
Proje Raporu: Hamming SEC-DED Kodlayıcı ve Hata Simülatörü

1. Giriş

Bu rapor, **BLM230 Bilgisayar Mimarisi Dersi Proje Ödevi** kapsamında geliştirilen "Hamming SEC-DED (Single-Error-Correcting Double-Error-Detecting) Kodlayıcı ve Hata Simülatörü" uygulamasını detaylandırmaktadır. Projenin temel amacı, Hamming SEC-DED kodunun çalışma prensiplerini görsel ve interaktif bir arayüzle kullanıcıya sunmaktır. Uygulama, 8, 16 ve 32 bitlik veriler üzerinde Hamming kodlama ve hata düzeltme/tespit yeteneklerini simüle etmektedir. Kullanıcı, yapay olarak hatalar oluşturabilir ve bu hataların nasıl tespit edilip düzeltildiğini gözlemleyebilir.

---

2. Proje Amaçları ve Kapsamı

Proje ödevi gereksinimleri doğrultusunda, geliştirilen simülatör aşağıdaki temel fonksiyonları yerine getirmektedir:

- **Veri Büyüklüğü Seçimi:** Kullanıcı, 8, 16 veya 32 bitlik veri boyutları arasında seçim yapabilir.
- **Hamming Kodu Üretimi:** Girilen ikili veri için otomatik olarak **Hamming SEC-DED kodu** hesaplanır ve oluşturulur. Bu süreç, parite bitlerinin doğru pozisyonlara yerleştirilmesini ve değerlerinin hesaplanmasını, ayrıca genel parite (global parity) bitinin eklenmesini içerir.
- **Yapay Hata Oluşturma:**
    - **Tek bitlik hata:** Hamming kodunda rastgele bir konumda tek bitlik hata oluşturulabilir.
    - **Çift bitlik hata:** Hamming kodunda rastgele iki farklı konumda çift bitlik hata oluşturulabilir.
    - **Manuel hata:** Kullanıcı tarafından belirtilen bir indekste bit hatası oluşturulabilir.

- **Hata Tespiti ve Düzeltme:**
    - Oluşturulan hataların **sendrom kelimesi** kullanılarak tespiti yapılır.
    - Uygulama, tespit edilen hatanın tek mi yoksa çift mi olduğunu kullanıcıya bildirir.
    - **Tek bitlik hatalar** için otomatik düzeltme mekanizması sunulur ve kullanıcı onayı ile düzeltme gerçekleştirilir.

- **Çift bitlik hatalar** tespit edilir ancak düzeltilemez.

- **Kullanıcı Dostu Arayüz:** Python'ın **Tkinter** kütüphanesi kullanılarak tasarlanmış, görsel öğelerle desteklenen bir arayüz ile uygulamanın kullanımı kolaylaştırılmıştır.

---

3. Teknik Detaylar ve Uygulama Mimarisi

Uygulama, temel olarak GUI sınıfı altında toplanan Tkinter arayüz bileşenleri ve Hamming kodlama/hata işleme mantığı ile çalışır.

3.1. Hata Oluşturma Fonksiyonları

- oneErrorButtonAction(): self.hammingListLast'ın rastgele bir indeksindeki biti tersine çevirir.
- twoErrorButtonAction(): self.hammingListLast'ın rastgele iki farklı indeksindeki bitleri tersine çevirir.
- manualErrorButtonAction(): Kullanıcının girdiği indekste ilgili biti tersine çevirir.
- updateHamming_Error(): Hamming kodunu arayüzdeki ilgili Entry alanında günceller.

3.2. Hata Tespiti ve Düzeltme

Hata Tespiti (control_action):

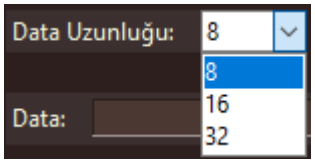
1. Mevcut Hamming kodu hammingArray olarak alınır.
2. Her parite biti pozisyonu için sendrom bitleri hesaplanır. Sendrom biti, o parite bitinin kontrol ettiği bitlerin XOR toplamının 0 olmaması durumunda 1'dir.
3. Tüm sendrom bitlerinin birleştirilmesiyle **sendrom kelimesi** (syndrome) oluşturulur. Sendrom kelimesinin değeri, hata pozisyonunu (1-tabanlı) gösterir.
4. Global parite biti de kontrol edilir.

Hata Düzeltme (fix_error):

1. Eğer control_action tarafından bir error_position belirlenmişse, bu pozisyondaki bit tersine çevrilir (0 ise 1, 1 ise 0 yapılır).
2. Güncellenen Hamming kodu arayüzde gösterilir ve hata mesajı "Hata düzeltildi!" olarak değiştirilir. "Düzelt" butonu tekrar gizlenir.

3.3. Kullanıcı Arayüzü (Tkinter)

- Uygulama, koyu renk temalı (koyu kestane, sıcak kahve tonları) bir arayüze sahiptir.
- **Veri Uzunluğu Seçimi:** ttk.Combobox ile 8, 16, 32 bit seçenekleri sunulur.



- **Veri Girişi:** tk.Entry alanı, yalnızca '0' ve '1' karakterlerini kabul eden bir doğrulama (validate_binary) ile korunmuştur. Girilen veri uzunluğu, seçilen bit sayısıyla sınırlıdır (dataLimit fonksiyonu).
- **Hamming Code Gösterimi:** Okunabilir (state='readonly') bir tk.Entry alanında gösterilir.
- **Hata Ekleme Butonları:** "Tek Hata Ekle", "Çift Hata Ekle" ve "İndex'e Hata Ekle" butonları, ilgili hata oluşturma fonksiyonlarını çağırır.
- **Hata Mesajı:** self.error_label etiketi, hata durumları ve işlem mesajları için kullanılır.
- **Kontrol ve Düzelt Butonları:** "Kontrol" butonu hataları tespit ederken, "Düzelt" butonu yalnızca tek hata tespit edildiğinde görünür hale gelir.

---

4. Geliştirme Ortamı ve Çalıştırma

- **Programlama Dili:** Python

- **GUI Kütüphanesi:** Tkinter
- **Gerekli Kütüphaneler:** tkinter, random

Uygulamayı Çalıştırma:

1. Proje dosyalarını GitHub deposundan klonlayın:

```
git clone https://github.com/AhmettCimen/Hamming-Code-Simulator.git
cd Hamming-Code-Simulator
```

2. Ana uygulamayı çalıştırın:

```
python gui.py
```

---

5. Sonuç ve Değerlendirme

Geliştirilen "Hamming SEC-DED Kodlayıcı ve Hata Simülatörü" uygulaması, proje ödevi gereksinimlerini büyük ölçüde karşılamaktadır. Kullanıcı dostu arayüzü sayesinde Hamming kodlama ve hata düzeltme/tespit mekanizmalarının anlaşılması kolaylaşmıştır.

Uygulamanın Güçlü Yönleri:

- Görsel ve etkileşimli bir öğrenme deneyimi sunar.
- Farklı veri uzunluklarını destekler (8, 16, 32 bit).
- Tek ve çift hata oluşturma senaryolarını başarıyla simüle eder.
- Tek bit hatalarını otomatik olarak düzeltme yeteneğine sahiptir.
- Veri girişi için doğrulama (sadece ikili karakterler) içerir.

Geliştirilebilecek Yönler:

- **Bellek Görselleştirmesi:** Proje tanımında "veriler bellekte saklanabilecek" ibaresi bulunmakta. Şu anki haliyle kodlanmış veri bir Entry kutusunda gösteriliyor; gerçek bir bellek simülasyonu (adresleme, okuma/yazma görselleştirmesi) eklenerek bu kısım daha da geliştirilebilir.
- **Geri Bildirimler:** Hata eklendiğinde veya düzeltildiğinde, kullanıcının daha belirgin görsel veya işitsel geri bildirimler alması sağlanabilir (örn. hatalı bitlerin kırmızı renkle vurgulanması).

6.2. Proje Kaynak Kodları ve Demo Videosu

- **GitHub:** https://github.com/AhmettCimen/Hamming-Code-Simulator.git
- **Demo Videosu:** **https://www.youtube.com/watch?v=gJVJ_yFkSG0**
