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

PROJE KONUSU:

SAYISAL TASARIM PROJESİ

ÖĞRENCİ ADI:

ZEHRA YARDIMCI

ÖĞRENCİ NUMARASI:

220502038

DERS SORUMLUSU:
PROF. DR. NEVCİHAN DURU

01.06.2024

1 GİRİŞ

1.1 Projenin amacı

Bu projenin amacı, basit mantık devrelerini tasarlamak için bir platform geliştirmektir. Bu platform, kullanıcıların çeşitli mantık kapıları, giriş çıkış elemanları ve bağlantı elemanları kullanarak devre tasarlamalarına ve simülasyon yapmalarına olanak tanır.

Projede gerçekleştirilmesi beklenenler:

- Mantık kapılarının eklenmesi için gerekli araçların sağlanması.
- Giriş ve çıkış elemanlarının eklenmesi ve özelliklerinin ayarlanabilmesi.
- Bağlantı elemanlarının eklenmesi ve devre elemanlarının birbirine bağlanabilmesi.
- Tasarım alanında yer alan elemanların özellik tablolarının oluşturulması ve bu özelliklerin sağ tıklama ile görüntülenip değiştirilebilmesi.
- Aynı devre elemanından birden fazla eklenebilmesi.
- Bağlantı hatlarının birleşme noktalarında bağlantı düğümlerinin kullanılması.
- Devre tasarımı tamamlandıktan sonra çalıştırma, reset ve durdur tuşları ile simülasyon yapılabilmesi.
- Simülasyon esnasında her bir elemanın giriş kutusunun seçilerek giriş değerinin değiştirilebilmesi.
- Çıkışta LED veya çıkış kutusu kullanılarak çıkış değerlerinin gösterilmesi ve LED'in yanıp sönmesi.

2 GEREKSİNİM ANALİZİ

2.1 Arayüz gereksinimleri

Kullanıcı Arayüzü Gereksinimleri:

Mantık Kapıları Ekleme Aracı:

- Kullanıcı, mantık kapılarını (AND, OR, NOT, NAND, NOR, XOR, XNOR) tasarım alanına sürükleyip bırakabilmelidir.
- Her bir mantık kapısı, etiketlenebilir ve giriş bağlantı sayısı ayarlanabilir olmalıdır.

Giriş/Çıkış Elemanları:

- Giriş elemanları ve çıkış elemanları tasarım alanına eklenebilmelidir.
- Giriş elemanları için etiket, renk ve başlangıç değeri belirlenebilmelidir.
- Çıkış elemanları için etiket ve renk belirlenebilmelidir.

Bağlantı Elemanları:

- Bağlantı hatları, devre elemanlarını birbirine bağlamak için kullanılabilmelidir.
- Bağlantı hatlarının renkleri ve etiketleri ayarlanabilir olmalıdır.
- Bağlantı noktalarında düğüm kullanılarak hatlar birleştirilebilmelidir.

Kontrol Tuşları:

- Çalıştır, reset ve durdur tuşları ile devrenin simülasyonu kontrol edilebilmelidir.
- Simülasyon sırasında kullanıcı giriş değerlerini değiştirebilmelidir.

Özellik Tabloları:

- Her bir devre elemanının özellikleri, sağ tıklama ile açılan bir özellik tablosunda görüntülenebilir ve değiştirilebilir olmalıdır.

- Tasarım alanına eklenen her bir elemanın özellik tablosuna erişilebilir olmalıdır.

Tasarım Alanı:

- Kullanıcı, tasarım alanında birden fazla aynı devre elemanını ekleyebilmelidir.

- Tasarım alanında devre elemanları serbestçe taşınabilir ve yerleştirilebilir olmalıdır.

2.2 Fonksiyonel gereksinimler

Mantık Kapılarının Eklenmesi:

- Kullanıcı, tasarım alanına AND, OR, NOT, NAND, NOR, XOR ve XNOR kapılarını ekleyebilmelidir.

- Her bir mantık kapısı için etiket ve giriş bağlantı sayısı ayarlanabilir olmalıdır.

Giriş Elemanlarının Eklenmesi:

- Kullanıcı, tasarım alanına butonlar ve anahtarlar gibi giriş elemanlarını ekleyebilmelidir.

- Giriş elemanları için etiket, renk ve başlangıç değeri belirlenebilir olmalıdır.

Çıkış Elemanlarının Eklenmesi:

- Kullanıcı, tasarım alanına LED'ler ve çıkış kutuları gibi çıkış elemanlarını ekleyebilmelidir.

- Çıkış elemanları için etiket ve renk belirlenebilir olmalıdır.

- LED'lerin yanma ve sönme durumu, çıkış değerine göre simüle edilebilmelidir.

Bağlantı Elemanlarının Eklenmesi:

- Kullanıcı, devre elemanlarını birbirine bağlamak için bağlantı hatları ekleyebilmelidir.
- Bağlantı hatları etiketlenebilir ve renkleri ayarlanabilir olmalıdır.
- Bağlantı noktalarında düğümler kullanılarak hatlar birleştirilebilir olmalıdır.

Kontrol Tuşları:

- Devrenin simülasyonunu başlatmak için çalıştır tuşu olmalıdır.
- Devreyi sıfırlamak için reset tuşu olmalıdır.
- Simülasyonu durdurmak için durdur tuşu olmalıdır.

Özellik Tabloları:

- Kullanıcı, tasarım alanına eklenen her bir devre elemanının özelliklerini görüntüleyip değiştirebilmelidir.
- Özellik tabloları, sağ tıklama ile açılabilmelidir.

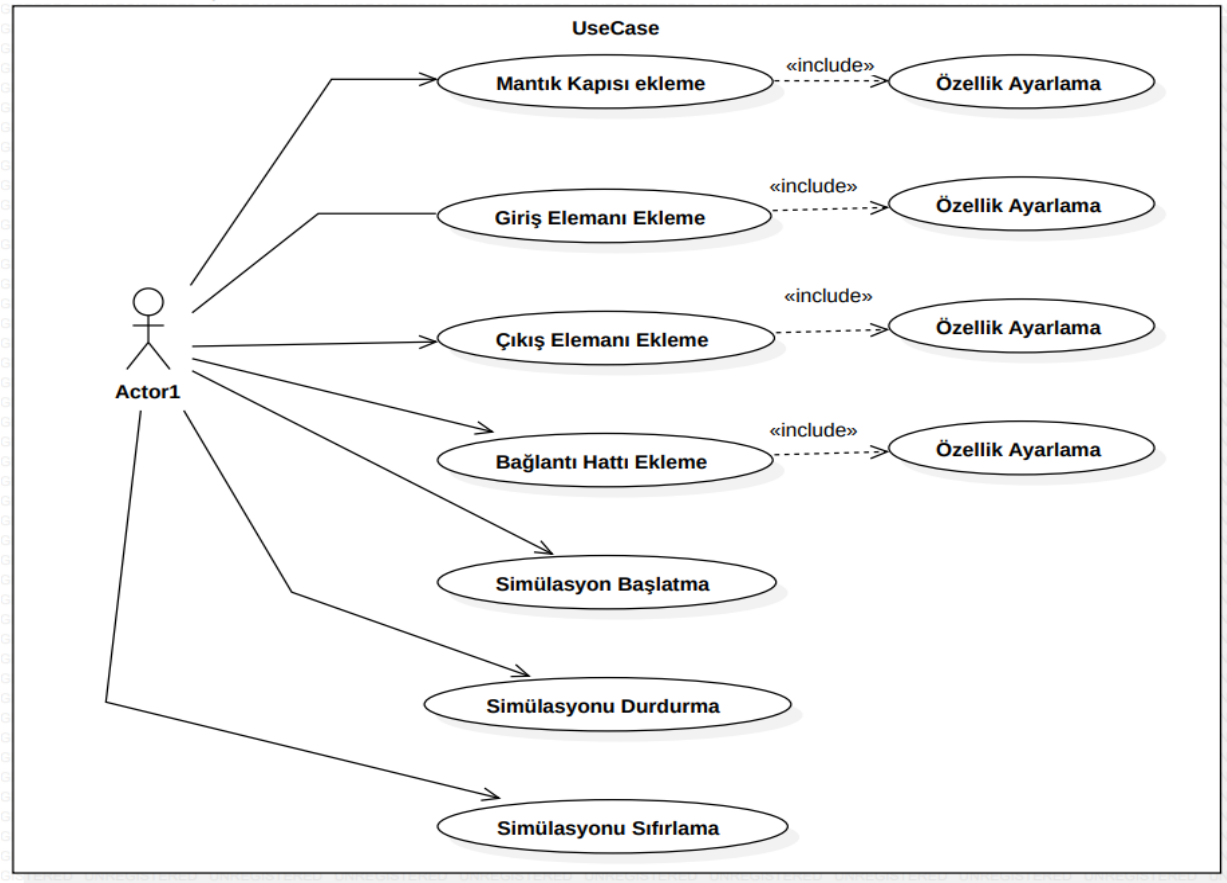
Simülasyon:

- Kullanıcı, simülasyon esnasında giriş elemanlarının değerlerini değiştirebilmelidir.
- Simülasyon sonucunda çıkış elemanlarının doğru şekilde yanıp sönmesi veya değer göstermesi sağlanmalıdır.

Tasarım Alanı:

- Tasarım alanında birden fazla aynı devre elemanı eklenebilir ve taşınabilir olmalıdır.
- Kullanıcı, tasarım alanında elemanları serbestçe yerleştirebilmelidir.

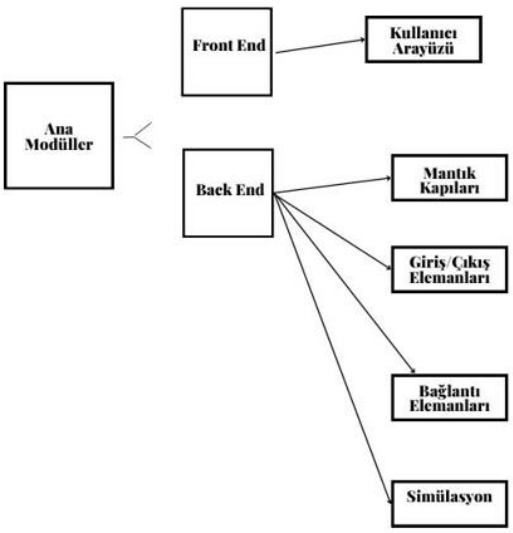
2.3 Use-Case diyagramı



3 TASARIM

3.1 Mimari tasarım

- Modül diyagramı:



Kullanıcı Arayüzü (UI) Katmanı:

- Kullanıcıların mantık kapılarını, giriş/çıkış elemanlarını ve bağlantı elemanlarını tasarım alanına ekleyebileceği ve düzenleyebileceği grafiksel kullanıcı arayüzü sağlanmalıdır.

- Kullanıcı arayüzü, simülasyon kontrol tuşlarını (çalıştır, reset, durdur) içermelidir.

- Elemanların özelliklerini görüntülemek ve değiştirmek için sağ tıklama menüleri bulunmalıdır.

Mantık Kapıları ve Elemanlar:

- Mantık kapıları (AND, OR, NOT, vb.) ve giriş/çıkış elemanları (LED'ler,giriş kutuları, çıkış kutuları) için sınıflar tanımlanmalıdır.

- Her bir mantık kapısı ve eleman, etiket, giriş bağlantı sayısı ve renk gibi özelliklere sahip olmalıdır.

- Mantık kapıları ve elemanlar, tasarım alanına sürükle bırak yöntemiyle eklenebilir olmalıdır.

Bağlantı Sistemi:

- Elemanlar arasındaki bağlantıları temsil eden bağlantı hatları ve düğümleri için sınıflar ve veri yapıları tanımlanmalıdır.

- Bağlantılar, tasarım alanında kullanıcı tarafından oluşturulabilmeli ve düzenlenebilir olmalıdır.

- Bağlantı düğümleri, birden fazla bağlantı hattının birleştirilebilmesi için kullanılmalıdır.

Simülasyon:

- Devrenin doğru çalışmasını sağlamak için bir simülasyon geliştirilmelidir.

- Simülasyon, mantık kapıları ve bağlantılar arasındaki veri akışını yönetmelidir.

- Kullanıcı giriş değerlerini değiştirdiğinde, simülasyon bu değişiklikleri algılamalı ve devreyi güncellemeli, sonuç olarak çıkış değerlerini hesaplamalıdır.

3.2 Kullanılacak teknolojiler

Yazılım Dili:

- Proje Python dilinde yazılmıştır.

Kullanılacak Harici Kütüphaneler

- **Tkinter:** Kullanıcı arayüzü oluşturmak için Tkinter kütüphanesi kullanılmıştır. Tkinter, Python'un standart GUI (grafiksel kullanıcı arayüzü) kütüphanesidir ve basit pencere uygulamaları geliştirmek için kullanılır.

- **import tkinter as tk**

- **from tkinter import ttk**

Diğer Teknolojiler

- **Nesne Yönelimli Programlama (OOP):** Projede mantık kapıları ve diğer elemanlar için nesne yönelimli programlama teknikleri kullanılmaktadır.

- Sınıflar ve nesneler kullanılarak mantık kapılarının davranışları ve özellikleri tanımlanmıştır. Örneğin, **LogicGate** sınıfı, mantık kapılarının temel özelliklerini ve davranışlarını tanımlar.

- **Etkinlik Bağlama:** Tkinter kullanılarak, kullanıcı etkileşimleri için etkinlik bağlama yöntemi kullanılmaktadır. Bu, fare tıklamaları ve sürükleme gibi kullanıcı etkileşimlerini yönetmek için kullanılır.

- **self.canvas.tag_bind(self.id, '<B1-Motion>', self.move)** gibi komutlar ile kullanıcı arayüzündeki elemanların hareketi sağlanmıştır.

3.3 Kullanıcı arayüzü tasarımı

Kullanıcı Arayüzü Tasarımı

- Kullanıcı arayüzü, kullanıcıların mantık devrelerini kolayca tasarlayabilmesi ve simüle edebilmesi için tasarlanmıştır.

- Tasarımda şu bileşenler yer almaktadır:

Tasarım Alanı:

- Kullanıcıların mantık kapılarını ve diğer devre elemanlarını ekleyebileceği geniş bir alan sağlar.

- Elemanlar, sürükle-bırak yöntemiyle tasarım alanına eklenebilir ve konumları değiştirilebilir.

Araç Çubuğu:

- Mantık kapıları (AND, OR, NOT vb.) ve diğer devre elemanları (giriş, çıkış, bağlantı) için araçlar içerir.

- Kullanıcı, araç çubuğundaki elemanları seçip tasarım alanına ekleyebilir.

Özellik Tablosu:

- Tasarım alanında yer alan her bir elemanın özelliklerini görüntülemek ve değiştirmek için kullanılabilir.

- Elemanlar sağ tıklanarak özellik tablosu açılabilir.

Simülasyon Kontrolleri:

- Çalıştır, reset ve durdur tuşları ile devrenin çalışması simüle edilir.

- Kullanıcı, simülasyon sırasında giriş değerlerini değiştirebilir ve çıkış değerlerini gözlemleyebilir.

Uygulamayı çalıştırmak için aşağıdaki adımları izleyebilirsiniz:

Gerekli Kütüphanelerin Yüklenmesi:

- Python ve Tkinter kütüphaneleri sisteminizde yüklü olmalıdır.

Kullanıcı Arayüzü ile Etkileşim:

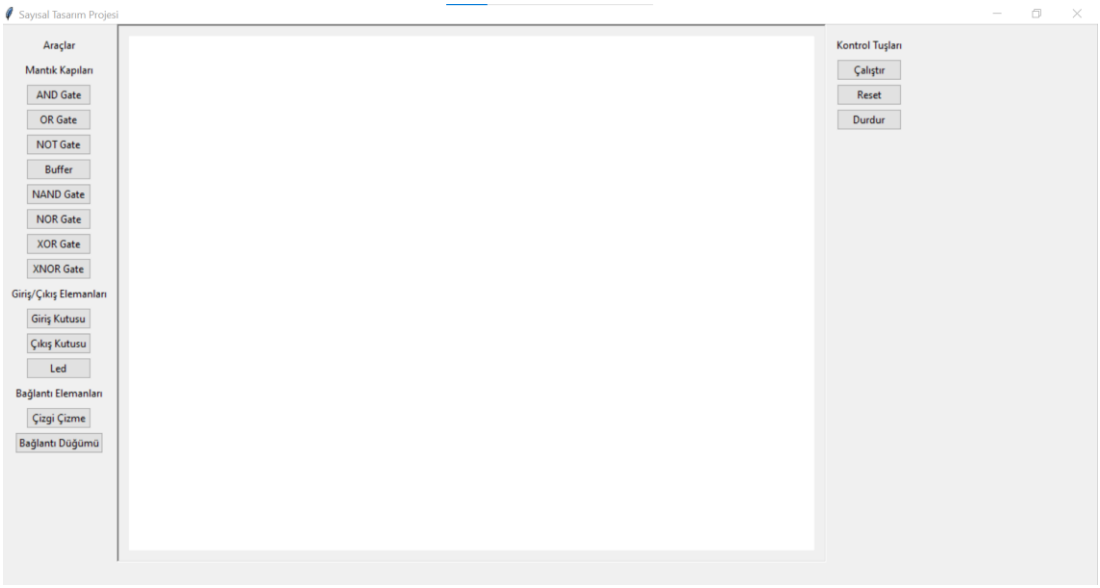
- Uygulama açıldıktan sonra, araç çubuğundaki elemanları seçip tasarım alanına ekleyerek devre tasarımınıza başlayabilirsiniz.

- Elemanların özelliklerini görüntülemek ve değiştirmek için sağ tıklama menülerini kullanabilirsiniz.

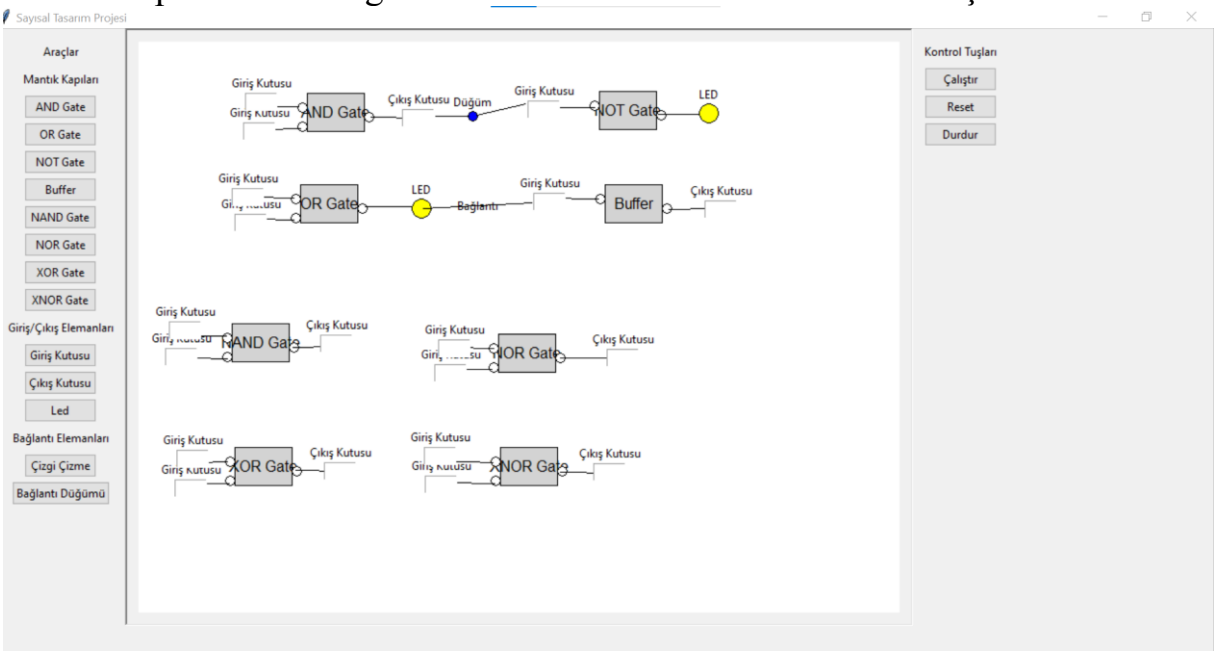
- Tasarımınızı tamamladıktan sonra, çalıştır tuşuna basarak devrenin simülasyonunu başlatabilirsiniz.

Ana Ekran:

Tasarım alanı ve araç çubuğu ile kullanıcı arayüzünün genel görünümü:

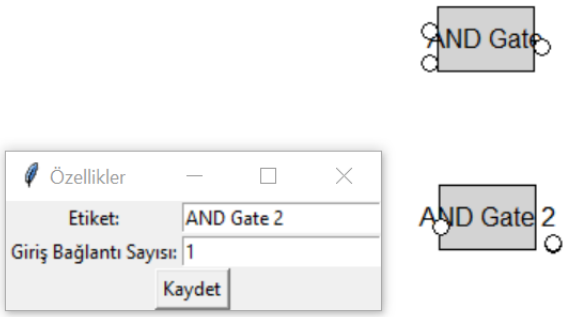


Mantık kapılarının ve diğer elemanların tasarım alanına eklenmiş hali:

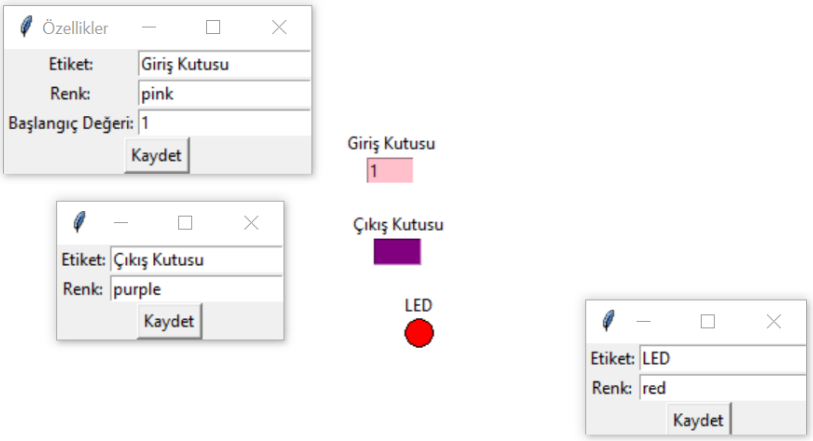


Eleman Özellikleri:

Bir mantık kapısının sağ tıklanarak özellik tablosunun açıldığı ekran:



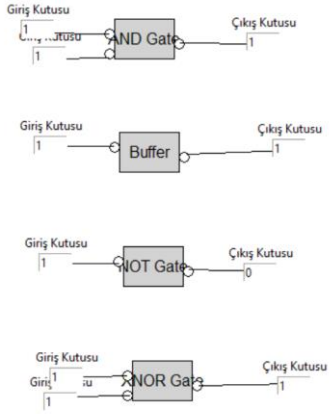
Giriş ve çıkış elemanlarının özelliklerinin gösterildiği ve düzenlendiği tablo :



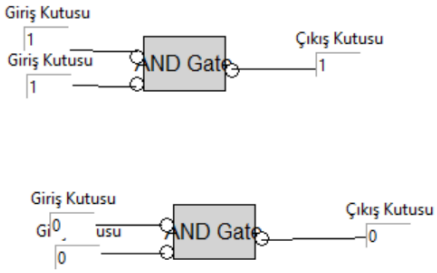
Bağlantı elemanlarının özelliklerinin gösterildiği tablosunun açıldığı ekran:

Simülasyon:

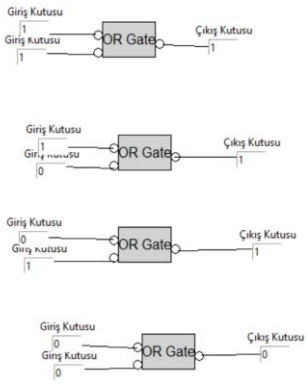
Çalıştır tuşuna basıldıktan sonra devrenin simülasyonunun başladığı ekran görüntüsü:



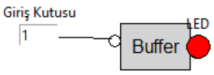
Giriş değerlerinin değiştirilmesi ve çıkış değerlerinin gözlemlendiği ekran görüntüsü:

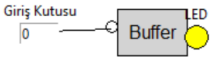


Tasarım alanına aynı devre elemanından birden fazla eklenebileceğinin görüntüsü:



Çıkış Elemanı olarak LED Kullanılabilirliği:





4 UYGULAMA

4.1 Kodlanan bileşenlerin açıklamaları

Sınıflar ve Fonksiyonlar

- **LogicGate Sınıfı**

  Mantık kapılarının temel özelliklerini ve davranışlarını tanımlar.

  Özellikler:

  - **canvas**: Mantık kapısının çizileceği Tkinter kanvas nesnesi.
  - **gate_type**: Mantık kapısının türü (AND, OR, NOT, vb.)
  - **x, y**: Mantık kapısının konum koordinatları.
  - **id**: Mantık kapısının dikdörtgen temsilcisi.
  - **text_id**: Mantık kapısının metin etiketi.
  - **input_ids**: Mantık kapısının giriş portlarının listesi.
  - **output**: Mantık kapısının çıkış değeri.
  - **connected_gates**: Mantık kapısına bağlı diğer kapıların listesi.

Yöntemler:

- **__init__(self, canvas, x, y, gate_type)**: Sınıfı başlatır ve mantık kapısını çizer.
- **create_input_output_ports(self)**: Mantık kapısının giriş ve çıkış portlarını oluşturur.
- **on_start_drag(self, event)**: Mantık kapısının sürüklenmesini başlatır.
- **move(self, event)**: Mantık kapısını taşır.
- **on_drop(self, event)**: Mantık kapısının taşınmasını durdurur.
- **show_properties(self, event)**: Mantık kapısının özelliklerini görüntüler.
- **start_connection(self, event)**: Bağlantı oluşturma işlemini başlatır.

o **BinaryGate Sınıfı (LogicGate'den Türemiş)**

İki girişli mantık kapılarını temsil eder.

Özellikler:

- **LogicGate** sınıfının tüm özelliklerini miras alır.
- İki giriş portu içerir.

Yöntemler:

- **__init__(self, canvas, x, y, gate_type)**: **LogicGate** sınıfının yapıcı metodunu çağırır ve iki giriş portu oluşturur.

o **AndGate Sınıfı (BinaryGate'den Türemiş)**

AND mantık kapısını temsil eder. İki girişten her ikisi de 1 olduğunda çıkışı 1 olan bir kapıdır.

Özellikler:

- **BinaryGate** sınıfının tüm özelliklerini miras alır.

- AND kapısı için özel çıkış değeri hesaplama içerir.

Yöntemler:

- **__init__(self, canvas, x, y)**: **BinaryGate** sınıfının yapıcı metodunu çağırır ve AND kapısını başlatır.
- **calculate_output(self)**: AND kapısının çıkış değerini hesaplar.

- **OrGate Sınıfı (BinaryGate'den Türemiş)**

OR mantık kapısını temsil eder. İki girişten en az biri 1 olduğunda çıkışı 1 olan bir kapıdır.

Özellikler:

- **BinaryGate** sınıfının tüm özelliklerini miras alır.
- OR kapısı için özel çıkış değeri hesaplama içerir.

Yöntemler:

- **__init__(self, canvas, x, y)**: **BinaryGate** sınıfının yapıcı metodunu çağırır ve OR kapısını başlatır.
- **calculate_output(self)**: OR kapısının çıkış değerini hesaplar.

- **NotGate Sınıfı (LogicGate'den Türemiş)**

NOT mantık kapısını temsil eder. Tek girişin tersini alarak çıkışı verir.

Özellikler:

- **LogicGate** sınıfının tüm özelliklerini miras alır.
- Tek giriş portu içerir.

Yöntemler:

- **__init__(self, canvas, x, y)**: **LogicGate** sınıfının yapıcı metodunu çağırır ve NOT kapısını başlatır.

- **calculate_output(self)**: NOT kapısının çıkış değerini hesaplar.

- **NandGate Sınıfı (BinaryGate'den Türemiş)**

NAND mantık kapısını temsil eder. AND kapısının tersidir; iki girişten her ikisi de 1 olduğunda çıkışı 0 olan bir kapıdır.

Özellikler:

- **BinaryGate** sınıfının tüm özelliklerini miras alır.
- NAND kapısı için özel çıkış değeri hesaplama içerir.

Yöntemler:

- **__init__(self, canvas, x, y)**: **BinaryGate** sınıfının yapıcı metodunu çağırır ve NAND kapısını başlatır.
- **calculate_output(self)**: NAND kapısının çıkış değerini hesaplar.

- **NorGate Sınıfı (BinaryGate'den Türemiş)**

NOR mantık kapısını temsil eder. OR kapısının tersidir; iki girişten en az biri 1 olduğunda çıkışı 0 olan bir kapıdır.

Özellikler:

- **BinaryGate** sınıfının tüm özelliklerini miras alır.
- NOR kapısı için özel çıkış değeri hesaplama içerir.

Yöntemler:

- **__init__(self, canvas, x, y)**: **BinaryGate** sınıfının yapıcı metodunu çağırır ve NOR kapısını başlatır.
-
- **calculate_output(self)**: NOR kapısının çıkış değerini hesaplar.

- **XorGate Sınıfı (BinaryGate'den Türemiş)**

XOR mantık kapısını temsil eder. İki girişten sadece biri 1 olduğunda

çıkışı 1 olan bir kapıdır.

Özellikler:

- **BinaryGate** sınıfının tüm özelliklerini miras alır.
- XOR kapısı için özel çıkış değeri hesaplama içerir.

Yöntemler:

- **__init__(self, canvas, x, y)**: **BinaryGate** sınıfının yapıcı metodunu çağırır ve XOR kapısını başlatır.
-
- **calculate_output(self)**: XOR kapısının çıkış değerini hesaplar.

- **XnorGate Sınıfı (BinaryGate'den Türemiş)**

XNOR mantık kapısını temsil eder. XOR kapısının tersidir; iki girişin aynı olduğu durumlarda çıkışı 1 olan bir kapıdır.

Özellikler:

- **BinaryGate** sınıfının tüm özelliklerini miras alır.
- XNOR kapısı için özel çıkış değeri hesaplama içerir.

Yöntemler:

- **__init__(self, canvas, x, y)**: **BinaryGate** sınıfının yapıcı metodunu çağırır ve XNOR kapısını başlatır.
- **calculate_output(self)**: XNOR kapısının çıkış değerini hesaplar.

- **Giriş Kutusu Sınıfı (InputBox)**

Giriş değerlerini almak için kullanılan kutuyu temsil eder. Kullanıcı tarafından manuel olarak giriş değeri belirlenebilir.

Özellikler:

- **canvas**: Giriş kutusunun çizileceği Tkinter kanvas nesnesi.

- **x, y**: Giriş kutusunun konum koordinatları.
- **id**: Giriş kutusunun dikdörtgen temsilcisi.
- **text_id**: Giriş kutusunun metin etiketi.
- **value**: Giriş kutusunun değeri (0 veya 1).

Yöntemler:

- **__init__(self, canvas, x, y, value)**: Sınıfı başlatır ve giriş kutusunu çizer.
- **toggle_value(self)**: Giriş kutusunun değerini değiştirir.
- **get_value(self)**: Giriş kutusunun mevcut değerini döndürür.
- **set_value(self, value)**: Giriş kutusunun değerini ayarlar.

o **Çıkış Kutusu Sınıfı (OutputBox)**

Çıkış değerlerini göstermek için kullanılan kutuyu temsil eder. Devrenin çıkış değerlerini kullanıcıya gösterir.

Özellikler:

- **canvas**: Çıkış kutusunun çizileceği Tkinter kanvas nesnesi.
- **x, y**: Çıkış kutusunun konum koordinatları.
- **id**: Çıkış kutusunun dikdörtgen temsilcisi.
- **text_id**: Çıkış kutusunun metin etiketi.
- **value**: Çıkış kutusunun değeri (0 veya 1).

Yöntemler:

- **__init__(self, canvas, x, y)**: Sınıfı başlatır ve çıkış kutusunu çizer.
- **update_value(self, value)**: Çıkış kutusunun değerini günceller ve gösterir.

- **get_value(self)**: Çıkış kutusunun mevcut değerini döndürür.

- **Bağlantı Sınıfı (Connection)**

  Mantık kapıları arasındaki bağlantıları temsil eder.

  Özellikler:

  - **canvas**: Bağlantının çizileceği Tkinter kanvas nesnesi.
  - **start_gate**: Bağlantının başladığı mantık kapısı.
  - **end_gate**: Bağlantının bittiği mantık kapısı.
  - **line_id**: Bağlantıyı temsil eden çizgi.

  Yöntemler:

  - **__init__(self, canvas, start_gate, end_gate)**: Bağlantıyı başlatır ve çizer.
  - **update_position(self)**: Bağlantının konumunu günceller.
  - **delete(self)**: Bağlantıyı kanvastan siler.

- **Düğüm Sınıfı (Node)**

  Bağlantı hatlarının birleştiği noktaları temsil eder.

  Özellikler:

  - **canvas**: Düğümün çizileceği Tkinter kanvas nesnesi.
  - **x, y**: Düğümün konum koordinatları.
  - **id**: Düğümü temsil eden oval.
  - **connections**: Düğüme bağlı bağlantıların listesi.

Yöntemler:

- **__init__(self, canvas, x, y)**: Sınıfı başlatır ve düğümü çizer.
- **add_connection(self, connection)**: Düğüme bağlantı ekler.
- **remove_connection(self, connection)**: Düğümden bağlantı kaldırır.
- **update_position(self)**: Düğümün konumunu günceller.
- **delete(self)**: Düğümü kanvastan siler.

- **Sayısal Tasarım Uygulaması Sınıfı (LogicDesignApp)**

Uygulamanın ana sınıfıdır ve tüm bileşenleri yönetir. Kullanıcı arayüzü ve mantık devresi simülasyonunu kontrol eder.

Özellikler:

- **root**: Tkinter ana penceresi.
- **canvas**: Tasarım alanı olarak kullanılan Tkinter kanvas nesnesi.
- **tools**: Araç çubuğu ve diğer arayüz bileşenleri.
- **elements**: Tasarım alanındaki tüm elemanların listesi.
- **connections**: Tüm bağlantıların listesi.

Yöntemler:

- **__init__(self, root)**: Sınıfı başlatır ve kullanıcı arayüzünü oluşturur.
- **create_widgets(self)**: Kullanıcı arayüzündeki bileşenleri oluşturur.
- **add_logic_gate(self, gate_type)**: Mantık kapısı ekler.
- **add_input_box(self)**: Giriş kutusu ekler.
- **add_output_box(self)**: Çıkış kutusu ekler.
- **add_connection(self, start_element, end_element)**: Bağlantı ekler.

- **run_simulation(self)**: Simülasyonu başlatır.
- **reset_simulation(self)**: Simülasyonu sıfırlar.
- **stop_simulation(self)**: Simülasyonu durdurur.

4.2 Görev dağılımı

Grup çalışması yapılmamıştır. Tüm yazılım kodları, tasarımı ve diğer gereksinimler bireysel olarak geliştirilmiştir.

4.3 Karşılaşılan zorluklar ve çözüm yöntemleri

Problem 1: Birden Fazla Kapıyı Birbirine Bağlayamamak

Geliştirme sürecinde karşılaşılan en büyük problemlerden biri, birden fazla mantık kapısını birbirine bağlayamamak oldu. Bağlantı çizgileri oluşturulmasına rağmen, bu çizgilerin mantıksal bağlantı işlevini yerine getirmiyor.

Çözüm Yöntemi: Bu durumu çözmek için çeşitli yöntemler denendi. Öncelikle, her mantık kapısının çıkış değerini diğer kapının giriş değeri olarak ayarlamaya çalıştım. Bu yaklaşım teorik olarak doğru olmasına rağmen, uygulamada beklenen sonucu vermedi.

Problem 2: Bağlantı Çizgisi Özelliğinin İşlevsiz Olması

- **Açıklama:** Bağlantı çizgileri görsel olarak doğru şekilde çizilmesine rağmen, bu çizgiler mantık kapıları arasındaki veri akışını sağlamada başarısız oldu.
- **Çözüm Yöntemi:** Bağlantıların doğru çalışmasını sağlamak için, mantık kapıları arasındaki veri akışını yönetmek adına bir bağlantı yönetim sistemi geliştirmeye çalıştım. Bu sistem, her bağlantının başlangıç ve bitiş noktalarını takip ederek, doğru kapılar arasında veri iletimini sağlamayı amaçladı. Ancak, bu çözüm de uygulamada beklenen sonucu vermedi.

4.4 Proje isterlerine göre eksik yönler

Projenin geliştirilmesi sürecinde, belirlenen fonksiyonel gereksinimlere uygun olarak çeşitli görevler yerine getirilmiş ve birçok işlev başarıyla kodlanmıştır. Ancak, bazı eksiklikler ve tamamlanamayan görevler de mevcuttur. Bu

eksiklikler aşağıda belirtilmiştir:

- **Mantık Kapıları Arasındaki Bağlantılar:**

- **Açıklama:** Birden fazla mantık kapısını birbirine bağlayarak, bağlantı çizgileri oluşturulmasına rağmen, bu çizgilerin mantıksal bağlantı işlevini yerine getirmesi sağlanamamıştır.

- **Eksik Yön:** Mantık kapılarının çıkış değerlerini diğer kapıların giriş değeri olarak doğru bir şekilde ayarlamak kodlanamamıştır.

5 TEST VE DOĞRULAMA

5.1 Yazılımın test süreci

Yazılım için bir test uygulaması geliştirilmiştir. Bu uygulama, **unittest** modülü kullanılarak oluşturulmuş ve proje bileşenlerini test etmek amacıyla tasarlanmıştır.

Test Edilen Bileşenler:

- **LogicGate Sınıfı:** Mantık kapılarının (AND, OR, NOT…) doğru çalışıp çalışmadığı test edilmiştir.

- **InputElement Sınıfı:** Giriş elemanlarının doğru şekilde çalışıp çalışmadığı test edilmiştir.

- **OutputElement Sınıfı:** Çıkış elemanlarının doğru şekilde çalışıp çalışmadığı test edilmiştir.

- **Connection Sınıfı:** Mantık kapıları arasındaki bağlantıların doğru çalışıp çalışmadığı test edilmiştir.

- **Simülasyon:** Mantık kapıları ve giriş elemanlarının kombinasyonlarıyla yapılan simülasyonun doğruluğu test edilmiştir.

Mantık Kapıları Testi:

test.py × main.py

```python
import unittest
from main import LogicGate, InputElement, OutputElement, Connection

class TestLogicCircuitPlatform(unittest.TestCase):

    def test_and_gate(self):
        and_gate = LogicGate( label: 'AND', 2)
        input1 = InputElement('Input1', 1)
        input2 = InputElement('Input2', 1)
        input1.value = 1
        input2.value = 1
        and_gate.inputs = [input1, input2]
        output = and_gate.calculate_output()
        self.assertEqual(output,  second: 1,  msg: "AND gate failed")

    def test_or_gate(self):
        or_gate = LogicGate( label: 'OR', 2)
        input1 = InputElement('Input1', 1)
        input2 = InputElement('Input2', 1)
        input1.value = 0
        input2.value = 1
        or_gate.inputs = [input1, input2]
        output = or_gate.calculate_output()
        self.assertEqual(output,  second: 1,  msg: "OR gate failed")
```

```python
    def test_not_gate(self):
        not_gate = LogicGate( label: 'NOT', 1)
        input1 = InputElement('Input1', 1)
        input1.value = 1
        not_gate.inputs = [input1]
        output = not_gate.calculate_output()
        self.assertEqual(output,  second: 0,  msg: "NOT gate failed")

    def test_buffer_gate(self):
        buffer_gate = LogicGate( label: 'BUFFER', 1)
        input1 = InputElement('Input1', 1)
        input1.value = 1
        buffer_gate.inputs = [input1]
        output = buffer_gate.calculate_output()
        self.assertEqual(output,  second: 1,  msg: "BUFFER gate failed")

    def test_nand_gate(self):
        nand_gate = LogicGate( label: 'NAND', 2)
        input1 = InputElement('Input1', 1)
        input2 = InputElement('Input2', 1)
```

```python
        input1.value = 1
        input2.value = 1
        nand_gate.inputs = [input1, input2]
        output = nand_gate.calculate_output()
        self.assertEqual(output,  second: 0,  msg: "NAND gate failed")

    def test_nor_gate(self):
        nor_gate = LogicGate( label: 'NOR', 2)
        input1 = InputElement('Input1', 1)
        input2 = InputElement('Input2', 1)
        input1.value = 0
        input2.value = 0
        nor_gate.inputs = [input1, input2]
        output = nor_gate.calculate_output()
        self.assertEqual(output,  second: 1,  msg: "NOR gate failed")

    def test_xor_gate(self):
        xor_gate = LogicGate( label: 'XOR', 2)
        input1 = InputElement('Input1', 1)
        input2 = InputElement('Input2', 1)
        input1.value = 1
        input2.value = 0
        xor_gate.inputs = [input1, input2]
        output = xor_gate.calculate_output()
```

```
        self.assertEqual(output, second: 1, msg: "XOR gate failed")

    def test_xnor_gate(self):
        xnor_gate = LogicGate( label: 'XNOR', 2)
        input1 = InputElement('Input1', 1)
        input2 = InputElement('Input2', 1)
        input1.value = 1
        input2.value = 1
        xnor_gate.inputs = [input1, input2]
        output = xnor_gate.calculate_output()
        self.assertEqual(output, second: 1, msg: "XNOR gate failed")
```

- **test_and_gate**: AND kapısının doğru çalışıp çalışmadığını kontrol eder.
- **test_or_gate**: OR kapısının doğru çalışıp çalışmadığını kontrol eder.
- **test_not_gate**: NOT kapısının doğru çalışıp çalışmadığını kontrol eder.
- **test_buffer_gate**: BUFFER kapısının doğru çalışıp çalışmadığını kontrol eder.
- **test_nand_gate**: NAND kapısının doğru çalışıp çalışmadığını kontrol eder.
- **test_nor_gate**: NOR kapısının doğru çalışıp çalışmadığını kontrol eder.
- **test_xor_gate**: XOR kapısının doğru çalışıp çalışmadığını kontrol eder.
- **test_xnor_gate**: XNOR kapısının doğru çalışıp çalışmadığını kontrol eder.

Giriş/Çıkış Elemanları Testi:

```
    def test_output_element(self):
        output_element = OutputElement('Output1')
        self.assertIsNotNone(output_element, msg: "OutputElement initialization failed")

    def test_input_element_initialization(self):
        input_element = InputElement('Input1', 1)
        self.assertEqual(input_element.label, second: 'Input1', msg: "InputElement label initialization failed")
        self.assertEqual(input_element.value, second: 1, msg: "InputElement value initialization failed")

    def test_input_element_value_change(self):
        input_element = InputElement('Input1', 1)
        input_element.value = 0
        self.assertEqual(input_element.value, second: 0, msg: "InputElement value change failed")
```

- **test_output_element**: Çıkış elemanının doğru şekilde oluşturulup oluşturulmadığını kontrol eder.

- **test_input_element_initialization**: Giriş elemanının doğru şekilde başlatılıp başlatılmadığını kontrol eder.

- **test_input_element_value_change**: Giriş elemanının değerinin değiştirilebildiğini kontrol eder.

Bağlantı Elemanları Testi:

```
96      def test_connection(self):
97          input1 = InputElement('Input1', 1)
98          input1.value = 1
99          and_gate = LogicGate( label: 'AND', 2)
100         and_gate.inputs = [input1, input1]
101         connection = Connection(input1, and_gate)
102         self.assertTrue(connection.transfer_signal(),  msg: "Connection failed")
```

- **test_connection**: Bağlantı elemanının sinyalleri doğru şekilde iletip iletmediğini kontrol eder.

Simülasyon Testi:

```
104     def test_simulation(self):
105         input1 = InputElement('Input1', 1)
106         input2 = InputElement('Input2', 1)
107         and_gate = LogicGate( label: 'AND', 2)
108         input1.value = 1
109         input2.value = 0
110         and_gate.inputs = [input1, input2]
111         output = and_gate.calculate_output()
112         self.assertEqual(output,  second: 0,  msg: "Simulation failed for AND gate with inputs 1, 0")
113
114 if __name__ == '__main__':
115     unittest.main()
116
```

- **test_simulation**: AND kapısı için bir simülasyon testi gerçekleştirir.

5.2 Yazılımın doğrulanması

Yazılımın Test Edilmesi ve Doğrulama:

- Test uygulaması ile yazılımın bileşenleri test edilmiş ve doğrulanmıştır.

Testler sonucunda elde edilen bulgular aşağıda belirtilmiştir.

- **Tam ve Doğru Çalışan Bileşenler:**

- **LogicGate Sınıfı:** Mantık kapıları beklenen sonuçları vermektedir.

AND Kapısı:

- Giriş değerleri [1, 1] olduğunda çıkış değeri 1 olarak doğru sonuç vermiştir.

- Giriş değerleri [1, 0] olduğunda çıkış değeri 0 olarak doğru sonuç vermiştir.

OR Kapısı:

- Giriş değerleri [0, 1] olduğunda çıkış değeri 1 olarak doğru sonuç vermiştir.

- Giriş değerleri [0, 0] olduğunda çıkış değeri 0 olarak doğru sonuç vermiştir.

NOT Kapısı:

- Giriş değeri 1 olduğunda çıkış değeri 0 olarak doğru sonuç vermiştir.

- Giriş değeri 0 olduğunda çıkış değeri 1 olarak doğru sonuç vermiştir.

BUFFER Kapısı:

- Giriş değeri 1 olduğunda çıkış değeri 1 olarak doğru sonuç vermiştir.

- Giriş değeri 0 olduğunda çıkış değeri 0 olarak doğru sonuç vermiştir.

NAND Kapısı:

- Giriş değerleri [1, 1] olduğunda çıkış değeri 0 olarak doğru sonuç vermiştir.

- Giriş değerleri [1, 0] olduğunda çıkış değeri 1 olarak doğru sonuç vermiştir.

NOR Kapısı:

- Giriş değerleri [0, 0] olduğunda çıkış değeri 1 olarak doğru sonuç vermiştir.

- Giriş değerleri [0, 1] olduğunda çıkış değeri 0 olarak doğru sonuç vermiştir.

XOR Kapısı:

- Giriş değerleri [1, 0] olduğunda çıkış değeri 1 olarak doğru sonuç vermiştir.
- Giriş değerleri [1, 1] olduğunda çıkış değeri 0 olarak doğru sonuç vermiştir.
- **XNOR Kapısı:**
- Giriş değerleri [1, 1] olduğunda çıkış değeri 1 olarak doğru sonuç vermiştir.
- Giriş değerleri [1, 0] olduğunda çıkış değeri 0 olarak doğru sonuç vermiştir.

OutputElement Sınıfı:

- Başarıyla başlatılmıştır.

InputElement Sınıfı:

- Başarıyla başlatılmış ve değerleri doğru şekilde değiştirilmiştir.

Eksik veya Hatalı Çalışan Bileşenler:

Connection Sınıfı:

- Bağlantıların mantıksal işlevi tam olarak sağlanamamıştır. Bağlantıların veri iletimi doğru şekilde gerçekleşmemektedir.

GİTHUB LİNK:

https://github.com/Zehrayardimci