8 Haziran 2025

BLM230 Bilgisayar Mimarisi

Hamming SEC DED

Doç. Dr. Erdem YAVUZ

ALEKS DULDA

21360859025

İçindekiler

1. Projenin Amacı

Bu projenin temel amacı, Hamming kodlama algoritmasını kullanarak veri güvenliğini sağlayan bir sistemin C# Windows Forms ortamında simüle edilmesidir. Hamming kodu ile tek bit hatalarının otomatik düzeltilmesi ve çift bit hatalarının tespit edilmesi hedeflenmiştir. Özellikle veri iletiminde karşılaşılabilecek bit hatalarının etkisini azaltmak ve kullanıcıya görsel olarak bu süreci anlatmak amaçlanmıştır.

2. Uygulamanın Kapsamı ve Hedefi

Proje, kullanıcıdan alınan 8, 16 veya 32 bitlik ikili veriyi Hamming kodu (SEC-DED) ile kodlar, yapay olarak kullanıcı tarafından girilen bit pozisyonlarında hata üretir ve bu hataları tespit edip gerekirse düzeltir. Kullanıcının arayüz üzerinden kolaylıkla veri girişi, hata üretimi ve hata kontrolü yapabilmesini sağlar.

Hedefler:

- Kullanıcı dostu bir arayüz sağlamak
- Eğitim ve test amaçlı simülasyon imkânı sunmak
- Hamming kodunun işleyişini adım adım göstermek
- Tek hata düzeltme ve çift hata tespiti (SEC-DED) mantığını göstermek

3. Kullanılan Platform ve Diller

- Programlama Dili: C#
- Uygulama Türü: Windows Forms (WinForms)
- Geliştirme Ortamı: Visual Studio 2022+
- Framework: .NET Framework / .NET Core (WinForms destekli)

Arayüzde kullanılan bileşenler arasında TextBox, ListBox, Button, Label gibi Windows Forms kontrolleri yer alır.

4. Hamming Kodu Nedir?

Hamming kodu, Richard Hamming tarafından geliştirilen bir hata tespit ve düzeltme algoritmasıdır. Temel amacı, veri iletiminde meydana gelen tek bitlik hataları tespit etmek ve düzeltmektir. Hamming kodu, veri bitleri arasına yerleştirilen "parity" (eşlik) bitleri ile çalışır. Bu parity bitleri, belirli bir konumda yer alan verilerin doğruluğunu kontrol etmek için kullanılır.

5. SEC (Single Error Correction) Nasıl Çalışır?

SEC algoritması, veri akışında meydana gelen **tek bir hatayı** tespit ederek düzeltme işlemi yapar. Veri içerisine belirli aralıklarla parity bitleri yerleştirilir ve bu bitlerin değerleri, belirli bir kurala göre hesaplanır. Alıcı tarafında parity bitlerinin kontrolü sonucunda hangi bitin hatalı olduğu bulunur. Hatalı pozisyon XOR işlemiyle tespit edilip otomatik olarak düzeltilir.

6. DED (Double Error Detection) Nasıl Eklenir?

Hamming kodu tek hata düzeltme üzerine kurulu olsa da **genel parity biti** eklenerek çift hata tespiti yapılabilir. Bu ek parity biti tüm bitlerdeki 1 sayısının çift/tek oluşunu kontrol eder. Eğer XOR sonucu ile parity biti kontrolü uyuşmazsa, bu durum sistemin çift hata olduğunu anlamasına yardımcı olur.

Not: Çift hata varsa veri düzeltilmez, kullanıcıya uyarı verilir

7. Genel Parity Biti Neden Gereklidir?

Genel parity biti, sistemin sadece tek hata değil aynı zamanda **çift hatayı da** tespit edebilmesi için eklenir. Hamming kodu varsayımsal olarak sadece tek hatayı düzeltebilir. Ancak bu durumda ikinci bir hata olduğunda yanlış düzeltme yapılabilir. Bu nedenle genel parity biti ile hatanın tek mi çift mi olduğu doğrulanır.

8. Kullanım Kılavuzu

Bu kılavuz, **Hamming SEC-DED Simülatörü** uygulamasının nasıl kullanılacağını adım adım açıklamakta ve karşılaşılabilecek farklı senaryolarda sistemin nasıl davrandığını örneklerle göstermektedir.

8.1. Uygulamayı Başlatma

- WinFormsApp1.sln dosyasını Visual Studio ile açın.
- F5 tuşuna basarak ya da **"Başlat (Start)"** butonuna tıklayarak uygulamayı çalıştırın.

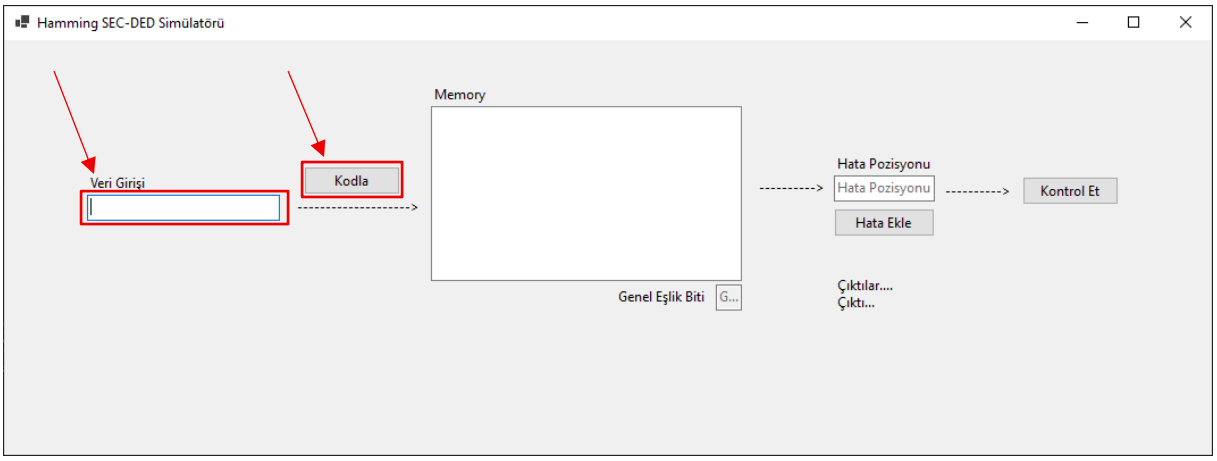
8.2. Veri Girişi

Adımlar:

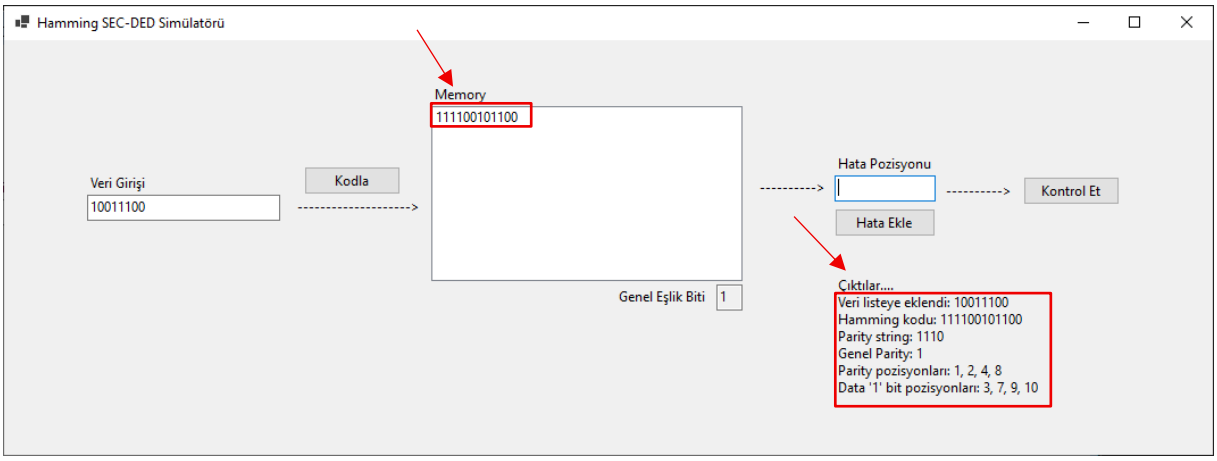
1. **Veri Girişi** alanına yalnızca 0 ve 1 karakterlerinden oluşan bir ikili veri yazın.
2. Veri uzunluğu **8, 16 veya 32 bit** olmak zorundadır. Aksi halde sistem uyarı verir.
3. "**Kodla**" butonuna tıklayın.

Sistem Tepkisi:

- Girilen veri Hamming kodlaması ile işlenir.
- Hamming kodu üretildikten sonra:
    - Kod belleğe eklenir.
    - ListBox içerisinde görünür hale gelir.
    - Genel parity biti otomatik hesaplanır ve arayüzde gösterilir.
    - Eşlik bitleri ve 1 olan bitlerin pozisyonları ayrıntılı olarak kullanıcıya sunulur.



EkranGörüntüsü 1 – Veri girişi



EkranGörüntüsü 2 – Veri Girişi ve Kodlama

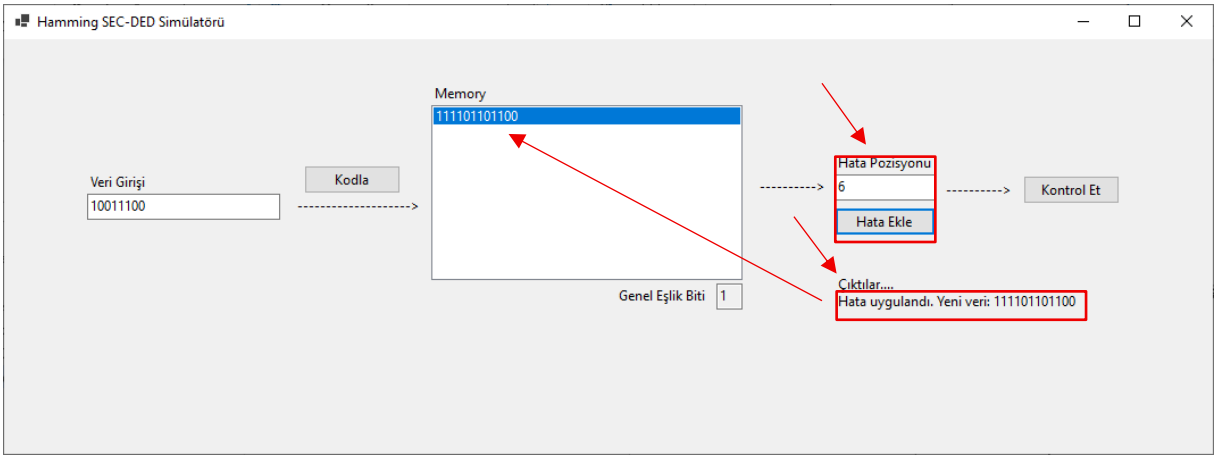
8.3. Hata Ekleme

Adımlar:

1. Bellek listesinde kodlanmış bir veri seçin.
2. Sağ kısımdaki "Hata Pozisyonu" alanına 1'den başlayarak bir bit pozisyonu girin.
3. "**Hata Ekle**" butonuna basın. (Bir şey yazılmazsa hata eklenmez!)

Sistem Tepkisi:

- Seçilen pozisyondaki bit terslenir (0 ↔ 1).
- Güncellenmiş veri bellekteki yerini alır.
- Kullanıcıya güncel veri gösterilir.



EkranGörüntüsü 3 - Hata Ekleme

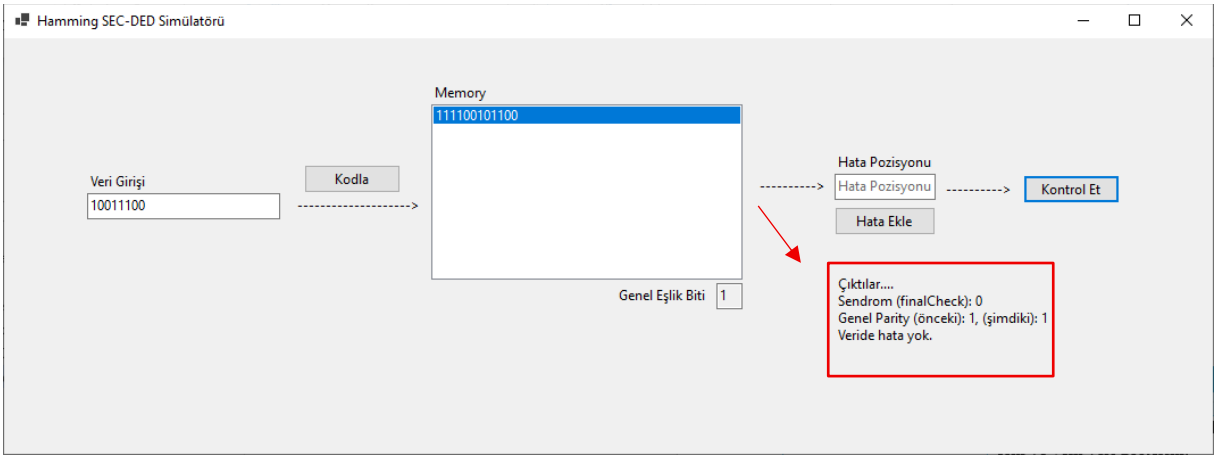
8.4. Hata Kontrolü ve Düzeltme

Adımlar:

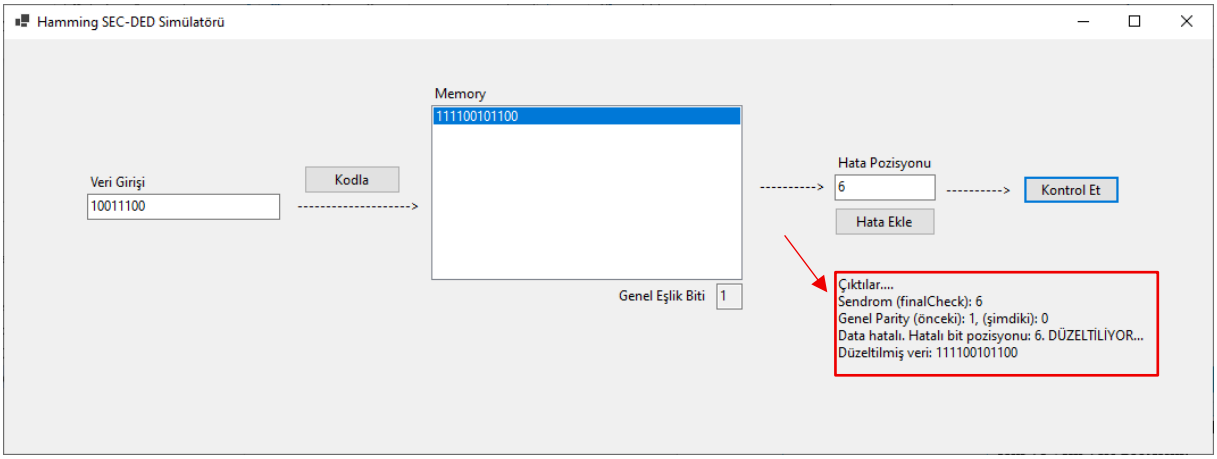
1. Bellekten bir veri seçin.
2. “Kontrol Et” butonuna basın.

Sistem Tepkisi ve Senaryolar:

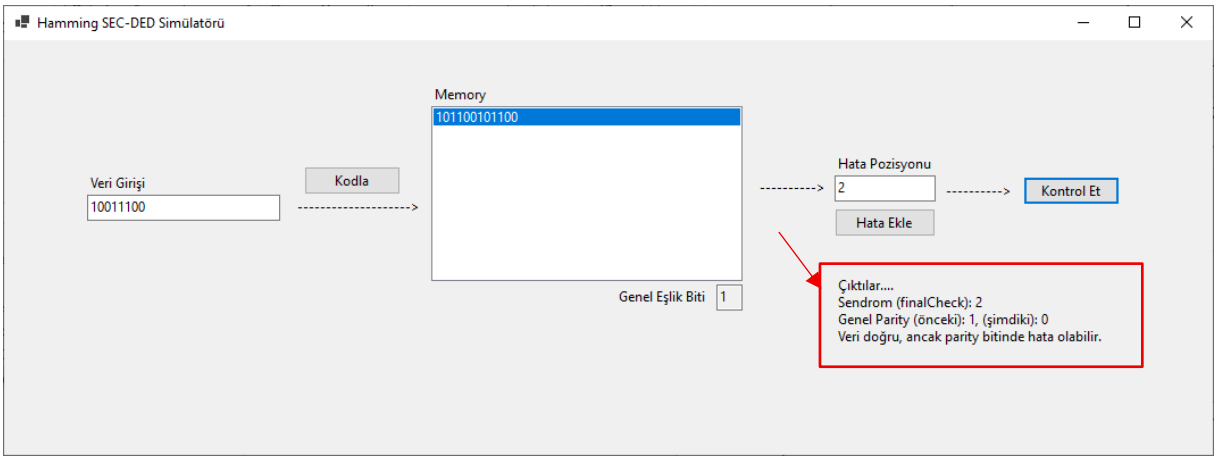
#	Durum	Açıklama
1	**Hiç Hata Yok**	Kod düzgünse: "Veride hata yok" mesajı verilir.
2	**Tek Hata Varsa**	Hatalı bit pozisyonu tespit edilir ve otomatik düzeltilir. Kullanıcıya eski ve yeni veri gösterilir.
3	**Parity Bitinde Hata**	Tek bit hatası parity alanındaysa sadece bilgi verilir, düzeltme yapılmaz.
4	**Çift Hata Varsa**	Sendrom değeri ve genel parity biti karşılaştırılır. Eğer uyuşmazlık varsa: "Çift hata tespit edildi, düzeltilemez" mesajı gösterilir.



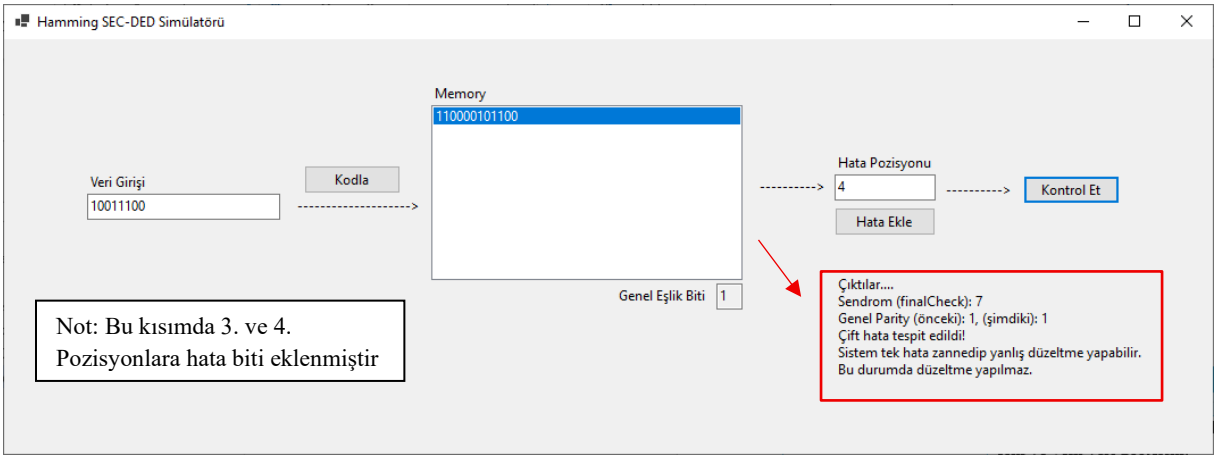
EkranGörüntüsü 4 – Senaryo 1



EkranGörüntüsü 5 – Senaryo 2



EkranGörüntüsü 6 – Senaryo 3



EkranGörüntüsü 7 – Senaryo 4

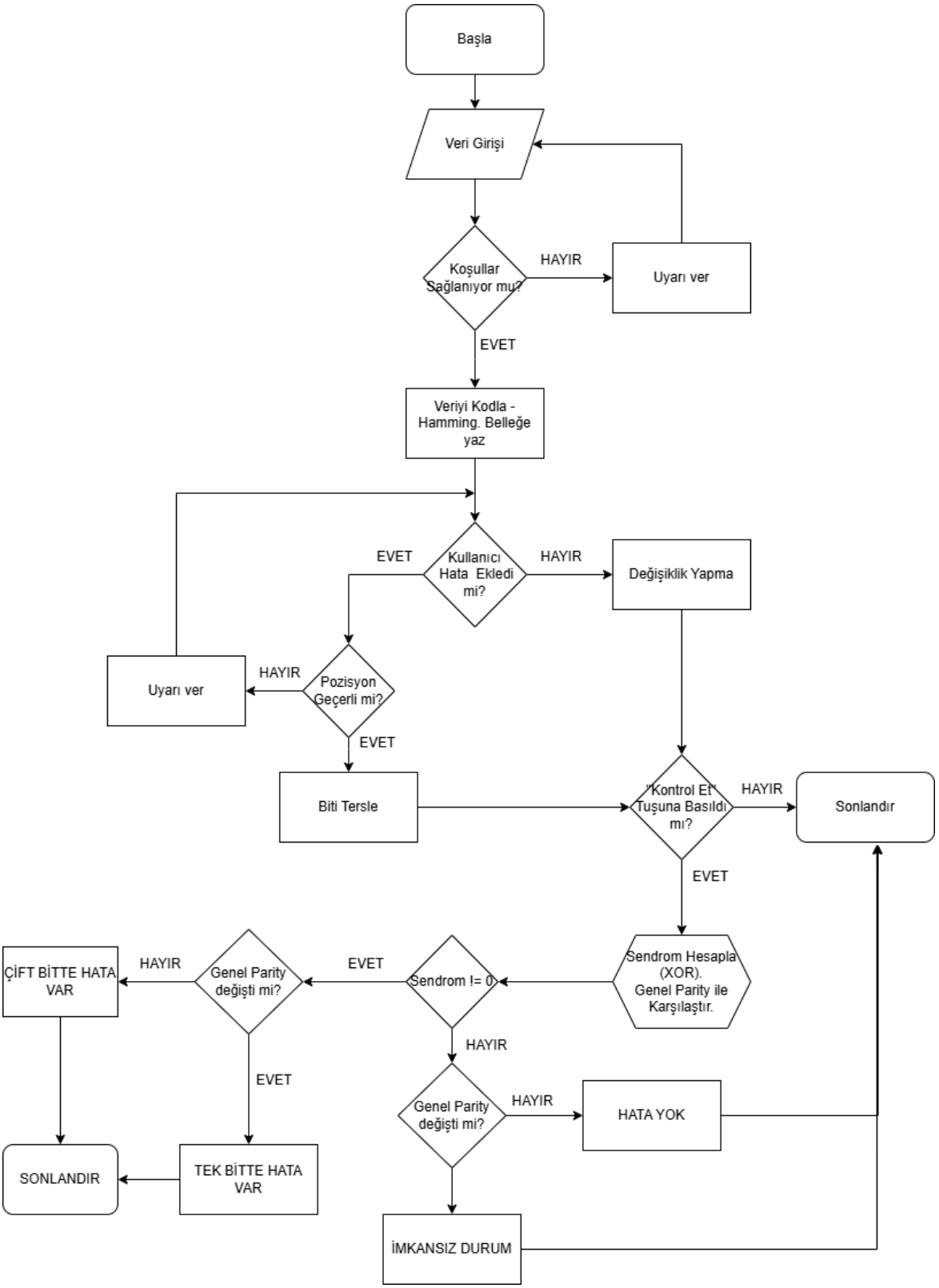
8.5. Bellek Yönetimi

- Her kodlanan veri belleğe eklenir.
- Hatalı ve düzeltilmiş veriler de güncellenmiş şekilde aynı listede tutulur.
- Bellekten bir veri seçilmeden hata ekleme veya kontrol işlemi yapılamaz.

8.6. Kullanıcı Arayüzü Açıklaması

Bileşen	Görevi
`inputTextBox`	Veri giriş alanı (0 ve 1)
`encodeButton`	Veri kodlama butonu
`memoryListBox`	Kodlanan verilerin saklandığı liste
`errorPosTextBox`	Hata ekleme için pozisyon girme alanı
`errorButton`	Belirtilen pozisyondaki biti tersler
`checkButton`	Hata kontrolü ve gerekirse düzeltme yapar
`generalParityTextBox`	Genel parity değerini gösterir
`outputLabel`	Sistem mesajlarını ve analiz sonuçlarını gösterir

9. FlowChart (Akış Diyagramı)



Şekil 1: Hamming SEC-DED algoritmasının yazılım üzerindeki akış diyagramı

10. Sonuç ve Değerlendirme

Bu proje kapsamında Hamming SEC-DED algoritması başarılı bir şekilde uygulanmış ve görsel bir simülasyon aracı geliştirilmiştir. Tek hata düzeltme ve çift hata tespit işlemleri, kullanıcı etkileşimli bir şekilde yönetilmiş ve her işlem için detaylı çıktı analizi yapılmıştır. Özellikle mühendislik öğrencileri için eğitim aracı olarak işlev görebilecek bu yazılım, algoritma mantığını somut olarak anlamaya büyük katkı sağlamaktadır.

11. Kaynakça

- Hamming, R. W. (1950). Error Detecting and Error Correcting Codes. Bell System Technical Journal, 29(2), 147–160..
- Stallings, W. (2016). Computer Organization and Architecture (10th ed.). Pearson Education.
- Microsoft Docs. (2024). System.Windows.Forms Namespace.
- Wikipedia contributors. (2024). Hamming code. Wikipedia, The Free Encyclopedia.