## Hamming SEC-DED Simülatörü Dokümanı

### 1. Giriş

Bu doküman, **Hamming SEC-DED (single-error-correcting, double-error-detecting) kodunun** nasıl çalıştığını ve geliştirilen simülatörün işleyişini kapsamlı bir şekilde açıklamaktadır. Proje, veri hatalarını tespit etmek ve düzeltmek için kullanılan **Hamming kodunu** simüle eden bir sistemdir.

Hamming kodları, **bilgi teorisi** ve **hata düzeltme kodlama teknikleri** içinde önemli bir yer tutmaktadır. Bellek hatalarının olasılığının yüksek olduğu sistemlerde veri güvenliğini sağlamak için kullanılan bir yöntemdir.

Bu simülatör, kullanıcının belirli boyutlardaki verileri **Hamming koduyla kodlamasına**, hatalar oluşturmasına ve bu hataları analiz etmesine olanak tanır.

# 2. Projenin Amacı

Proje aşağıdaki hedefleri gerçekleştirmek için tasarlanmıştır:

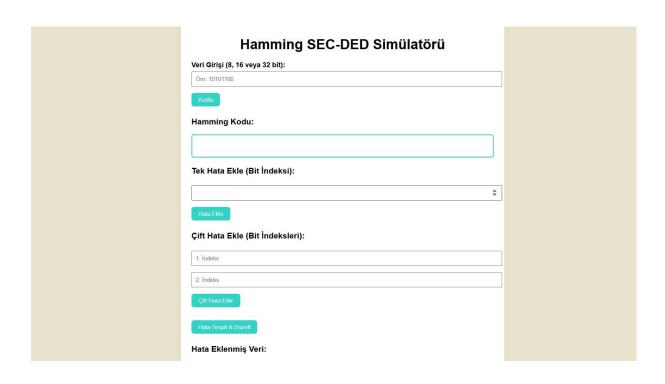
- Hamming kodlarını öğrenmek: Kullanıcılar, SEC-DED (tek hatayı düzeltebilir, çift hatayı tespit edebilir) kodlama prensibini deneyimleyebilir.
- **Hata düzeltme mekanizmalarını anlamak:** Bellek hatalarını simüle ederek nasıl tespit ve düzeltildiklerini analiz edebilir.
- **Kullanıcı dostu bir deneyim sunmak:** Etkileşimli bir arayüz ile hataların nasıl eklendiği ve düzeltildiği gösterilebilir.

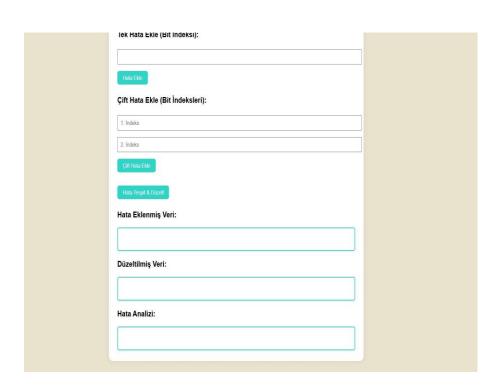
# 3. Kullanıcı Arayüzü

Simülatör, kullanıcıların **görsel olarak veri girişi yapmasına** ve hata eklemesine olanak tanıyan bir **HTML, CSS ve JavaScript** tabanlı **etkileşimli arayüz** sunar.

# Arayüzde Bulunan Bölümler

- Veri Girişi Alanı: Kullanıcıdan 8, 16 veya 32 bitlik veri girişi alır.
- Kodlama Butonu: Kullanıcının girdiği veriyi Hamming koduyla şifreler ve hata düzeltme için gerekli bitleri ekler.
- Tek Hata Ekleme: Kullanıcı, belirli bir bit pozisyonuna tek hata ekleyebilir.
- Çift Hata Ekleme: Kullanıcı, iki farklı bit pozisyonuna hata ekleyebilir.
- Hata Analizi: Eklenen hataların konumu ve tespit edilen hatalar hakkında bilgi verir.
- Hata Düzeltme: Sistem tek hataları otomatik olarak düzeltir, çift hatalar yalnızca tespit edilebilir ancak düzeltme garantili değildir.





## 4. Teknik Detaylar

# Hamming Kodlama Algoritması

Hamming kodları, belirli bir **veri uzunluğuna** göre **parite bitleri ekleyerek** hataları düzelten bir algoritmadır.

#### Adımlar:

# 1. Veri Uzunluğu ve Parite Bitlerinin Belirlenmesi:

- o Kullanıcıdan gelen veri uzunluğu (8, 16 veya 32 bit) analiz edilir.
- o Hamming kodunda gerekli **parite bitleri (r)** hesaplanır.
- o **2^r≥m+r+1** eşitsizliğini sağlayan minimum r değeri bulunur.

## 2. Veri Yerleşimi:

- Hamming kodunda, veri bitleri ile parite bitleri belirli bir düzen içinde yerleştirilir.
- o Parite bitleri, hata tespiti için özel konumlarda saklanır.

# 3. Parite Bitlerinin Hesaplanması:

o Her parite biti, kendisine bağlı olan bitlerden XOR işlemi ile hesaplanır.

# 4. Genel Parite Kontrolü:

• **Genel parite biti**, veri içerisindeki tüm bitlerin XOR işlemi ile oluşturulmasını sağlar.

# Hata Ekleme Mekanizması

Hata simülasyonu yapmak için **kullanıcının belirlediği bir bit değiştirilebilir.** Çift hata ekleme seçeneği ile iki farklı bit değiştirilebilir.

- Tek hata ekleme: Kullanıcının belirlediği tek bir bit ters çevrilir (0 -> 1 veya 1 -> 0).
- **Çift hata ekleme:** Kullanıcının belirlediği **iki farklı bit** ters çevrilir (0 -> 1 veya 1 -> 0).

# Hata Tespiti ve Düzeltme

Hata tespit ve düzeltme **sendrom analizi** ile yapılır.

#### 1. Sendrom kelimesi hesaplanır:

- Her **parite biti** tekrar hesaplanır ve hata olup olmadığı kontrol edilir.
- o Eğer sendrom değeri **0** ise, hata bulunmaz.

- Eğer sendrom değeri sıfırdan büyükse, tek hata tespit edilmiştir ve düzeltilebilir.
- Eğer genel parite hatalı ancak sendrom 0 ise, çift hata bulunmaktadır ve düzeltilemez.

# 2. Hata Düzeltme:

- o Tek hata tespit edilirse, hata bulunan bit ters çevrilerek düzeltilir.
- o Çift hata tespit edilirse, yalnızca **tespit edilir**, ancak düzeltilmez.

| 0110010000011          |  |
|------------------------|--|
| Düzeltilmiş Veri:      |  |
| 0110010000111          |  |
| lata Analizi:          |  |
| Hata 3. bitte eklendi. |  |
| Hata J. Ditte Chichai. |  |

| Çift Hata Ekle (Bit İndel | sleri): |  |
|---------------------------|---------|--|
| 1                         |         |  |
| 4                         |         |  |
| Çift Hata Ekle            |         |  |
| Hata Tespit & Düzelt      |         |  |
| Hata Eklenmiş Veri:       |         |  |
| 1110010000011             |         |  |
| Düzeltilmiş Veri:         |         |  |
| 1110010000011             |         |  |
| Hata Analizi:             |         |  |
|                           |         |  |

# **Kod Yapısı**

Simülatör şu kod bileşenleri üzerine inşa edilmiştir:

- **HTML:** Kullanıcı arayüzünün temel yapısını oluşturur.
- CSS: Görsel öğeleri düzenler ve simülatörü estetik hale getirir.
- JavaScript: Hamming kodlama, hata ekleme ve hata düzeltme işlevlerini yürütür.

# 5. Örnek Kullanım Senaryoları

Aşağıda simülatörün nasıl çalıştığını gösteren bazı örnek senaryolar verilmiştir.

# Senaryo 1: Veri Kodlama

**Girdi:** 10101100 (8 bit) **Kodlanmış Çıktı:** 01110110100

# Senaryo 2: Tek Hata Ekleme

Girdi: 01110110100 Hata Ekleme: 5. bit ters çevrildi Hata Eklenmiş Çıktı: 01110010100

# Senaryo 3: Hata Tespiti ve Düzeltme

Hata Analizi: Tek hata bulundu Düzeltilmiş Çıktı: 01110110100

Senaryo 4: Çift Hata Ekleme

Hata Eklenmiş Çıktı: 01110000100 Hata Analizi: Çift hata tespit edildi ancak düzeltilemedi.

# 6. Sonuç ve Kullanım Alanları

Hamming SEC-DED simülatörü, hata düzeltme kodlarının nasıl çalıştığını öğrenmek, hata tespit mekanizmalarını anlamak ve pratik deneyim kazanmak için kullanılır. Bellek sistemleri, veri iletişimi ve hata düzeltme algoritmaları üzerinde çalışanlar için yararlı bir araçtır.

#### 7. Görsel Eklenebilecek Alanlar

# Hamming Kodlama Algoritması

- Veri Yerleşimi: Hamming kodunda veri ve parite bitlerinin düzenini gösteren bir şema.
- Parite Hesaplama: Parite bitlerinin nasıl hesaplandığını gösteren bir örnek.

#### Hata Ekleme Mekanizması

- Tek Hata Ekleme: Bir bitin ters çevrilmesini gösteren görsel.
- Çift Hata Ekleme: İki bitin ters çevrilmesini gösteren görsel.

# Hata Tespiti ve Düzeltme

- Sendrom Analizi: Hata tespit sürecini gösteren bir akış diyagramı.
- Hata Düzeltme: Tek hatanın nasıl düzeltildiğini gösteren bir örnek.

#### LİNKLER:

https://www.youtube.com/watch?v=NCJMXv1PZ9o

https://github.com/AsmBrk/Hamming\_Sim