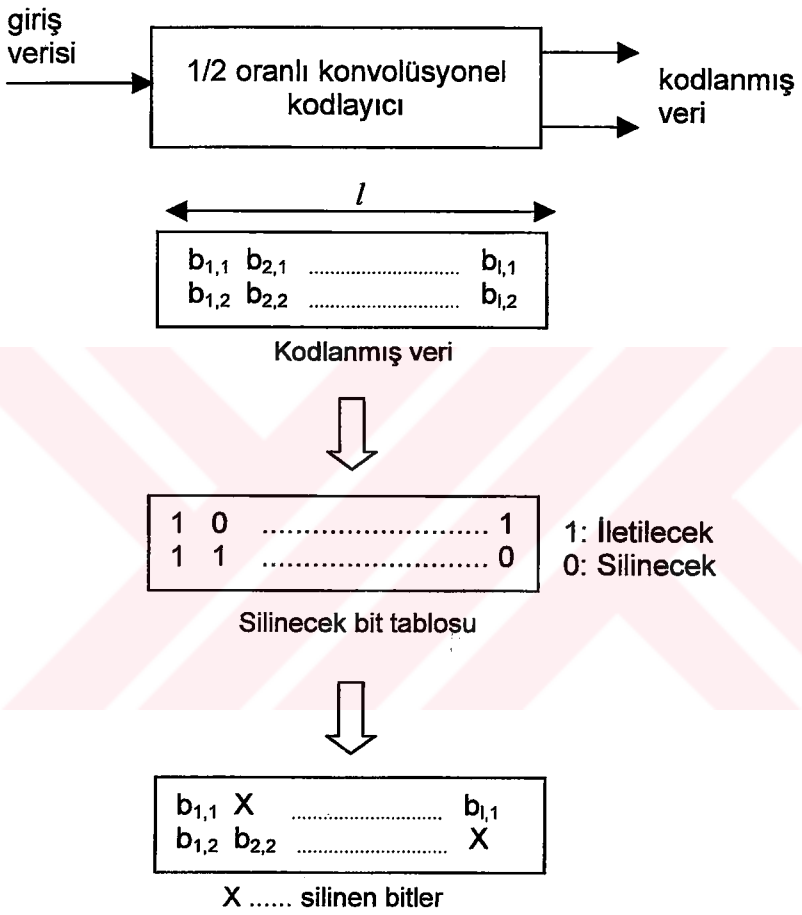


Şekil 4.8 Dış ters-serpiştirici yapısı

4.2.3 İç kodlama

Dizgede, iç kodlamada, 1/2 oranına ve 64 adet duruma (state) sahip ana konvolüsyonel koddan türeyen bir dizi delinmiş (punctured) konvolüsyonel kod kullanılmaktadır. Delinmiş konvolüsyonel kodlar, düşük hızlı konvolüsyonel kodların giriş bloğunda yer alan bir kısım bitlerin periyodik olarak silinmesiyle elde edilmektedir. Bu şekilde yüksek oranlı Viterbi kod çözücü yapısı basitleştirilmiş olur. Bununla ilgili blok diyagram şekil 4.11'de verilmiştir.

Şekil 4.9'da 1/2 oranına sahip bir koddan yüksek oranlı kodların elde edilmesi gösterilmektedir. l bloktan ($2l$ bit) özellikle seçilmiş m bit, periyodik olarak silinir. Silme işleminde, silinecek bitlerin konumlarını belirten tablodan yararlanılır. m değeri olarak $(l-1)$ seçildiğinde $(n-1)/n$ oranlı bir kod elde edilmiş olur. Bu şekilde oluşturulmuş kod kelimelerinin çözülmesi de benzer bir yoldan yapılır. Kodlamada silinen bitlerin yerlerine anlamsız bitler yerleştirilir. Bu işlemden sonra kod oranı yeniden 1/2 yapılmış olur ve kod çözme işlemi bu orana göre yapılır.



Şekil 4.9 1/2 konvolüsyonel koddan delinmiş kodun üretilmesi

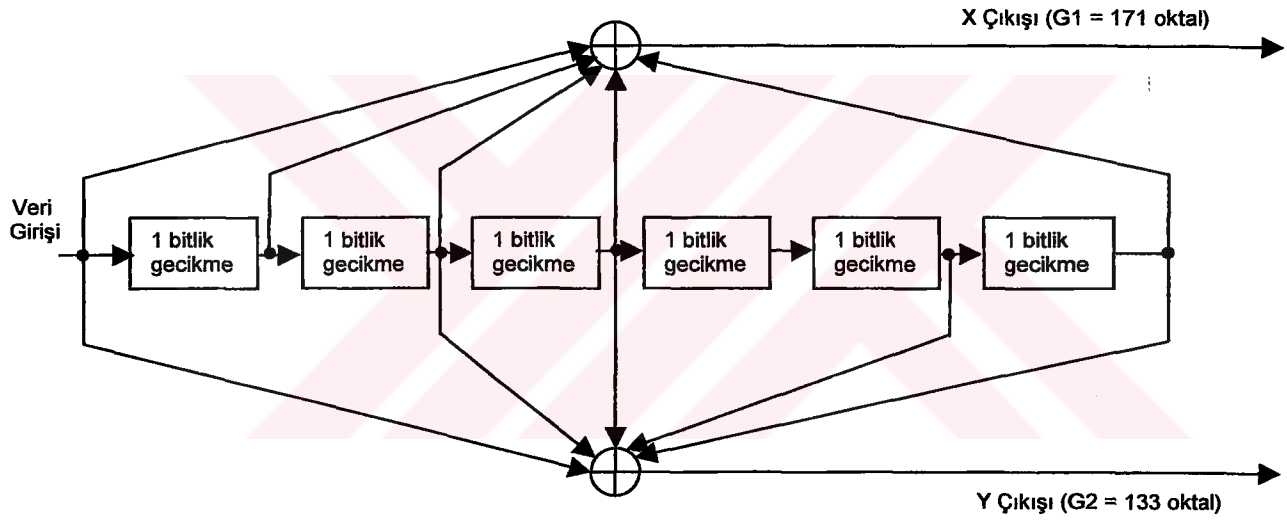
Böyle bir yapı kullanarak elde edilebilecek kod oranları ve bu oranları elde etmek için kullanılabilecek delme desenleri tablo 4.1'de verilmiştir.

Ana kod bloğunun yapısı şekil 4.10'da gösterilmiştir. Burada, X çıkışı için üreteç polinomu G1=171_{oktal} ve Y çıkışı içinse G2=133_{oktal} 'dır.

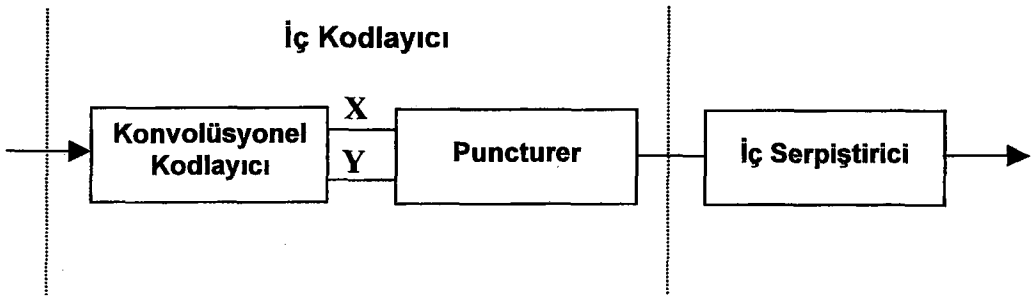
Tablo 4.1 Gerçekleştirilebilecek kod oranları için, paralelden seriye çevirme işleminden sonraki delme deseni ve iletilen dizi değerleri

Kod Oranı (r)	Delme Deseni	İletilen Dizi
1/2	X: 1 Y: 1	X_1Y_1
2/3	X: 1 0 Y: 1 1	$X_1Y_1Y_2$
3/4	X: 1 0 1 Y: 1 1 0	$X_1Y_1Y_2X_3$
5/6	X: 1 0 1 0 1 Y: 1 1 0 1 0	$X_1Y_1Y_2X_3Y_4X_5$
7/8	X: 1 0 0 0 1 0 1 Y: 1 1 1 1 0 1 0	$X_1Y_1Y_2X_3Y_4X_5Y_6X_7$

Konvolüsyonel kodlama bloğunun çıkışındaki ilk bit X_1'e karşı düşmektedir.



Şekil 4.10 Ana konvolüsyonel kod bloğu



Şekil 4.11 İç kodlama ve serpiştirme

4.2.4 İç serpiştirme

Dizgede, önce bitlere ve daha sonra da simgelere olmak üzere iki tip serpiştirme işlemi uygulanmaktadır.

4.2.4.1 Bit düzeyinde serpiştirme

Giriş bit dizisi veya dizileri (hiyerarşik iletim modu kullanılıyorsa), v adet alt bit dizisine ayrıştırılır. Alt bit dizisi sayısı seçilen modülasyon yöntemine bağlıdır. QPSK için $v=2$, 16-QAM için $v=4$ ve 64-QAM için $v=6$'dır. Hiyerarşik olmayan iletim modunda mevcut tek giriş bit dizisi, v adet alt bit dizisine ayrıştırılmaktadır. Hiyerarşik iletim şeklinde ise yüksek öncelikli bit dizisi 2 adet alt diziye, düşük öncelikli bit dizisiyse $(v-2)$ adet alt diziye ayrıştırılmaktadır.

Burada ayrıştırma (demultiplexing), giriş bitlerini (x_{d_i}) çıkış bitlerine (b_{e,d_0}) dönüştürme işlemi olarak tanımlanmaktadır.

Hiyerarşik olmayan iletim modunda :

$$x_{di} = b_{[di(\mathrm{mod})v](div)(v/2)+2[di(\mathrm{mod})(v/2)],di(div)v} \tag{4.4}$$

Hiyerarşik iletim modunda :

$$\begin{aligned} x'_{di} &= b_{di(\mathrm{mod})2,di(div)2} \\ x''_{di} &= b_{[di(\mathrm{mod})(v-2)](div)((v-2)/2)+2[di(\mathrm{mod})((v-2)/2]+2,di(div)(v-2)} \end{aligned} \tag{4.5}$$

Burada :

x_{di} hiyerarşik olmayan modda ayrıştırıcı girişini,

x'_{di} yüksek öncelikli ayrıştırıcı girişini,

x''_{di} hiyerarşik modda düşük öncelikli girişi,

d_i giriş bit sayısını,

b_{e,d_0} ayrıştırıcı çıkışını,

e ayrıştırılan bit dizisi sayısını $(0 \le e < v)$,

d_0 ayrıştırıcı çıkışındaki bit dizisindeki bit numarasını,

mod mutlak alma işlemini,

div bölme işlemini temsil etmektedir.

Ayrıştırma işlemi sonucunda, değişik modülasyon yöntemleri için aşağıdaki eşleştirmeler (demapping) oluşmaktadır:

QPSK:

$x_0 \longrightarrow b_{0,0}$

$x_1 \longrightarrow b_{1,0}$

16-QAM hiyerarşik olmayan iletimde:

$x_0 \longrightarrow b_{0,0}$

$x_1 \longrightarrow b_{2,0}$

$x_2 \longrightarrow b_{1,0}$

$x_3 \longrightarrow b_{3,0}$

16-QAM hiyerarşik iletimde:

$x_0' \longrightarrow b_{0,0}$

$x_1' \longrightarrow b_{1,0}$

$x_0'' \longrightarrow b_{2,0}$

$x_1'' \longrightarrow b_{3,0}$

64-QAM hiyerarşik olmayan iletimde:

$x_0 \longrightarrow b_{0,0}$

$x_1 \longrightarrow b_{2,0}$

$x_2 \longrightarrow b_{4,0}$

$x_3 \longrightarrow b_{1,0}$

$x_4 \longrightarrow b_{3,0}$

$x_5 \longrightarrow b_{5,0}$

64-QAM hiyerarşik iletimde:

$x_0' \longrightarrow b_{0,0}$

$x_1' \longrightarrow b_{1,0}$

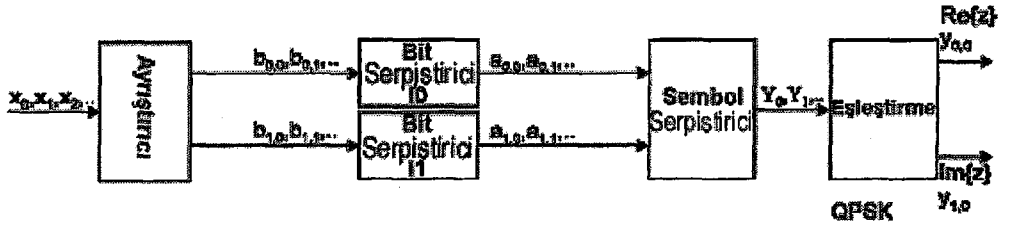
$x_0'' \longrightarrow b_{2,0}$

$x_1'' \longrightarrow b_{3,0}$

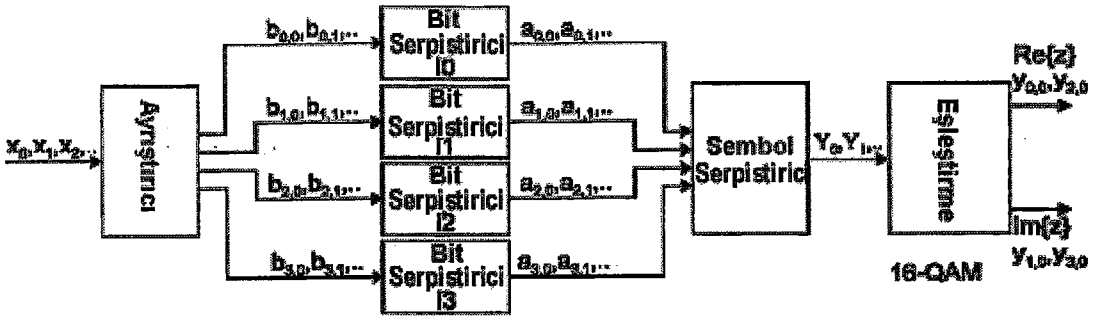
$x_2'' \longrightarrow b_{3,0}$

$x_3'' \longrightarrow b_{5,0}$

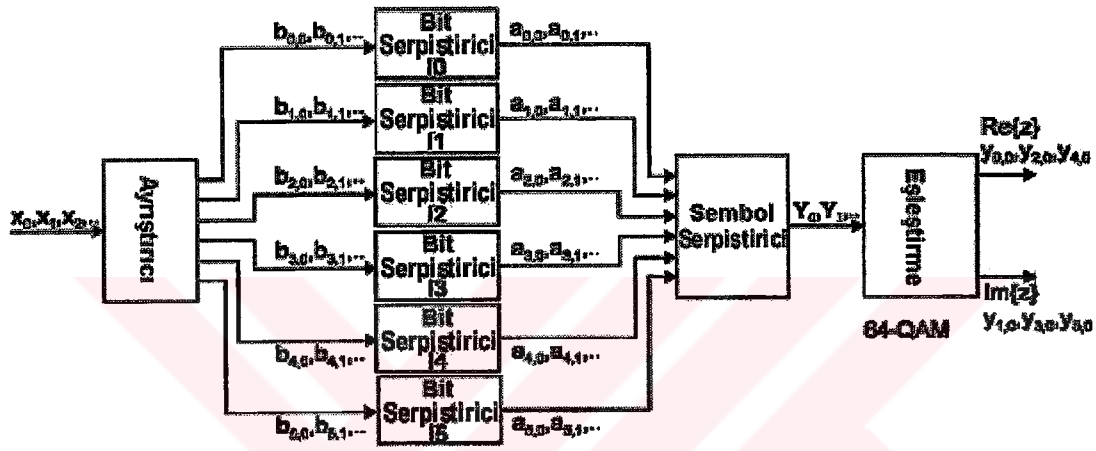
Eşleştirmenin hiyerarşik iletim modunda nasıl gerçekleştirildiği şekil 4.12, şekil 4.13 ve şekil 4.14 'te, hiyerarşik olmayan iletim modlarında nasıl gerçekleştirildiği ise şekil 4.15 ve şekil 4.16'da gösterilmiştir.



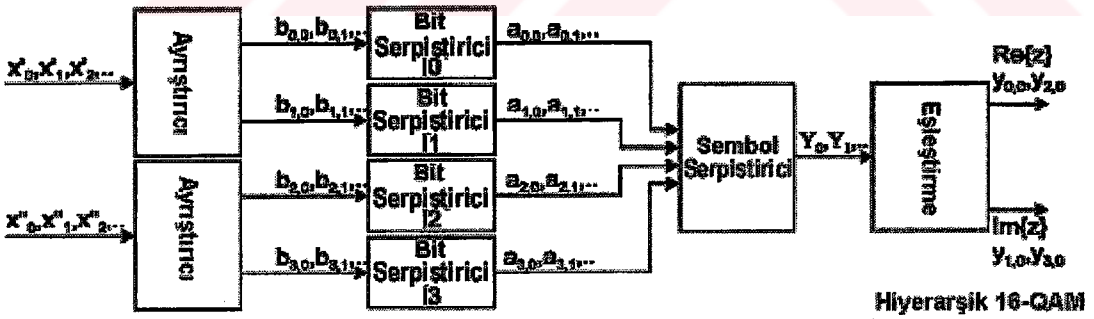
Şekil 4.12 Hiyerarşik olmayan QPSK iletim modunda giriş bitlerinin çıkış modülasyon simgelerine eşleştirilmesi



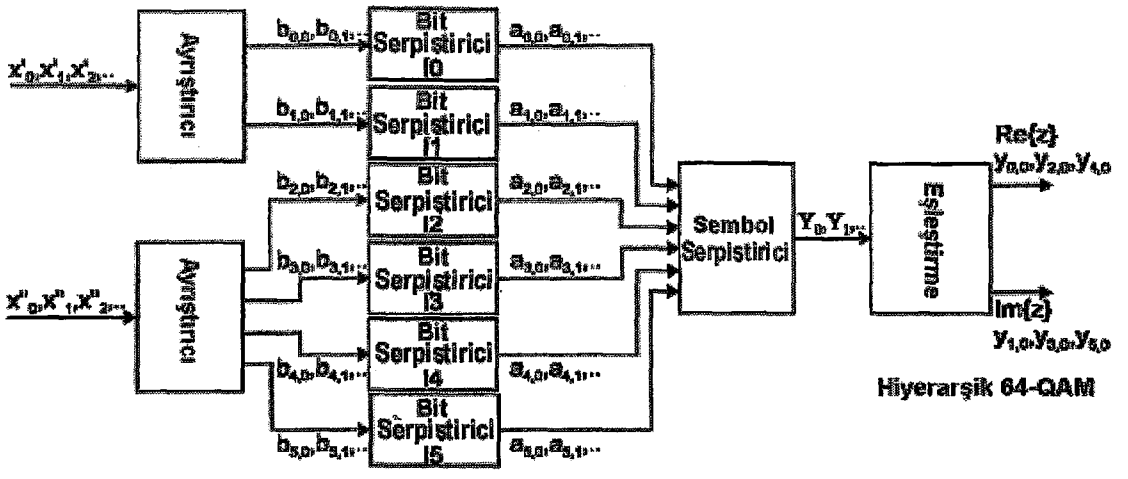
Şekil 4.13 Hiyerarşik olmayan 16-QAM iletim modunda giriş bitlerinin çıkış modülasyon simgelerine eşleştirilmesi



Şekil 4.14 Hiyerarşik olmayan 64-QAM iletim modunda giriş bitlerinin çıkış modülasyon simgelerine eşleştirilmesi



Şekil 4.15 Hiyerarşik 16-QAM iletim modlarında, giriş bitlerinin çıkış modülasyon simgelerine eşleştirilmesi



Şekil 4.16 Hiyerarşik 64-QAM iletim modlarında, giriş bitlerinin çıkış modülasyon simgelerine eşleştirilmesi

Her bit alt dizisi farklı bir bit serpiştirici bloğu tarafından işlenmektedir. Bu yüzden, v değişkenine bağlı olarak, serpiştirici sayısı altıya kadar çıkabilmektedir. Bit serpiştiriciler I0,I1,...,I5 olarak isimlendirilmişlerdir. I0 ve I1 QPSK kullanıldığında, I0,...,I3 16-QAM kullanıldığında, I0,I1,...,I5 ise 64-QAM kullanıldığında devreye girmektedir.

Dizgede yer alan serpiştirici sayısı bit alt dizisi sayısına eşit olmalıdır. Ayrıştırıcıdan çıkan her alt dizi farklı bir bit serpiştiricisi tarafından işlenir.

Her serpiştiricinin blok uzunluğu birbirine eşit ve 126 bit uzunluğundadır. Serpiştirme yöntemi her serpiştirici için farklıdır. Blok uzunluğu 126 bit olduğundan, blok serpiştirme işlemi, her COFDM simgesi için, 2K modunda oniki kez, 8K modunda kırksekiz defa tekrarlanır.

Serpiştiricilere giren bit vektörleri şu şekilde tanımlanmaktadır :

$$B(e) = \left(b_{e,0}, b_{e,1}, b_{e,2}, \ldots, b_{e,125}\right) \quad , e = 0,1,2,\ldots,(v-1) \tag{4.6}$$

Serpiştirici çıkış vektörleri ise şu şekilde tanımlanmaktadır :

$$A(e) = \left(a_{e,0}, a_{e,1}, a_{e,2}, \ldots, a_{e,125}\right) \tag{4.7}$$

$$a_{e,w} = b_{e,He(w)} \quad , w = 0,1,2,\ldots,125 \tag{4.8}$$

$H_e(w)$, her bir serpiştirici için farklı olarak tanımlanmış bir permütasyon işlevidir :

$$
\begin{aligned}
I0: &\quad H_0(w) = w \\
I1: &\quad H_1(w) = (w+63)\bmod 126 \\
I2: &\quad H_2(w) = (w+105)\bmod 126 \\
I3: &\quad H_3(w) = (w+42)\bmod 126 \\
I4: &\quad H_4(w) = (w+21)\bmod 126 \\
I5: &\quad H_5(w) = (w+84)\bmod 126
\end{aligned}
\tag{4.9}
$$

Her adımda, serpiştirici çıkışlarından birer bit alınarak v bitlik simgeler oluşturulmaktadır.

Bit serpiştirici bloğunun çıkışındaki simge şu şekilde ifade edilebilir :

$$y'_w = (a_{0,w}, a_{1,w}, \ldots, a_{v-1,w}) \tag{4.10}$$

Dış serpiştirmede olduğu gibi, bu işlem de zaman uzamında etkili olmaktadır.

4.2.4.2 Simge düzeyinde serpiştirme

Simge serpiştiricinin amacı, v bitten oluşan dizileri 2K modunda 1512 veya 8k modunda 6048 taşıyıcıya eşleştirmektir. Simge serpiştirici, 1512 veya 6048 adet veri simgesi üzerinde işlem yapmaktadır.

2K modunda, herbiri 126 veri bloğundan (v bitlik) meydana gelen 12 grup, blok blok işlenerek $Y' = (y'_0, y'_1, y'_2, \ldots, y'_{1511})$ biçiminde bir vektör haline getirilir. Benzer şekilde, 8k modunda, herbiri 126 veri bloğundan (v bitlik) meydana gelen 48 grup, blok blok işlenerek $Y' = (y'_0, y'_1, y'_2, \ldots, y'_{6047})$ biçiminde bir vektör haline getirilir.

Serpiştirilmiş $Y = (y_0, y_1, y_2, \ldots, y_{Nmaks-1})$ vektörü şu şekilde tanımlanmaktadır :

$y_{H(q)} = y'_q$, çift numaralı COFDM simgelerinde (q=0,1,...,N_{maks}-1)

$y_q = y'_{H(q)}$, tek numaralı COFDM simgelerinde (q=0,1,...,N_{maks}-1)

2K modunda N_{maks} = 1512 ve 8K modunda N_{maks} = 6048 olmaktadır.

COFDM simgesinin, içinde bulunduğu COFDM çerçevesindeki sırasını belirleyen simge indisi hakkında, COFDM çerçeve yapısının anlatıldığı bölümde bilgi verilecektir.

$H(q)$ permütasyon işlevi aşağıdaki algoritmayla tanımlanmaktadır :

$$
\begin{aligned}
&q = 0; \\
&for(\imath = 0; \imath < M_{maks}; i = i + 1) \\
&\{H(q) = (\imath \bmod 2).2^{N_r - 1} + \sum_{j=0}^{N_r - 2} R_i(j).2^j \\
&if(H(q) < N_{maks}) \quad q = q + 1;\}
\end{aligned}
\tag{4.11}
$$

R_i', $(N_r - 1)$ adet bitten oluşan ikili dizidir. Burada $N_r = \log_2 M_{maks}$ ve 2K modunda $M_{maks} = 2048$, 8K modunda $M_{maks} = 8192$ olmaktadır. R_i''nin aldığı değerler ,

i=0,1 : $R_i'[N_r - 2, N_r - 3, \ldots, 1, 0] = 0, 0, \ldots, 0, 0$ (4.12)

i=2 : $R_i'[N_r - 2, N_r - 3, \ldots, 1, 0] = 0, 0, \ldots, 0, 1$ (4.13)

2<i< M_{maks} : { $R_i'[N_r - 3, N_r - 4, \ldots, 1, 0] = R_{i-1}'[N_r - 2, N_r - 3, \ldots, 2, 1]$

2K modunda: $R_i'[9] = R_{i-1}'[0] \oplus R_{i-1}'[3]$ (4.14)

8K modunda: $R_i'[11] = R_{i-1}'[0] \oplus R_{i-1}'[1] \oplus R_{i-1}'[4] \oplus R_{i-1}'[6]$ }

şeklinde olmaktadır.

Tablo 4.2 ve tablo 4.3'te verilen bit permütasyonları kullanılarak R_i vektörü elde edilebilir.

Tablo 4.2 2K modunda bit permütasyonları

R_i' vektörü bit yerleşimleri	9	8	7	6	5	4	3	2	1	0
R_i vektörü bit yerleşimleri	0	7	5	1	8	2	6	9	3	4

Tablo 4.3 8K modunda bit permütasyonları

R_i' vektörü bit yerleşimleri	11	10	9	8	7	6	5	4	3	2	1	0
R_i vektörü bit yerleşimleri	5	11	3	0	10	8	6	9	2	4	1	7

y, y''ne benzer şekilde v bitten oluşmaktadır :

$$y_{q'} = \left(y_{0,q'}, y_{1,q'}, \ldots, y_{v-1,q'}\right) \tag{4.15}$$

Burada q' simge serpiştiricinin çıkışındaki simge numarasını temsil etmektedir.

Bu çıkış y değerleri verinin işaret uzayına eşleştirilmesinde kullanılır.

Simge düzeyinde serpiştirme işlemi frekans uzamında etkili olmaktadır.

4.2.5 İşaret uzayı ve eşleştirme

Dizge, iletimde COFDM'i kullanmaktadır. COFDM çerçevesindeki veri taşıyıcılarının eşleştirlmesinde, QPSK, 16-QAM, 64-QAM, düzensiz (non-uniform) 16-QAM ve düzensiz 64-QAM kullanılmaktadır. Bütün yöntemlerde Gray eşleştirilmesi kullanılmaktadır. Söz konusu tüm modülasyon yöntemleri için eşleştirmenin nasıl uygulanacağı Ek A'da yer alan şekillerde gösterilmiştir.

Bütün bu yöntemlerde $y_{u,q'}$, karmaşık modülasyon simgesi z'yi oluşturan bitleri temsil etmektedir.

Hiyerarşik olmayan iletimde, iç serpiştiricinin çıkışındaki, v bitlik bloklardan oluşan veri dizisi, z ile gösterilen karmaşık sayıya eşleştirilir. Hiyerarşik iletimde ise veri dizileri şekil 4.12, şekil 4.13 ve şekil 4.14'te gösterildiği gibi şekillendirilir ve şekil A.1, şekil A.2 ve şekil A.3'te gösterilen uygun bir eşleştirme yöntemi uygulanır.

Hiyerarşik 64-QAM'da iç serpiştirici çıkışında yer alan $y_{0,q'}$ ve $y_{1,q'}$ bitleri yüksek öncelikli bitlerdir. Bu iki bit kullanılarak eşleştirilecek işaretin, işaret uzayının hangi çeyreğinde yer aldığı belirlenir. $y_{2,q'}$, $y_{3,q'}$, $y_{4,q'}$ ve $y_{5,q'}$ bitleri ise düşük öncelikli bitlerdir. Bu dört bit kullanılarak, işaret uzayının yüksek öncelikli bitlerle belirlenmiş olan çeyreğinde yer alan 16 işaretten birisi seçilerek eşleştirme yapılır.

Hiyerarşik 16-QAM'da iç serpiştiricinin çıkışında yer alan $y_{0,q'}$ ve $y_{1,q'}$ bitleri yüksek öncelikli bitlerdir. $y_{2,q'}$ ve $y_{3,q'}$ bitleri ise düşük öncelikli bitlerdir. Eşleştirme mantığı hiyerarşik olmayan 64-QAM'dakiyle aynıdır.

4.3 COFDM Çerçeve Yapısı

İletilen işaret çerçeveler halinde düzenlenmektedir. Her çerçeve 68 COFDM simgesi içermektedir ve toplam T_F uzunluğuna sahiptir. Her simge, 8K modunda 6817 taşıyıcı, 2K modunda ise 1705 taşıyıcıdan oluşmaktadır ve toplam T_S uzunluğuna sahiptir. COFDM simgesi, kaynağa ait gerçek bilginin taşındığı, T_U uzunluğuna sahip bir kısım ve Δ uzunluğuna sahip koruma aralığından meydana gelmektedir. COFDM'in incelendiği bölümde de anlatıldığı gibi, koruma aralığında taşınan bilgi, diğer kısımda taşınan bilgiden alınmakta ve çevrimsel devamlık sağlanmaktadır. Tablo 4.6'da gösterildiği gibi, 4 farklı koruma aralığı uzunluğu seçilebilir.

COFDM çerçevesinde yer alan simgeler 0'dan 67'ye kadar numaralandırılmıştır. Tüm simgeler veri ve referans bilgisi içermektedir.

İletilen veriye ek olarak COFDM çerçevesinde :

- Saçılmış (scattered) pilot taşıyıcıları
- Devamlı (continual) pilot taşıyıcıları
- TPS (transmission parameter signalling) taşıyıcıları

bulunmaktadır.

Saçılmış pilot taşıyıcılar kanal kestiriminde ve evre gürültüsünün takibinde, devamlı pilot taşıyıcılar frekans, zaman ve çerçeve eş zamanlamasında, TPS taşıyıcıları ise iletim modu bilgisinin iletilmesinde kullanılmaktadır.

Taşıyıcılar, $k \in [K_{\min}; K_{maks}]$ indisi ile gösterilmektedir. 2K modunda $K_{\min} = 0$ ve $K_{maks} = 1704$, 8K modunda ise $K_{\min} = 0$ ve $K_{maks} = 6816$ olmaktadır. Bitişik taşıyıcılar arasındaki mesafe $1/T_U$, $K_{\min}$ ve K_{maks} taşıyıcıları arasındaki mesafe ise $(K-1)/T_U$ olmaktadır. Dizgede koruma aralığı süresi olarak simge periyodunun 1/4'ü, 1/8'i, 1/16'sı ve 1/32'si seçilebilmektedir. 8 MHz'lik kanalda, 2K ve 8K modlarında, COFDM parametrelerinin alabileceği değerler tablo 4.4 ve tablo 4.6'da gösterilmiştir.

Tablo 4.4 8 MHz'lik kanallarda sahip olunan COFDM parametre değerleri

Parametre	8K Modu	2K Modu
Taşıyıcı sayısı K	6817	1705
K_{min} numaralı taşıyıcının değeri	0	0
K_{maks} numaralı taşıyıcının değeri	6816	1704
T_U süresi	896 μsn	224 μsn
Taşıyıcı aralığı ($1/T_U$)	1116 Hz	4464 Hz
K_{min} ve K_{maks} aralığı	7,61 MHz	7,61 MHz

Dizgenin sahip olduğu temel periyot, 8MHz'lik kanallarda $7/64\mu sn$, 7 MHz'lik kanallarda $1/8\mu sn$ ve 6 MHz'lik kanallarda $7/48\mu sn$ olmaktadır. Temel dizge periyotu, dizge saat frekansının tersi olarak düşünülebilir. Dizge saatinin değiştirilmesiyle iletim bant genişliği ve bit hızı değişmektedir.

İletilen işaret şu şekilde ifade edilebilir :

$$s(t) = \mathrm{Re}\left\{ e^{j2\pi f_c t} \sum_{m=0}^{\infty} \sum_{l=0}^{67} \sum_{k=K_{\min}}^{K_{maks}} c_{m,l,k} \times \psi_{m,l,k}(t) \right\} \tag{4.16}$$

$$\psi_{m,l,k} = \begin{cases} e^{j2\pi \frac{k'}{T_U}(t-\Delta-l\times T_S-68\times m\times T_S)} & (l+68\times m)\times T_S \le t \le (l+68\times m+1)\times T_S \\ 0 & dışında \end{cases}$$

Burada:

k taşıyıcı numarasını,

l COFDM simge numarasını,

m iletilen çerçeve numarasını,

K iletilmiş olan taşıyıcı sayısını,

T_S simge periyodunu,

T_U taşıyıcı aralığının(taşıyıcılar arasındaki aralığın) tersini,

Δ koruma aralığı süresi,

f_c RF işaretin merkez frekansını,

k' merkez frekansına göre taşıyıcı indisini, $k' = k - (K_{maks} + K_{\min})/2$,

$c_{m,l,k}$ m çerçevesinde yer alan $(l+1)$ numaralı veri simgesinin karmaşık karşılığını ifade etmektedir.

$c_{m,l,k}$, ek A'da verilen işaret uzaylarındaki z noktalarının normalleştirilmiş modülasyon değerlerini alabilir. Normalleştirme çarpanı ile $E[c \times c^*] = 1$ yapılmaktadır.

Tablo 4.5 Veri simgelerinin normalleştirme çarpanları

Modülasyon yöntemi		Normalleştirme çarpanı
QPSK	-	$c = z/\sqrt{2}$
16-QAM	$\alpha = 1$	$c = z/\sqrt{10}$
64-QAM	$\alpha = 2$	$c = z/\sqrt{20}$
	$\alpha = 4$	$c = z/\sqrt{52}$
	$\alpha = 1$	$c = z/\sqrt{42}$
	$\alpha = 2$	$c = z/\sqrt{60}$
	$\alpha = 4$	$c = z/\sqrt{108}$

Tablo 4.6 8 MHz'lik kanallarda geçerli olan süreler

Mod	8K				2K			
Koruma Aralığı (Δ/T_U)	1/4	1/8	1/16	1/32	1/4	1/8	1/16	1/32
Simge Süresi (T_U)	$8192 \times T$ $896\mu sn$				$2048 \times T$ $224\mu sn$			
Koruma Aralığı Süresi (Δ)	$2048 \times T$ $224\mu sn$	$1024 \times T$ $112\mu sn$	$512 \times T$ $56\mu sn$	$256 \times T$ $28\mu sn$	$512 \times T$ $56\mu sn$	$256 \times T$ $28\mu sn$	$128 \times T$ $14\mu sn$	$64 \times T$ $7\mu sn$
Toplam Simge Süresi $(T_S = \Delta + T_U)$	$10240 \times T$ $1120\mu sn$	$9216 \times T$ $1008\mu sn$	$8704 \times T$ $952\mu sn$	$8448 \times T$ $924\mu sn$	$2560 \times T$ $280\mu sn$	$2304 \times T$ $252\mu sn$	$2176 \times T$ $238\mu sn$	$2112 \times T$ $231\mu sn$

4.4 Referans İşaretlerin Üretilmesi

COFDM çerçevesinde yer alan çeşitli taşıyıcılar, değerleri ve konumları alıcı tarafından bilinen referans bilgilerle modüle edilmektedir. Referans bilgisini taşıyan taşıyıcılar yükseltilmiş (boosted) güç seviyesiyle iletilirler. Bunlara saçılmış veya devamlı pilot taşıyıcıları adı verilir.

Her dört COFDM simgesinde bir devamlı pilotlarla saçılmış pilotlar çakışır. Kaynak bitlerini taşıyan veri taşıyıcılarının sayısı her simge için aynıdır: 2K modunda 1512 adet, 8K modunda 6048 adet.

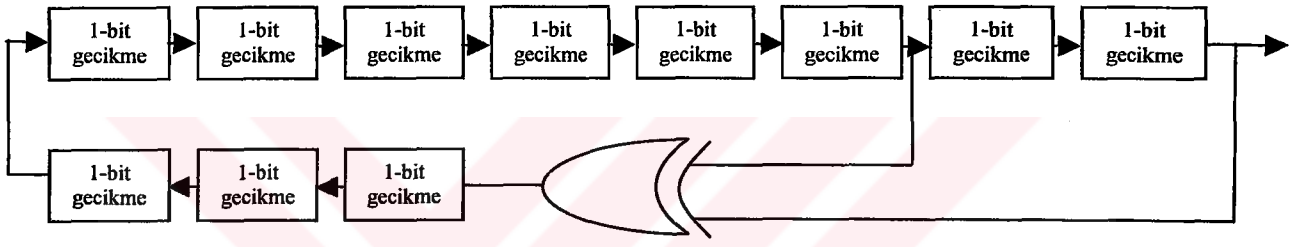
Saçılmış veya devamlı pilot taşıyıcılarda taşınan bilgi, PRBS dizisinden faydalanılarak üretilmektedir.

4.4.1 Referans dizisinin tanımı

Devamlı ve saçılmış pilotlar, taşıyıcı indisleri k'ya göre, bir w_k PRBS dizisine bağlı olarak modüle edilmektedir.

PRBS dizisi şekil 4.17'de gösterildiği biçimde üretilmektedir.

PRBS, üretilen ilk bit, ilk aktif taşıyıcıya karşı gelecek şekilde koşullanır.



Şekil 4.17 PRBS dizisinin üretilmesi

PRBS üreteç polinomu olarak $X^{11} + X^2 + 1$ kullanılmaktadır.

4.4.2 Saçılmış pilotların yerleşimi

Saçılmış pilotlarda, referans dizisinden alınan referans bilgisi iletilmektedir.

Saçılmış pilotlar daima yükseltilmiş (boosted) güç seviyesinde iletilmektedir. Kullanılan modülasyon şu şekilde ifade edilebilir:

$$\mathrm{Re}\{c_{m,l,k}\} = 4/3 \times 2(1/2 - w_k) \qquad (4.17)$$

$$\mathrm{Im}\{c_{m,l,k}\} = 0 \qquad (4.18)$$

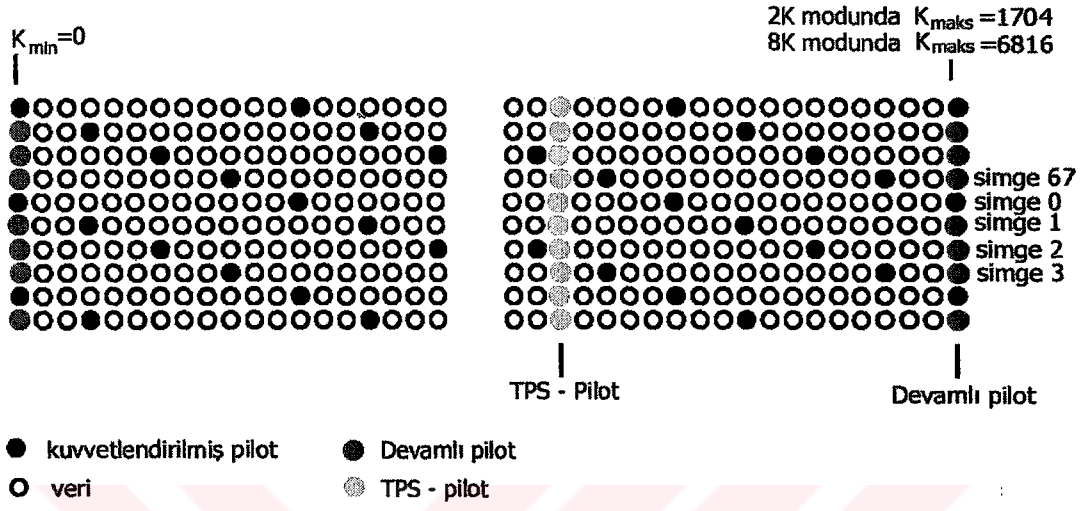
Burada m çerçeve indisi, k taşıyıcıların frekans indisi ve l simgelerin zaman indisini temsil etmektedir. w_k, PRBS dizisindeki k. elemanın değeridir.

Saçılmış pilot taşıyıcı numaralarını temsil eden k'nın alabileceği değerler aşağıdaki ifade ile bulunabilir:

$$k = K_{\min} + 3 \times (l \bmod 4) + 12p \qquad (4.19)$$

burada $p \geq 0, k \in [K_{\min}; K_{maks}]$'tır. p, k'nın alacağı değeri, $[K_{\min}; K_{maks}]$ aralığı dışına çıkarmayacak şekilde her değeri alabilir.

Pilot yerleşim düzeni şekil 4.18'de gösterilmiştir.



Şekil 4.18 DVB-T çerçeve yapısı

4.4.3 Devamlı pilot taşıyıcıların yerleşimi

Saçılmış pilotlara ek olarak 8K modunda 177, 2K modunda 45 devamlı pilot eklenir. Bu taşıyıcılar, tüm simgelerde yer almalarından dolayı "devamlı" olarak adlandırılmaktadırlar.

Devamlı pilotlar yükseltilmiş güç seviyesinde iletilmektedir. Saçılmış pilotların modülasyonunda olduğu gibi aynı referans dizisinden yararlanılmaktadır. Bu taşıyıcıların modülasyon ifadesi şu şekilde verilebilir :

$$\mathrm{Re}\{C_{m,l,k}\} = 4/3 \times 2(1/2 - w_k) \qquad (4.20)$$

$$\mathrm{Im}\{C_{m,l,k}\} = 0 \qquad (4.21)$$

Tüm veri taşıyıcıları normalize edilmiştir. Bu nedenle $E[c \times c^*] = 1$ 'dir.

2K ve 8K modlarında geçerli olan devamlı pilot taşıyıcı indislerinin değerleri tablo 4.7'de verilmektedir.

Tüm devamlı ve saçılmış taşıyıcıların yükseltilmiş güç seviyesinde iletilmelerinden dolayı $E[c \times c^*] = 16/9$ olmaktadır.

Devamlı ve saçılmış pilot taşıyıcı değerlerinin hem zaman hem de frekans uzamında ara değerlerinin bulunmasıyla kanal kestirimi ve eşitleme oldukça hızlı bir şekilde gerçekleştirilebilmektedir.

Tablo 4.7 Devamlı pilot taşıyıcıların taşıyıcı indisleri

2K Modu	8K Modu
0 48 54 87 141 156 192 201 255 279 282 333 432 450 483 525 531 618 636 714 759 765 780 804 873 888 918 939 942 969 984 1050 1101 1107 1110 1137 1140 1146 1206 1269 1323 1377 1491 1683 1704	0 48 54 87 141 156 192 201 255 279 282 333 432 450 483 525 531 618 636 714 759 765 780 804 873 888 918 939 942 969 984 1050 1101 1107 1110 1137 1140 1146 1206 1269 1323 1377 1491 1683 1704 1752 1758 1791 1845 1860 1896 1905 1959 1983 1986 2037 2136 2154 2187 2229 2235 2322 2340 2418 2463 2469 2484 2508 2577 2592 2622 2643 2646 2673 2688 2754 2805 2811 2814 2841 2844 2850 2910 2973 3027 3081 3195 3387 3408 3456 3462 3495 3549 3564 3600 3609 3663 3687 3690 3741 3840 3858 3891 3933 3939 4026 4044 4122 4167 4173 4188 4212 4281 4296 4326 4347 4350 4377 4392 4458 4509 4515 4518 4545 4548 4554 4614 4677 4731 4785 4899 5091 5112 5160 5166 5199 5253 5268 5304 5313 5367 5391 5394 5445 5544 5562 5595 5637 5643 5730 5748 5826 5871 5877 5892 5916 5985 6000 6030 6051 6054 6081 6096 6162 6213 6219 6222 6249 6252 6258 6318 6381 6435 6489 6603 6795 6816

4.5 TPS (Transmission Parameter Signalling)

TPS taşıyıcıları, iletim şekliyle ilgili, kanal kodlaması ve modülasyonu gibi, işaretleşme parametrelerinin iletilmesinde kullanılır. TPS, 2K modunda her bir COFDM simgesinde yer alan paralel 17 TPS taşıyıcı, 8K modunda ise her bir COFDM simgesinde yer alan 68 taşıyıcı üzerinden iletilmektedir. Aynı COFDM simgesinde yeralan tüm TPS taşıyıcıları aynı farksal kodlanmış bilgi bitini taşır. 2K ve 8K modlarındaki TPS taşıyıcıların indisleri tablo 4.8'de gösterilmiştir.

Tablo 4.8 TPS taşıyıcı indisleri

2K Modu	8K Modu
34 50 209 346 413 569 595 688 790 9011073 1219 1262 1286 1469 1594 1687	34 50 209 346 413 569 595 688 790 901 1073 1219 1262 1286 1469 1594 1687 1738 1754 1913 2050 2117 2273 2299 2392 2494 2605 2777 2923 2966 2990 3173 3298 3391 3442 3458 3617 3754 3821 3977 4003 4096 4198 4309 4481 4627 4670 4694 4877 5002 5095 5146 5162 5321 5458 5525 5681 5707 5800 5902 6013 6185 6331 6374 6398 6581 6706 6799

TPS taşıyıcılarında,

a) Modülasyon (QAM söz konusuysa α 'nın aldığı değer de iletilir)
b) Hiyerarşi bilgisi
c) Koruma aralığı
d) İç kod oranları
e) İletim modu
f) Çerçeve numarası (çerçevenin süper çerçeve içerisinde kaçıncı sırada yer aldığı)

bilgileri taşınmaktadır.

TPS, birbirini izleyen 68 COFDM simgesi (1 COFDM çerçevesi) üzerinde tanımlanmaktadır. 4 COFDM çerçevesi, COFDM süper çerçevesini oluşturmaktadır. TPS taşıyıcılarının iletiminde farksal modülasyon uygulanmaktadır ve koşullama her COFDM çerçevesinin ilk simgesinde yapılmaktadır. Bununla ilgili ayrıntılı bilgi TPS modülasyonunun anlatıldığı bölümde verilecektir.

Her COFDM simgesi 1 TPS biti taşımaktadır. Bir COFDM çerçevesine karşı düşen her TPS bloğu 68 bit içermektedir. Bitlerin dağılımı ve taşıdığı bilgiler şu şekilde verilmektedir :

- 1 adet koşullama biti
- 16 adet eş zamanlama biti
- 37 adet bilgi biti
- 14 adet hata koruması için kullanılan bit

37 bilgi bitinden 23'ü kullanılmaktadır. Gelecekte kullanılmak üzere ayrılmış olan 14 bitin sıfır yapılması gerekmektedir.

4.5.1 TPS iletim şekli

Bitlerin alabileceği değerler ve anlamları tablo 4.9'da verilmektedir.

Tablo 4.9 TPS ile iletilebilecek değerler

Bit numarası	Desen	Amaç/İçerik
S_0	Aşağıda verilmiştir	Koşullama
S_1- S_{16}	0011010111101110 veya 1100101000010001	Eş zamanlama kelimesi
S_{17}- S_{22}	010111	Uzunluk göstergesi
S_{23}, S_{24}	Tablo 4.10	Çerçeve numarası
S_{25}, S_{26}	Tablo 4.11	İşaret uzayı
S_{27}, S_{28}, S_{29}	Tablo 4.12	Hiyerarşi bilgisi
S_{30}, S_{31}, S_{32}	Tablo 4.13	Kod oranı (yüksek öncelikli dizi)
S_{33}, S_{34}, S_{35}	Tablo 4.14	Kod oranı (düşük öncelikli dizi)
S_{36}, S_{37}	Tablo 4.14	Koruma aralığı
S_{38}, S_{39}	Tablo 4.15	İletim modu
S_{40}- S_{53}	Hepsi 0	Gelecekte kullanılacak
S_{54}- S_{67}	BCH kodu	Hata koruması

4.5.1.1 Koşullama

S_0 , farksal 2-PSK modülasyon yönteminde kullanılan koşullama bitidir. PRBS dizisi kullanılarak şu şekilde modüle edilir :

$$\begin{aligned} \mathrm{Re}\{c_{m,l,k}\} &= 2(1/2 - w_k) \\ \mathrm{Im}\{c_{m,l,k}\} &= 0 \end{aligned} \qquad (4.22)$$

4.5.1.2 Eş zamanlama

Eş zamanlama kelimesi 1-16 numaralı bitlerden oluşmaktadır.

Süper çerçevede yer alan birinci ve üçüncü TPS bloklarının sahip olduğu eş zamanlama kelimesi 001101011101110 şeklindedir. İkinci ve dördüncü TPS blokları ise 1100101000010001 eş zamanlama kelimesine sahiptir.

4.5.1.3 TPS uzunluk bilgisi

TPS 'in ilk altı bitinde, gönderilen TPS bilgisinin kaç bit uzunlukta olduğu bilgisi taşımaktadır. Şu an için 010111 değeri taşınmaktadır.

4.5.1.4 Çerçeve numarası

Dört COFDM çerçevesi bir süper çerçeveyi oluşturmaktadır. Süper çerçeve içinde yer alan çerçeveler 0'dan 3'e kadar numaralandırılmaktadır. Bitlerin, bu numaralara göre alacağı değerler tablo 4.10'da gösterilmiştir.

Tablo 4.10 TPS'te çerçeve numarası iletimi

s_{23} ve s_{24} bitleri	Çerçeve Numarası
00	Süper çerçevedeki 1 numaralı çerçeve
01	Süper çerçevedeki 2 numaralı çerçeve
10	Süper çerçevedeki 3 numaralı çerçeve
11	Süper çerçevedeki 4 numaralı çerçeve

4.5.1.5 İşaret uzayı bilgisi

Tablo 4.11'de gösterildiği gibi, işaret uzayı bilgisi 2 bit ile ifade edilmektedir. Alıcı aynı zamanda, modülasyon yöntemini belirleyebilmek amacıyla, tablo 4.12'de verilen hiyerarşi bilgisini de almalıdır.

Tablo 4.11 TPS'te işaret uzayı bilgisi iletimi

s_{25} ve s_{26} bitleri	İşaret Uzayı Karakteristikleri
00	QPSK
01	16-QAM
10	64-QAM
11	Kullanılmıyor

4.5.1.6 Hiyerarşi bilgisi

Hiyerarşi bilgisi, iletimin hiyerarşik olup olmadığını, eğer hiyerarşikse α 'nın değerini içerir . Tablo 4.12'de bitlerin alabileceği değerler ve anlamları gösterilmiştir.

Tablo 4.12 TPS'te hiyerarşi bilgisi iletimi

s_{27}, s_{28} ve s_{29} bitleri	α
000	Hiyerarşik olmayan iletim
001	α =1
010	α =2
011	α =4
100	Kullanılmıyor
101	Kullanılmıyor
110	Kullanılmıyor
111	kullanılmıyor

4.5.1.7 Kod oranı bilgisi

Hiyerarşik olmayan kanal kodlaması ve modülasyonu kullanılıyorsa, bir tek kod oranı bilgisinin gönderilmesine ihtiyaç vardır. Bu durumda, tablo 4.13'te değerleri verilmiş, kod oranını gösteren üç biti takiben 000 bit dizisi kullanılır.

Hiyerarşik iletim söz konusuysa iki farklı modülasyona iki farklı kod oranı uygulanabilir. Bu durumda önce, yüksek öncelikli bit dizisinin kod oranı iletilir, bunu düşük öncelikli bit dizisinin kod oranı takip eder.

Tablo 4.13 TPS'te kod oranı bilgisi iletimi

s_{27}, s_{28} ve s_{29} bitleri (yüksek öncelikli stream) **s_{27}, s_{28} ve s_{29} bitleri (düşük öncelikli stream)**	**Kod oranı**
000	1/2
001	2/3
010	3/4
011	5/6
100	7/8
101	Kullanılmıyor
110	Kullanılmıyor
111	Kullanılmıyor

4.5.1.8 Koruma aralığı bilgisi

İşaretleşmede koruma aralığı değerlerinin ne şekilde iletildiği tablo 4.14'te gösterilmiştir.

Tablo 4.14 TPS'te koruma aralığı bilgisi iletimi

s_{36} ve s_{37} bitleri	**Koruma aralığı değerleri (Δ / T_U)**
00	1/32
01	1/16
10	1/8
11	1/4

4.5.1.9 İletim modu bilgisi

İletim modu bilgisi 2 bit ile taşınır. İletilebilecek değerler tablo 4.15'te verilmiştir.

Tablo 4.15 TPS'te iletim modu bilgisi iletimi

s_{38} ve s_{39} bitleri	**İletim modu**
00	2K modu
01	8K modu
10	Kullanılmıyor
11	Kullanılmıyor

4.5.2 TPS'te hataya karşı koruma

Eş zamanlama verisi ve bilgi taşıyan 53 bite, orjinal BCH(127,113,t=2) kodundan türetilmiş, kısaltılmış BCH(67,53,t=2) kodunun 14 denklik biti eklenmektedir.

Kod üreteç polinomu şu şekilde verilmektedir :

$$h(x) = x^{14} + x^9 + x^8 + x^6 + x^5 + x^4 + x^2 + x + 1 \qquad (4.23)$$

4.5.3 TPS modülasyonu

TPS taşıyıcıları normalleştirilmiş güç seviyesinde iletilirler (Sahip oldukları enerji, tüm veri taşıyıcılarının enerjilerinin ortalamasına eşittir , yani $E[c \times c^*]=1$ 'dir.).

Her TPS taşıyıcısı DBPSK modülasyonu ile modüle edilir. DBPSK her TPS bloğunun başında koşullanır. l. simgenin k. taşıyıcısının farksal modülasyonu için aşağıdaki kural uygulanır:

$$\text{Eğer } s_1 = 0, \quad \mathrm{Re}\{c_{m,l,k}\} = \mathrm{Re}\{c_{m,l-1,k}\}; \mathrm{Im}\{c_{m,l,k}\} = 0 \tag{4.24}$$

$$\text{Eğer } s_1 = 1, \quad \mathrm{Re}\{c_{m,l,k}\} = -\mathrm{Re}\{c_{m,l-1,k}\}; \mathrm{Im}\{c_{m,l,k}\} = 0 \tag{4.25}$$

Bütün bu bilgilerin ışığında DVB-T'de değişik modülasyon yöntemleri, kod oranları ve koruma aralığı uzunluğu değerleri için iletilebilecek bit hızları tablo 4.16'da verilmiştir. Tablo 4.17'de ise [1] numaralı kaynakta Gauss,Rayleigh ve Rician kanallar için yapılmış benzetim sonuçları yer almaktadır.

Tablo 4.16 2K modunda çalışan hiyerarşik olmayan dizgelerde, 8MHz'lik kanallarda elde edilebilecek bit hızları (Mbit/sn)

Modülasyon	Kod oranı	Bit hızı (Mbit/sn)			
		Δ/T_U = 1/4	Δ/T_U = 1/8	Δ/T_U = 1/16	Δ/T_U = 1/32
QPSK	1/2	4,98	5,53	5,85	6,03
	2/3	6,64	7,37	7,81	8,04
	3/4	7,46	8,29	8,78	9,05
	5/6	8,29	9,22	9,76	10,05
	7/8	8,71	9,68	10,25	10,56
16-QAM	1/2	9,95	11,06	11,71	12,06
	2/3	13,27	14,75	15,61	16,09
	3/4	14,93	16,59	17,56	18,10
	5/6	16,59	18,43	19,52	20,11
	7/8	17,42	19,35	20,49	21,11
64-QAM	1/2	14,93	16,59	17,56	18,10
	2/3	19,91	22,12	23,42	24,13
	3/4	22,39	24,88	26,35	27,14
	5/6	24,88	27,65	29,27	30,16
	7/8	26,13	29,03	30,74	31,67

Tablo 4.17 DVB tarafından, benzetim sonucu elde edilen, farklı kod oranları ve modülasyon yöntemleri için Rayleigh, Ricean ve Gauss kanallarında eşik C/N değerleri

		Viterbi kod çözücüden sonra $2x10^{-4}$ bit-hata oranı değerini elde etmek için gereken C/N(dB)		
Modülasyon	**Kod oranı**	**Gauss Kanalı**	**Ricean Kanalı**	**Rayleigh Kanalı**
QPSK	1/2	3,1	3,6	5,4
QPSK	2/3	4,9	5,7	8,4
QPSK	3/4	5,9	6,8	10,7
QPSK	5/6	6,9	8,0	13,1
QPSK	7/8	7,7	8,7	16,3
16-QAM	1/2	8,8	9,6	11,2
16-QAM	2/3	11,1	11,6	14,2
16-QAM	3/4	12,5	13,0	16,7
16-QAM	5/6	13,5	14,4	19,3
16-QAM	7/8	13,9	15,0	22,8
64-QAM	1/2	14,4	14,7	16,0
64-QAM	2/3	16,5	17,1	19,3
64-QAM	3/4	18,0	18,6	21,7
64-QAM	5/6	19,3	20,0	25,3
64-QAM	7/8	20,1	21,0	27,9

4.6 Şebeke Yapıları

4.6.1 Boşluk doldurucular (gap-fillers)

COFDM yönteminin işaret yansımalarıyla baş edebildiği, hatta kimi durumda yansımaları bir avantaj haline çevirdiğini gördük. Bu durumda, dizgenin, yansımaların yapay yollarla, kasıtlı olarak yaratıldığı bir ortamda çalıştırılması düşünülebilir. Bu fikirden yola çıkılarak yapılacak bir uygulama, pasif yansıtıcılar veya küçük aktif röleler kullanarak, yansıtılan işaretin taşıyıcı frekansını değiştirmeden, kuvvetlendirilip işaretin normalde ulaşmadığı bölgelere yeniden yayınlamak olabilir. Boşluk doldurucular ilave bir frekansa ihtiyaç duymamaktadırlar ve yapılarının basitliği maliyetlerini düşük tutmaktadır.

4.6.2 Yoğun şebekeler

Bir bölgede yer alan tüm vericilerin birbiriyle eş zamanlı bir biçimde, aynı işareti aynı frekansta iletmesi durumunda alıcı girişinde alınan güç, her vericiden gelen işaret güçlerinin toplamı olacaktır. Fakat, birden fazla vericinin varlığı, kanalın yapısını frekans seçici hale getirmektedir.

Değişik noktalardan gelen işaretler, benzer işaretin yansımaları olarak algılanacaktır (yapay yansıma) ve gecikme süreleri koruma aralığı süresinden küşük kaldığı

takdirde bu işaretler yapıcı bir şekilde birbirlerini etkileyecektir. Bu yöntemi kullanarak çok sayıda vericinin birbirleriyle bağlanması pek çok avantaja sahiptir:

- Aşırı güçlü vericilere ihtiyaç duyulmayacağından, yayın altyapısının maliyeti düşecektir
- Düşük bir yatırım bedeliyle yeni bir servisin yayına sokulması mümkün olmaktadır. Daha sonra, yavaş bir şekilde servis sayısında ve kapsama alanında büyüme gerçekleştirilebilir
- Kapsama alanı sınırlarında bitişik kanal girişiminin kontrolü daha iyi gerçekleştirilir. Bu sayede, frekans tekrar kullanma uzaklığı düşürülmüş olur.

4.6.3 Tek frekanslı şebekeler

Vericilerin birbirleriyle eş zamanlı olarak aynı frekanstan yayın yapması fikri, birbirlerine yakın vericilerden bir yayının iletimi için, farklı taşıyıcı frekansı kullanılması gerektiği günümüz analog televizyon yayın uygulamasıyla bir tezat oluşturmaktadır. Fransa'da 45 adet 8 MHz'lik kanalın 5 ulusal kanal tarafından doldurulduğu düşünülürse (1995 yılındaki durum) COFDM'in kullanıldığı DVB-T dizgesi, ayrı ayrı kullanılabilecek bant sayısını 9 katına çıkartmaktadır.

Tek frekanslı şebekelerde vericinin gücünün yanı sıra koruma aralığı uzunluğu da önemlidir. Nispeten büyük koruma aralığı değerleri, komşu vericiler arasındaki mesafenin büyük tutulmasını sağlar. Örneğin $200\mu sn$'lik bir koruma aralığıyla 60 km'lik mesafeye çıkılabilir ($200\mu snx300000km/sn = 60km$).

Uzun yansıma gecikmelerini karşılayabilmek, vericiler arasındaki mesafeyi arttırıp daha geniş alanı daha az vericiyle kapsayabilmek için büyük koruma aralığı değerlerine sahip olan 8K iletim modu uygun olmaktadır.

5. DVB-T TEMEL DİZGE BENZETİMİ

Benzetimde DVB-T temel dizgesinin temel bantta ve Beyaz Gauss gürültülü (AWGN – Additive White Gaussian Noise) kanallardaki başarımı incelenmiştir.

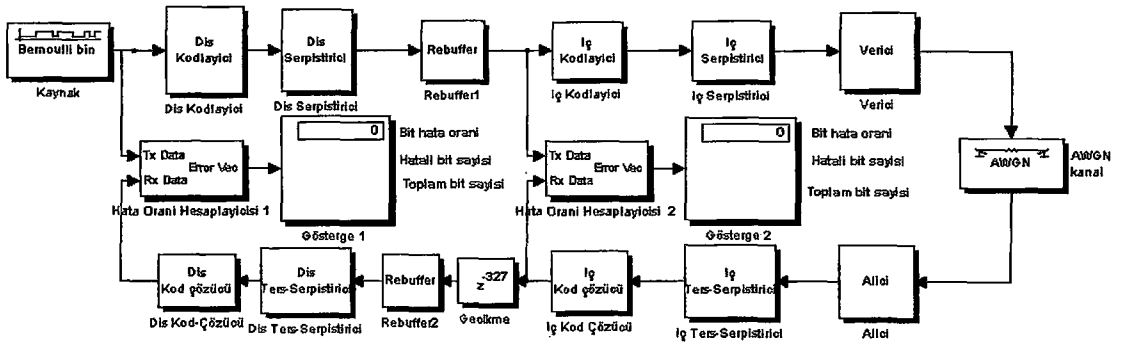
Benzetim için Matlab 5.3 (Sürüm 11) kullanılmıştır. Tasarımda, Simulink kütüphanesinde yer alan bloklardan yararlanılmıştır. Kullanılan birçok bloğun bu sürüme ait olmasından dolayı, benzetimin geriye doğru uyumluluğu yoktur. Benzetimde, temel DVB-T dizgesinin daha iyi anlaşılması ve anlatılması ve Gauss kanalındaki davranışının ortaya koyulması amaçlanmıştır ve DVB-T şartnamesine (EN 300 744) sadık kalınmıştır. Benzetimi yapılan dizgede FEC oranı olarak 3/4 işletme modu olarak 2K modu seçilmiş, 64-QAM eşleştirmesi kullanılmıştır.

Benzetim parametrelerinde yer alan çözücü seçeneklerinde tip olarak değişken adım("variable step") ve ayrık (discrete) değerleri seçilmiştir.

Takibeden bölümlerde, benzetimde kullanılan bloklar ve işlevleri hakkında bilgi verilecektir.

5.1 Benzetimde Kullanılan Bloklar

Tasarlanan benzetici şekil 5.1'de gösterilmektedir.



Şekil 5.1 DVB-T Benzetimi

5.1.1 Kaynak

Kaynak olarak Matlab'in Haberleşme Kütüphanesi'nde (Communication Toolbox) bulunan "Bernoulli Random Binary Generator" bloğu kullanılmıştır. Bir sonraki blok

olan dış kodlayıcı için gerekli olan 1504 bit üretilmektedir. Blok için girilmesi gereken parametre isimleri ve değerleri tablo 5.1'de gösterilmiştir.

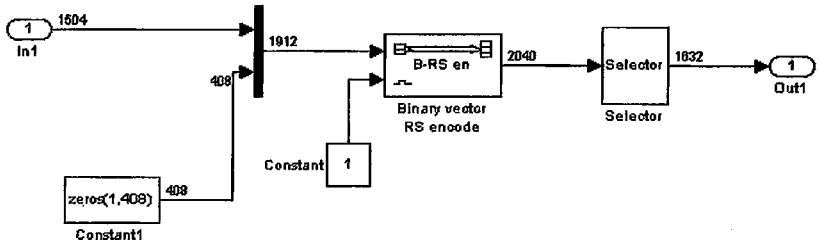
Tablo 5.1 İşaret kaynağı bloğu için girilmesi gereken parametre değerleri

Parametre	Açıklama	Değer
Sıfır gelme olasılığı	Sıfır gelme olasılığını belirler.Vektör olarak girilirse çıkışta, olasılık vektörüyle aynı uzunluğa sahip, ikili(binary) vektörler üretilir	zeros(1,1504)+0.5
Tohum (Seed)	Rastgele sayı üretecinin davranışını belirler	floor(1e5*rand(1,1504))
Örnekleme Zamanı	Kaç saniyede bir çıkış üretileceği belirlenir	1

zeros(M,N) işleviyle MxN boyutunda sıfır matrisi oluşturulmaktadır.

5.1.2 Dış kodlayıcı

Dış kodlamada kısaltılmış Reed-Solomon RS(204,188,t=8) kodlaması kullanılmıştır. Bloğun çekirdeğinde Matlab haberleşme kütüphanesinde yer alan "Binary vector I/O RS encode" bloğu bulunmaktadır. Bu çekirdek bloğun parametreleri olarak mesaj uzunluğu için 239, kod uzunluğu için 255 kullanılmıştır.Bu kod, 255 sekizli içerisinden 8 rastgele hatalı sekizliyi düzeltme özelliğine sahiptir. Kısaltılmış RS(204,188,t=8) kodu, kullanılan çekirdek RS kodlama bloğuna girilen her bir veri bloğunun sonuna 51 adet sıfır sekizli eklemek ve kodlayıcının çıkışında, eklenen bu sekizlileri süzgeçlemek suretiyle elde edilebilir.



Şekil 5.2 Dış kodlayıcı

Her benzetim adımında, dış kodlayıcı bloğuna 1504 bit girmekte 1632 bit çıkmaktadır.

5.1.3 Dış serpiştirici

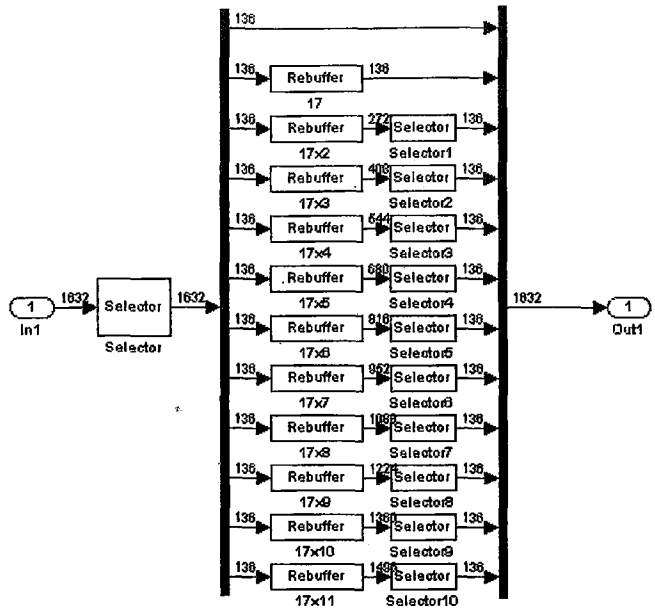
EN 300 744'te önerildiği gibi, giriş veri dizisine bir anahtar aracılığıyla, dairesel olarak bağlanan 12 daldan oluşmaktadır. j numaralı dal, jxM adet (M=17=N/I,N=204) hücreden oluşan İGİÇ kaydırmalı kaydedici olarak tasarlanmaktadır.

Girişe gelen bitlerin, her benzetim adımında, bir dala bir sekizli gelecek şekilde düzenlenmelerinde tablo 5.2'deki koşullama komutları kullanılır:

Tablo 5.2 Dış serpiştirici bloğu koşullaması

```
ilkonyedi = [];
 ayrisma = [];
 for dal = 1:12,
     for cell = 1:17,
          for bit = 1:8,
                ayrisma =[ayrisma ((8*(dal-1)) + bit)];
          end;
        dal = dal + 12;
      end;
 end;

 for j = 1:136,
       ilkonyedi = [ilkonyedi j];
 end;
```

Oluşturulan "ayrisma" vektörü, girişte yer alan, Matlab Simulink kütüphanesindeki (toolbox) seçme ("selector") bloğunun bileşenler ("elements") parametresinin değeri olarak kullanılır. Bu sayede, her bir dala bir sekizli getirilmiş olur. Kullanılan, Matlab DSP kütüphanesinde yer alan, tampon ("rebuffer") bloklarında tampon boyutu ("buffer size") parametresi için $17x8x(j-1)$, örtüşme ("buffer overlap") parametresi için $17x8x(j-2)$, ilk değer ("initial conditions") parametresi için ise 0 değerleri kullanılmıştır. Burada j, dal numarasını temsil etmektedir. Tampon blokları için çerçeve tabanlı giriş seçeneği işaretlenmiş ve kanal sayısı olarak da 1 seçilmiştir. Örtüşme parametresi, mevcut tamponda, önceki tampona ait kaç adet örneğin tutulacağını belirler.



Şekil 5.3 Dış Serpiştirici

Her benzetim adımında, dış serpiştirici bloğuna 1632 bit girmekte ve 1632 bit çıkmaktadır.

5.1.4 İç kodlayıcı

İç kodlayıcı 2 blok kullanılarak tasarlanmıştır: Matlab haberleşme kütüphanesinde bulunan konvolüsyonel kodlayıcı ("Convolutional Encoder") ve delme("puncture") bloğu.

Konvolüsyonel kodlama bloğunun üreteç polinomları bir çıkış için 171_{OKTAL} ve diğer çıkış için 133_{OKTAL} olmaktadır. Kodlayıcı 1/2 oranına sahip kodlar üretmektedir ve 64 duruma(state) sahiptir. Kodlayıcı bloğunun "constraint length" parametresi değeri 7 olarak seçilmiştir.

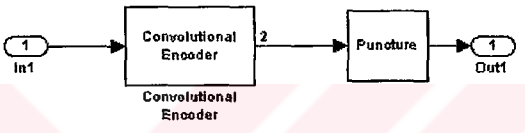
Kodlayıcının ardından kullanılan delme bloğu ile 1/2'den daha yüksek kod oranları elde edilebilmektedir. Kodlayıcı bloğundan çıkan 2 bitlik veri paketleri önce 6 bitlik paketler haline getirilir. Daha sonra, delme bloğunun "Delme Matrisi" parametresi için girilen değer ([1 0 1;1 1 0]) kullanılır.

Tablo 5.3 Delme bloğu koşullaması

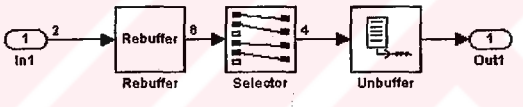
```
[insize,buflen] = size(pmat);
outsize = sum(sum(pmat));
n = 0;
outpos = [];
for j = 1:buflen,
   for i = 1:insize,
      n = n+1;
      if(pmat(i,j)), outpos = [outpos n];end;
   end;
end;
```

Tablo 5.3'te verilen blok koşullama koduyla, iletilecek ve iletilmeyecek (silinecek) bit konumları belirlenir ve buna göre giriş bitleri, seçme bloğuyla seçilir.

İşlenen bitler, delme bloğu çıkışında, "Matlab Simulink DSP" kütüphanesinde yer alan "unbuffer" bloğuyla seriye çevrilir.



Şekil 5.4 İç kodlayıcı



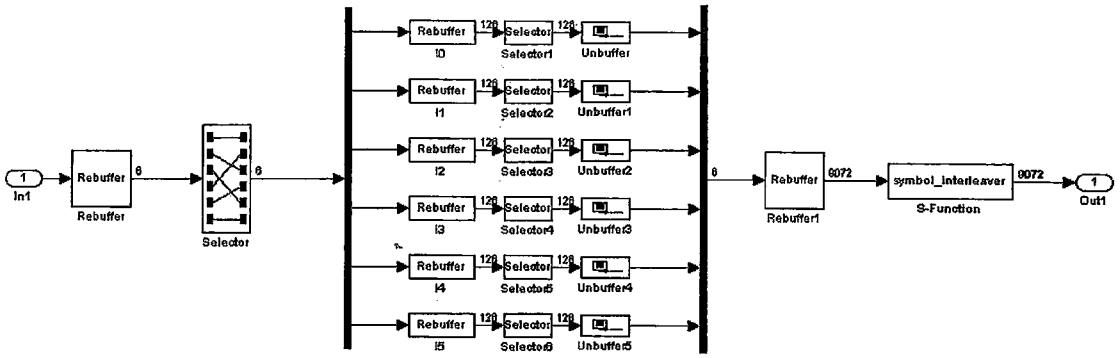
Şekil 5.5 İç kodlayıcı delme bloğu

Her benzetim adımında, iç kodlayıcı bloğuna 1 bit girmekte ve 1 bit çıkmaktadır.

5.1.5 İç serpiştirici

Bu blokta bit ve simge düzeyinde serpiştirme işlemi gerçekleştirilir. Seri olarak gelen veri, 64-QAM eşleştirmesi kullanılacağından, tampon bloğu aracılığıyla altışar bitlik paketler halinde işlenir. Tampon bloğundan hemen sonra kullanılan seçme bloğuyla ayrıştırma işlemi gerçekleştirilir. Ayrıştırma işleminde, seçme bloğu bileşenler("elements") parametresi için [1 4 2 5 3 6] değeri kullanılmaktadır. Ayrıştırılan 6 bitlik veri paketleri, her bir dala, her benzetim adımında 1 bit girecek şekilde 6'ya bölünür. Her bir dala giren bitler, tampon bloğuyla 126 bitlik paketler haline getirilir ve pakete ait bitler, bileşenler parametre değerleri iç serpiştirici blok koşullamasında belirlenen vektörlerden oluşan seçicilerle serpiştirilir. Serpiştirme işleminden sonra bir çoğullayıcı ile dallardan gelen bitler birleştirilerek, her bir

benzetim adımında, bir simge oluşturulur. 6 bitlik bu paketler 9072'lik bir uzunluğa sahip tampon bloğuyla birleştirilir. Bu şekilde 1512 taşıyıcıya dağıtılacak olan bitler serpiştirilmiş ve biraraya getirilmiş olur.



Şekil 5.6 İç serpiştirici

Blok koşullamasında kullanılan kodun bölümleri ve ilgili açıklamalar aşağıda verilmektedir :

Tablo 5.4a Değişkenlerin koşullaması

```
so = 0 ;
DosyaHandle = fopen('SymbNo.txt','w+');
fwrite( DosyaHandle , so , 'integer*1' );
fclose(DosyaHandle);
```

Kodun, tablo 5.4a'da verilen bölümünde "symbno.txt" isimli dosya içeriği 0 olarak koşullanmaktadır. Daha sonraki bloklarda bu değer kullanılacak ve her yeni simge üretiminde 1 arttırılacaktır.

Tablo 5.4b Bit serpiştirmede kullanılan seçme bloklarının davranışının belirlenmesi

```
interleaver0 = [];
interleaver1 = [];
interleaver2 = [];
interleaver3 = [];
interleaver4 = [];
interleaver5 = [];

for k=0:125,
    interleaver0 = [interleaver0 (k+1)];
    interleaver1 = [interleaver1 mod((k+63),126)+1];
    interleaver2 = [interleaver2 mod((k+105),126)+1];
    interleaver3 = [interleaver3 mod((k+42),126)+1];
    interleaver4 = [interleaver4 mod((k+21),126)+1];
    interleaver5 = [interleaver5 mod((k+84),126)+1];
end;
```

Bu kod bloğu sayesinde bit serpiştirmede kullanılan seçme ("selector") bloklarının bileşenler parametresi için girilmesi gereken değer, EN 300 744'te gösterildiği şekilde belirlenmektedir.

Tablo 5.4c R_i ve R_i' değişkenlerinin koşullanması

```
hq = zeros(1,1512);
 R(2048,10) = 0;
 RI(2048,10) = 0;
```

Bu blok ile diğer kısımlarda kullanılan değişkenler koşullanmaktadır. R vektörü, iç serpiştirme ile ilgili bölümde bahsedilen R_i vektörüne, RI vektörü ise aynı bölümdeki R_i' vektörüne denk düşmektedir.

Tablo 5.4d H(q)'nun bulunmasında kullanılan R_i vektörünün elde edilmesi

```
RI(3,1) = 1;
 for satir = 4:2048
      for sutun = 1:9,
            RI(satir,sutun) = RI((satir-1),(sutun+1));
           end;
           RI(satir,10) = XOR(RI((satir-1),1),RI((satir-1),4));
 end;

 R(:,1) = RI(:,10);
 R(:,2) = RI(:,7);
 R(:,3) = RI(:,5);
 R(:,4) = RI(:,2);
 R(:,5) = RI(:,1);
 R(:,6) = RI(:,8);
 R(:,7) = RI(:,4);
 R(:,8) = RI(:,9);
 R(:,9) = RI(:,6);
 R(:,10) = RI(:,3);
```

Burada, simge serpiştirme ile ilgili bölümde de anlatıldığı gibi, permütasyon işlevi H(q)'nun bulunmasında kullanılan R_i vektörü elde edilmektedir.

Tablo 5.4e H(q)'nun elde edilmesi

```
q = 1;
 for g=0:2047,
      aratoplam = 0;
      for j=1:10,
           aratoplam = aratoplam + R((g+1),j)*power(2,(j-1));
      end;
     hq(q) = mod(g,2)*power(2,10) + aratoplam;
     if hq(q) < 1512
               q = q + 1;
     end
 end;

 DosyaHandle = fopen('hq.txt','w+');
 fwrite( DosyaHandle , hq , 'integer*2' );

 fclose(DosyaHandle);
```

Burada, ilgili bölümde verilen algoritma kullanılarak h(q) işlevi değerleri bulunmakta ve bu değerler simge serpiştirme bloğunda kullanılmak üzere dosyaya kaydedilmektedir.

Bit serpiştirme ve çoğullama işlemlerinden sonra 9072 bitlik veri paketleri simge serpiştirme bloğuna girmektedir. Simge serpiştirme bloğu Matlab işlevleri kullanılarak oluşturulmuş, içeriği Ek b'de verilen, bir s-işlevinden ibarettir. Bu işlevde çıkış değerinin belirlendiği kod aşağıda verilmektedir.

Tablo 5.5 Blok çıkışının belirlenmesi

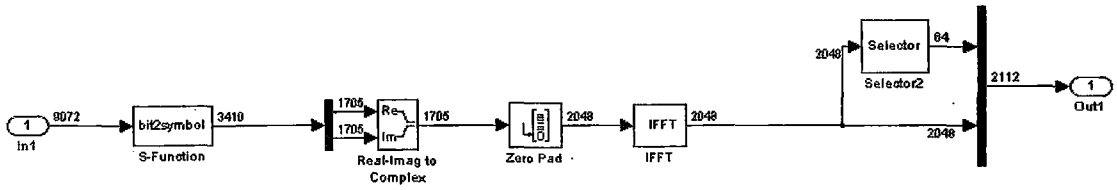
```
DosyaHandle = fopen('SymbNo.txt','r');
 SembolNo = fread(DosyaHandle,1,'integer*1');
 fclose(DosyaHandle);

 DosyaHandle = fopen('hq.txt','r');
 hq = fread(DosyaHandle,'integer*2');
 fclose(DosyaHandle);
 % selector davranisini belirle ================================

 disp('interleaver SembolNo');
 disp(SembolNo);

 for symbno = 0:1511,
        if mod(SembolNo,2) == 1
          for r = 1:6
             sys(6*symbno + r) = u(6*hq(symbno+1) + r);
          end;
        end

        if mod(SembolNo,2) == 0
          for r = 1:6
             sys(6*hq(symbno+1) + r) = u(6*symbno + r);
          end;
        end
 end;
```

Burada iç serpiştirici bloğunun koşullanmasıyla elde edilen ve dosyaya yazılan H(q) işlevi değerleri ilgili dosyadan okunup bir değişkene atanmaktadır. Bu işlemden sonra OFDM simge numarasının tek veya çift olmasına göre simge serpiştirme gerçekleştirilmektedir ve hem bit hem de simge düzeyinde serpiştirilmiş veri 9072 bitlik bloklar halinde blok dışına verilmektedir.

Her benzetim adımında, iç serpiştirme bloğuna 1 bit girmekte ve 9072 bit çıkmaktadır.

5.1.6 Verici

Verici bloğunda, referans işaretlerin eklenmesi , 64-QAM simgelerinin oluşturulması, bu simgelere ters HFD dönüşümü uygulanması ve koruma aralığının eklenmesi işlemleri gerçekleştirilmektedir.

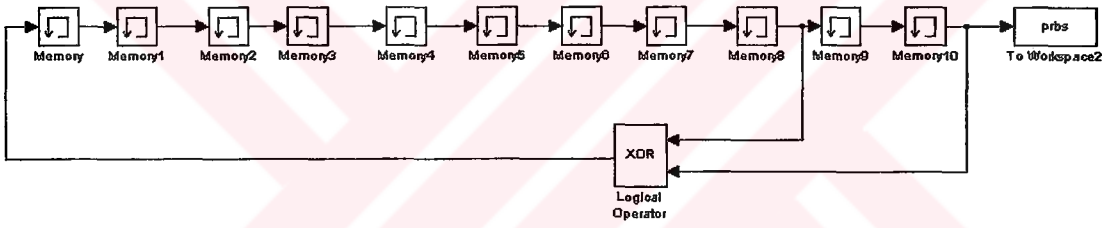


Şekil 5.7 Verici

Bu bloğa giren 9072 bitlik paketler, simge oluşturma bloğu tarafından işlenir ve her bir 6 bit, Ek A'da gösterildiği şekilde ilgili simge değerlerine eşleştirilir ve simgelerin sanal ve gerçel kısımları $1/\sqrt{42}$ ile çarpılarak normalleştirilir. Bu işlevle saçılmış ve devamlı pilot taşıyıcı bitleri de eşleştirilmektedir.

Bu işlevin içeriği Ek C'de verilmiştir.

Bu işlevde kullanılan wk vektörünün elemanlarının değerleri şekil 5.8'de gösterilen PRBS üreteç modeli kullanılarak elde edilmiştir.



Şekil 5.8 Matlab PRBS üreteç modeli

Alıcı ve verici blokları arasında eş zamanlamayı sağlamak için, işlem gören COFDM simge numarası bir dosyaya kaydedilmektedir.

Simge oluşturma bloğu çıkışında simgelerin gerçel ve sanal kısımları tek bir blokta toplanır. Gerçel ve sanal kısımlar daha sonra bir ayrıştırıcı ile ayrıştırılır ve Matlab Simulink kütüphanesinde yer alan "Real-Imag to Complex" bloğuyla 1705 adet, veri ve referans bilgisi taşıyıcılarına karşı düşen, karmaşık değer elde edilir. Bu değerleri içeren vektör, 2048 noktalı ters HFD bloğuna girileceğinden, bu vektörün sonuna, Matlab DSP kütüphanesinde yer alan "Zero Pad" bloğuyla, 343 adet sıfır eklenir.

Ters HFD işlemi sonucunda elde edilen vektörün sonunda yer alan 64 eleman, koruma aralığı elde etmek amacıyla, vektörün başına eklenmekte ve vektör uzunluğu 2112'ye çıkmaktadır ($\Delta/T_U = 1/32$).

Her benzetim adımında, verici bloğuna 9072 bit girmekte ve 2112 karmaşık işaret çıkmaktadır.

5.1.7 Kanal

Benzetimde, dizgenin, sabit antenle alıştaki başarımını ölçmek için 'Beyaz Gauss Gürültü'lü kanaldaki davranışı incelenmiş ve bu amaçla Matlab haberleşme kütüphanesinde yer alan "AWGN channel" bloğu kullanılmıştır. Bu blokla ilgili olarak tohum("Seed") parametresi için 123, mod parametresi için "işaret-gürültü oranı", giriş işaret güç parametresi için 1/2101 değerleri kullanılmıştır. Doğrudan, benzetim üzerinden bir COFDM simgesinin ortalama üretilme zamanının 4,162 sn. olduğu ölçülmüş ve T_s parametresi için bu değer kullanılmıştır.

Benzetim, farklı E_S/N_0 (dB) değerleri için tekrarlanmıştır. Bit hata oranı ve E_b/N_0 eğrisinin elde edilmesi amacıyla E_S/N_0 (dB) değerlerinden $10\log(6) = 7{,}782$ değeri çıkartılmıştır ($E_b = E_S/6$).

Matlab 5'le birlikte gelen AWGN kanal bloğunda kullanılan, "toolbox\comm\commsfun" dizininde bulunan "scomawgnchan" adlı s-işlevi tek giriş ve tek çıkış için tasarlandığından düzeltilmesi gerekmektedir. İşlevin değiştirilmiş hali Ek F'de yer almaktadır.

Aynı işlevin kullandığı, "toolbox\comm\commasks" dizininde bulunan "commblkawgnchan.m" dosyasında da "varargout{1} = floor(1e5*rand(2,1));" satırı yerine "varargout{1} = floor(1e5*rand(4224,1));" satırı yazılmalıdır.

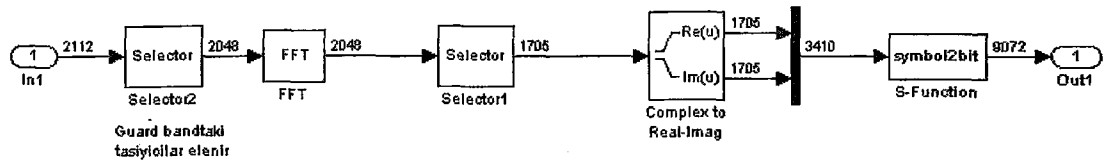
Söz konusu değişiklikler yapılmadan benzetim çalışmayacaktır.

5.1.8 Alıcı

Alıcı bloğu girişinde kullanılan seçme bloğuyla koruma aralığındaki taşıyıcılar elenir. Elemeden sonra elde edilen vektör HFD bloğuna girilir. Bu işlemden sonra, verici bloğunda eklenen sıfırlar, seçme bloğuyla elenir ve elde edilen karmaşık işaretlerden oluşan 1705 uzunluğundaki vektör, Matlab Simulink kütüphanesinde yer alan "Complex to Real-Imag" bloğuyla gerçel ve sanal kısımlarına ayrılır. Gerçel ve sanal değerler çoğullayıcı bloğuyla tek bir vektörde toplanır ve 3410 uzunluğundaki bu vektör simge dönüştürme bloğuna girilir.

Simge dönüştürme bloğu bir s-işlevinden oluşmaktadır. Bu işlev Ek D'de verilmiştir.

Bu blokta referans taşıyıcılar değerlendirmeye alınmamakta ve elenmektedir.

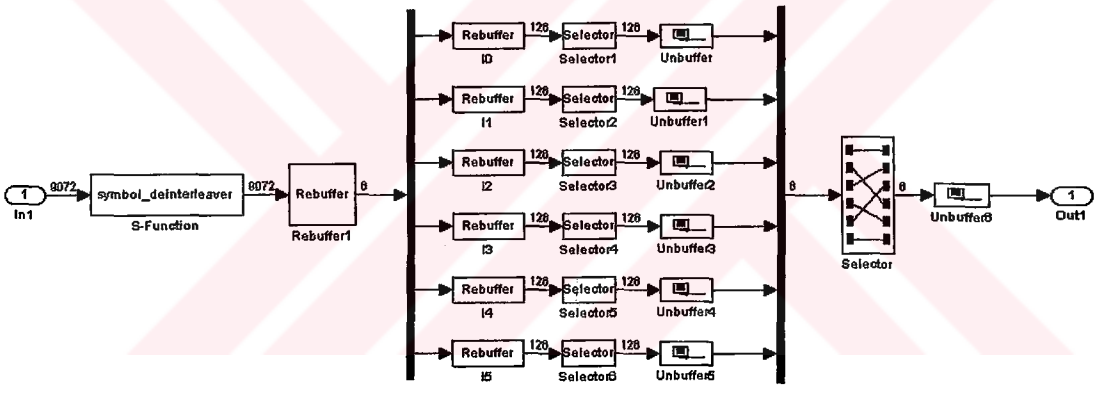


Şekil 5.9 Alıcı

Her benzetim adımında, alıcı bloğuna 2112 karmaşık işaret girmekte ve 9072 bit çıkmaktadır.

5.1.9 İç ters-serpiştirici

Bu bloğa giren, 1512 adet 64-QAM simgesine karşı düşen 9072 bit, simge ters-serpiştirici bloğuyla ters-serpiştirilir. Bu bir s-işlevi bloğudur ve içeriği Ek E'de verilmiştir. H(q) değerleri iç serpiştirici bloğu tarafından hesaplanıp bir dosyaya yazılmıştır. Bu yüzden bu işlevde yeniden bir hesaplama yapılmamakta, değerler doğrudan dosyadan alınmaktadır.



Şekil 5.10 İç ters-serpiştirici

Ters-serpiştiriciden çıkan 9072 bitlik vektörler, bir tampon bloğuyla 6'şar bitlik paketlere bölünür ve bir ayrıştırıcı bloğuyla her bir dala 1 bit gelecek şekilde, 6 daldan oluşan bit ters-serpiştiriciye girilir. Bit ters-serpiştiricide, her bir dalda kullanan seçme bloklarına ait "bileşenler" parametre değerleri, iç ters-serpiştirici bloğunun koşullanması sırasında elde edilir. İç ters-serpiştirici koşullama kodu açıklamalarla beraber aşağıda verilmektedir :

Tablo 5.6a'da verilen kod bloğuyla kullanılan vektörler koşullanmaktadır.

Tablo 5.6b'de verilen kodla ise bit ters-serpiştiricisinde yer alan seçme bloklarının davranışları belirlenmektedir.

Tablo 5.6a İç ters-serpiştirici bloğunun koşullanması

```
interleaver0 = [];
interleaver1 = [];
interleaver2 = [];
interleaver3 = [];
interleaver4 = [];
interleaver5 = [];
deinterleaver0 = zeros(1,126);
deinterleaver1 = zeros(1,126);
deinterleaver2 = zeros(1,126);
deinterleaver3 = zeros(1,126);
deinterleaver4 = zeros(1,126);
deinterleaver5 = zeros(1,126);

R(2048,10) = 0;
RI(2048,10) = 0;
```

Tablo 5.6b Seçme bloklarının davranışlarının belirlenmesi

```
for k=0:125,
    interleaver0 = [interleaver0 (k+1)];
    interleaver1 = [interleaver1 mod((k+63),126)+1];
    interleaver2 = [interleaver2 mod((k+105),126)+1];
    interleaver3 = [interleaver3 mod((k+42),126)+1];
    interleaver4 = [interleaver4 mod((k+21),126)+1];
    interleaver5 = [interleaver5 mod((k+84),126)+1];
end;

for k=1:126,
    deinterleaver0(interleaver0(k)) = k ;
    deinterleaver1(interleaver1(k)) = k ;
    deinterleaver2(interleaver2(k)) = k ;
    deinterleaver3(interleaver3(k)) = k ;
    deinterleaver4(interleaver4(k)) = k ;
    deinterleaver5(interleaver5(k)) = k ;
end;
```

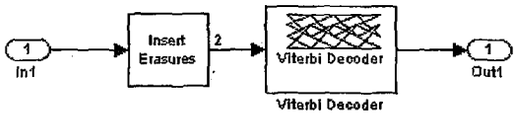
Seçme bloklarından sonra veri, "unbuffer" bloğuyla seriye çevrilmekte ve her bir daldan gelen bitler çoğullanıp 6 bitlik paketler haline getirilmektedir. Bu paketler, daha sonra bir seçme bloğundan ibaret olan birleştiriciye girilmektedir. Birleştirici, iç serpiştirici bloğunda yer alan ayrıştırıcının yaptığı işlemin tersini gerçekleştirir. Bu bloğun "bileşenler" parametresinin değeri olarak [1 3 5 2 4 6] kullanılmaktadır.

Birleştirici bloğundan sonra veri seriye çevrilmektedir.

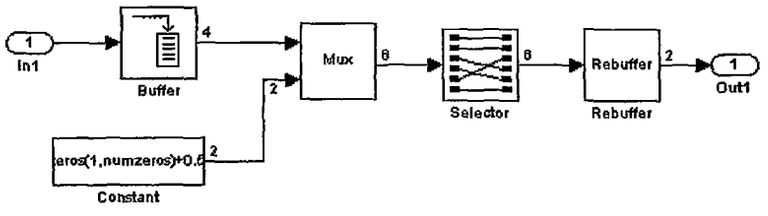
Her benzetim adımında, iç ters-serpiştirici bloğuna 9072 bit girmekte ve 1 bit çıkmaktadır.

5.1.10 İç kod çözücü

İç kod çözücü, iki bloktan oluşmaktadır: Sıfır ekleme bloğu ve Viterbi kod çözücü bloğu.



Şekil 5.11 İç kod çözücü



Şekil 5.12 İç kod çözücü sıfır ekleme bloğu

Sıfır ekleme bloğunun "delme matrisi" parametresi için, iç kodlayıcı bloğunda yer alan "Delme" alt bloğuna girilmiş olan matris kullanılmaktadır ([1 0 1;1 1 0]). Sıfır ekleme bloğuna giren bitler dörder bitlik bloklar haline getirilir. Daha sonra dört bitlik bloklara iki adet 0,5 değeri eklenir (0 ya da 1 değil de 0,5 gibi bir ara değer eklenmesinin nedeni, sıfır ekleme bloğundan sonra gelen Viterbi kod çözme bloğunda sert karar verme yönteminin kullanılıyor olmasıdır). Oluşan altı elemanlı paketler, "bileşenler" parametresi kod çözücü koşullama kodu ile belirlenen ve sonradan eklenen elemanları (0,5 değerli), iç kodlama bloğu tarafından silinen elemanların yerine yerleştiren seçme bloğuna girer.

Seçme bloğundan sonra, veri paketleri, Viterbi kod çözücüye girmek amacıyla, tampon bloğu kullanılarak, iki elemanlı bloklar haline getirilir.

Viterbi kod çözücüde üreteç polinomları olarak konvolüsyonel kodlama bloğundaki değerler ($[171\ 133]_{oktal}$) kullanılır. Kod çözücüde sert karar verme yöntemi kullanılmaktadır.

Tablo 5.7 Sıfır ekleme bloğu koşullama kodu

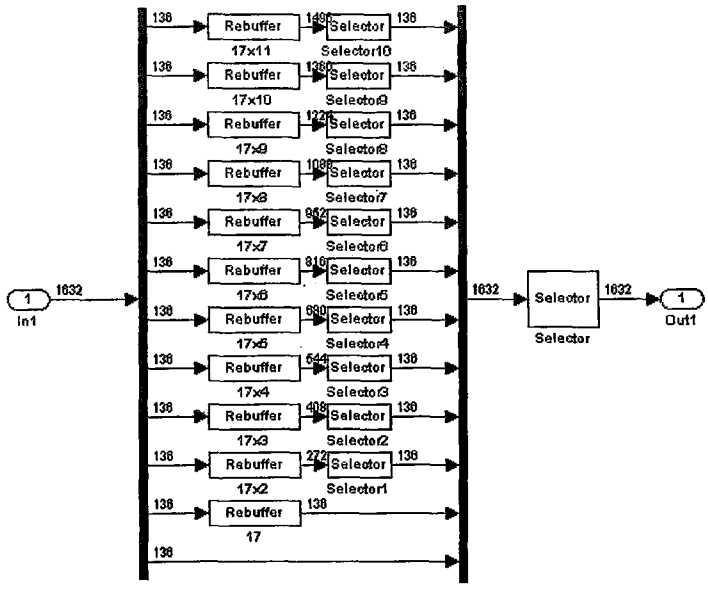
```
[insize,buflen] = size(pmat);
% buflen, bloğa ait "delme matrisi" parametre değeridir
outsize = sum(sum(pmat));
numzeros = insize*buflen-outsize;
n1 = 0;
n2 = outsize;
shuffle = [];
for j = 1:buflen,
   for i = 1:insize,
      if(pmat(i,j))
         n1 = n1+1;
         shuffle = [shuffle n1];
      else
         n2 = n2+1;
         shuffle = [shuffle n2];
      end;
   end;
end;
```

Her benzetim adımında, iç kod çözücü bloğuna 1 bit girmekte ve 1 bit çıkmaktadır.

5.1.11 Dış ters–serpiştirici

İç kod çözücüden çıkan bitler, doğrudan dış ters-serpiştiriciye girilemez. Öncelikle kod çözücü bloğu çıkışına kadar işlenen bit sayısını, giriş ve çıkış arasında evre kayması oluşmaması ve dolayısıyla ters-serpiştiricinin doğru çalışması için,1632'nin bir katı yapmamız gerekmektedir. Bu amaçla yapılan incelemede, iç kodlama bloğuna girişle iç kod çözücü bloğundan çıkış arasında 15993 bitlik bir kaymanın söz konusu olduğu görülmüştür. Buradan yola çıkılarak 15993'ü 1632'nin bir katı olan 16320 yapmak için eklenmesi gereken olan bit sayısının 327 olduğu, kod çözücü çıkışında her bir adımda 1 bit işlendiğinden blok çıkışındaki verinin 327 örnek periyodu boyunca geciktirilmesi gerektiği ortaya çıkmıştır. Bu amaçla, DSP kütüphanesinde yer alan "Integer Delay" bloğu kullanılmıştır.

Gecikme bloğundan hemen sonra veri dizisi 1632 bitten oluşan paketler haline getirilmekte ve dış ters-serpiştiriciye girilmektedir.



Şekil 5.13 Dış ters-serpiştirici

Bu blokta koşullama için tablo 5.8'de verilen kod kullanılmaktadır:

Tablo 5.8 Dış ters-serpiştirici bloğu koşullama kodu

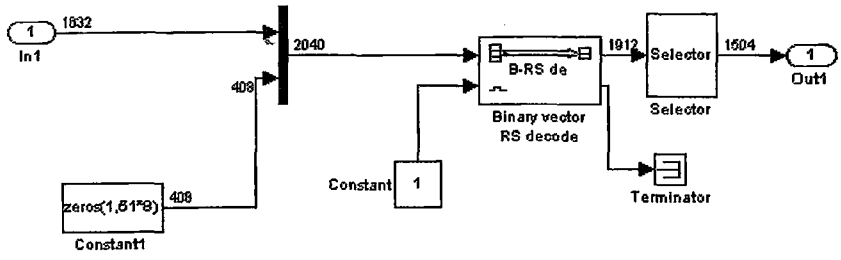
```
ayrisma_deint = [];
 ilkonyedi = [];
 ayrisma = [];
 for dal = 1:12,
     for cell = 1:17,
          for bit = 1:8,
                ayrisma =[ayrisma ((8*(dal-1)) + bit)];
          end;
        dal = dal + 12;
      end;
 end;

 for k  = 1:1632,
            ayrisma_deint(ayrisma(k)) = k;

 end;
```

Serpiştirici bloğunda kullanılan dalların sırası, $m \leq 6$ olduğu sürece m numaralı dal $(12-m+1)$ numaralı dal ile yer değiştirecek şekilde, değiştirilir. Blok çıkışında yer alan seçme alt bloğu, dış serpiştiricide yer alan, ayrıştırma işleminden sorumlu, seçme bloğunun tam tersi işlemi yaparak ("bileşenler" parametre değeri olarak ayrisma_deint vektörü kullanılmak suretiyle) bit dizilişini düzeltir.

Her benzetim adımında, dış ters-serpiştirici bloğuna 1632 bit girmekte ve 1632 bit çıkmaktadır.

5.1.12 Dış kod çözücü

Bu bloğun çekirdeğinde Reed-Solomon RS(255,239,t=8) kod çözücü bloğu yer almaktadır. Kodlamada kullanılan kısaltılmış RS(204,188,t=8) kodunu elde etmek için, bloğa girilen 1632 bitlik veriye 408 adet sıfır eklenir. Ortaya çıkan 2040 bitlik dizi RS(255,239,t=8) kod çözücü bloğa girilir. Kod çözücü çıkışında elde edilen 1912 bitlik dizinin ilk 188 sekizlisi alınarak blok çıkışına verilir.



Şekil 5.14 Dış kod çözücü

Her benzetim adımında, dış kod çözücü bloğuna 1632 bit girmekte ve 1504 bit çıkmaktadır.

5.2 Bit Hata Oranının Hesaplanması

Bit hata oranı hesabı için Matlab haberleşme kütüphanesinde yer alan "Error Rate Calculation" bloğundan yararlanılmıştır. İç kodlayıcı ile iç kod çözücü arasına yerleştirilen hata oranı ölçme bloğu için 15993 örneklik bir gecikme değeri kullanılmıştır. Veri kaynağı ile dış kod çözücü arasına yerleştirilen hata oranı ölçme bloğunda ise 23 örneklik bir gecikme değeri kullanılmıştır.

Gecikmeler, kullanılan tampon("rebuffer","buffer" ve "unbuffer") bloklarından kaynaklanmaktadır. Giriş ve çıkış arasındaki gecikme, benzetimde iletilen ve alınan bit dizilerinin incelenmesiyle bulunmuştur.

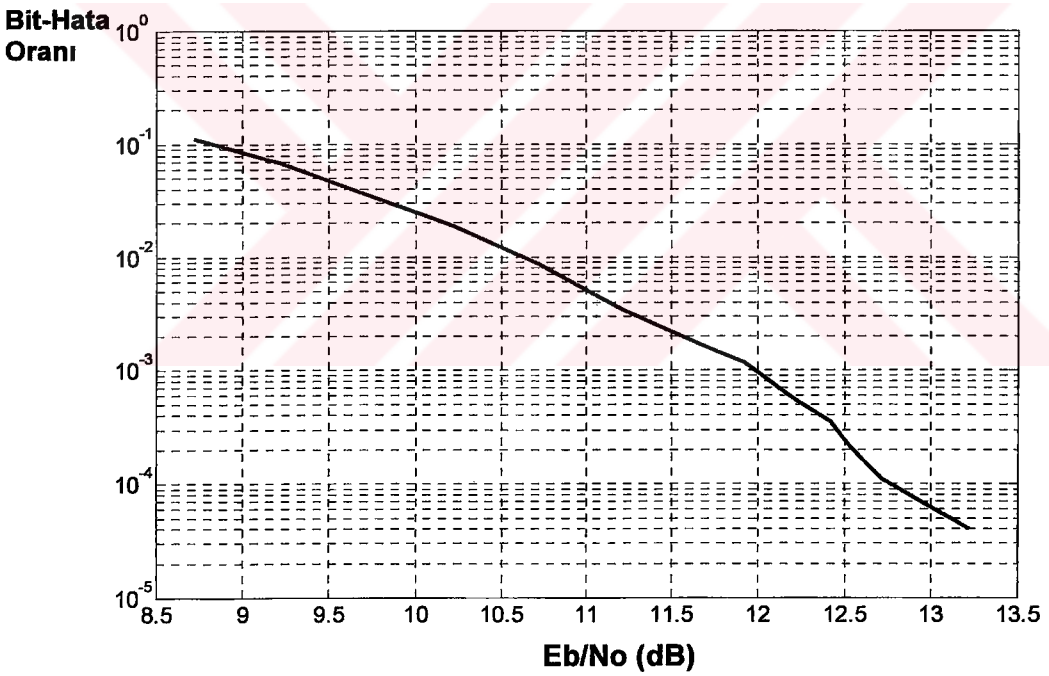
5.3 Benzetim Sonuçları

14 farklı E_S / N_0 değeri için Viterbi kod çözücü ve Reed-Solomon kod çözücüden sonra meydana gelen bit-hata oranları ölçülmüştür. Her bir ölçüm için ortalama 400000 bit kullanılmıştır. Sonuçlar tablo 5.9'da verilmiştir.

Tablo 5.9'daki E_S / N_0 değerlerinden $10\log(6) = 7{,}782$ değeri çıkartılarak ($E_b = E_S/6$) elde edilen Bit-Hata Oranı - E_b / N_0 eğrisi şekil 5.15'te gösterilmiştir.

Tablo 5.9 Benzetim sonucunda bulunan Es/No ve bit-hata oranı değerleri

E_S / N_0 (dB)	Viterbi Kod Çözücü Çıkışındaki Bit-Hata Oranı	Reed-Solomon Kod Çözücü Çıkışındaki Bit-Hata Oranı
16,5	0,1111	0,1147
17,0	0,06833	0,07048
17,5	0,03571	0,03645
18,0	0,01904	0,01609
18,5	0,008604	0,004687
19,0	0,00344	0,000856
19,2	0,002482	0,0004535
19,5	0,001576	0,000143
19,7	0,001167	≈0
20,0	0,000543	≈0
20,2	0,0003542	≈0
20,3	0,0002227	≈0
20,5	0,0001109	≈0
21,0	0,00004044	≈0



Şekil 5.15 Benzetim sonucu elde edilen "Eb/No – Bit-Hata Oranı" eğrisi

DVB-T dizge eşiği, Reed-Solomon kod çözme işleminden önce $2x10^{-4}$ değerinde bir bit-hata oranını sağlayan E_b / N_0 değeri olarak tanımlanmaktadır. Bu şart sağlandığı takdirde Reed-Solomon kod çözücü çıkışındaki bit-hata oranı $1x10^{-11}$ civarında olacaktır (Quasi Error Free Alış) ve bu da her bir kaç saatte bir, bir hata olasılığı anlamına gelmektedir.

Benzetimi yapılan dizgenin eşik E_b / N_0 değeri, şekil 5.15 üzerinden, 12,55 dB olarak bulunmuştur.

Tablo 4.17'de gösterildiği gibi, DVB tarafından aynı parametrelerle yapılan benzetim sonucunda AWGN kanallarda C/N değeri 18,0 dB olarak bulunmuştur.

$$E_b / N_0 = C/N - 10\log(R_b / BW) \qquad (5.1)$$

Bu eşitlikte R_b, kanaldan iletilen bit sayısını, BW ise iletim bant genişliğini temsil etmektedir.

(5.1) eşitliği kullanılarak, kod oranı 3/4, modülasyon yöntemi 64-QAM, $\Delta / T_U = 1/32$ iken E_b / N_0 değeri $18 - 10\log(27{,}14/8) = 12{,}69dB$ olarak bulunur.

Tasarlanan benzeticiyle bulunan eşik E_b / N_0 değeri ile DVB'ninki arasında 0,14 dB'lik bir fark vardır. Aradaki farkın kanal ve Viterbi kod çözücü blok parametrelerinden kaynaklandığı düşünülmektedir. Farkın %3'lük bir sapmaya denk düştüğü göz önünde bulundurularak benzeticinin doğru çalıştığı sonucuna varılmıştır.

6. SONUÇ

Bu çalışmada, uzun bir süredir Avrupa'nın gündemini meşgul eden, pek çok ülkede kullanılmaya başlanmış, pek çok ülkede deneme yayınları sürmekte olan sayısal karasal televizyon yayın dizgesi DVB-T, anahtar özellikleriyle birlikte açıklanmıştır.

Yapılan araştırmada, DVB-T dizgesinin karasal kanallardaki başarımının, kullanılan COFDM modülasyon yöntemine dayandığı, COFDM'in şiddetli çoklu yol yayılmasına karşı oldukça dirençli olduğu; dizgenin taşıyıcı sayısı, iletim modu (hiyerarşik ve hiyerarşik olmayan), iç kod oranı ve koruma aralığı süresi gibi ayarlanabilir parametrelere sahip olması nedeniyle hemen hemen her türlü iletim şartında kullanılabildiği görülmüştür.

İşaretin içine referans taşıyıcılar gömmek suretiyle zaman ve frekans uzamında ara değer bulma işlemiyle, eşitlemenin ve kanal kestiriminin oldukça hızlı bir şekilde gerçekleştirildiği ve koruma aralığının etkin bir biçimde kullanılmasıyla spektrum veriminin önemli ölçüde arttırıldığı ortaya koyulmuştur.

Dizgenin daha iyi anlaşılması ve anlatılması amacıyla, Matlab 5.3 kullanılarak benzetimi yapılmış, tüm dizge blokları Matlab Simulink kullanılarak tasarlanmış ve benzetime dahil edilmiştir.

İç kod oranı olarak 3/4, modülasyon yöntemi olarak 64-QAM, iletim modu olarak 2K ve Δ / T_U oranı olarak 1/32 seçilmiş ve seçilen bu parametrelere göre benzetim 14 farklı E_S / N_0 değeri için gerçekleştirilmiştir. Bu değerler için Viterbi kod çözücü çıkışındaki bit-hata oranı ölçülmüştür.

E_S / N_0 değerleri E_b / N_0'a dönüştürülmüş ve E_b / N_0 - Bit-hata oranı eğrisi elde edilmiştir. Bu eğriden faydalanılarak, DVB tarafından $2x10^{-4}$ olarak belirlenen, Viterbi kod çözücü çıkışındaki bit hata oranı eşik değerine karşı düşen E_b / N_0 değeri olarak 12,55 dB bulunmuştur. DVB'nin yaptığı benzetim sonuçlarıyla arada 0,14 dB'lik bir sapma olduğu görülmüş, bu sapmanın kullanılan kanal ve de Viterbi kod çözücü blokları arasındaki farklılıktan kaynaklanabileceği düşünülmüş ve sunulan benzeticinin, DVB-T dizgesinin AWGN kanallardaki başarımının ölçülmesinde kullanılabileceği sonucuna varılmıştır.

KAYNAKLAR

[1] **EN 300 744 V1.2.1**, 1997. Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television, *European Telecommunications Standards Institue.*

[2] **Bingham, John A. C.**, May 1990. Multicarrier Modulation for Data Transmission: An Idea Whose Time Has Come, *IEEE Communications Magazine*, 5-14.

[3] **Floch, B. L., Alard, M. and Berrou, C.**, 1995. Coded Orthogonal Frequency Division Multiplex, *Proceedings of the IEEE,* **Vol. 83**, **No. 6,** 982-996.

[4] **Cimini, L. J., JR.**, 1985. Analysis and Simulation of a Digital Mobile Channel Using Orthogonal Frequency Division Multiplexing, *IEEE Transactions on Communications*, **Vol. COM-33, No.7**, 665-675.

[5] **Zou, W. Y. and Wu, Y.**, 1995. COFDM: An Overview, *IEEE Transactions on Broadcasting*, **Vol.41, No. 1**, 1-8.

[6] **Casas, E. F. and Leung C.**, 1991. OFDM for Data Communication Over Mobile Radio FM Channels – Part I: Analysis and Experimental Results, *IEEE Transactions on Communications,* **Vol.39, No. 5**, 783-793.

[7] **O'Leary, S.**, 1997. Hierarchical Transmission and COFDM Systems, *IEEE Transactions on Broadcasting,* **Vol.43, No.2**, 166-174.

[8] **Weste, N. and Skellern, D.J.**, October 1998. VLSI for OFDM, *IEEE Communication Magazine*, 127-131.

[9] **Armada, A. G., Bardon B. and Calvo M.**, 1998. Parameter Optimization and Simulated Performance of a DVB-T Digital Television Broadcasting System, *IEEE Transactions on Broadcasting,* **Vol.44, No.1**, 131- 138.

[10] **Lee, W., Park H. and Park, J.**, 1999. Viterbi Decoding Method Using Channel State Informatiin in COFDM System, *IEEE Transactions on Consumer Electronics,* **Vol. 45, No.3,** 533-537.

[11] **Lee, W., Park, H., Kang, K. and Kim, K.**, 1998. Performance Analysis of Viterbi Decoder Using Channel State Information in COFDM System, *IEEE Transactions on Broadcasting,* **Vol.44, No.4**, 488-495.

[12] **Wu, Y.**, 1999. Performance Comparison of ATSC 8-VSB and DVB-T COFDM Transmission Systems for Digital Television Terrestrial Broadcasting, *IEEE Transactions on Consumer Electronics,* **Vol.45, No.3**, 916-924.

[13] **Arrinda, A., Velez M. M., Angueira, P., Vega, D. and Ordiales, J.L.**, 1999. Digital Terrestrial Television (COFDM 8K System) Field Trials and Coverage Measurement in Spain, *IEEE Transactions on Broadcasting*, **Vol.45, No.2**, 171-176.

[14] **Stott, J.H.**, 1998. The How and Why of COFDM, *EBU Technical Review*

[15] **Stott, J.H.**, 1997. Explaining some of the magic of COFDM, *Proceedings of 20th International Television Symposium*

[16] **Moller, L.G.**, 1997. COFDM and the Choice of Parameters for DVB-T, *Proceedings of 20th International Television Symposium*

[17] **Weck C. and Schramm R.**, 1997. Receiving DVB-T: Results of Field Trials and Coverage Considerations, *Proceedings of 20th International Television Symposium*

[18] **Chini, A., Wu, Y., El-Tanany, M. and Mahmoud S.**, 1998. Filtered Decision Feedback Channel Estimation for OFDM-Based DTV Terrestrial Broadcasting System, *IEEE Transactions on Broadcasting*, **Vol.44, No.1**, 2-11.

[19] **Ligeti, A. and Zander, J.**, 1999. Minimal Cost Coverage Planning for Single Frequency Networks, *IEEE Transactions on Broadcasting*,**Vol.45, No.1,** 78-87.

[20] **Shelswell, P.**, 1996. The COFDM Modulation System: The Heart of Digital Audio Broadcasting, *BBC Research and Development Report*

[21] **Yasuda, Y., Kashiki, K. and Hirata, Y.**, 1984. High-Rate Punctured Convolutional Codes for Soft Decision Viterbi Decoding, *IEEE Transactions on Communications*, **Vol.COM-21, No.3**, 315-319.

[22] **Nokes, C., Laflin, N. and Darlington, D.**, 2000. All-Digital Planning and Digital Switch-Over, *BBC Research and Development Presentation.*

[23] **Nokes, C. R. and Mitchell, J. D.**, 1999. Potential Benefits of Hierarchical Modes of the DVB-T Specification, *BBC Research and Development*

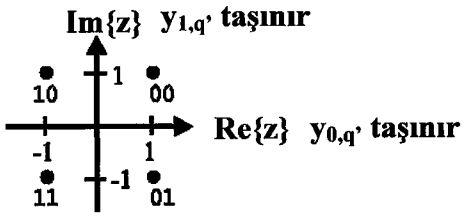
[23] **Bahai, A. R. S. and Saltzberg B. R.**, 1999. Multi-Carrier Digital Communications Theory and Applications of OFDM, Kluwer Academic/Plenum Publishers, New York.

[24] **Bruin, R. and Smits, J.**, 1999. Digital Video Broadcasting Technology, Standards and Regulations, Artech House Inc., Norwood.

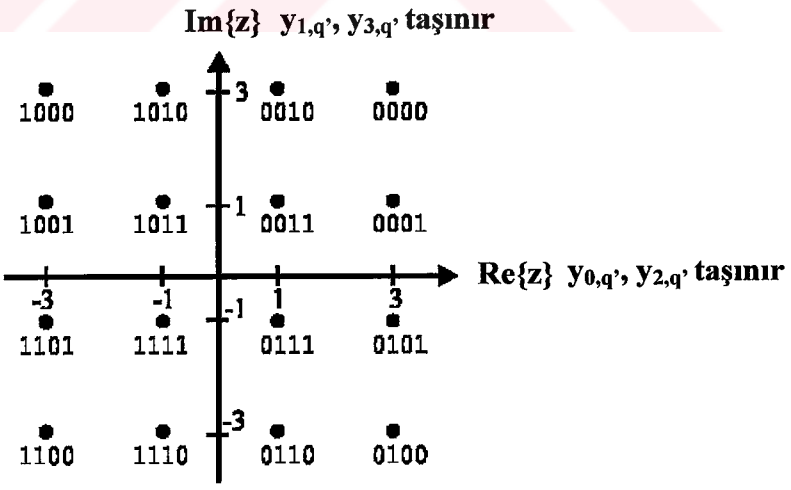
[25] **Whitaker, J, Benson, B.**, 2000. Standard Handbook of Video and Television Engineering, McGraw-Hill, New York.

Ek A

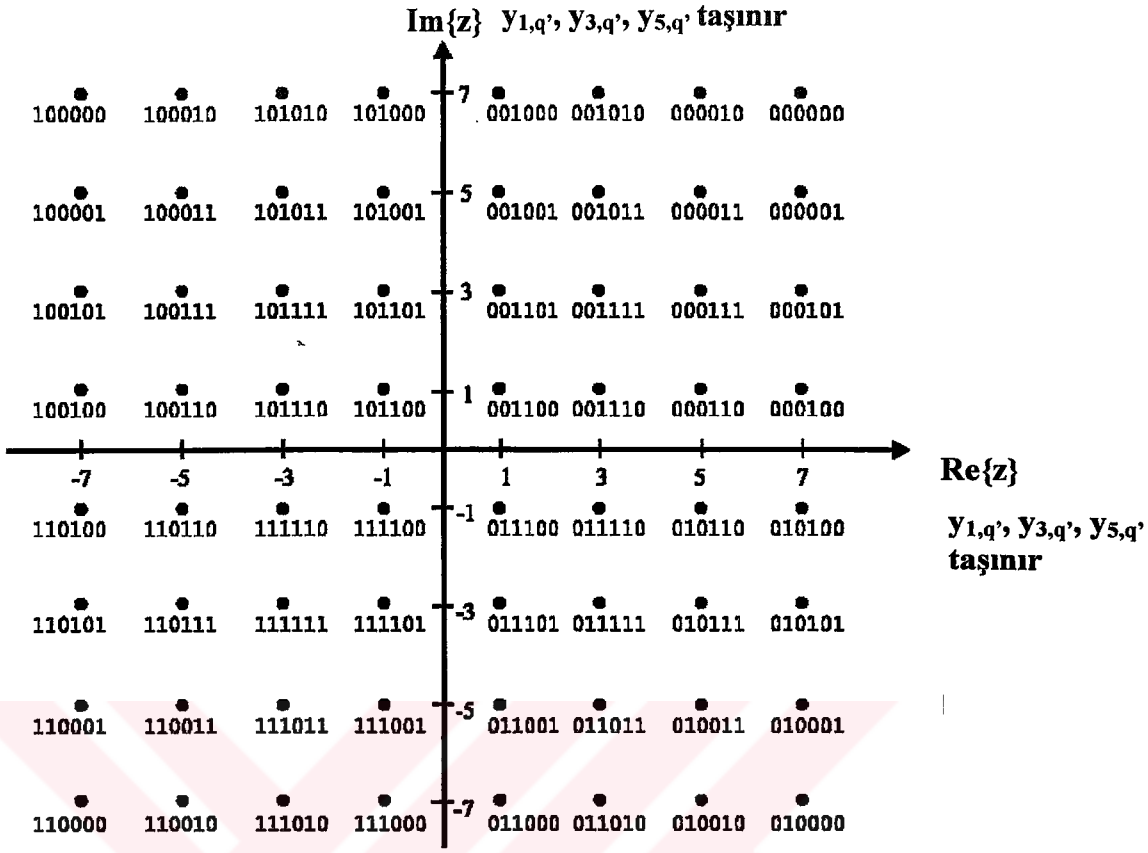
İŞARET UZAYLARI VE EŞLEŞTİRME



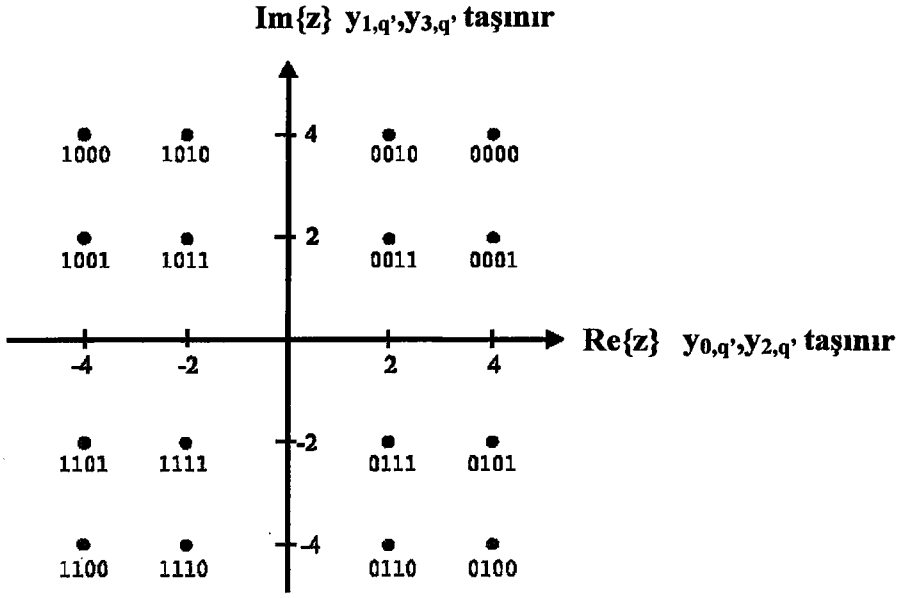
Şekil A.1 Hiyerarşik olmayan modda QPSK işaret uzayı (α = 1)



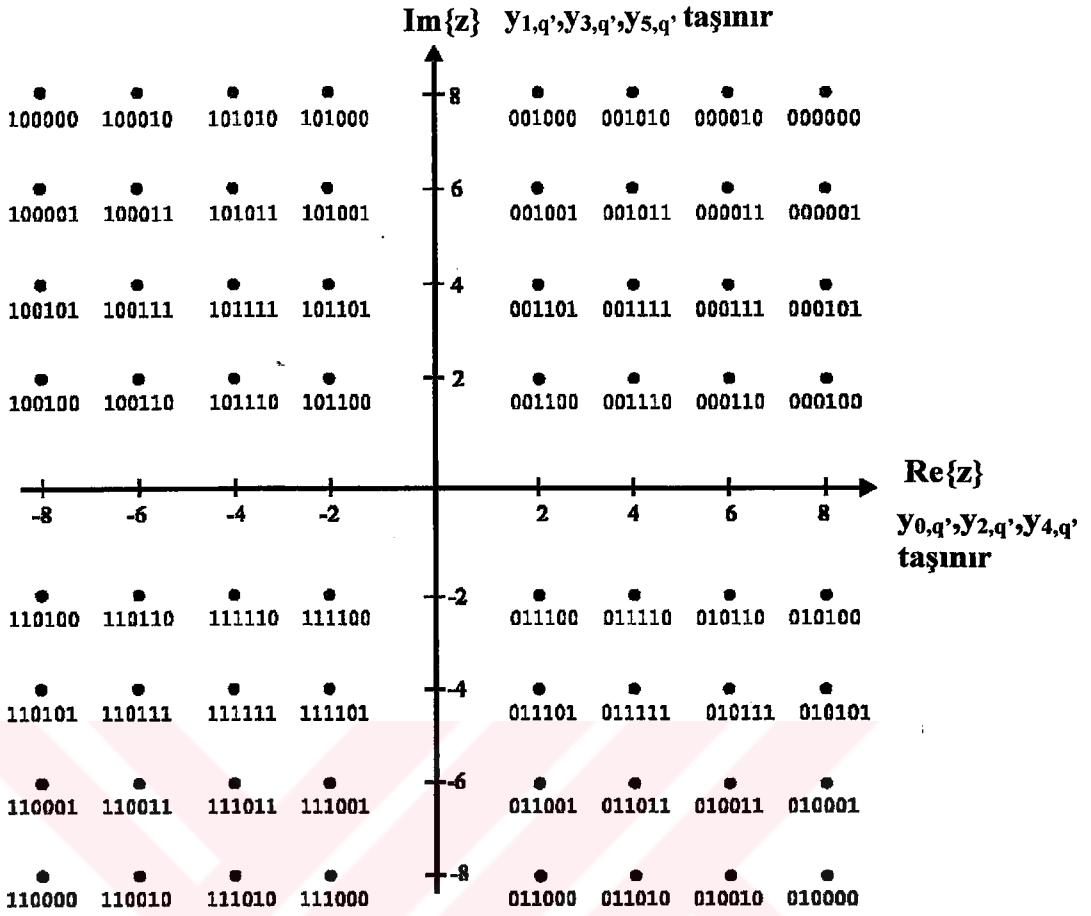
Şekil A.2 Hiyerarşik olmayan modda 16-QAM işaret uzayı (α = 1)



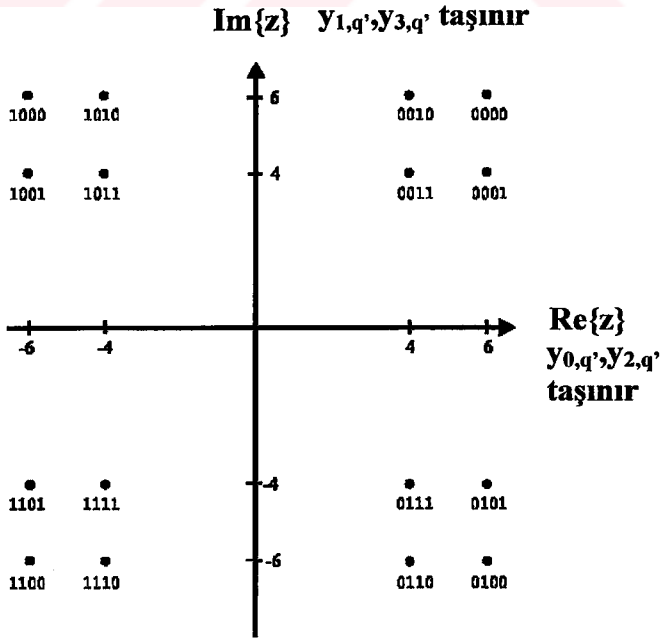
Şekil A.3 Hiyerarşik olmayan modda 64-QAM işaret uzayı (α = 1)



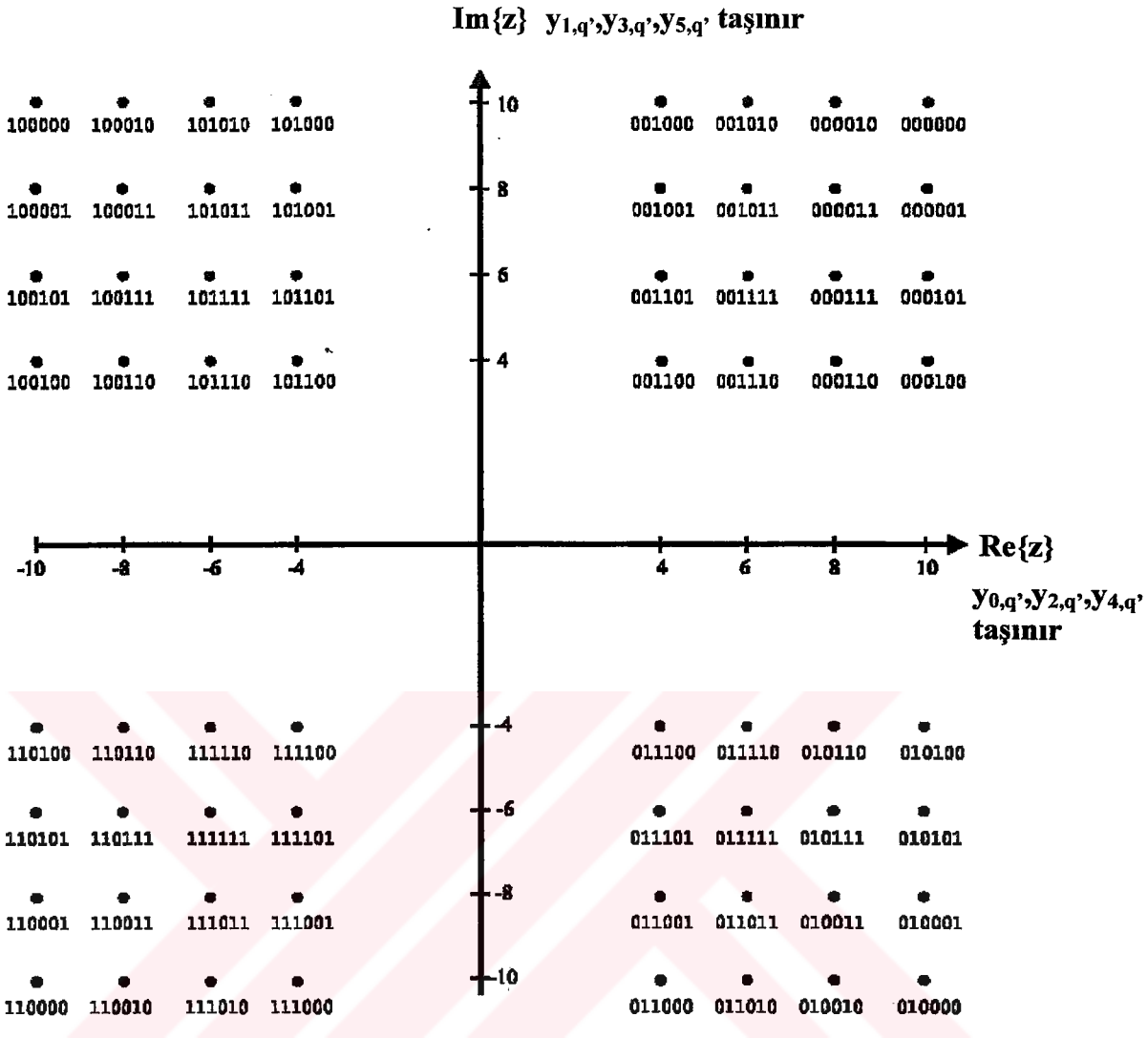
Şekil A.4 Düzenli olmayan 16-QAM işaret uzayı(α = 2)



Şekil A.5 Düzenli olmayan 64-QAM işaret uzayı(α = 2)



Şekil A.6 Düzenli olmayan 16-QAM işaret uzayı(α = 4)



Şekil A.7 Düzenli olmayan 64-QAM işaret uzayı(α = 4)

Ek B

SİMGE SERPİŞTİRİCİ S-İŞLEVİ

```
function [sys,x0,str,ts] = symbol_interleaver(t,x,u,flag)
% Simge serpistirme

switch flag,

  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;

  %%%%%%%%%%%%%%%%
  % Derivatives %
  %%%%%%%%%%%%%%%%
  case 1,
   sys=mdlDerivatives(t,x,u);

  %%%%%%%%%%
  % Update %
  %%%%%%%%%%
  case 2,
   sys=mdlUpdate(t,x,u);

  %%%%%%%%%%
  % Outputs %
  %%%%%%%%%%
  case 3,
   sys=mdlOutputs(t,x,u);

  %%%%%%%%%%%%%%%%%%%%%%%%
  % GetTimeOfNextVarHit %
  %%%%%%%%%%%%%%%%%%%%%%%%
  case 4,
   sys=mdlGetTimeOfNextVarHit(t,x,u);

  %%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%
  case 9,
   sys=mdlTerminate(t,x,u);

  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
```

```
   error(['Unhandled flag = ',num2str(flag)]);
end

%=============================================================
% mdlInitializeSizes
%=============================================================
function [sys,x0,str,ts]=mdlInitializeSizes

sizes = simsizes;

sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1512*6; % 1512 adet 6 bitlik sembol
sizes.NumInputs      = -1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
double(sys);
%
% initialize the initial conditions
%
x0  = [];

%
% str is always an empty matrix
%
str = [];

%
% initialize the array of sample times
%
ts  = [-1 0];

% end mdlInitializeSizes
%
%=============================================================
% mdlDerivatives
%=============================================================
function sys=mdlDerivatives(t,x,u)

sys = [];

% end mdlDerivatives
%
%=============================================================
% mdlUpdate
%=============================================================
function sys=mdlUpdate(t,x,u)

sys = [];

% end mdlUpdate
```

```
%=============================================================
% mdlOutputs
%=============================================================

function sys=mdlOutputs(t,x,u)

DosyaHandle = fopen('SymbNo.txt','r');
SembolNo = fread(DosyaHandle,1,'integer*1');
fclose(DosyaHandle);

DosyaHandle = fopen('hq.txt','r');
hq = fread(DosyaHandle,'integer*2');
fclose(DosyaHandle);
% selector davranisini belirle ===================================

disp('interleaver SembolNo');
disp(SembolNo);

for symbno = 0:1511,
     if mod(SembolNo,2) == 1
       for r = 1:6
          sys(6*symbno + r) = u(6*hq(symbno+1) + r);
      end;
     end

     if mod(SembolNo,2) == 0
       for r = 1:6
         sys(6*hq(symbno+1) + r) = u(6*symbno + r);
      end;
     end
end;
% =============================================================
% end mdlOutputs
%=============================================================
% mdlGetTimeOfNextVarHit
%=============================================================
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;
sys = t + sampleTime;
%=============================================================
% end mdlGetTimeOfNextVarHit
%=============================================================
% mdlTerminate
%=============================================================
function sys=mdlTerminate(t,x,u)
sys = [];
% end mdlTerminate
```

Ek C

İŞARET EŞLEŞTİRMESİ, REFERANS BİLGİLERİNİN EKLENMESİ, SİMGELERİN OLUŞTURULMASI İŞLEMLERİNİ GERÇEKLEŞTİREN S-İŞLEVİ

```
function [sys,x0,str,ts] = bit2symbol(t,x,u,flag)
% bit2symbol
switch flag,

  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;

  %%%%%%%%%%%%%%%%%
  % Derivatives %
  %%%%%%%%%%%%%%%%%
  case 1,
   sys=mdlDerivatives(t,x,u);

  %%%%%%%%%%
  % Update %
  %%%%%%%%%%
  case 2,
   sys=mdlUpdate(t,x,u);

  %%%%%%%%%%
  % Outputs %
  %%%%%%%%%%
  case 3,
   sys=mdlOutputs(t,x,u);

  %%%%%%%%%%%%%%%%%%%%%%%%%%
  % GetTimeOfNextVarHit %
  %%%%%%%%%%%%%%%%%%%%%%%%%%
  case 4,
   sys=mdlGetTimeOfNextVarHit(t,x,u);

  %%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%
  case 9,
   sys=mdlTerminate(t,x,u);

  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
```

```
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
%=============================================================
% mdlInitializeSizes
%=============================================================
function [sys,x0,str,ts]=mdlInitializeSizes

sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 3410; %1705 tasiyiciyi temsilen (gercel ve sanal kisimlarini)
sizes.NumInputs      = -1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
double(sys);

% initialize the initial conditions
x0  = [];
str = [];
ts  = [-1 0];

% end mdlInitializeSizes
%=============================================================
% mdlDerivatives
%=============================================================
function sys=mdlDerivatives(t,x,u)

sys = [];

% end mdlDerivatives
%=============================================================
% mdlUpdate
%=============================================================
function sys=mdlUpdate(t,x,u)
sys = [];
% end mdlUpdate

%=============================================================
% mdlOutputs
%=============================================================
function sys=mdlOutputs(t,x,u)

DosyaHandle = fopen('SymbNo.txt','r');
SembolNo = fread(DosyaHandle,1,'integer*1');
fclose(DosyaHandle);

DosyaHandle = fopen('SymbNo.txt','w');
fwrite(DosyaHandle, mod( (SembolNo + 1) , 68 ) ,'integer*1');
fclose(DosyaHandle);

TP = [];
```

```
CikisVektoru = 33 * ones( 1 , 3410 );
Tasiyicilar = ones( 1 , 3024 );
% 45 adet devamli pilot
ContinualPilots = [ 0 48 54 87 141 156 192 201 255 279 282 333 432 450 483 525
531 618 636 714 759 765 780 804 873 888 918 939 942 969 984 1050 1101 1107
1110 1137 1140 1146 1206 1269 1323 1377 1491 1683 1704 ];
% 17 adet TPS tasiyicisi
TPSCarriers = [ 34 50 209 346 413 569 595 688 790 901 1073 1219 1262 1286
1469 1594 1687 ];
% wk 1800'luk bir vektor
wk = [ 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 0
0 1 1 0 0 0 1 1 1 1 1 1 1 1 0 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 0 1 1
0 0 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 1 1 1 1 1 0 1 1 1 1
1 0 0 0 1 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1 1 1 0 0 0 0 1 1 0 0 1 0 1 1
0 1 1 1 1 1 0 0 1 1 0 1 0 0 0 0 1 1 1 1 1 0 0 1 0 1 1 0 0 0 1 1 1 0 0 1 1 1 0 1 1 0 1 1 1 1
0 1 0 1 1 0 1 0 0 1 0 0 0 1 1 0 0 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 1 1 0 1 0 1 0
0 0 1 1 1 0 0 0 0 1 0 1 1 0 1 1 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1 0 1 0 0 1 0 1 0 0 1 0 0
1 1 0 0 0 1 1 0 1 1 1 1 1 0 1 1 1 0 1 0 0 0 1 0 1 0 1 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0 1 0 0
0 1 1 1 1 0 1 0 1 0 1 1 0 0 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 1 1 1 0 1
1 0 0 1 0 0 0 1 0 1 1 1 1 0 1 0 1 0 0 1 0 0 1 0 0 0 0 1 1 0 1 1 0 1 0 0 1 1 1 0 1 1 0 0 1
1 1 0 1 0 1 1 1 1 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0
0 0 0 0 1 1 0 1 1 0 0 0 0 1 1 1 0 1 1 1 0 0 1 1 0 1 0 1 0 1 1 1 1 1 0 0 0 0 1 0 0 0 1 1
0 0 0 1 0 1 0 1 1 1 1 0 1 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1
1 1 1 1 1 0 1 1 0 1 0 0 0 0 1 0 1 1 0 0 1 0 0 1 0 0 1 1 1 1 0 1 1 0 1 1 1 0 0 1 0 1 1 0 1
0 1 1 1 0 0 1 1 0 0 0 1 0 1 1 1 1 1 1 0 1 0 0 1 0 0 0 0 0 1 0 0 1 1 0 1 0 0 1 0 1 1 1 1 0 0
1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 0 0 0 0 0 1 0 1 0 1 1 0 0 0 1 0 0 0 0 1 1 1 0 1 0 1 0
0 1 1 0 1 0 0 0 0 1 1 1 1 0 0 1 0 0 1 1 0 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 0 0 0 0 0 1 0 0
0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 1 0 0 0 0 1 1 0 0 0 0 0 1 0 1 1 1 1 0 0 0 1 0 0 1 0 0 1 1 0 1
0 1 1 0 1 1 1 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 1 1 0 0 1 1 1 1 0 1 0 1 1 1 1 0 0 1 0 0 0 1 0 0
1 1 1 0 1 0 1 0 1 1 1 1 0 1 0 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 0
0 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 1 1 1 0 1 1 0 1 0 0 1 0 1 0 1 1 0 0 1 1
0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 0 1 1 0 1 1 1 1 0 0 1
1 1 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 0 0 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1 0 1 0 1 0 1 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 1 0 1 0 1 0 0 1
0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 1 1 0 0 1 0 0 0 1 1 0 1 1 1 0 1 0 1 1 1 0 1 0 1 0 0 0 1
0 1 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 1 1 0 1 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0 1 1 1
1 0 0 1 0 1 1 1 0 0 1 1 1 0 0 1 0 1 1 1 1 0 1 1 1 0 0 1 0 0 1 0 1 0 1 1 1 0 1 1 0 0 0 0 1
0 1 0 1 1 1 0 0 1 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 0 1 0 0 1 1 0 1 1 0 0 0 1 1 1 1 0 1 1
1 0 1 1 0 0 1 0 1 0 1 0 1 1 1 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0
0 0 1 1 1 0 1 1 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 0 1 0 1 0 0 1 0
1 1 0 0 0 0 1 1 0 0 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 0 0 1 1
1 1 1 1 0 1 0 1 1 0 0 0 0 1 0 0 0 1 1 1 0 0 1 0 1 0 1 1 0 1 1 1 0 0 0 0 1 1 0 1 0 1 1 0 0
1 1 1 0 0 0 1 1 1 1 1 0 1 1 0 1 1 0 0 0 1 0 1 1 0 1 1 1 1 0 1 0 0 1 1 0 1 0 1 0 0 1 1 1 1 0
0 0 0 1 1 1 0 0 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1
1 0 1 0 0 0 1 0 0 1 1 0 0 1 0 1 0 1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0 1 0 0 1 1 1 1
1 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 0 1 0 1 1 0 1 1 0 0 0 0 0 1 1
0 1 1 1 0 0 0 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 0 1 1 0 1 1 0 0 1 0 1 1 1 0 1 1 1 1 0 0 1 0 1
0 1 0 0 1 1 1 0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 1 0 1 1 1 0 0 0 1 0 1 0 1 0 1 1 0
1 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 1 1 1 1 0 1 0 0 1 1 0 0 0 1 0 0 1 1 1 1 1 0 1 0 1 1 1 0
0 0 1 0 0 0 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 0 0 0 1 1 0 0 1
1 1 1 0 ];
ScatteredPilots = [];
SembolIndeks = SembolNo;
disp('verici sembol indeks');
disp(SembolIndeks);
```

```
% Scattered pilot indekslerini belirle
for p = 0:1700,
  k = ( 3 * mod( SembolIndeks , 4 ) ) + ( 12 * p ) ;
  if k <= 1704
    if k ~= 0
      ScatteredPilots = [ ScatteredPilots k ] ;
    end
  else
    break;
  end
end;

 % Sacilmis pilotlarin yerlestirilmesi
 for i =  1:142,
   CikisVektoru( 2 * ScatteredPilots( i ) + 1 ) = ( 8 / 3 ) * ( 0.5 -  wk( ScatteredPilots( i
) + 1 ) );
   CikisVektoru( 2 * ScatteredPilots( i ) + 2 ) = 0;
 end
% Devamli pilot tasiyicilarin yerlestirilmesi
Cakisan = 0;
for i = 1:45,
  if CikisVektoru( 2 * ContinualPilots( i ) + 1 ) == 33
    Cakisan = Cakisan + 1;
  end;
  CikisVektoru( 2 * ContinualPilots( i ) + 1 ) = ( 8 / 3 ) * ( 0.5 -  wk( ContinualPilots( i )
+ 1 ) );
  CikisVektoru( 2 * ContinualPilots( i ) + 2 ) = 0;
 end;

%TPS tasiyicilarinin yerlestirilmesi
 for i = 1:17
   CikisVektoru( 2 * TPSCarriers( i ) + 1 ) = 1 - ( 2 * wk( TPSCarriers( 1 ) + 1 ) );
   CikisVektoru( 2 * TPSCarriers( i ) + 2 ) = 0;
 end;

% Veri tasiyicilarinin bulunmasi (yerlestirme islemi sonra)
for i = 1:1512,
  switch bin2dec( sprintf( '%d%d' , u(i*6-3) , u(i*6-1) ) )
  case 0,
    if u( i * 6 - 5 ) == 0
      Tasiyicilar( 2 * i - 1 ) = ( 7 / sqrt( 42 ) );
    else
      Tasiyicilar( 2 * i - 1 ) = ( -7 / sqrt( 42 ) );
    end
  case 1,
    if u( i * 6 - 5 ) == 0
      Tasiyicilar( 2 * i - 1 ) = ( 5 / sqrt( 42 ) );
    else
      Tasiyicilar( 2 * i - 1 ) = ( -5 / sqrt( 42 ) );
    end
  case 2,
    if u( i * 6 - 5 ) == 0
      Tasiyicilar( 2 * i - 1 ) = ( 1 / sqrt( 42 ) );
    else
```

```
      Tasiyicilar( 2 * i - 1 ) = ( -1 / sqrt( 42 ) );
    end
  case 3,
    if u( i * 6 - 5 ) == 0
      Tasiyicilar( 2 * i - 1 ) = ( 3 / sqrt( 42 ) );
    else
      Tasiyicilar( 2 * i - 1 ) = ( -3 / sqrt( 42 ) );
    end
  end;

  switch bin2dec( sprintf( '%d%d' , u( i * 6 - 2 ) , u( i * 6 ) ) )
  case 0,
    if u( i * 6 - 4 ) == 0
      Tasiyicilar( 2 * i ) = ( 7 / sqrt( 42 ) );
    else
      Tasiyicilar( 2 * i ) = ( -7 / sqrt( 42 ) );
    end
  case 1,
    if u( i * 6 - 4 ) == 0
      Tasiyicilar( 2 * i ) = ( 5 / sqrt( 42 ) );
    else
      Tasiyicilar( 2 * i ) = ( -5 / sqrt( 42 ) );
    end
  case 2,
    if u( i * 6 - 4 ) == 0
      Tasiyicilar( 2 * i ) = ( 1 / sqrt( 42 ) );
    else
      Tasiyicilar( 2 * i ) = ( -1 / sqrt( 42 ) );
    end
  case 3,
    if u( i * 6 - 4 ) == 0
      Tasiyicilar( 2 * i ) = ( 3 / sqrt( 42 ) );
    else
      Tasiyicilar( 2 * i ) = ( -3 / sqrt( 42 ) );
    end
  end;
end;

%Cikis vektoru taranir, 33 bulunan yere ve bir sonrasina ilgili tasiyici degerleri
yerlestirilir
TasiyiciNo = 0;
for d = 1:3410,
  if CikisVektoru( d ) == 33
    TasiyiciNo = TasiyiciNo + 1;
    CikisVektoru( d ) = Tasiyicilar( TasiyiciNo ) ;
    TP = [ TP d ];
  end
end
DosyaHandle = fopen('TasiyiciPozisyon.txt','w');
fwrite(DosyaHandle, TP ,'integer*2');
fclose(DosyaHandle);
disp( 'Verici tasiyici sayisi' );
disp( TasiyiciNo );
disp( length( TP ) );
```

```
for i = 1:3410,
  sys( i ) = CikisVektoru( i );
end

% end mdlOutputs
%
%=============================================================
% mdlGetTimeOfNextVarHit
%=============================================================
function sys=mdlGetTimeOfNextVarHit(t,x,u)

sampleTime = 1;
sys = t + sampleTime;

% end mdlGetTimeOfNextVarHit
%
%=============================================================
% mdlTerminate
%=============================================================
function sys=mdlTerminate(t,x,u)
sys = [];
% end mdlTerminate
```

Ek D

SİMGELERDEN İLGİLİ BİTLERİN ELDE EDİLMESİNDE KULLANILAN S-İŞLEVİ

```
function [sys,x0,str,ts] = symbol2bit(t,x,u,flag)
% symbol2bit Gelen simgeler ilgili bitlere cevrilir
switch flag,

  %%%%%%%%%%%%%%%%%%
  % Initialization                %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;

  %%%%%%%%%%%%%%%%
  % Derivatives                  %
  %%%%%%%%%%%%%%%%
  case 1,
   sys=mdlDerivatives(t,x,u);

  %%%%%%%%%%
  % Update          %
  %%%%%%%%%%
  case 2,
   sys=mdlUpdate(t,x,u);

  %%%%%%%%%%
  % Outputs          %
  %%%%%%%%%%
  case 3,
   sys=mdlOutputs(t,x,u);

  %%%%%%%%%%%%%%%%%%%%%%%%
  % GetTimeOfNextVarHit                    %
  %%%%%%%%%%%%%%%%%%%%%%%%
  case 4,
   sys=mdlGetTimeOfNextVarHit(t,x,u);

  %%%%%%%%%%%%
  % Terminate              %
  %%%%%%%%%%%%
  case 9,
   sys=mdlTerminate(t,x,u);

  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags                %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
   error(['Unhandled flag = ',num2str(flag)]);
```

```
end
%=============================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%=============================================================
function [sys,x0,str,ts]=mdlInitializeSizes

sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1512*6;    % Tasiyicilar (6 bitlik)
sizes.NumInputs      = -1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
double(sys);

% initialize the initial conditions

x0  = [];
str = [];
ts  = [-1 0];

% end mdlInitializeSizes
%=============================================================
% mdlDerivatives
%=============================================================
function sys=mdlDerivatives(t,x,u)

sys = [];

% end mdlDerivatives
%=============================================================
% mdlUpdate
%=============================================================
function sys=mdlUpdate(t,x,u)

sys = [];
% end mdlUpdate
%=============================================================
% mdlOutputs
%=============================================================
function sys=mdlOutputs(t,x,u)

IslemZamani = 41691;

DosyaHandle = fopen('SymbNo.txt','r');
SembolNo = fread(DosyaHandle,1,'integer*1');
fclose(DosyaHandle);

CikisVektoru = 33 * ones( 1 , 3410 );
Tasiyicilar = ones( 1 , 3024 );

% 45 adet devamli pilot
```

```
ContinualPilots = [ 0 48 54 87 141 156 192 201 255 279 282 333 432 450 483 525
531 618 636 714 759 765 780 804 873 888 918 939 942 969 984 1050 1101 1107
1110 1137 1140 1146 1206 1269 1323 1377 1491 1683 1704 ];

% 17 adet TPS tasiyicisi
TPSCarriers = [ 34 50 209 346 413 569 595 688 790 901 1073 1219 1262 1286
1469 1594 1687 ];

% wk 1800'luk bir vektor
wk = [ 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 0
0 1 1 0 0 0 1 1 1 1 1 1 1 1 0 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 0 1 1
0 0 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 1 1 1 1 1 0 1 1 1 1
1 0 0 0 1 0 1 0 0 0 1 1 0 1 0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 1 0 1 1
0 1 1 1 1 1 0 0 1 1 0 1 0 0 0 1 1 1 1 1 0 0 1 0 1 1 0 0 0 1 1 1 0 0 1 1 1 0 1 1 0 1 1 1 1
0 1 0 1 1 0 1 0 0 1 0 0 0 1 1 0 0 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 1 1 0 1 0 1 0
0 0 1 1 1 0 0 0 0 1 0 1 1 0 1 1 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1 0 1 0 0 1 0 1 0 0 1 0 0
1 1 0 0 0 1 1 0 1 1 1 1 1 0 1 1 1 0 1 0 0 0 1 0 1 0 1 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0 1 0 0
0 1 1 1 1 0 1 0 1 0 1 1 0 0 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 1 1 1 0 1
1 0 0 1 0 0 0 1 0 1 1 1 1 0 1 0 1 0 0 1 0 0 1 0 0 0 0 1 1 0 1 1 0 1 0 0 1 1 1 0 1 1 0 0 1
1 1 0 1 0 1 1 1 1 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0
0 0 0 0 1 1 0 1 1 0 0 0 0 1 1 1 0 1 1 1 0 0 1 1 0 1 0 1 0 1 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1
0 0 0 1 0 1 0 1 1 1 1 0 1 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1
1 1 1 1 1 0 1 1 0 1 0 0 0 0 1 0 1 1 0 0 1 0 0 1 0 0 1 1 1 1 0 1 1 0 1 1 1 0 0 1 0 1 1 0 1
0 1 1 1 0 0 1 1 0 0 0 1 0 1 1 1 1 1 1 1 0 1 0 0 1 0 0 0 0 1 0 0 1 1 0 1 0 0 1 0 1 1 1 1 0 0
1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 0 0 0 0 0 1 0 1 0 1 1 0 0 0 1 0 0 0 0 1 1 1 0 1 0 1 0
0 1 1 0 1 0 0 0 0 1 1 1 1 0 0 1 0 0 1 1 0 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 0 0 0 0 0 1 0 0
0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 1 0 0 0 0 1 1 0 0 0 0 0 1 0 1 1 1 1 0 0 0 1 0 0 1 0 0 1 1 0 1
0 1 1 0 1 1 1 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 1 0 0 1 1 1 1 0 1 0 1 1 1 1 0 0 1 0 0 0 1 0 0
1 1 1 0 1 0 1 0 1 1 1 0 1 0 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 0 1 1 0 1 0 1 0 1 0 1 1 1 0 0
0 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 1 1 1 0 1 1 0 1 0 0 1 0 1 0 1 1 0 0 1 1
0 0 0 0 1 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 0 1 1 0 1 1 1 1 0 0 1
1 1 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 0 0 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1 0 1 0 1 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 1 0 1 0 1 0 0 1
0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 1 1 0 0 1 0 0 0 1 1 0 1 1 1 0 1 0 1 1 1 0 1 0 1 0 0 0 1
0 1 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0 1 1 1
1 0 0 1 0 1 1 1 0 0 1 1 1 0 0 1 0 1 1 1 1 0 1 1 1 0 0 1 0 0 1 0 1 0 1 1 1 0 1 1 0 0 0 0 1
0 1 0 1 1 1 0 0 1 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 0 1 0 0 1 1 0 1 1 0 0 0 1 1 1 1 0 1 1
1 0 1 1 0 0 1 0 1 0 1 0 1 1 1 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0
0 0 1 1 1 0 1 1 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 1 0 1 0 0 1 0
1 1 0 0 0 0 1 1 0 0 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 0 0 1 1
1 1 1 1 0 1 0 1 1 0 0 0 0 1 0 0 0 1 1 1 0 0 1 0 1 0 1 1 0 1 1 1 0 0 0 0 1 1 0 1 0 1 1 0 0
1 1 1 0 0 0 1 1 1 1 1 0 1 1 0 1 1 0 0 0 1 0 1 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 0 0 1 1 1 1 0
0 0 0 1 1 1 0 0 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1
1 0 1 0 0 0 1 0 0 1 1 0 0 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0 1 0 0 1 1 1 1
1 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 0 1 0 1 1 0 1 1 0 0 0 0 0 1 1
0 1 1 1 0 0 0 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 0 1 1 0 1 1 0 0 1 0 1 1 1 0 1 1 1 1 0 0 1 0 1
0 1 0 0 1 1 1 0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 1 0 1 1 1 0 0 0 1 0 1 0 1 0 1 1 0
1 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 1 1 1 1 0 1 0 0 1 1 0 0 0 1 0 0 1 1 1 1 1 0 1 0 1 1 1 0
0 0 1 0 0 0 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 0 0 0 1 1 0 0 1
1 1 1 0 ];

ScatteredPilots = [];
SembolIndeks = SembolNo - 1; % bit2symbol'de 1 arttirilmisti

disp('alici sembol indeks');
```

```
disp(SembolIndeks);

% Scattered pilot indekslerini belirle
for p = 0:1700,
  k = ( 3 * mod( SembolIndeks , 4 ) ) + ( 12 * p ) ;
  if k <= 1704
    if k ~= 0
      ScatteredPilots = [ ScatteredPilots k ] ;
    end
  else
    break;
  end
end;

 %Scatteredlari yerlestir
 ScatteredSize = size( ScatteredPilots );
 for i =  1:142,
    CikisVektoru( 2 * ScatteredPilots( i ) + 1 ) = 1;
    CikisVektoru( 2 * ScatteredPilots( i ) + 2 ) = 1;
 end
%Devamli pilot tasiyicilarini yerlestir
for i = 1:45,
   CikisVektoru( 2 * ContinualPilots( i ) + 1 ) = 1;
   CikisVektoru( 2 * ContinualPilots( i ) + 2 ) = 1;
 end;

%TPS tasiyicilarini yerlestir
for i = 1:17
    CikisVektoru( 2 * TPSCarriers( i ) + 1 ) = 1;
    CikisVektoru( 2 * TPSCarriers( i ) + 2 ) = 1;
 end;

% Cikis vektorunu taranir, 33 bulunan yere ve bir sonrasina ilgili tasiyici degerleri
% yerlestirilir
TasiyiciNo = 0;
DosyaHandle = fopen('TasiyiciPozisyon.txt','r');
TP = fread(DosyaHandle,'integer*2');
fclose(DosyaHandle);

for t = 1:3024,
   Tasiyicilar( t ) = u( TP( t ) );
end

disp('Alici:');
disp(TasiyiciNo);

%Cikis tasiyici bitlerinin belirlenmesi
for i = 1:1512,

   if(Tasiyicilar(2*i-1) >= (6/sqrt( 42 ))) % gercel kisim

      if(Tasiyicilar(2*i) >= (6/sqrt( 42 ))) %sanal kisim
         sys((i*6-5):(i*6)) = [0 0 0 0 0 0];
      elseif(Tasiyicilar(2*i) < (6/sqrt( 42 )) & Tasiyicilar(2*i) >= (4/sqrt( 42 )))
         sys((i*6-5):(i*6)) = [0 0 0 0 0 1];
```

```
    elseif(Tasiyicilar(2*i) < (4/sqrt( 42 )) & Tasiyicilar(2*i) >= (2/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 0 0 1 0 1];
    elseif(Tasiyicilar(2*i) < (2/sqrt( 42 )) & Tasiyicilar(2*i) >= 0)
      sys((i*6-5):(i*6)) = [0 0 0 1 0 0];
    elseif(Tasiyicilar(2*i) < 0 & Tasiyicilar(2*i) >= (-2/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 1 0 1 0 0];
    elseif(Tasiyicilar(2*i) < (-2/sqrt( 42 )) & Tasiyicilar(2*i)>= (-4/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 1 0 1 0 1];
    elseif(Tasiyicilar(2*i) < (-4/sqrt( 42 )) & Tasiyicilar(2*i)>= (-6/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 1 0 0 0 1];
    elseif(Tasiyicilar(2*i) < (-6/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 1 0 0 0 0];
    end

  elseif(Tasiyicilar(2*i-1) < (6/sqrt( 42 )) & Tasiyicilar(2*i-1) >= (4/sqrt( 42 )))

    if(Tasiyicilar(2*i) >= (6/sqrt( 42 ))) %sanal kisim
      sys((i*6-5):(i*6)) = [0 0 0 0 1 0];
    elseif(Tasiyicilar(2*i) < (6/sqrt( 42 )) & Tasiyicilar(2*i) >= (4/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 0 0 0 1 1];
    elseif(Tasiyicilar(2*i) < (4/sqrt( 42 )) & Tasiyicilar(2*i) >= (2/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 0 0 1 1 1];
    elseif(Tasiyicilar(2*i) < (2/sqrt( 42 )) & Tasiyicilar(2*i) >= 0)
      sys((i*6-5):(i*6)) = [0 0 0 1 1 0];
    elseif(Tasiyicilar(2*i) < 0 & Tasiyicilar(2*i) >= (-2/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 1 0 1 1 0];
    elseif(Tasiyicilar(2*i) < (-2/sqrt( 42 )) & Tasiyicilar(2*i)>= (-4/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 1 0 1 1 1];
    elseif(Tasiyicilar(2*i) < (-4/sqrt( 42 )) & Tasiyicilar(2*i)>= (-6/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 1 0 0 1 1];
    elseif(Tasiyicilar(2*i) < (-6/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 1 0 0 1 0];
    end

  elseif(Tasiyicilar(2*i-1) < (4/sqrt( 42 )) & Tasiyicilar(2*i-1) >= (2/sqrt( 42 )))

    if(Tasiyicilar(2*i) >= (6/sqrt( 42 ))) %sanal kisim
      sys((i*6-5):(i*6)) = [0 0 1 0 1 0];
    elseif(Tasiyicilar(2*i) < (6/sqrt( 42 )) & Tasiyicilar(2*i) >= (4/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 0 1 0 1 1];
    elseif(Tasiyicilar(2*i) < (4/sqrt( 42 )) & Tasiyicilar(2*i) >= (2/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 0 1 1 1 1];
    elseif(Tasiyicilar(2*i) < (2/sqrt( 42 )) & Tasiyicilar(2*i) >= 0)
      sys((i*6-5):(i*6)) = [0 0 1 1 1 0];
    elseif(Tasiyicilar(2*i) < 0 & Tasiyicilar(2*i) >= (-2/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 1 1 1 1 0];
    elseif(Tasiyicilar(2*i) < (-2/sqrt( 42 )) & Tasiyicilar(2*i)>= (-4/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 1 1 1 1 1];
    elseif(Tasiyicilar(2*i) < (-4/sqrt( 42 )) & Tasiyicilar(2*i)>= (-6/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 1 1 0 1 1];
    elseif(Tasiyicilar(2*i) < (-6/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 1 1 0 1 0];
    end
  elseif(Tasiyicilar(2*i-1) < (2/sqrt( 42 )) & Tasiyicilar(2*i-1) >= 0)
    if(Tasiyicilar(2*i) >= (6/sqrt( 42 ))) %sanal kisim
```

```
      sys((i*6-5):(i*6)) = [0 0 1 0 0 0];
    elseif(Tasiyicilar(2*i) < (6/sqrt( 42 )) & Tasiyicilar(2*i) >= (4/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 0 1 0 0 1];
    elseif(Tasiyicilar(2*i) < (4/sqrt( 42 )) & Tasiyicilar(2*i) >= (2/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 0 1 1 0 1];
    elseif(Tasiyicilar(2*i) < (2/sqrt( 42 )) & Tasiyicilar(2*i) >= 0)
      sys((i*6-5):(i*6)) = [0 0 1 1 0 0];
    elseif(Tasiyicilar(2*i) < 0 & Tasiyicilar(2*i) >= (-2/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 1 1 1 0 0];
    elseif(Tasiyicilar(2*i) < (-2/sqrt( 42 )) & Tasiyicilar(2*i)>= (-4/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 1 1 1 0 1];
    elseif(Tasiyicilar(2*i) < (-4/sqrt( 42 )) & Tasiyicilar(2*i)>= (-6/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 1 1 0 0 1];
    elseif(Tasiyicilar(2*i) < (-6/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [0 1 1 0 0 0];
    end
  elseif(Tasiyicilar(2*i-1) < 0 & Tasiyicilar(2*i-1) >= (-2/sqrt( 42 )))
    if(Tasiyicilar(2*i) >= (6/sqrt( 42 ))) %sanal kisim
      sys((i*6-5):(i*6)) = [1 0 1 0 0 0];
    elseif(Tasiyicilar(2*i) < (6/sqrt( 42 )) & Tasiyicilar(2*i) >= (4/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 0 1 0 0 1];
    elseif(Tasiyicilar(2*i) < (4/sqrt( 42 )) & Tasiyicilar(2*i) >= (2/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 0 1 1 0 1];
    elseif(Tasiyicilar(2*i) < (2/sqrt( 42 )) & Tasiyicilar(2*i) >= 0)
      sys((i*6-5):(i*6)) = [1 0 1 1 0 0];
    elseif(Tasiyicilar(2*i) < 0 & Tasiyicilar(2*i) >= (-2/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 1 1 1 0 0];
    elseif(Tasiyicilar(2*i) < (-2/sqrt( 42 )) & Tasiyicilar(2*i)>= (-4/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 1 1 1 0 1];
    elseif(Tasiyicilar(2*i) < (-4/sqrt( 42 )) & Tasiyicilar(2*i)>= (-6/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 1 1 0 0 1];
    elseif(Tasiyicilar(2*i) < (-6/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 1 1 0 0 0];
    end
  elseif(Tasiyicilar(2*i-1) < (-2/sqrt( 42 )) & Tasiyicilar(2*i-1)>= (-4/sqrt( 42 )))
    if(Tasiyicilar(2*i) >= (6/sqrt( 42 ))) %sanal kisim
      sys((i*6-5):(i*6)) = [1 0 1 0 1 0];
    elseif(Tasiyicilar(2*i) < (6/sqrt( 42 )) & Tasiyicilar(2*i) >= (4/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 0 1 0 1 1];
    elseif(Tasiyicilar(2*i) < (4/sqrt( 42 )) & Tasiyicilar(2*i) >= (2/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 0 1 1 1 1];
    elseif(Tasiyicilar(2*i) < (2/sqrt( 42 )) & Tasiyicilar(2*i) >= 0)
      sys((i*6-5):(i*6)) = [1 0 1 1 1 0];
    elseif(Tasiyicilar(2*i) < 0 & Tasiyicilar(2*i) >= (-2/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 1 1 1 1 0];
    elseif(Tasiyicilar(2*i) < (-2/sqrt( 42 )) & Tasiyicilar(2*i)>= (-4/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 1 1 1 1 1];
    elseif(Tasiyicilar(2*i) < (-4/sqrt( 42 )) & Tasiyicilar(2*i)>= (-6/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 1 1 0 1 1];
    elseif(Tasiyicilar(2*i) < (-6/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 1 1 0 1 0];
    end
  elseif(Tasiyicilar(2*i-1) < (-4/sqrt( 42 )) & Tasiyicilar(2*i-1)>= (-6/sqrt( 42 )))
    if(Tasiyicilar(2*i) >= (6/sqrt( 42 ))) %sanal kisim
      sys((i*6-5):(i*6)) = [1 0 0 0 1 0];
```

```
    elseif(Tasiyicilar(2*i) < (6/sqrt( 42 )) & Tasiyicilar(2*i) >= (4/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 0 0 0 1 1];
    elseif(Tasiyicilar(2*i) < (4/sqrt( 42 )) & Tasiyicilar(2*i) >= (2/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 0 0 1 1 1];
    elseif(Tasiyicilar(2*i) < (2/sqrt( 42 )) & Tasiyicilar(2*i) >= 0)
      sys((i*6-5):(i*6)) = [1 0 0 1 1 0];
    elseif(Tasiyicilar(2*i) < 0 & Tasiyicilar(2*i) >= (-2/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 1 0 1 1 0];
    elseif(Tasiyicilar(2*i) < (-2/sqrt( 42 )) & Tasiyicilar(2*i)>= (-4/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 1 0 1 1 1];
    elseif(Tasiyicilar(2*i) < (-4/sqrt( 42 )) & Tasiyicilar(2*i)>= (-6/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 1 0 0 1 1];
    elseif(Tasiyicilar(2*i) < (-6/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 1 0 0 1 0];
    end
  elseif(Tasiyicilar(2*i-1) < (-6/sqrt( 42 )))
    if(Tasiyicilar(2*i) >= (6/sqrt( 42 ))) %sanal kisim
      sys((i*6-5):(i*6)) = [1 0 0 0 0 0];
    elseif(Tasiyicilar(2*i) < (6/sqrt( 42 )) & Tasiyicilar(2*i) >= (4/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 0 0 0 0 1];
    elseif(Tasiyicilar(2*i) < (4/sqrt( 42 )) & Tasiyicilar(2*i) >= (2/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 0 0 1 0 1];
    elseif(Tasiyicilar(2*i) < (2/sqrt( 42 )) & Tasiyicilar(2*i) >= 0)
      sys((i*6-5):(i*6)) = [1 0 0 1 0 0];
    elseif(Tasiyicilar(2*i) < 0 & Tasiyicilar(2*i) >= (-2/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 1 0 1 0 0];
    elseif(Tasiyicilar(2*i) < (-2/sqrt( 42 )) & Tasiyicilar(2*i)>= (-4/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 1 0 1 0 1];
    elseif(Tasiyicilar(2*i) < (-4/sqrt( 42 )) & Tasiyicilar(2*i)>= (-6/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 1 0 0 0 1];
    elseif(Tasiyicilar(2*i) < (-6/sqrt( 42 )))
      sys((i*6-5):(i*6)) = [1 1 0 0 0 0];
    end
  end
end

% end mdlOutputs
%=============================================================
% mdlGetTimeOfNextVarHit
%=============================================================
function sys=mdlGetTimeOfNextVarHit(t,x,u)

sampleTime = 1;
sys = t + sampleTime;
% end mdlGetTimeOfNextVarHit
%=============================================================
% mdlTerminate
%=============================================================
function sys=mdlTerminate(t,x,u)
sys = [];
% end mdlTerminate
```

Ek E

SİMGE TERS-SERPİŞTİRİCİ S-İŞLEVİ

```
function [sys,x0,str,ts] = symbol_deinterleaver(t,x,u,flag)
% Ic Ters-Dizileyici

switch flag,

 %%%%%%%%%%%%%%%%%%
 % Initialization %
 %%%%%%%%%%%%%%%%%%
 case 0,
   [sys,x0,str,ts]=mdlInitializeSizes;

 %%%%%%%%%%%%%%%%
 % Derivatives %
 %%%%%%%%%%%%%%%%
 case 1,
  sys=mdlDerivatives(t,x,u);

 %%%%%%%%%%
 % Update %
 %%%%%%%%%%
 case 2,
  sys=mdlUpdate(t,x,u);

 %%%%%%%%%%
 % Outputs %
 %%%%%%%%%%
 case 3,
  sys=mdlOutputs(t,x,u);

 %%%%%%%%%%%%%%%%%%%%%%%%
 % GetTimeOfNextVarHit %
 %%%%%%%%%%%%%%%%%%%%%%%%
 case 4,
  sys=mdlGetTimeOfNextVarHit(t,x,u);

 %%%%%%%%%%%%
 % Terminate %
 %%%%%%%%%%%%
 case 9,
  sys=mdlTerminate(t,x,u);

 %%%%%%%%%%%%%%%%%%%%
 % Unexpected flags %
 %%%%%%%%%%%%%%%%%%%%
 otherwise
```

```
   error(['Unhandled flag = ',num2str(flag)]);
end
%=============================================================
% mdlInitializeSizes
%=============================================================
function [sys,x0,str,ts]=mdlInitializeSizes

sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1512*6; % 1512 adet sembol = 9072 adet bit
sizes.NumInputs      = -1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
double(sys);

% initialize the initial conditions

x0  = [];
str = [];
ts  = [-1 0];
% end mdlInitializeSizes

%=============================================================
% mdlDerivatives
%=============================================================
function sys=mdlDerivatives(t,x,u)

sys = [];

% end mdlDerivatives
%=============================================================
% mdlUpdate
%=============================================================
function sys=mdlUpdate(t,x,u)

sys = [];
% end mdlUpdate
%=============================================================
% mdlOutputs
%=============================================================
function sys=mdlOutputs(t,x,u)

DosyaHandle = fopen('SymbNo.txt','r');
SembolNo = fread(DosyaHandle,1,'integer*1');
fclose(DosyaHandle);

DosyaHandle = fopen('hq.txt','r');
hq = fread(DosyaHandle,'integer*2');
fclose(DosyaHandle);

SembolNo = SembolNo - 1;
```

```
disp('deinterleaver SembolNo');
disp(SembolNo);

for symbno = 0:1511,
      if mod(SembolNo,2) == 1
        for r = 1:6
           sys(6*hq(symbno+1) + r) = u(6*symbno + r);
       end;
      end

      if mod(SembolNo,2) == 0
        for r = 1:6
          sys(6*symbno + r) = u(6*hq(symbno+1) + r);
       end;
      end
end;
% end mdlOutputs
%=============================================================
% mdlGetTimeOfNextVarHit
%=============================================================
function sys=mdlGetTimeOfNextVarHit(t,x,u)

sampleTime = 1;
sys = t + sampleTime;

% end mdlGetTimeOfNextVarHit
%=============================================================
% mdlTerminate
%=============================================================
function sys=mdlTerminate(t,x,u)
sys = [];
% end mdlTerminate
```

Ek F

AWGN KANAL BLOĞUNDA KULLANILAN S-İŞLEVİ

```
/*
* SCOMAWGNCHAN Communications Toolbox S-Function for additive white
* gaussian noise channel. Used in combination with Simulink
* built-in gaussian noise source.
*
* Modified by: Trefor Delve, May, December 1998
*                    A.Koray Yalcin , April 2001
* Original author: Mike Mulligan
*  Copyright (c) 1996-99 by The MathWorks, Inc.
*/

#define S_FUNCTION_NAME scomawgnchan
#define S_FUNCTION_LEVEL 2

#define DATA_TYPES_IMPLEMENTED

#include "simstruc.h"
#include <math.h>

enum {INPORT_SIGNAL=0, INPORT_NOISE, INPORT_VARIANCE,
NUM_INPORTS};
enum {OUTPORT_SIGNAL=0, NUM_OUTPORTS};

enum {MODEC=0, ESNO_DBC,SIGNAL_POWERC, SYMBOL_PERIODC,
FIELD_VARIANCEC, NUM_ARGS};
enum {STD = 0, NUM_DWORK};

#define MODE                    (ssGetSFcnParam(S,MODEC))
#define ESNO_DB                 (ssGetSFcnParam(S,ESNO_DBC))
#define SIGNAL_POWER    (ssGetSFcnParam(S,SIGNAL_POWERC))
#define SYMBOL_PERIOD   (ssGetSFcnParam(S,SYMBOL_PERIODC))
#define FIELD_VARIANCE  (ssGetSFcnParam(S,FIELD_VARIANCEC))

/* --- Function prototypes --- */
static void setNoiseStdDev(SimStruct *S);


/* Function: mdlCheckParameters  */
#ifdef MATLAB_MEX_FILE
#define MDL_CHECK_PARAMETERS
static void mdlCheckParameters(SimStruct *S)
{
   static char_T err_msg[256];

   /* --- Check the mode --- */
```

```
  if ( ((int)mxGetPr(MODE)[0] < 0) || ((int)mxGetPr(MODE)[0] > 2))
  {
    sprintf(err_msg, "Mode input is outside of expected range");
    ssSetErrorStatus(S,err_msg);
    return;
  }

        /* --- Check the parameters required for the Es/No mode --- */
        if((int)mxGetPr(MODE)[0] == 0)
        {

                /* --- Check the ESNO_DB field --- */
                if (mxGetN(ESNO_DB)*mxGetM(ESNO_DB) < 1)
                {
                  sprintf(err_msg, "Es/No parameter not defined");
                  ssSetErrorStatus(S,err_msg);
                  return;
                }

                if ((mxGetN(ESNO_DB) > 1) || (mxGetM(ESNO_DB) > 1) ||
mxIsComplex(ESNO_DB))
                {
                  sprintf(err_msg, "Es/No parameter may only be a scalar.");
                  ssSetErrorStatus(S,err_msg);
                  return;
                }

                /* --- Check the SIGNAL_POWER field --- */
                if (mxGetN(SIGNAL_POWER)*mxGetM(SIGNAL_POWER) < 1)
                {
                  sprintf(err_msg, "Signal power parameter not defined");
                  ssSetErrorStatus(S,err_msg);
                  return;
                }

                if ((mxGetN(SIGNAL_POWER) > 1) || (mxGetM(SIGNAL_POWER) >
1) || mxIsComplex(SIGNAL_POWER) || (mxGetPr(SIGNAL_POWER)[0] < 0.0))
                {
                  sprintf(err_msg, "Signal power parameter may only be a real
positive scalar.");
                  ssSetErrorStatus(S,err_msg);
                  return;
                }

                /* --- Check the SYMBOL_PERIOD field --- */
                if (mxGetN(SYMBOL_PERIOD)*mxGetM(SYMBOL_PERIOD) < 1)
                {
                  sprintf(err_msg, "Symbol period parameter not defined");
                  ssSetErrorStatus(S,err_msg);
                  return;
                }
```

```
                if ((mxGetN(SYMBOL_PERIOD) > 1) || (mxGetM(SYMBOL_PERIOD)
> 1) || mxIsComplex(SYMBOL_PERIOD) || (mxGetPr(SYMBOL_PERIOD)[0] <=
0.0))
                {
                  sprintf(err_msg, "Symbol period parameter may only be a real
positive scalar.");
                  ssSetErrorStatus(S,err_msg);
                  return;
                }

                /* --- End of if((int)mxGetPr(MODE)[0] == 0) --- */
        }
        else if((int)mxGetPr(MODE)[0] == 1)
        {
                /* --- Check the parameters required for the Variance mode (variance
from the Mask) --- */
                if (mxGetN(FIELD_VARIANCE)*mxGetM(FIELD_VARIANCE) < 1)
                {
                  sprintf(err_msg, "Variance parameter not defined");
                  ssSetErrorStatus(S,err_msg);
                  return;
                }

                if ((mxGetN(FIELD_VARIANCE) > 1) || (mxGetM(FIELD_VARIANCE)
> 1) || mxIsComplex(FIELD_VARIANCE) || (mxGetPr(FIELD_VARIANCE)[0] < 0.0))
                {
                  sprintf(err_msg, "Variance parameter may only be a real, positive
scalar.");
                  ssSetErrorStatus(S,err_msg);
                  return;
                }
                /* --- End of else if((int)mxGetPr(MODE)[0] == 1) */
        }
        else if((int)mxGetPr(MODE)[0] == 2)
        {
                /* --- No check required when the variance is supplied from the port---
*/

                /* --- End of else if((int)mxGetPr(MODE)[0] == 2) */
        }
        else
        {
                sprintf(err_msg, "Unrecognized mode");
                ssSetErrorStatus(S,err_msg);
                return;
        }

} /* end mdlCheckParameters */
#endif



/* Function: mdlInitializeSizes */
static void mdlInitializeSizes(SimStruct *S)
  {
```

```
        ssSetNumSFcnParams(S, NUM_ARGS);

#if defined(MATLAB_MEX_FILE)
   if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) return;
   mdlCheckParameters(S);
   if (ssGetErrorStatus(S) != NULL) return;
#endif

   if (!ssSetNumInputPorts(S, NUM_INPORTS)) return;

/*   ssSetInputPortWidth(          S, INPORT_SIGNAL, 1);*/
   ssSetInputPortWidth(          S, INPORT_SIGNAL, DYNAMICALLY_SIZED);
   ssSetInputPortDirectFeedThrough(S, INPORT_SIGNAL, 1);
   ssSetInputPortComplexSignal(   S, INPORT_SIGNAL, COMPLEX_INHERITED);
        ssSetInputPortReusable(               S, INPORT_SIGNAL, 0);


/*   ssSetInputPortWidth(          S, INPORT_NOISE, 2);*/
   ssSetInputPortWidth(          S, INPORT_NOISE, 4224);
   ssSetInputPortDirectFeedThrough(S, INPORT_NOISE, 1);
   ssSetInputPortComplexSignal(   S, INPORT_NOISE, COMPLEX_NO);
        ssSetInputPortReusable(               S, INPORT_NOISE, 0);

   ssSetInputPortWidth(          S, INPORT_VARIANCE, 1);
   ssSetInputPortDirectFeedThrough(S, INPORT_VARIANCE, 1);
   ssSetInputPortComplexSignal(   S, INPORT_VARIANCE, COMPLEX_NO);
        ssSetInputPortReusable(               S, INPORT_VARIANCE, 0);


   /* Outputs: */
   if (!ssSetNumOutputPorts(S,1)) return;

/*   ssSetOutputPortWidth(       S, OUTPORT_SIGNAL, 1);*/
   ssSetOutputPortWidth(       S, OUTPORT_SIGNAL, DYNAMICALLY_SIZED);
   ssSetOutputPortComplexSignal(S, OUTPORT_SIGNAL,
COMPLEX_INHERITED);
        ssSetOutputPortReusable(    S, OUTPORT_SIGNAL, 0);

        if (!ssSetNumDWork(S, DYNAMICALLY_SIZED)) return;

} /* End of mdlInitializeSizes(SimStruct *S) */


/* Function: mdlInitializeSampleTimes  */
static void mdlInitializeSampleTimes(SimStruct *S)
{
   ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
   ssSetOffsetTime(S, 0, FIXED_IN_MINOR_STEP_OFFSET);
}


/* Function: mdlInitializeConditions */
#define MDL_INITIALIZE_CONDITIONS
static void mdlInitializeConditions(SimStruct *S)
```

```
{
        /* --- Evaluate the parameters --- */
        setNoiseStdDev(S);
}



/* Function: mdlProcessParameters */
#define MDL_PROCESS_PARAMETERS   /* Change to #undef to remove function
*/
#if defined(MDL_PROCESS_PARAMETERS) && defined(MATLAB_MEX_FILE)
static void mdlProcessParameters(SimStruct *S)
{
        /* --- Evaluate the parameters --- */
        setNoiseStdDev(S);
}
#endif /* MDL_PROCESS_PARAMETERS */




/* Function: setNoiseStdDev */
static void setNoiseStdDev(SimStruct *S)
{
        /* --- Re-evaluate the std based on the new parameters --- */
        real_T Ts    = ssGetSampleTime(S,0);
        real_T *std  = (real_T *)ssGetDWork(S, STD);
   char_T  err_msg[256];

        switch((int)mxGetPr(MODE)[0])
        {
        case 0:
                /* --- Es/No mode chosen --- */
                *std =
sqrt((mxGetPr(SIGNAL_POWER)[0]*mxGetPr(SYMBOL_PERIOD)[0])/(Ts*pow(10.0
,(mxGetPr(ESNO_DB)[0]/10.0))));
                break;
        case 1:
                /* --- Variance is to be taken from the field --- */
                *std  = sqrt(mxGetPr(FIELD_VARIANCE)[0]);
                break;
        case 2:
                /* --- Variance is to be taken from the port --- */
                *std  = 0.0;
                break;
        default:
                sprintf(err_msg, "Unrecognized mode");
                ssSetErrorStatus(S,err_msg);
                return;
                break;
        }
}       /* end of setNoiseStdDev(SimStruct *S) */
```

```
/* Function: mdlStart */
#define MDL_START
static void mdlStart (SimStruct *S)
{

        /* ---    Check that if the Es/No mode is selected, the sample
                        time is not continuous
        */
#ifdef MATLAB_MEX_FILE
        if ( ((int)mxGetPr(MODE)[0] == 0) && (ssGetSampleTime(S, 0) ==
CONTINUOUS_SAMPLE_TIME) )
        {
           ssSetErrorStatus(S,"In Es/No mode, the block input and output must have
discrete sample times");
                return;
        }
#endif
}


/* Function: mdlOutputs */
static void mdlOutputs(SimStruct *S, int_T tid)
{
      int aky;
      const boolean_T    c0      = (ssGetInputPortComplexSignal(S,
INPORT_SIGNAL) == COMPLEX_YES);
        const real_T workStd  = *(real_T *)ssGetDWork(S, STD);
/* KoRaY */
      const int_T girisUzunlugu = ssGetInputPortWidth(S, INPORT_SIGNAL);
/* KoRaY */

        static char_T err_msg[256];
   InputRealPtrsType  noise   = ssGetInputPortRealSignalPtrs(S, INPORT_NOISE);
   InputRealPtrsType  var     = ssGetInputPortRealSignalPtrs(S,
INPORT_VARIANCE);

        real_T          std     = 0.0;


        switch((int)mxGetPr(MODE)[0])
        {
        case 0:
        case 1:
                /* --- Es/No or Variance from mask mode chosen --- */
                std  = workStd;
                break;

        case 2:

                /* --- Variance from port chosen --- */
                if ( **var < 0.0)
                {
                        sprintf(err_msg, "Variance must be >= 0.0");
                        ssSetErrorStatus(S,err_msg);
                        return;
```

```
                }

                std = sqrt(**var);
                break;

        default:
                sprintf(err_msg, "Unrecognized mode");
                ssSetErrorStatus(S,err_msg);
                return;
                break;

        }

   if (!c0) {
      /*
      * Real Gaussian Noise Channel
      */
      InputRealPtrsType  uptr = ssGetInputPortRealSignalPtrs(S,
INPORT_SIGNAL);
      real_T            *y    = ssGetOutputPortRealSignal(S, OUTPORT_SIGNAL);

      *y = **uptr + std*(**noise);

   } else {
      /*
       * Complex Gaussian Noise Channel
       */
/*       InputPtrsType      uptr = ssGetInputPortSignalPtrs(S, INPORT_SIGNAL);
      creal_T           *y    = (creal_T *)ssGetOutputPortSignal(S,
OUTPORT_SIGNAL);
      const creal_T     *u    = (creal_T *)*uptr;

                std = std/sqrt(2.0);

      y->re = u->re + std*(**noise++);
      y->im = u->im + std*(**noise);
*/
      InputPtrsType      uptr = ssGetInputPortSignalPtrs(S, INPORT_SIGNAL);
      creal_T           *y    = (creal_T *)ssGetOutputPortSignal(S,
OUTPORT_SIGNAL);
      std = std/sqrt(2.0);
/********************* eklenen kisim *********************/
      for(aky=0;aky<girisUzunlugu;aky++)
      {
            const creal_T     *u    = (creal_T *)(*uptr++);

            y->re = u->re + std*(**noise++);
            y->im = u->im + std*(**noise);

            y++;
            noise++;
      }
/*********************************************************/
   }
}
```

```
/* Function: mdlSetWorkWidths */

#ifdef MATLAB_MEX_FILE
#define MDL_SET_WORK_WIDTHS
static void mdlSetWorkWidths(SimStruct *S)
{

   ssSetNumDWork(S, NUM_DWORK);

   ssSetDWorkWidth(       S, STD, 1);
   ssSetDWorkDataType(    S, STD, SS_DOUBLE);
   ssSetDWorkComplexSignal(S, STD, COMPLEX_NO);

}
#endif


static void mdlTerminate(SimStruct *S)
{
}

#ifdef  MATLAB_MEX_FILE
#define MDL_SET_INPUT_PORT_COMPLEX_SIGNAL
static void mdlSetInputPortComplexSignal(SimStruct *S, int_T portIdx,
                                         int_T iPortComplexSignal)
{
   if (portIdx != 0) {
      ssSetErrorStatus(S,"Invalid port index for complex propagation.");
      return;
   }
   ssSetInputPortComplexSignal(S, portIdx, iPortComplexSignal);
   ssSetOutputPortComplexSignal(S, portIdx, iPortComplexSignal);

}

#define MDL_SET_OUTPUT_PORT_COMPLEX_SIGNAL
static void mdlSetOutputPortComplexSignal(SimStruct *S, int_T portIdx,
                                          int_T oPortComplexSignal)
{
   if (portIdx != 0) {
      ssSetErrorStatus(S,"Invalid port index for complex propagation.");
      return;
   }
   ssSetOutputPortComplexSignal(S, portIdx, oPortComplexSignal);
   ssSetInputPortComplexSignal(S, portIdx, oPortComplexSignal);

}
#endif

#ifdef  MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

ÖZGEÇMİŞ

A. Koray Yalçın 8 Temmuz 1975 tarihinde Ankara'da doğmuştur. 1992 yılında İstanbul Bahçelievler Lisesi'nden mezun oldukta sonra İstanbul Teknik Üniversitesi (İ.T.Ü.), Elektrik - Elektronik Fakültesi, Elektronik ve Haberleşme mühendisliği bölümünde yüksek öğrenimine başlamıştır. 1997 yılında mezun olduktan sonra, İ.T.Ü Elektronik ve Haberleşme mühendisliği bölümünde yüksek lisans eğitimine başlamıştır. Aynı yıl İ.T.Ü. Çoğulortam Merkezi'nde araştırma görevlisi olarak göreve başlamıştır. 1998 yılında BEKO Elektronik firması için KOSGEB Vakfı-ETA bünyesinde başlayan QPSK ve QAM STB alıcı ön katı tasarımı projesine katılmıştır. Aralık 1999 tarihinden itibaren Star Digital firmasında tesis şefi olarak çalışmaya başlamıştır ve hala bu görevini sürdürmektedir.