BURSA TEKNİK ÜNİVERSİTESİ

Bursa Teknik Üniversitesi

Bilgisayar Mühendisliği

BLM0230 Bilgisayar Mimarisi

Ad Soyad	Edem Makhsudov
Bölüm	Bilgisayar Mühendisliği
Öğrenci No	22360859373
Proje Konusu	Hamming SEC-DED Code Simülatörü
Programlama Dili	Python

2025

İÇİNDEKİLER

1. PROJE AMACI VE KAPSAMI

Bu projede, **Hamming SEC-DED** (Single Error Correcting, Double Error Detecting) algoritması kullanılarak hata tespit ve düzeltme **simülatörü geliştirilmiştir.** Simülatör aşağıdaki ana işlevleri gerçekleştirmektedir:

- 8, 16 ve 32 bitlik veriler üzerinde Hamming SEC-DED kodlama
- Bellek için otomatik parity bit hesaplama
- Yapay hata oluşturma ve test simülasyonu
- Sendrom hesaplama ile hata tespit ve düzeltme
- Görsel kullanıcı dostu arayüz

2. TEORİK ALTYAPI

2.1 Hamming SEC-DED Algoritması

Hamming kodu, Richard Hamming tarafından 1950'de geliştirilen ve tek bit hatalarını düzeltebilen, çift bit hatalarını tespit edebilen bir hata düzeltme kodudur. SEC-DED versiyonu, overall parity biti eklenerek çift bit hatalarının tespit edilebilmesini sağlar.

Algoritma Adımları:

1. Veri bit sayısı (m) için gerekli parity bit sayısı (r) hesaplanır: 2^r ≥ m + r + 1
2. Parity bitleri 2'nin kuvvetleri olan pozisyonlara yerleştirilir (1, 2, 4, 8, ...)
3. Her parity bit, binary pozisyon AND işlemi ile belirlenen veri bitlerini kontrol eder
4. Overall parity bit, tüm bitlerin XOR'u olarak hesaplanır

2.2 Hata Tespit ve Düzeltme Mantığı

Sendrom hesaplama ile hata pozisyonu belirlenir ve SEC-DED kurallarına göre:

- **Sendrom = 0, Overall Parity = 0:** Hata yok
- **Sendrom = 0, Overall Parity = 1:** Overall parity bitinde hata
- **Sendrom ≠ 0, Overall Parity = 1:** Tek bit hatası (düzeltilebilir)
- **Sendrom ≠ 0, Overall Parity = 0:** Çift bit hatası (tespit edilebilir, düzeltilemez)

3. YAZILIM MİMARİSİ

3.1 Modüler Tasarım

Proje 4 ana modülden oluşmaktadır:

`config.py`: Renk paleti ve stil ayarları

```
COLORS = {
    'data_bit': '#4ecdc4',        # Veri bitleri (turkuaz)
    'parity_bit': '#ff6b6b',      # Parity bitleri (kırmızı)
    'overall_parity': '#ff9ff3', # Overall parity (pembe)
    'error_bit': '#ffd93d'        # Hata bitleri (sarı)
    # …
}
```

`hamming_logic.py`: Hamming algoritma implementasyonu

- **`calculate_parity_bits()`:** Parity bit sayısı hesaplama
- **`encode_data()`:** SEC-DED kodlama
- **`detect_and_correct_error()`:** Hata tespit ve düzeltme

`hamming_simulator.py`: GUI ve kullanıcı etkileşimi

- Tkinter tabanlı görsel arayüz
- Renkli bit görselleştirme
- Interaktif hata simülasyonu

`main.py`: Uygulama başlatıcı

3.2 Algoritmik Implementasyon

```
def encode_data(data):
    m = len(data)  # Veri bit sayısı
    r = HammingLogic.calculate_parity_bits(m)
    total_length = m + r + 1  # +1 overall parity için

    # Parity bitlerini hesapla
    for parity_pos in parity_positions:
        parity_value = 0
        for i in range(1, total_length):
            if (i & parity_pos) != 0:  # AND işlemi ile kontrol
                parity_value ^= int(encoded_bits[i - 1])
        encoded_bits[parity_pos - 1] = str(parity_value)

    # Overall parity hesapla
    overall_parity = 0
    for bit in encoded_bits[:-1]:
        overall_parity ^= int(bit)
    encoded_bits[-1] = str(overall_parity)
```

4. KULLANICI ARAYÜZÜ VE ÖZELLİKLER

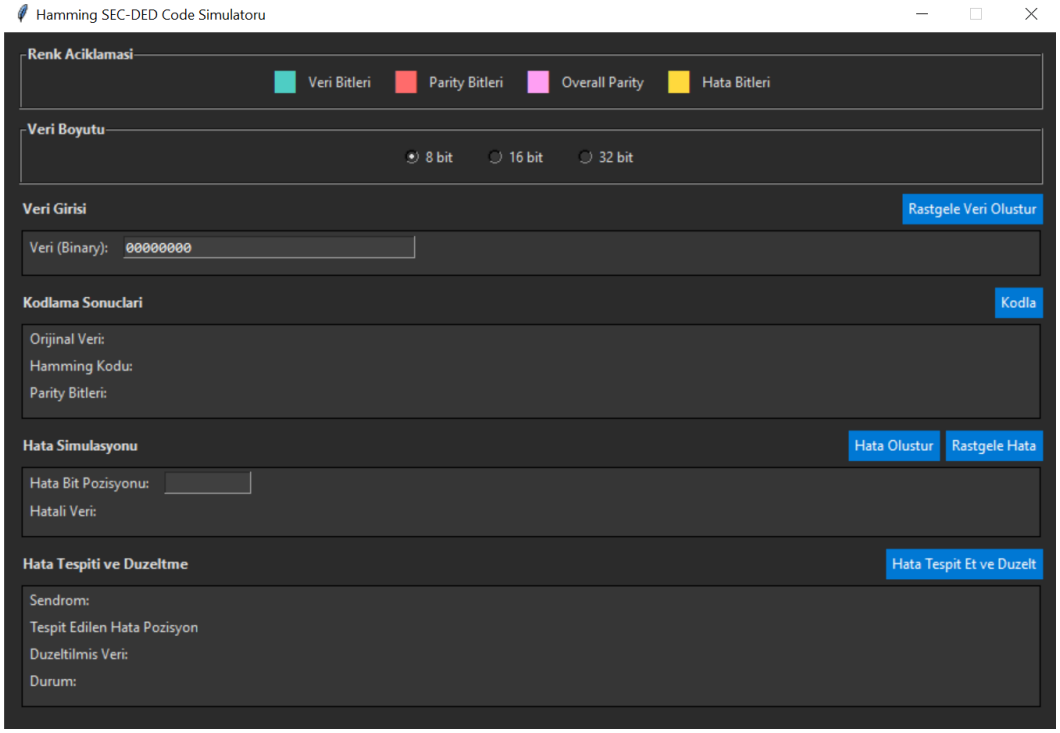
4.1 Görsel Tasarım

- **Renk Kodlaması:** Her bit tipi farklı renkle görselleştirilir
- **Modüler Arayüz:** Kodlama, hata simülasyonu ve düzeltme bölümleri ayrı
- **Real-time Görselleştirme:** Bit değişiklikleri anlık olarak görülür

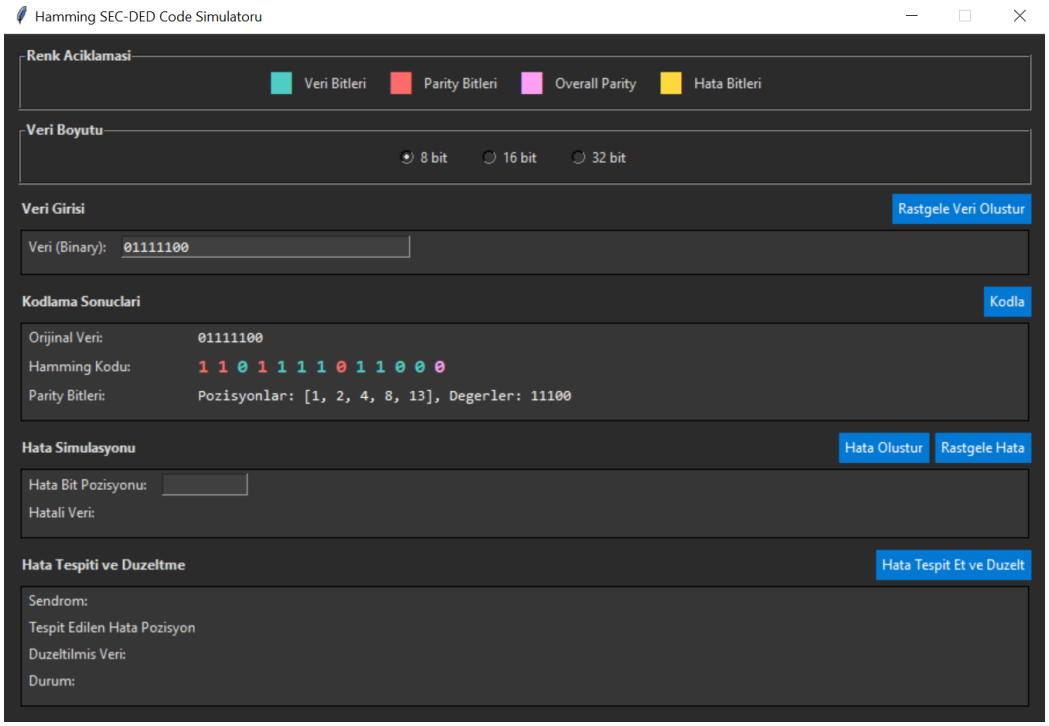
4.2 Fonksiyonellik

1. **Veri Boyutu Seçimi:** 8, 16, 32 bit radio button ile seçim
2. **Veri Girişi:** Binary veri girişi ile validasyon
3. **Rastgele Veri:** Test için otomatik veri üretimi
4. **Hata Simülasyonu:** Manuel/rastgele hata oluşturma
5. **Hata Düzeltme:** Sendrom hesaplama ve otomatik düzeltme

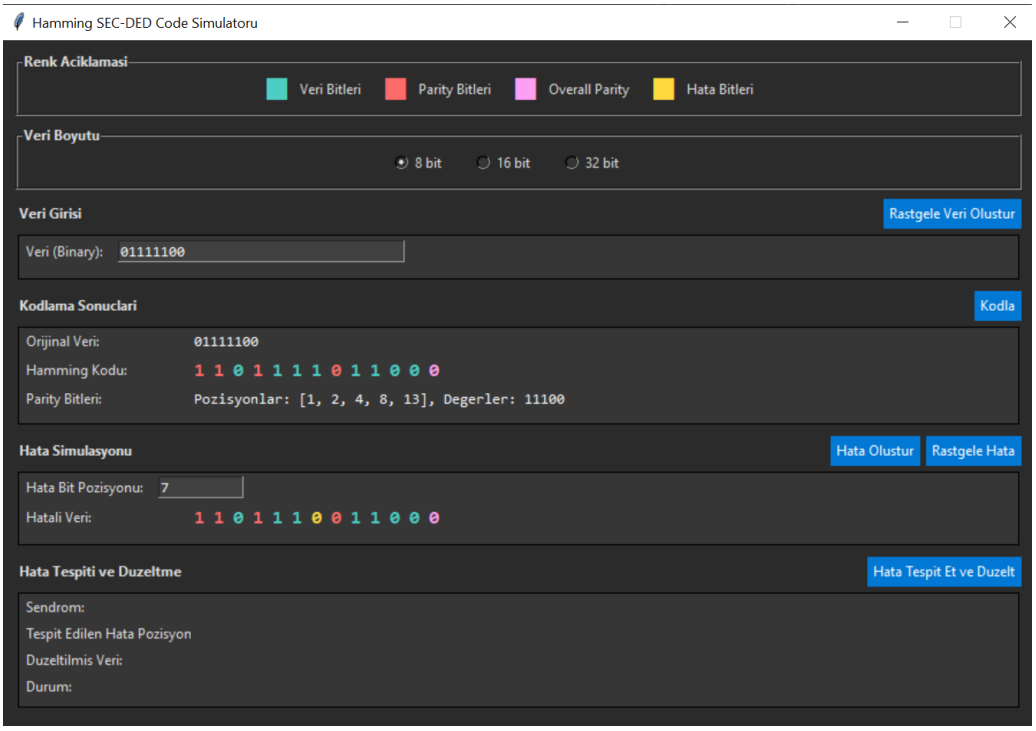
5. EKRAN GÖRÜNTÜLERİ



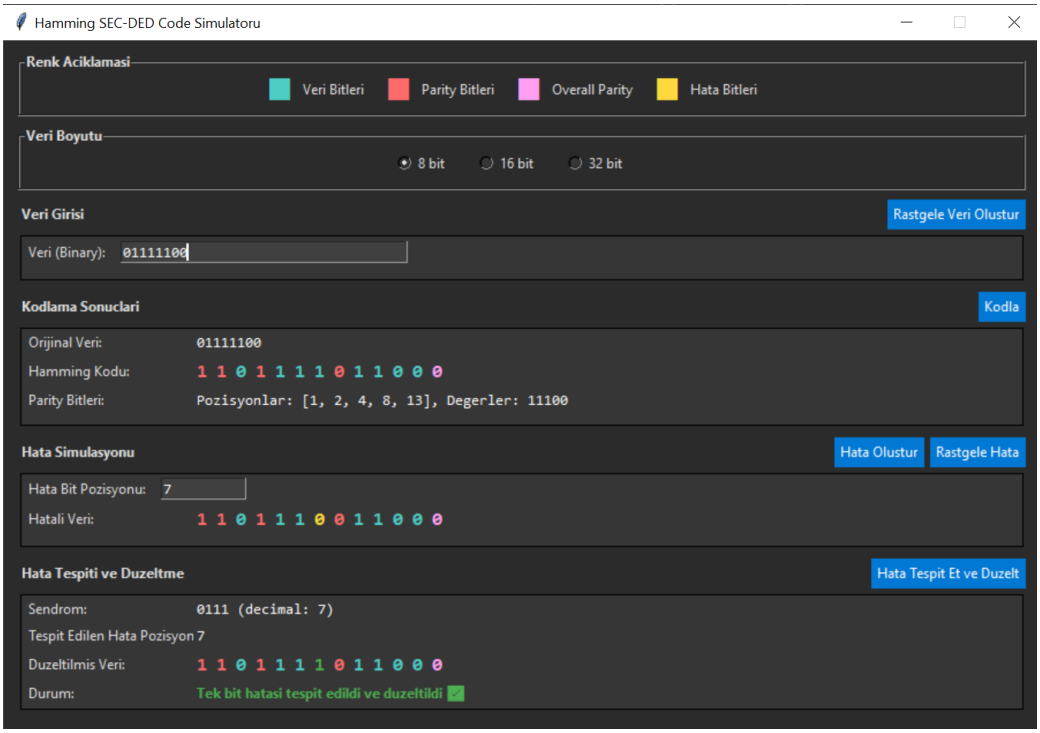
Ana arayüz ve renk açıklaması



8-bit veri kodlama örneği



Hata simülasyonu



Başarılı hata düzeltme

6. SONUÇ VE DEĞERLENDİRME

Geliştirilen Hamming SEC-DED simülatörü, bellek sistemlerinde kullanılan hata düzeltme tekniklerinin eğitim amaçlı görselleştirilmesini başarıyla sağlamaktadır.

Başarılan Hedefler:

- 8, 16, 32 bit veri desteği
- Doğru SEC implementasyonu
- Görsel kullanıcı dostu arayüz
- Tam hata simülasyonu
- Sendrom hesaplama doğruluğu

Teknik Özellikler:

- Python Tkinter GUI framework
- Modüler kod yapısı
- Renkli bit görselleştirme
- Real-time hata tespit/düzeltme

Hamming SEC-DED Code Simülatörü projesi, belirlenen tüm teknik gereksinimleri başarıyla karşılamış ve eğitimsel bir araç olarak yüksek kalitede geliştirilmiştir. Proje, teorik bilgiyi pratik uygulamayla birleştirerek, bilgisayar mimarisi alanında hata düzeltme kodlarının öğrenilmesine önemli katkı sağlamaktadır.

7. KAYNAKLAR VE BAĞLANTILAR

GitHub Repository: *https://github.com/makhsudov/HammingCodeSimulator*
Demo Video: *https://youtu.be/DwC2nwy0Hlw*

Online Kaynaklar

1. **Wikipedia:** "Hamming Code - Error Detection and Correction" *https://en.wikipedia.org/wiki/Hamming_code*
2. **GeeksforGeeks:** "Hamming Code in Computer Network" *https://www.geeksforgeeks.org/hamming-code-in-computer-network/*
3. **Claude:** *https://claude.com/*