HATA SEZME VE DÜZELTME TEKNİKLERİ

4.Hata Sezme ve düzeltme

- Veri paketleri iletilirken bazı bitleri bozulabilir. Bu olasılık çoğu uygulamalar (text v.b) tarafından kabul edilmez. Paketin içerisindeki 1 bit bozulsa dahi, tüm verinin yanlış anlaşılmasına neden olabilir. Bunun için iletişim yapılırken bozulma olup olmadığının anlaşılması için hata sezme teknikleri ve düzeltme teknikleri kullanılır. Alıcı tarafta düzeltme işlemi başarılamazsa, paketin yeniden gönderilmesi kaçınılmazdır.

  Veri iletiminde genel olarak iki tip hata olabilir.

1- Patlama hatası (burst error): Çevre koşulları nedeniyle bir süre alıcıya gerçek olmayan anlamsız bilgiler gelir. En az 2 veya daha fazla bit bozulabilir. Bu süre 1-100 ms arası olabilir.(yıldırım v.s)

2- Rasgele hata (random errror): İletim yolundaki elektriksel gürültü nedeniyle bilgi katarı içinde rastgele bir bitin bozulması söz konusudur.

İletim hatalarının sebepleri 3 ana kategoride incelenebilir

- **Girişim (Interference):**

– ortamdaki elektromanyetik radyasyon (elektrik motorları v.b)
– Kablolardan iletilen sinyallerin ve radyo iletiminin bozulmasına sebep olan gürültü ye sebep olur

- **(Bozulma) Distortion:**

– Bütün fiziksel sistemler (medya) sinyalleri bozarlar.
– Bir fiber optik boyunca bir sinyal hareket ederken, sinyal dağılır(disperse)
– Kablolar capacitance (kapasitans) ve inductance (indüktans) özelliklerine sahiptir

- Diğer frekanslardaki sinyallerin girmesine izin verirken, bazı frekanslardaki sinyalleri bloke eder.

– Bir kabloyu büyük bir metal nesnenin yanına yerleştirmek, kablonun içinden geçen sinyallerin frekansını değiştirebilir.
– Metal objeler radyo dalgalarının frekansını bloke edebilir.

- **(Zayıflama)Attenuation :**

– Bir sinyal ortamlar arasından geçerken, sinyal zayıflar.

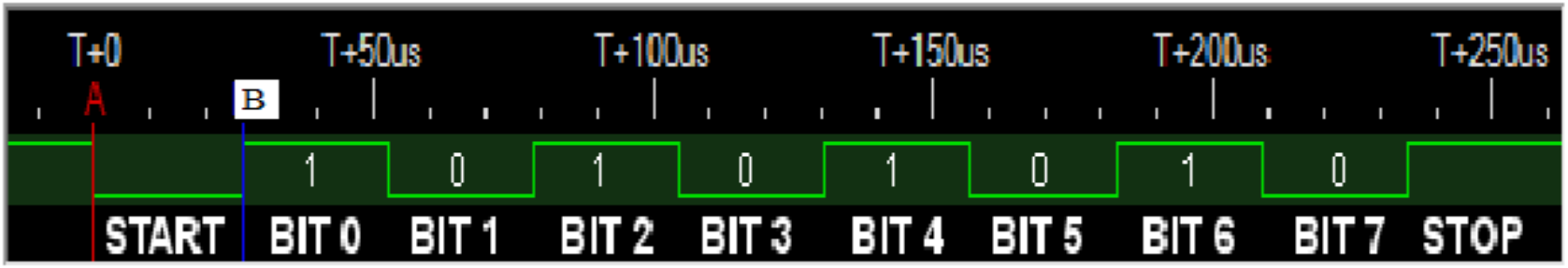
- Kablo veya fiber optiklerin üzerindeki sinyaller uzun mesafelerde zayıflar, bir radyo sinyali uzaklıkla birlikte zayıflar.

Veri üzerine iletim hatalarının etkisi

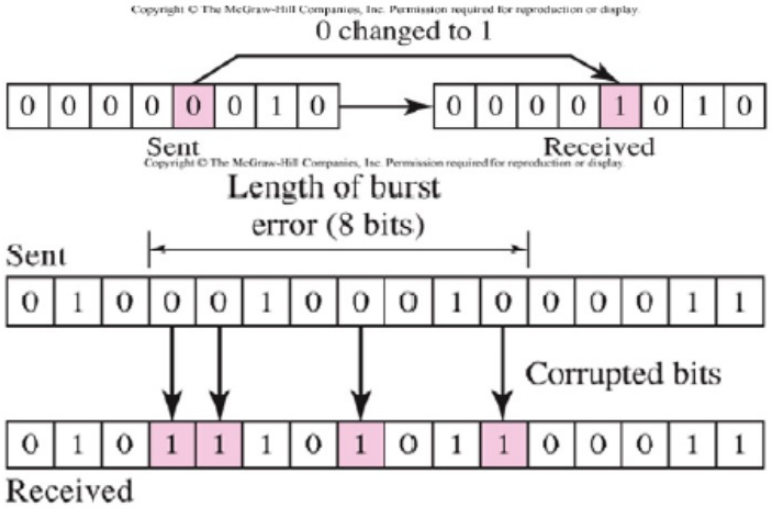
- Tabloda iletim hatalarının veriye olan etkisi gösterilmiştir.
  - Örnek olarak; çok kısa süren girişim, spike olarak adlandırılır, tek bitlik hataya sebep olur
  - Uzun süren girişim veya bozulma burst (birden fazla bitlik) hataları üretebilir
  - Bazen bir sinyal açık olarak 1 veya 0 seviyesinde değildir,fakat belirsiz bir alana düşer ki bu erasure (silinmiş yer) olarak adlandırılır.

Type Of Error	Description
Single Bit Error	A single bit in a block of bits is changed and all other bits in the block are unchanged (often results from very short-duration interference)
Burst Error	Multiple bits in a block of bits are changed (often results from longer-duration interference)
Erasure (Ambiguity) Silme (Belirsizlik)	The signal that arrives at a receiver is ambiguous (does not clearly correspond to either a logical 1 or a logical 0 (can result from distortion or interference)

Hatırlatma: Her bit iletim (transmision) hızına bağlı olarak belli bir süre için aynı konumda kalmalıdır. Başlangıç bitinden sonra veri bitleri gelir. Veri biti sırası en az anlamlı bitten(LSB) en anlamlı bite(MSB) doğrudur. Bu örnekte ikili kod ”01010101” olduğuna göre ilk önce 1 biti iletilmelidir. Örnekte 1 bitlik süre (A ve B arasındaki süre) 25 mikrosaniye olursa bu iletişim hattının hızı 40.000 bps olur !!!!!!!.



- 1kbps hızında iletişim yapılırken 10 msn noise oluşursa, 10 bit bozulur.
- 1Mbps hızında iletişim yapılırsa toplam 10.000 bit bozulur.



Single bit error

Burst errors

Hata bulma (sezme) olasılıkları

Pb : İletişim hattından İletilen tek bitlik bir bilginin, belirli bir zaman aralığındaki bozulma olasılığıdır.

Bit hata oranı (Bit Error Rate – BER) olarak da bilinir.

P1 : Bir çerçevenin hatasız olarak alınabilme olasılığı.

P2 : Hata bulma protokolu kullanılarak, bir çerçevenin bir yada daha çok bulunamamıs hatalı bitle alınma olasılığı.

P3 : Hata bulma protokolu kullanılarak, bir frame'in bir ya da daha çok sayıda bulunmus hatalı bitle alınma olasılığı.

Hata bulma protokolu olmadan;

$$P_1 = (1 - P_b)^F$$ Frame'in hatasız alınma olasılığı

$$P_2 = 1 - P_1$$

$$P_3 = 0$$

F: Her çerçevedeki bit sayısı

BER arttıkça frame'in hatasız alınma olasılığı düser.
Frame uzunluğu arrtıkça hatasız alınma olasılığı düşer

İletişim Ortamlarının Ortalama BER Değerleri

Probability of Single-Bit Error – bit error rate (BER):

- wireless medium: $p_b = 10^{-3}$
- copper-wire: $p_b = 10^{-6}$
- fibre optics: $p_b = 10^{-9}$

Örnek

- BER=10^{-6}
- Sürekli kullanılan 64Kbps'lik bir iletişim kanalında frame uzunluğu 1000-bit
- Kullanıcı şartı günde en fazla 1 frame hatalı olarak alınacak. Bu hat kullanılabilir mi?
- $64000*60*60*24/1000=5.529*10^6$ paket/gün
- 1 paketin hatalı olma olasılığı $P_2=1/5.529*10^6 = 0.18*10^{-6}$ (sağlamamız gereken şart)
- $P_b = 10^{-6}$ olduğuna göre $P_1= (1- P_b)^F = 0.999$
- $P_2= 1- P_1 = 10^{-3}$

Örnek

F (Çerçeve)= 500 bytes (4000 bits)

P_b = BER = 10^{-6}

$P_1 = (1- 10^{-6})^{4000} = 0.9960$ Çerçevenin hatasız alınma olasılığı

$P_2 = 1-(1- 10^{-6})^{4000} = 0.0040$ Çerçevenin hatalı alınma olasılığı

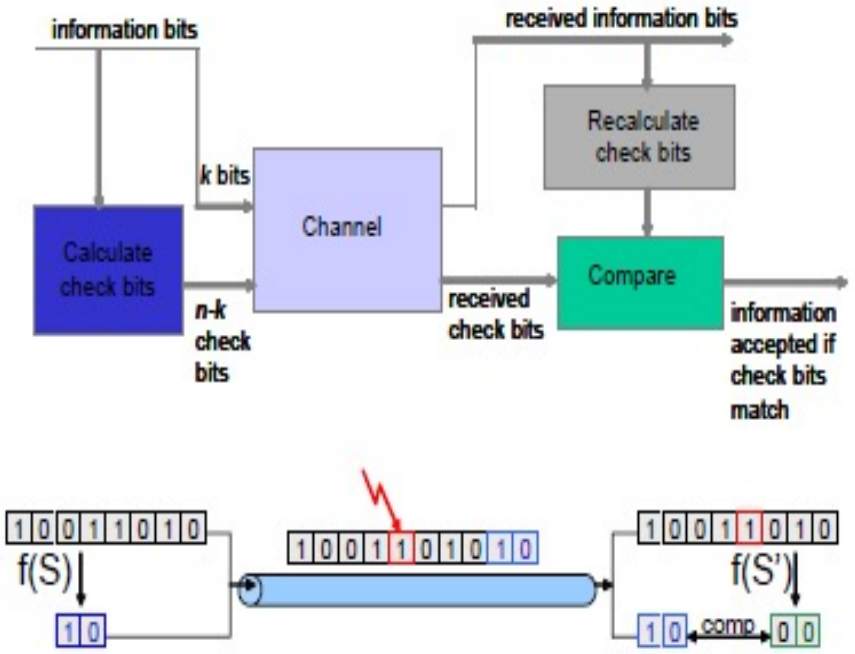
Her 1000 çerçeveden 4 tanesinin hatalı alınma olasılığı var.

100kbps taşıma hızına göre, 1 dakikada taşınan paketlerden kaç tanesinin bozuk alınma ihtimali vardır.?

Cevap 6 adet.

Hata bulma İşlemi

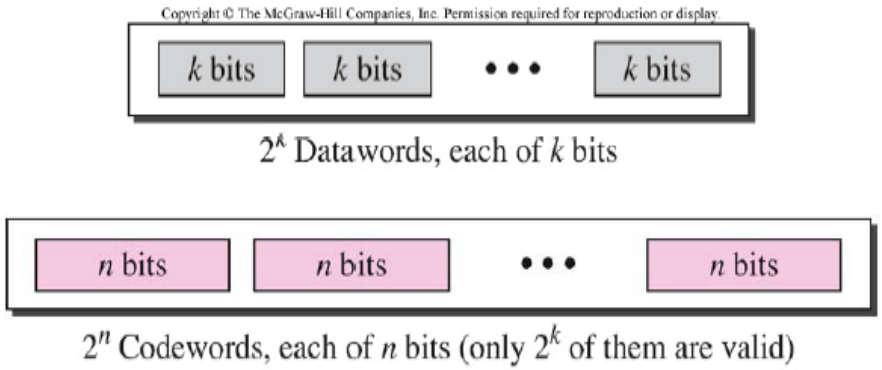
Hata denetimi için genellikle blok kodlama ve ***lineer blok kodlama*** *kullanılır. Blok kodlamada gönderilecek mesajlar* ***k-bitlik*** *veri bloklarına bölünür. Bu bloklara* ***DATAWORDs*** *denir. Herbir dataword'e* ***r*** *bitlik* ***redundant*** *(hata bulma, düzeltme için) bitleri eklenir. Bu bitler alıcıda çıkartılır. Şimdi orijinal bir veri bloğunun boyutu* ***n = k + r*** *olmuştur. Yeni bloğa* ***CODEWORDs*** *denir.*



Gönderici: Gönderilecek frame (dataword) (*k-bit) için, veri bitlerinden bir hata* bulma kodu (check bits) bitlerini (r) hesaplar. Hesaplanan bu hata bulma kodu bitlerini veri bitlerine ekler (*n = k + r) (Codewords).*

Alıcı : Hata bulma kodunu ve veri bitlerini birbirinden ayırır. Alınan veri bitlerinden hata bulma kodunu hesaplar. Hesapladığı hata bulma kodu ile alınan hata bulma kodunu karşılastırır. Tam eşleşme varsa r bitlerini atarak datawordu oluşturur. Tam eşlesme yoksa hata vardır.

Lineer blok kodlar



- Blok kodlamada mesaj *k-bit bloklara (dataword) bölünür.*
- Her bloğa *r-bit redundant bit eklenir.*
- Oluşan *n=k+r bit* ***codeword olarak adlandırılır.***
- Toplam 2^k dataword ve 2^n codeword üretilebilir. Ancak 2^k adet codeword kullanılır.

İki codeword arasındaki XOR sonucu yine geçerli bir codeword oluşturursa bu kodlar lineer blok kodlardır.

- Aşağıdaki tabloda herhangi iki codeword arasında XOR işlemi yapılırsa, yine geçerli bir codeword oluşmaktadır.

Dataword	Codeword
00	000
01	011
10	101
11	110

Dataword	Codeword
00	00000
01	01011
10	10101
11	11110

Hata Sezme ve Düzeltme

- Uygulamada 2 popüler hata denetimi stratejisi vardır.

1- FEC (Error Correction code - Forward Error Correction-İleri yönlü hata denetimi):

- Hata denetimi yapmak için gönderilecek veri kümesine ek bitler (redundant) ekler. Hatayı bulur ve gerekirse alıcıda düzeltmeye çalışır. Böylelikle bir hız kaybı olmaz. Çok gürültülü ortamlarda kullanılmaz. FEC stratejisi, yeniden iletimin çok zor veya imkansız olduğu bağlantılarda ve veri kümesinin küçük olduğu uygulamalarda kullanılır.
- FEC için en çok kullanılan birkaç önemli algoritma;

a) Katlamalı kodlar **b) BCH kodlar**

c) Hamming Kodlar **d) Reed-Salamon kodları**

2- ARQ (Automatic Repeat Request- Otomatik tekrar isteği): Hatanın sezme ve , bozulan verinin yeniden iletilmesi için alıcı taraftan istekte bulunulması (retransmisyon) işlemidir. Uygulamada bu teknik kullanılır,çünkü aynı hatayı tespit için gerekli bit sayısı, düzeltmek için kullanılan bit sayısından çok daha azdır. Önemli ARQ (hata sezme) algoritmaları;

a) CRC kodları

b)Seri Eşlik (Parity)

c)Blok Eşlik

d) Checksum (Modül Toplamı- Kontrol toplamı)

Hata Sezme kodlamaları

Eşlik (Parity) biti üretimi ve kullanılması – Parity Check

Bir çerçevedeki 1 veya birden fazla bitin bozulduğunu anlamak için gönderilecek sözcüklerden hesap edilen bit veya bitler ekleme işlemidir. Lineer blok kodlama yapısındadır.

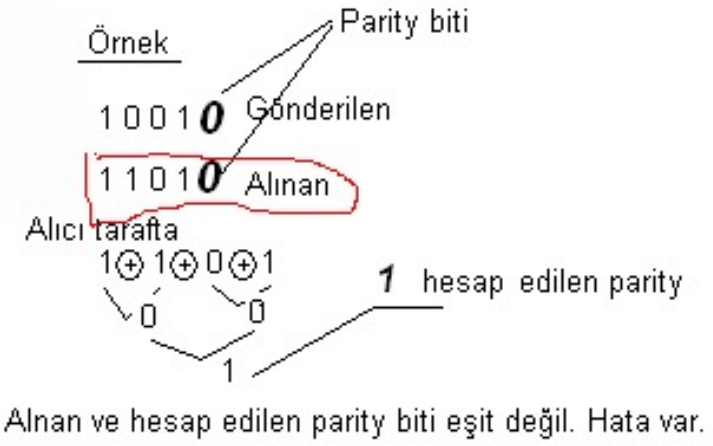
A- Sözcükte çift sayıda 1 var ise eşlik biti 0

B- Sözcükte tek sayıda 1 var ise eşlik biti 1 olarak hesap edilir.

Yanda BCD sözcüklerden elde edilen enine ve boyuna parity bitleri verilmektedir.

Parity bitlerinin donanımsal oluşturulması, sözcükteki bitlerin exor'lanması ile gerçekleştirilir.

ABCD	Çift eşlik	Tek Eşlik	
0000	0	1	
0001	1	0	
0010	1	0	
0011	0	1	Boyuna parity
0100	1	0	
0101	0	1	
0110	0	1	
0111	1	0	
1000	1	0	
1001	0	1	
Çift Eşlik 0001			
Tek Eşlik 1110			
Enine parity			

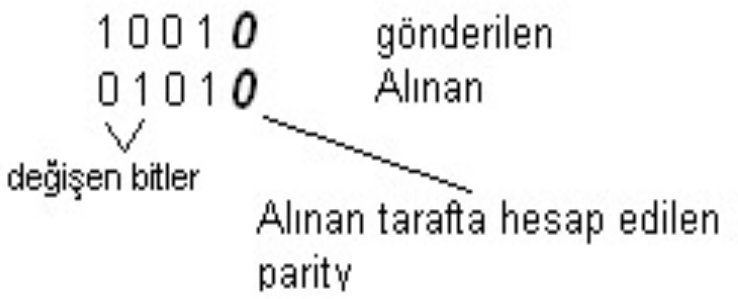


Alnan ve hesap edilen parity biti eşit değil. Hata var.

Eşlik kontrol yöntemi genellikle, küçük bit sayılı verilerde hata sezmek için kullanılır.

Parity biti oluşturma

- Tek bir parity biti, tekbir hatayı sezebilir.
- Birden fazla bit hatası, tekbir parity biti ile sezilemeyebilir.
- Yani iki adet parity biti iki adet hatayı sezebilir veya tekbir hatayı sezip düzeltebilir.
- *Tekbir eşlik bitinin kullanıldığı yerler, bir sözcükte tekbir hatanın yüksek ihtimalle kabul edilebileceği yerlerde uygulanabilir.*



Gönderieln ve hesp edilen parity biti aynı olmasına rağmen, hatayı sezemedi.

Parity biti oluşturma

- Örnek: 1975 (BCD kodunda) sayısını iletişim hattı üzerinden iletelim

P.B

1 0 0 0 1 1 (n.zaman)

9 1 0 0 1 0 (n+1.zaman)

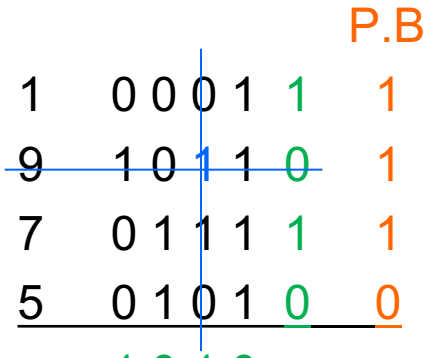
7 0 1 1 1 1 (n+2. zaman)

5 0 1 0 1 0 (n+3.zaman)

1 0 1 0 enine parity bitleri

Alınan tarafta da veri bu şekilde alınıp, hesaplanan ve alınan parityler eşit ise hata yoktur.

Hatalı durum (Alınan tarafta)



1 0 0 0 enine parity bitleri

Kırmızılar hesaplanan parity bitleri olsun. Alınan parity bitleri ile farklı, o halde hata vardır.

Hata düzeltme
Hamming Kodlama

- Daha çok, hatayı alıcıda düzeltmek için kullanılan en yaygın olarak kullanılan yöntemdir. Her bir sözcüğün yolda j bitinin bozulması durumunda, hatayı düzeltmenin mümkün olduğu kod'a j bit hata bağışıklığı olan kod denir. Bu kodlamada lineer blok kodlama yapısındadır. Genelde minimum 3 adet ek bit kullanır.
- Örneğin 4 bitlik data ve 3 adet parity biti kullanan Hamming (7,4) kodunda, 7 bitlik bir kod sözcüğü elde edilir. 4 bit veri biti, 3 bit parity biti olarak.
- 3 bitlik bir hamming kodu 1 bitlik hata düzelten veya 2 bitlik hatayı sezen bir kodlamadır.
- Genellikle sözcüğün sadece tek bir bitinin yolda bozulabileceği kabul edilen uygulamalarda kullanılır.

Hamming Kodlama

- Örneğin, aynı anda yalnız 1 bitin bozulduğu varsayılarak, **(a_0,a_1,a_2,a_3)** veri bitlerini ve **(a_4,a_5,a_6**) ise Hamming bitlerini ifade etsin. Bu durumda 7 bitlik bir sözcük **(codeword)** elde edilir.
- Düzeltme bitleri aşağıdaki formulasyonla hesap edilir;

$$a_4 = a_0 \oplus a_1 \oplus a_2$$

$$a_5 = a_1 \oplus a_2 \oplus a_3$$

$$a_6 = a_2 \oplus a_3 \oplus a_4$$

Bu şekilde elde edilen bitler ile sözcük oluşturulup alıcıya gönderilir.

$$a_0,a_1,a_2,a_3\ ,a_4,a_5,a_6$$

Hamming Kodlama

Alıcı tarafta okunan bitler,

$a_0', a_1', a_2', a_3', a_4', a_5', a_6'$ olsun. Alınan bitlere göre hesaplanan düzeltme bitleri;

$$a_4'' = a_0' \oplus a_1' \oplus a_2'$$
$$a_5'' = a_1' \oplus a_2' \oplus a_3'$$
$$a_6'' = a_2' \oplus a_3' \oplus a_4'$$

şeklindedir.Alıcı tarafta alınan ve hesaplanan bitlerin karşılaştırılması;

$$\mathbf{s_4 = a_4' \oplus a_4'' = a_0' \oplus a_1' \oplus a_2' \oplus a_4'}$$
$$\mathbf{s_5 = a_5' \oplus a_5'' = a_1' \oplus a_2' \oplus a_3' \oplus a_5'}$$
$$\mathbf{s_6 = a_6' \oplus a_6'' = a_2' \oplus a_3' \oplus a_4' \oplus a_6'}$$

Sonucunda $\mathbf{s_4, s_5, s_6}$ bitlerinin sonuçları 0 olursa gönderilen ve alınan sözcük doğrudur.

Hamming Kodlama

s_4,s_5,s_6 bitlerinin geriye kalan 7 kombinasyonu ise, hatanın hangi bitte olduğunu sezmeye ve düzeltmeye yarar.

s4s5s6	hatalı bit
0 0 0	hata yok
0 0 1	a6
0 1 0	a5
0 1 1	a3
1 0 0	a0
1 0 1	a4
1 1 0	a1
1 1 1	a2

$a_4'' = a_0' \oplus a_1' \oplus a_2'$

$a_5'' = a_1' \oplus a_2' \oplus a_3'$

$a_6'' = a_2' \oplus a_3' \oplus a_4'$

$s_4 = a_4' \oplus a_4''$

$s_5 = a_5' \oplus a_5''$

$s_6 = a_6' \oplus a_6''$

	a_0	a_1	a_2	a_3	a_4	a_5	a_6
s_4	X	X	X		X		
s_5		X	X	X		X	
s_6			X	X	X		X

Hamming kodlaması Örnek

	a_0	a_1	a_2	a_3	a_4	a_5	a_6
Gönderilen	1	0	0	1	0	0	1

	a'_0	a'_1	a'_2	a'_3	a'_4	a'_5	a'_6
Alınan	1	0	0	0	0	0	1

Gönderilen Paritiy

$a_4 = a_0 \oplus a_1 \oplus a_3 = 1 \oplus 0 \oplus 1 = 0$

$a_5 = a_0 \oplus a_2 \oplus a_3 = 1 \oplus 0 \oplus 1 = 0$

$a_6 = a_1 \oplus a_2 \oplus a_3 = 0 \oplus 0 \oplus 1 = 1$

Alınanla hesaplanan Parity

$a''_4 = a'_0 \oplus a'_1 \oplus a'_3 = 1 \oplus 0 \oplus 0 = 1$

$a''_5 = a'_0 \oplus a'_2 \oplus a'_3 = 1 \oplus 0 \oplus 0 = 1$

$a''_6 = a'_1 \oplus a'_2 \oplus a'_3 = 0 \oplus 0 \oplus 0 = 0$

$s_4 = a_4' \oplus a_4'' = a_0' \oplus a_1' \oplus a_3' \oplus a_4' = 1 \oplus 0 \oplus 0 \oplus 0 = 1$

$s_5 = a_5' \oplus a_5'' = a_0' \oplus a_2' \oplus a_3' \oplus a_5' = 1 \oplus 0 \oplus 0 \oplus 0 = 1$

$s_6 = a_6' \oplus a_6'' = a_1' \oplus a_2' \oplus a_3' \oplus a_6' = 0 \oplus 0 \oplus 0 \oplus 1 = 1$

Bu tabloya göre a_3 biti hatalı alınmıştır.

	a_0	a_1	a_2	a_3	a_4	a_5	a_6
s_4	x	x		x	x		
S_5	x		x	x		x	
s_6		x	x	x			x

S_4	S_5	S_6	Hata
0	0	0	Yok
0	0	1	a_6
0	1	0	a_5
0	1	1	a_2
1	0	0	a_4
1	0	1	a_1
1	1	0	a_0
1	1	1	a_3

Automatic Repeat reQuest (ARQ) Mechanisms

- Bir taraf diğer tarafa mesaj gönderdiği zaman, diğer taraf mesajı aldığına dair geriye küçük onay (acknowledgement) ACK mesajı gönderir.
  - Mesela, eğer A B ye mesaj gönderdiyse, B de A ya ACK gönderir
  - ACK alındıktan sonra, A mesajın doğru şekilde yerine ulaştığını anlar
  - Eğer T zaman aralığında ACK gelmezse, A mesajın kaybolduğunu farzeder ve mesajın bir kopyasını yeniden gönderir
- ARQ hata saptanan veri'nin üstesinden gelmek için önemlidir
  - fakat hata doğrulama yöntemi değildir
  - Gönderim hatalarının saptanması için çoğu bilgisayar ağları CRC'yi kullanır
- ARQ şeması gönderimin garantiye alınmasını salar, eğer gönderim sırasında hata oluşmuşsa
  - Eğer hata oluşmuşsa alıcı paketi atar
  - Ve gönderici bir kopyasını tekrar gönderir

ARQ'ya örnek

Çevrimli Fazlalık Sınaması (CRC: Cyclic Redundancy Check)

- CRC yöntemi, uzun veri dizilerindeki hataların sezilmesi için kullanılır.

- **CRC (Çevrimli Fazlalık Sınaması)** kodlamasında, gönderilecek veri katarından hesaplanan bir sınama bit katarı (Cylic kod) oluşturulur. Ve verinin sonuna eklenir. *Cyclic kodlar özel lineer blok kodlardır. Her cyclic kod rotate edilebilir. Rotate edildikten sonra oluşan kod geçerli bir codeword'tür. 1011000 codeword left-shift yapılırsa codeword 0110001 olur ve geçerlidir.*
- Cyclic redundancy check (CRC), LAN ve WAN ağlarda kullanılır.
- Bu yöntem esasta donanımsal olarak gerçekleştirilmeye dayanır. Yazılımsal gerçekleştirilmeye göre donanımsal olarak daha hızlı çalışmaktadır.
- CRC bitlerinin gerçek zamanda hesaplamak için donanım desteği veren iletişim chip'leri mevcuttur. XOR kapısı ve Kaydırmalı kaydedici (Shift Register) kullanılarak donanımsal olarak gerçekleştirilebilir.

Cyclic Redundancy Codes (CRCs)

- Kanal kodlama formu olarak bilinen Cyclic Redundancy Code (CRC) yüksek hızlı ağlarda kullanılır
- CRC'nin asıl özelliği aşağıda özetlenmiştir

İsteğe Bağlı uzunlukta mesaj	Veri sözcüğünün boyutu sabit değildir, bu da bir CRC'nin keyfi bir uzunluktaki mesaja uygulanabileceği anlamına gelir.
Mükemmel hata tespiti	**Hesaplanan değer bir mesajdaki bitlerin sırasına bağlı olduğundan, bir CRC mükemmel hata tespit yeteneği sağlar.**
Hızlı donanımsal gerçekleme	**Onun sofistike matematiksel temeline rağmen, bir CRC hesaplaması son derece hızlı bir şekilde donanımla gerçekleştirilebilir.**

CRC kodlamanın temel özellikleri

CRC

- Gönderilecek veri paketi, n adet bite sahip ise, n'inci dereceden bir polinom olarak düşünülür.

- Bu polinomun katsayıları, her bir orijinal veri bitidir.
- Bu polinom, x^p ile çarpıldıktan sonra üreteç polinomuna bölünür (p üreteç fonksiyonun en yüksek derecesidir. CRC'nin bit sayısı p'ye eşittir.)

- Kalan bölümündeki polinom CRC kodunun bitlerini oluşturur.*'CRC* bitleri orijinal verinin sonuna eklenerek, birlikte alıcıya gönderilir.

- Alıcı,kendisine gelen bu bit dizisinden oluşturacağı polinomu, üreteç polinomuna böldüğü zaman kalan 0 ise, gelen veride bir bozulma yoktur.

CRC Kodlama oluşturma adımları (yazılımsal)

CRC katarını hesaplama yöntemi:

1- Gönderilecek veri katarı P(x) gibi bir polinom ile gösterilsin. (Katsayıları 1 veya 0). Aktarılacak bilginin uzunluğu n bit ise; polinom aşağıdaki gibi olur.

$$P(x) = b_{n-1}\, x^{n-1} + b_{n-2}\, x^{n-2} + \ldots\ldots\ldots\ldots + b_1.x^1 + b_0.x^0$$

Yola enson çıkarılacak bit değeri

Buna göre **1 0 1 0 0 1 0 1 1 1** şeklindeki orijinal veri katarından;

$$P(x) = 1.x^9 + 1.x^7 + 1.x^4 + 1.x^2 + 1.x^1 + 1.x^0$$

$$P(x) = x^9 + x^7 + x^4 + x^2 + x^1 + 1$$

Polinomu oluşturulsun.

CRC kodlama adımları

2- **Bu P(x) polinomu x^p ile çarpılır.** ***(P' genellikle üreteç fonksiyonun en üst derecesidir).*** **Bu işlem genellikle veri kodlarının sonlarına eklenmiş p tane 0 ekleme işlemidir**.

$x^p . P(x) = 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0 \ldots\ldots 0\ 0\ 0\ 0$ p tane 0

- G(x) bazı üreteç polinomları $x^{16} + x^{15} + x^2 + 1$

  $x^{16} + x^{12} + x^5 + 1$

  $x^{12} + x^{11} + x^3 + x^2 + x + 1$

  $x^4 + x^2 + x + 1$

Ad	ÜRETEÇ Polinom	Uygulama
CRC-8	X^8+x^2+x+1	ATM başlık
CRC-10	$X^{10}+x^9+x^5+x^4+x^2+1$	ATM AAL
CRC-16	$X^{16}+x^{12}+x^5+1$	HDLC
CRC-32	$X^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$	LAN lar

CRC kodlama adımları

3- $\mathbf{X^p.P(x)}$ polinomu , p. dereceden G(x) adı verilen üreteç (genarating) polinomuna bölünür. Üreteç polinomları belirli hata sezme özellikleri bulunan standart polinomlardır. Biçimi hususunda iletişim başlamadan önce gönderici ve alıcı anlaşmış olmalıdır.

4- Gönderici; $\mathbf{X^p.P(x) / G(x)}$ bölümünü hesaplar. Bu bölme işleminde:

Bölüm : Q(x)
Kalan : R(x) olsun.

- **P:** üretec fonksiyonun en yüksek derecesi.

$$\mathbf{X^p.P(x) = Q(x).G(x) + R(x)} \quad \text{olur.}$$

- Bu işlemde 2 tabanı (mod 2) aritmetiği (elde göz önüne alınmadan toplama) kullanılırsa (mod2 aritmetiğinde toplama ve çıkarma işlemleri aynı sonucu verir);

- $$\mathbf{X^p.P(x) - R(x) = Q(x).G(x) + R(x)}$$

F.1

$$\mathbf{X^p.P(x) + R(x) = Q(x).G(x)} \quad \text{olmalıdır.}$$

CRC kodlama adımları

- Buna göre gönderici alıcıya ; P(x) polinomuna karşı düşen bit dizisi yerine **$X^p.P(x) + R(x)$** polinomuna karşı düşen bit dizisini göndersin.
  (*Bu polinom ,aslında asıl veri katarının arkasına eklenmiş p uzunluğunda ek bir diziden ibaretdir.)*

- Alıcıya gönderilen toplam diziye ilişkin polinom ,G(x) polinomunun tam katıdır (F1'e göre). Ayrıca G(x) polinomu alıcı tarafından'da bilinmektedir.

- Eğer yolda herhangi bir bitin bozulmadığını ve **Q(x).G(x)**'in alıcıya aynen ulaştığını varsayarsak; Alıcı kendine gelen bit dizisine karşılık düşen polinomu G(x)'e böldüğünde, bölme sonucunda kalan olmamalıdır.

- Alıcı kalanın = 0 olması durumunda hatasız iletim olduğunu anlar, gelen bit dizisinde en sondaki p tanesini atar, geriye kalan n bit bilgiyi işler.

CRC kodlama sayısal örnek:

- CRC hata sezme yöntemi kullanılacak bir iletişimde, gönderilecek bilgi bitleri dizisi **“ 1 0 1 1 0 1 0 1 ”** şeklindedir.

  Üreteç polinomu olarak **$G(x) = x^3 + x + 1$** ise, veriye eklenecek CRC denetim bitlerini hesaplayınız ve gönderilecek bit dizisini belirtiniz.

Çözüm:

1- Verilen diziden polinom oluşturulur.

Verilen orijinal dizi: 1 0 1 1 0 1 0 1

P(x) polinomunun terimleri: x^7 x^6 x^5 x^4 x^3 x^2 x^1 x^0

$$P(x) = 1.x^7 + 0.x^6 + 1.x^5 + 1.x^4 + 0.x^3 + 1.x^2 + 0.x^1 + 1.x^0 = x^7 + x^5 + x^4 + x^2 + 1$$

2- Polinomu üreteç G(x) polinomunun en yüksek dereceli terimiyle çarp:

$$x^3 .P(x) = x^3.(x^7 + x^5 + x^4 + x^2 + 1) = x^{10} + x^8 + x^7 + x^5 + x^3$$

3 - $x^3 .P(x)$ polinomunu verilen G(x) polinomuna bölelim.

$x^3 .P(x)$ $x^{10} + x^8 + x^7 + x^5 + x^3$ | $x^3 + x + 1$ Bölen : **G(x)**

$+ \quad x^{10} + x^8 + x^7$ | $x^7 + x^2$ Bölüm : **Q(x)**

$x^5 + x^3$

$+ \quad x^5 + x^3 + x^2$

x^2 : Kalan : **R(x)**

4- Gönderilecek bit dizisini tanımlayan polinom

$T(x) = Q(x).G(x) = x^3 .P(x) + R(x) = x^{10} + x^8 + x^7 + x^5 + x^3 + x^2$

$T(x) = 1.x^{10} + 0.x^9 + 1.x^8 + 0.x^6 + 1.x^5 + 0.x^4 + 1.x^3 + 1.x^2 + 0.x^1 + 0.x^0$

Gönderilen bit dizisi: **1 0 1 1 0 1 0 1 1 0 0**

CRC kodlama Sayısal Örnek

Alıcıya gelen bit katarı **1 0 1 1 0 1 0 1 1 0 0** olsun. Alıcı buradan oluşturacağı polinomu üreteç fonksiyonuna böler. Kalan 0 ise veri hatasız gelmiştir. CRC bitlerini atar. Orijinal veriyi elde etmiştir.

$x^{10}+x^8+x^7+x^5+x^3+x^2$ | $x^3 + x + 1$

$x^{10}+x^8+x^7$ | x^7+x^2

$x^5+x^3+x^2$

$x^5+x^3+x^2$

0 0 0

Üreteç Fonksiyonlarının seçimi önemlidir. Üreteç polinomu hangi tür hataların tespit edilebileceğini belirtir.

- CRC yöntemiyle şunları belirleyebiliriz:
  - Tüm tek bit hatalarını
  - Tüm çift bit hatalarını
  - Tüm tek sayılı bit hatalarını
  - R'den küçük tüm hata patlamalarını (R üreteç polinomun derecesidir)
  - R'den büyük çok fazla hata patlamasını

Hata Tipi	16 ikil CRC Hatayı Bulma Olasılığı	32 ikil CRC Hatayı Bulma Olasılığı
Tek ikil hataları	1.0	1.0
İki ikil hataları	1.0	1.0
Tek sayılı hatalar	1.0	1.0
16 ya da 32'deb (CRC uzunluğundan) daha küçük hata patlamaları	1.0	1.0
CRC uzunluğuna eşit (16,32) hata patlamaları	1-(1/215)	1-(1/231)
CRC uzunluğundan daha büyük sayıda hata patlamaları	1-(1/216)	1-(1/232)

CRC Etkisi

Uygulamada en çok kullanılan 16 bit ve 32 CRC uygulamalarıdır

8.13 Cyclic Redundancy Codes (CRC)

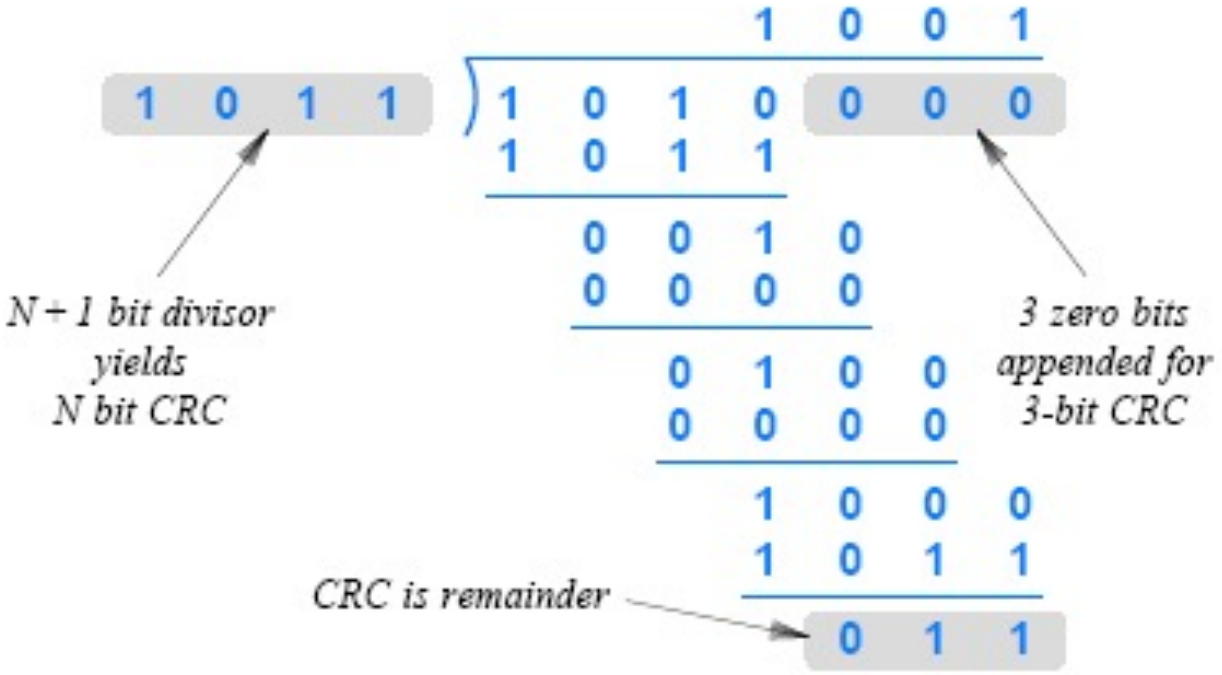
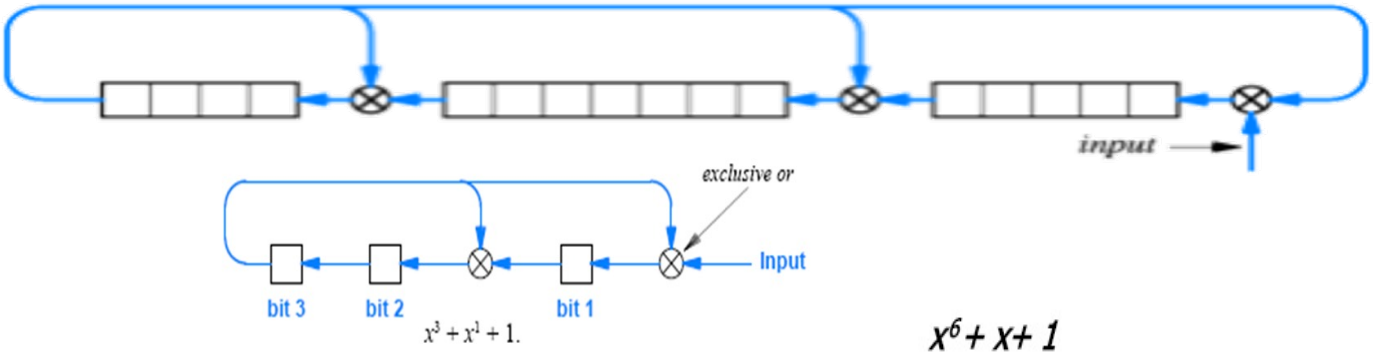


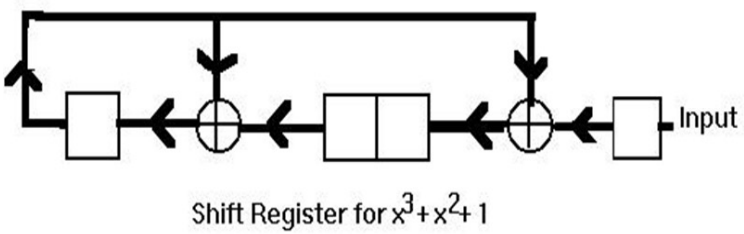
Figure 8.12 Illustration of a CRC computation viewed as the remainder of a binary division with no carries.

CRC kod oluşturma (Donanımsal)

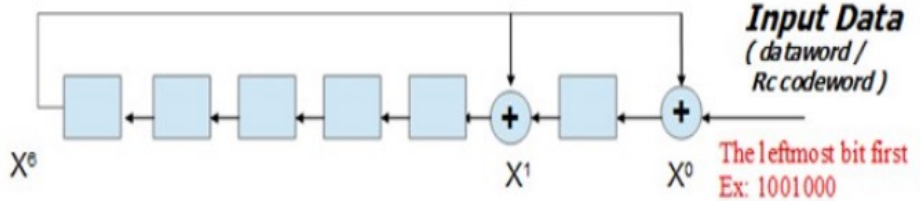
- Tasarımcı istediği büyüklükte bir CRC katarı oluşturabilir. Ethernet teknolojisi için 32 bit en uygunudur. CRC bit sayısı, kullanılan üreteç polinomunun en yüksek derecesine eşittir.
- Tasarımcı ShifREG ve XOR kapılarını kullanarak devreyi oluşturmalıdır. Kullanılacak shiftreg bit sayısı üreteç fonksiyonun en üst derecesine eşit olacaktır.
- Üreteç polinomunda değeri 1 olan her katsayıyı temsil eden shiftREG biriminin önüne Ex-OR bağlacı konur. En yüksek dereceli katsayı geri beslenir.
- Örnek: 16 bit'lik CRC kodu oluşturan üreteç fonk. $X^{16} + x^{12} + x^5 + 1$ için.



$x^6 + x + 1$



Shift Register for $x^3 + x^2 + 1$



Örnek

Orijinal veri : 1011001 **$P(x) = x^6 + x^4 + x^3 + 1$**

Üreteç polinomu **$G(x) = x^4 + x + 1$**

$x^4.(x^6+x^4+x^3+1) = x^{10}+x^8+x^7+x^4$ | x^4+x+1

$x^{10}+x^7+x^6$ | $x^6+x^4+x^2+x$

$x^8+x^6+x^4$

$x^8+x^5+x^4$

x^6+x^5

$x^6+x^3+x^2$

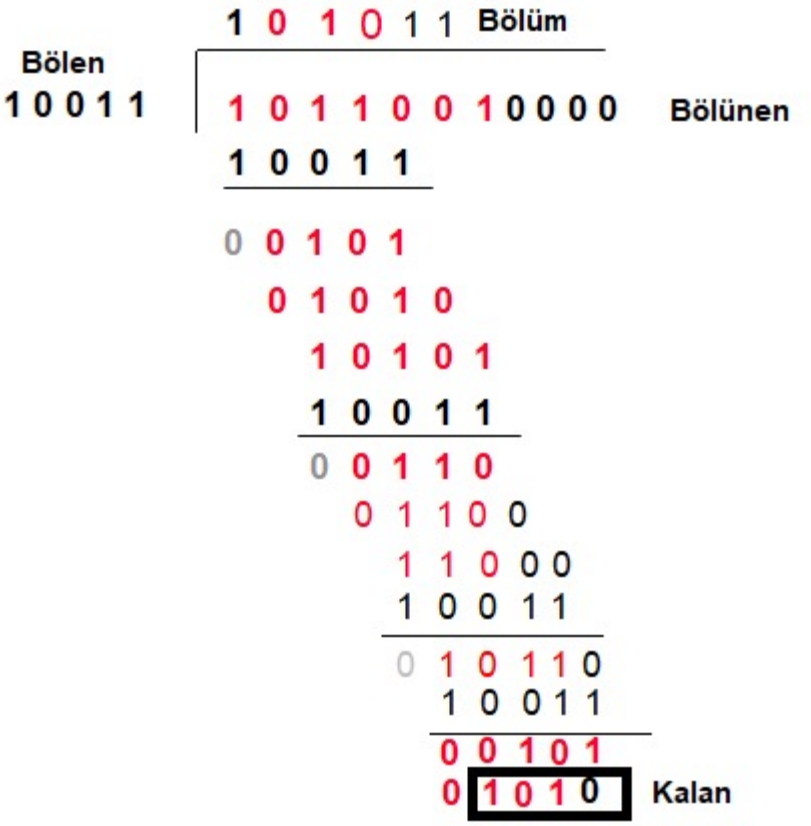
$x^5+x^3+x^2$

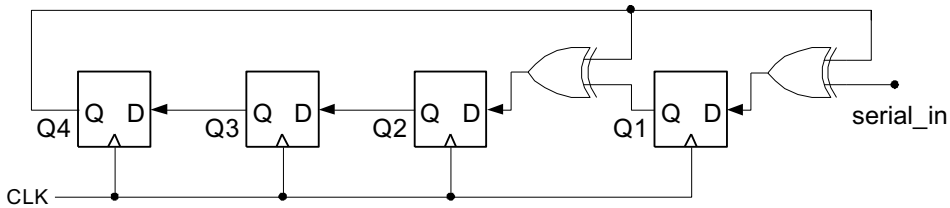
x^5+x^2+x

x^3+x^1……..CRC bitleri 1010 olur

Gönderilen : 10110011010

Polynomsal Bölme

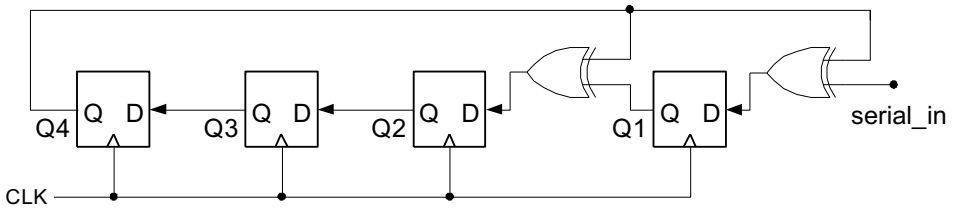




MSB sıfırsa, sola kaydırın, bir sonraki biti getirin.
MSB 1 ise, bölen ile XOR'la ve sonra sola kaydır.

Mod2 art. (Eldesiz çıkarma-toplama-XOR)

CRC kodlama

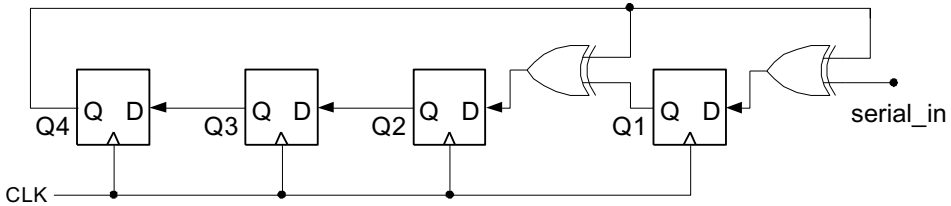


1 0 1 1 0 0 1 0 0 0 0

0	0	0	0	
0	0	0	1	0 1 1 0 0 1 0 0 0 0
0	0	1	0	1 1 0 0 1 0 0 0 0
0	1	0	1	1 0 0 1 0 0 0 0
1	0	1	1	0 0 1 0 0 0 0
0	1	0	1	0 1 0 0 0 0
1	0	1	0	1 0 0 0 0
0	1	1	0	0 0 0 0
1	1	0	0	0 0 0
1	0	1	1	0 0
0	1	0	1	0
1	0	1	0	

Message sent: 1 0 1 1 0 0 1 1 0 1 0

CRC decoding



1 0 1 1 0 0 1 1 0 1 0

0 0 0 0

0 0 0 1 0 1 1 0 0 1 1 0 1 0
0 0 1 0 1 1 0 0 1 1 0 1 0
0 1 0 1 1 0 0 1 1 0 1 0
1 0 1 1 0 0 1 1 0 1 0
0 1 0 1 0 1 1 0 1 0
1 0 1 0 1 1 0 1 0
0 1 1 0 1 0 1 0
1 1 0 1 0 1 0
1 0 0 1 1 0
0 0 0 0 0
0 0 0 0

Checksum

- Checksum bir ASCII metin veya alafanümerik bir text dosyasının sabit uzunlukta bir hash'inin (özetinin) çıkarılması işlemidir.
- Farklı şekillerde gerçekleştirilebilir.

- CheckSum8 Xor : veriler 8'er bit olarak ayrılır. Biribirleriyla XOR'lanır. Son xor değeri checksum'dır. Örn: abcabc'nin Checsum8 Xor = dd çıkar.

- CheckSum8 Modulo 256 : Veriler 8 bitlik guruplara ayrılır. Bunlar toplanır. 8 bit işlem yapılıyorsa eldeler göz önüne alınmaz. Sonuç 256 ya bölünür. Mod 256 sonuçtur. Örn: aaaaaaaa 'in checksum8 Modulo 256 sonucu = A8'dir. (bayt) word sonucu.

  Sum of Bytes % 256

- CheckSum8 2s Complement

  0x100 - Sum Of Bytes

  veri 8 bitlik guruplara ayrılık ikilik tabanda eldesiz toplanır. En son toplamın 2'ye tümleyeni alınır.

  Örnek aaaaaaaa 'nın checksum'ı= 0x58

16-bitlik checksum

- Checksum hesaplamak için, gönderici
  - 16-bitlik tamsayılara sayısal değerler ekler
  - Ve sonucu gönderir
- Mesajı onaylamak için, alıcı aynı işlemi yeniden hesaplamalıdır
  - Checksum 1'in tümleyeni aritmetiğine göre hesaplanır
  - Ve 32 ve 64 bit tam sayı aritmetiği yerine 16'lık kullanılır.

- Döngü aşamasında, overflow (taşma)oluşabilir
  - Bunun için algoritma, overflow olan veriyi geri toplama ekler
- Neden checksum toplamın tersi aritmetiğine göre hesaplanırda, toplama göre hesaplanmaz?

- Not:2'ye tümleme sonuç için 8 bitlik blokları topla, oluşan elde bitini kullanma, enson toplam değerinin 2'ye tümleme metoduna göre düzenle
- Örnek: 0xaaaa nın 8 bitlik checeksum'ı 0xac çıkmalı 0xaaaaaaaa'nın checksum'ı 0x58 çıkmalı

Internette (IP'de) kullanılan 16-bitlik checksum

- Checksum, özel kodlama şeması IP paketlerinde önemli bir rol oynar.
  - Genelde internet checksum olarak bilinir,
  - sonuç 16-bit'in 1'e göre tümleyeni alınır ve checsum elde edilmiş olur.
- Internet checksum dataword üzerinde eşit veri büyüklüğünü empoze etmez (limitlemez).
  - Algoritma mesajların keyfi uzunlukta olmasına izin verir
  - Ve bütün mesaj üzerinden checksum hesaplaması yapılmasına izin verir
- Internet checksum mesajlar içerisindeki verilere 16-bitlik tamsayılar şeklinde muamele eder, aşağıdaki Şekil bunu gösterir

Varsayalım ki 4x16 bitlik bir veri için 16 bitlik checksum yapacağız.
1000 0110 0101 1110
1010 1100 0110 0000
0111 0001 0010 1010
1000 0001 1011 0101