T.C
KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR/YAZILIM MÜHENDİSLİĞİ

PROJE KONUSU:
Lab-II 3.Proje

ÖĞRENCİ ADI: Kaan ÖZDEMİR-Emin Kayra ERTEKİN
ÖĞRENCİ NUMARASI: 220501042- 220501032
https://github.com/eminkay/
https://github.com/kaanozdmr

DERS SORUMLUSU:
PROF. DR./DR. ÖĞR. ÜYESİ Ercan ÖLÇER

TARİH:02.06.2024

1 GİRİŞ

1.1 Projenin amacı

- Bu proje, dijital devre elemanlarını ve lojik kapılarını simüle etmek ve etkileşimli bir şekilde çalıştırmak için bir grafiksel kullanıcı arayüzü geliştirmeyi amaçlar. Pygame kütüphanesi kullanılarak gerçekleştirilen bu proje, kullanıcıların dijital devreler oluşturmasına, simüle etmesine ve analiz etmesine olanak tanır. Temel mantık kapılarının (AND, OR, NOT, XOR, NAND, NOR, XNOR, Buffer) davranışlarını görsel olarak göstermeye yardımcı olur.
- Bu projede sizden basit mantık devrelerini tasarlamak için bir platformu geliştirmeniz beklenmektedir. Program
- ile ilgili kısıtlar aşağıdaki gibidir:
- Geliştirilecek olan platformun araçlar kısmında aşağıdaki özellikler bulunmalıdır:
- Mantık kapıları (Her bir mantık kapısını eklemek için kullanılacak araçlar)
  Her bir aracın kendine ait bir özellik tablosu olmalıdır:
- Mantık kapıları için; etiket(B) ve giriş bağlantı sayısı.
- Giriş çıkış elemanları için; etiket(B), renk(B) ve başlangıç(sadece giriş elemanları için) değeri.
- Bağlantı elemanları için; etiket(B) ve renk(B).
- Yanında (B) etiketi bulunanlar bonus olarak değerlendirilecektir.
- Özellik tablosundaki değerler tasarım alanına yerleştirilen elemanlar üzerinde sağ fare tuşu ile
- tıklanarak görüntülenebilir ve değiştirilebilir.
- Tasarım alanına aynı devre elemanından birden fazla eklenebilir.
- Bağlantı hatlarının herhangi bir noktada birleştirilmesi gerekiyorsa bağlantı düğümü kullanılmalıdır.
- Tasarım işlemi bittikten sonra platformda yer alan çalıştır, reset ve durdur tuşları ile devrenin çalışması
- simüle edilmelidir.
- Simülasyon esnasında kullanıcı her bir elemanın giriş kutusunu seçerek giriş değerini değiştirebilir.
- Kullanıcı devrenin çıkışında led veya çıkış kutusu kullanabilir. Çıkış kutusu çıkış değerini göstermelidir.
- Led konulduğunda ise devrenin çıkış değerine göre ledin yanması ve sönmesi sağlanmalıdır.

2 GEREKSİNİM ANALİZİ

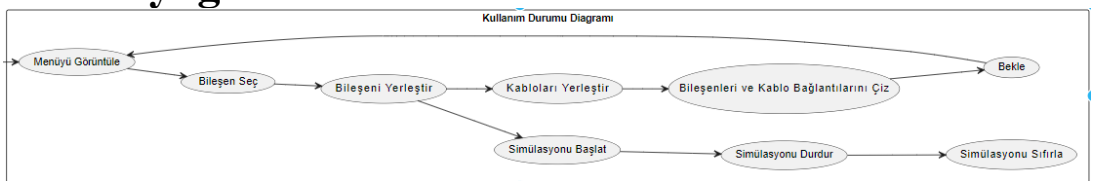
2.1 Arayüz gereksinimleri

- Ekranın üst kısmında, farklı kapı türleri ve diğer bileşenlerin (AND, NOT, OR, NOR, XOR, XNOR, NAND, BUFF, Düğme, Çıkış, LED, Kablolar, Mouse, Düğüm) yer aldığı bir menü çubuğu vardır.
- Kapı ve bileşenlerin görselleri kullanıcı tarafından sürüklenip bırakılabilmektedir.
- Menü çubuğunun altında, "Çalıştır", "Reset" ve "Durdur" butonları yer almaktadır

2.2 Fonksiyonel gereksinimler

- Devre elemanlarını ve kabloları ekleyebilmek.
- Eklenen devre elemanlarını ve kabloları silme veya değiştirme yeteneği.
- Devreyi simüle ederek, elemanların ve kabloların gerçek zamanlı durumunu görebilmek.
- Simülasyonun gerçekçi sonuçlar üretmesi ve gerçek devrelerin davranışlarını taklit etmesi.
- Kullanıcı dostu arayüzle kullanıcıların kolayca devre oluşturabilmesi.

2.3 Use-Case diyagramı

- 

3 TASARIM

3.1 Mimari tasarım

- **Sınıf Yapısı:** Kod, devre elemanlarını temsil etmek için farklı sınıflar kullanır. Her sınıf, ilgili elemanın çizimini yapabilir ve durumunu kontrol edebilir.
- **Devre Elemanlarının Oluşturulması:** Her bir devre elemanı, türüne bağlı olarak farklı giriş ve çıkışlara sahip olabilir. Bu bilgi, elemanların oluşturulması ve bağlantılarının yönetilmesi sırasında önemlidir.
- **Kablo Yönetimi:** Kablo nesneleri, bağlı oldukları devre elemanlarını ve durumlarını izler. Kullanıcı, kabloları sürükleyerek ve bırakarak devre elemanları arasında bağlantılar oluşturabilir.
- **Simülasyon Kontrolleri:** Kullanıcı, devreyi simüle etmek için düğmelere tıklayabilir. Bu işlevsellik, devrenin nasıl çalıştığını görselleştirmek için kullanılır.
- **Ekran Güncelleme ve Kullanıcı Etkileşimi:** Ekran, kullanıcı

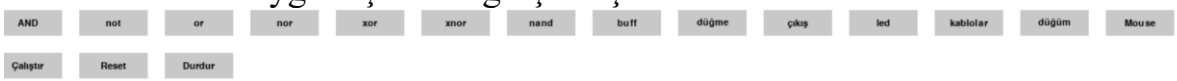
etkileşimlerine ve devre elemanlarının durumlarına göre düzenli olarak güncellenir.

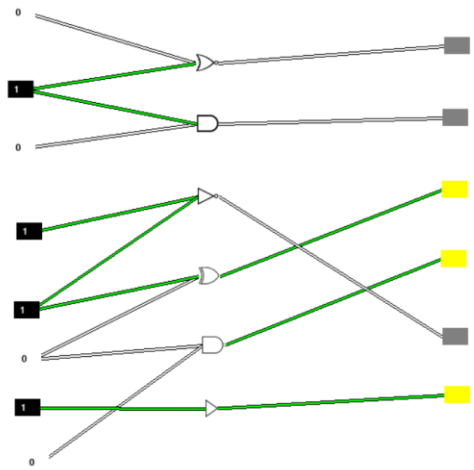
3.2 Kullanılacak teknolojiler

- Kodumuz python dilinde yazılmıştır.
- Kodumuzda **Pygame** kütüphanesi kullanılmıştır , Python programlama dilinde oyun ve çoklu ortam uygulamaları geliştirmek için kullanılan açık kaynaklı bir kütüphanedir. Temel olarak, oyun geliştirme için bir araç seti sunar ve grafik, ses, girdi kontrolü ve oyun mantığı gibi birçok farklı bileşeni kapsar.
- **pygame.locals** modülü, Pygame kütüphanesinin çeşitli sabitlerini ve özel veri tiplerini içeren bir alt modüldür. Bu sabitler, Pygame oyun geliştirme sürecinde kullanılan çeşitli olaylar, tuşlar, fare düğmeleri vb. için sembolik isimler içerir.

3.3 Kullanıcı arayüzü tasarımı

- **Pencere Oluşturma**: Pygame kütüphanesi kullanılarak bir pencere oluşturulur.
- **Elemanları Çizme**: Menüdeki elemanlar, fare ile seçilip sürüklendikten sonra ekrana çizilir.
- **Bağlantılar Oluşturma**: Kullanıcı, elemanlar arasında bağlantılar kurabilir.
- **Simülasyon Başlatma ve Durdurma**: Kullanıcı, devre simülasyonunu başlatabilir, durdurabilir veya sıfırlayabilir.
- **Kullanıcı Etkileşimi İzleme**: Fare tıklamaları ve fare hareketleri izlenir ve uygun işlemler gerçekleştirilir.





4 UYGULAMA

4.1 Kodlanan bileşenlerin açıklamaları

- **Kapı Sınıfları: andkapı, nandkapı, orkapı, norkapı, xorkapı,**

xnorKapı, **notkapı**, **bufferkapı**, **çıkışkapı**, **düğme**, **led** ve **düğüm**. Her bir sınıf, belirli bir mantıksal kapı türünü temsil eder. Her kapı sınıfı, kapının türüne göre girişlerini ve çıkışlarını yönetir. Ayrıca, kapının durumunu kontrol etmek için bir **kontrol** yöntemi içerir.

- **Düğüm Sınıfı (düğüm):** Bu sınıf, devrede bir düğümü temsil eder. Düğümler, kapıların giriş ve çıkışlarını birbirine bağlamak için kullanılır.
- **Kablo Sınıfı (kablo):** Bu sınıf, kapılar arasındaki bağlantıyı temsil eder. Bağlantıların durumunu takip eder ve devrenin çizimini yaparken bu bağlantıları görselleştirir.
- **Çizim ve Simülasyon Fonksiyonları: Menu**, **öğeyerleştir**, **kabloyerleştir**, **nesneler** vb. adlı fonksiyonlar, kullanıcı arayüzünü oluşturmak ve devreyi çizmek için kullanılır. Ayrıca, simülasyonu başlatmak, duraklatmak veya sıfırlamak için gerekli işlevselliği sağlarlar.
- **Ana Döngü: while True** döngüsü, programın sürekli olarak çalışmasını sağlar. Bu döngü, kullanıcı etkileşimlerini dinler, çizimleri günceller ve simülasyonu yönetir.
-

4.2 Görev dağılımı

- Kod dağılımı şu şekildedir: kodun %60'ına yakınını Kaan ÖZDEMİR, kalan %40 ve raporu ise Emin Kayra ERTEKİN yapmıştır.

4.3 Karşılaşılan zorluklar ve çözüm yöntemleri

- Sağ tıklandığında ekranda doğruluk tablosunu yazdıramadık, bulduğumuz çözüm ise kapılara sağ tıklandığında terminalde doğruluk tablosu yazdırılıyor.

4.4 Proje isterlerine göre eksik yönler

- Bonus özellikler ve etiket yapılmadı.
-