# 

T.C.

CUMHURİYET ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

# Sayısal elektronik simülatör uygulaması

## ERHAN BİLGİN

## 2014123071

### *Özet:*

Projenin genel amacı lojik kapıları kullanarak simülasyonunu sağlayabileceğimiz bir ortam yaratmaktır.

***Anahtar Kelimeler:*** lojik kapılar , html5, javascript ,canvas

*Abstract :*

Createing a simulation environment using the purpose logic gates of the project.

***Keywords:*** logic gates , html5, javascript ,canvas

## Giriş

Projede kullanılan teknolojiler şunlardır ;

1-) Html5

2-) Javascript

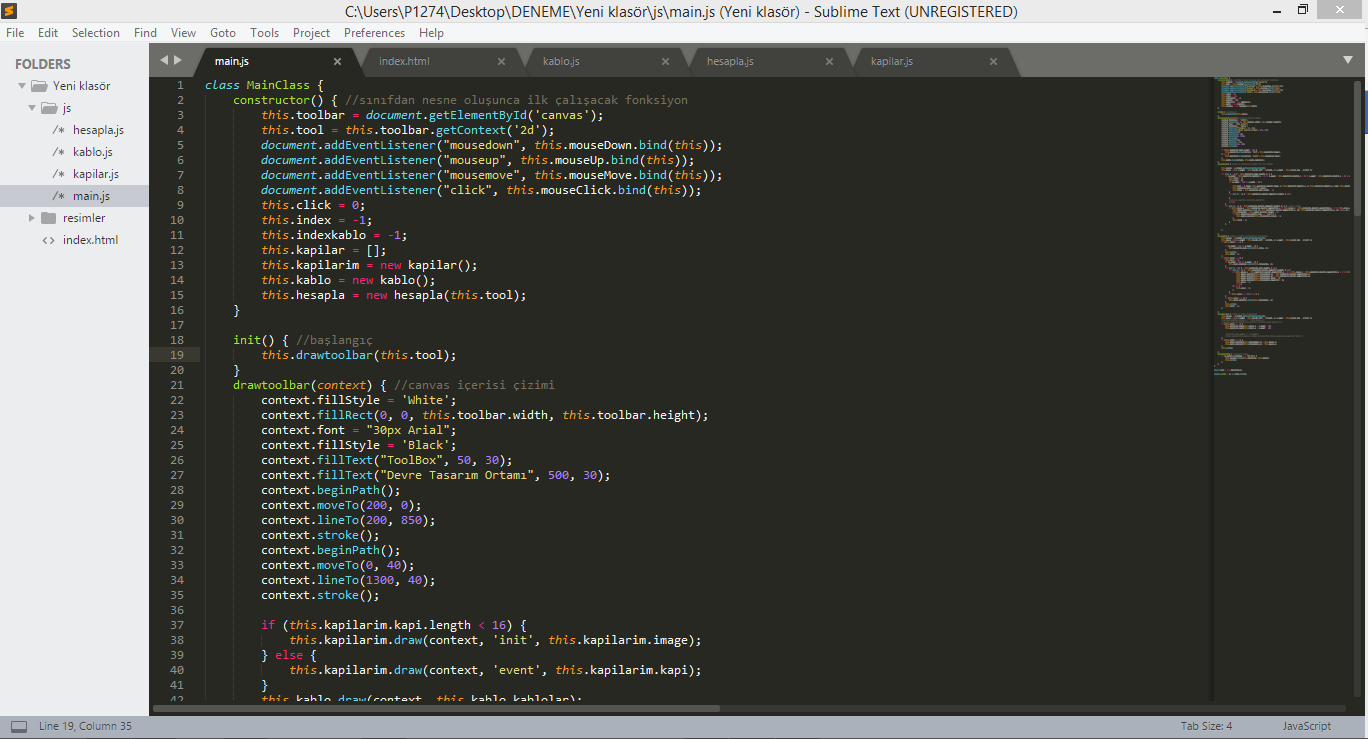
Projeyi yaparken tüm sistemlerde ve bilgisayarlara yüklenme gereksinimi olmadan sağlayacak olan html ve javascript teknolojilerini kullanarak yaptım. Html den bahsedecek olursam Html(Hypertext markup language ) web sayfaları hazırlamak için kullanılan bir işaretleme dilidir. Aslında html5 ile gelen canvas özelliğini projede ağırlıklı olarak kullandım. Canvas nedir sorusuna gelirsek canvas dilimizde tuval anlamına gelmektedir. Html5 in canvas etiketi sayesinde bize bir tuval yani çizimlerimizi oluşturabileceğimiz bir çizim ortamı sağlamaktadır. Tabi bize bir tuval verdi ama elimizde fırçamız yok fırça görevi üstlenen dil ise javascript dir. Bir diğer adı aslında ecmascript dir. Javascript bir script dili olduğu için herhangi bir yükleme gerekmeksizin web ortamlarında programlama yapabilmemize yarar. Javascript kodlarken ecmascript6 ile gelen class(sınıf) yapısı kullanarak kodlamalarımı yaptım.

## UYGULAMA

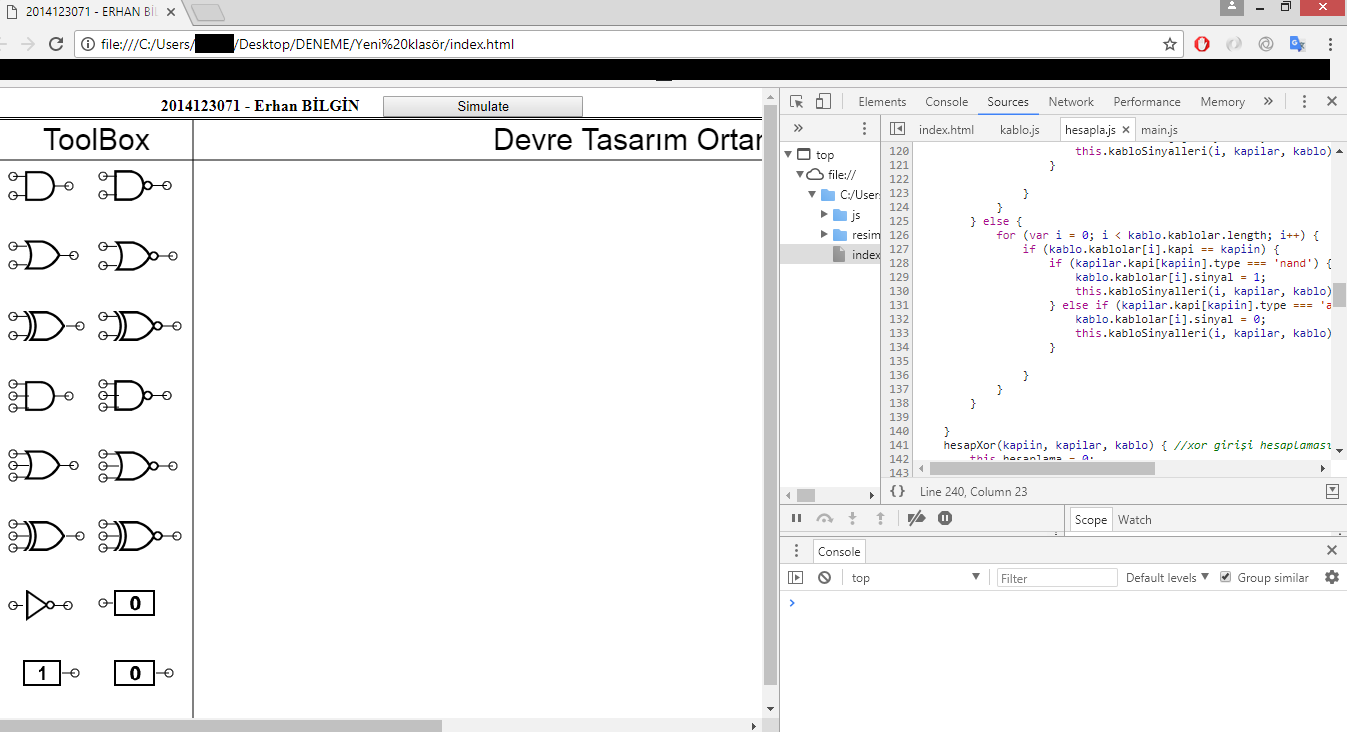
Uygulamayı için kullanılan ide ve diller

* Html5 ve javascript için Sublime Text 3 (Şekil 1.1)
* Web ortamında uygulamayı görmek için Google Chrome uygulaması (Şekil 1.2)

Projenin Geliştirildiği ortam



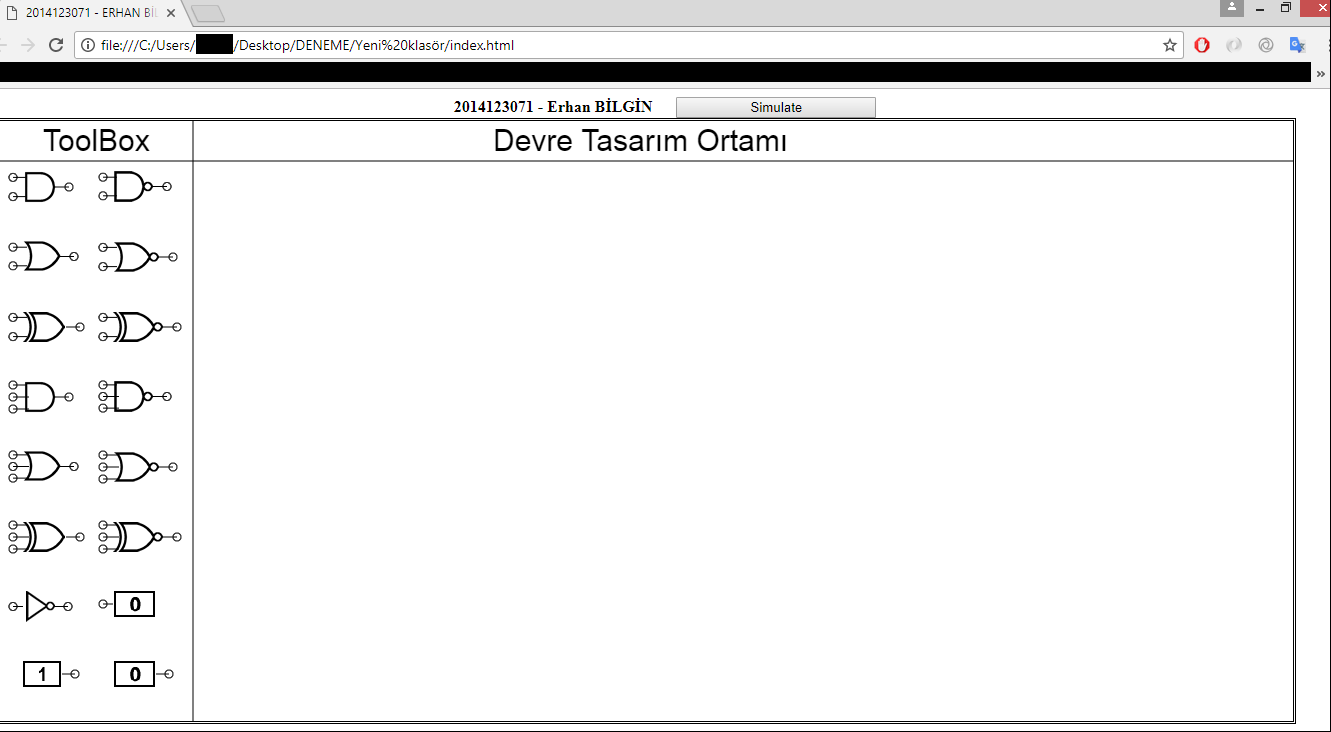
**Şekil 1.1 :** Sublime Text 3 geliştirme ortamı



**Şekil 1.2 :** Google Chrome uygulaması

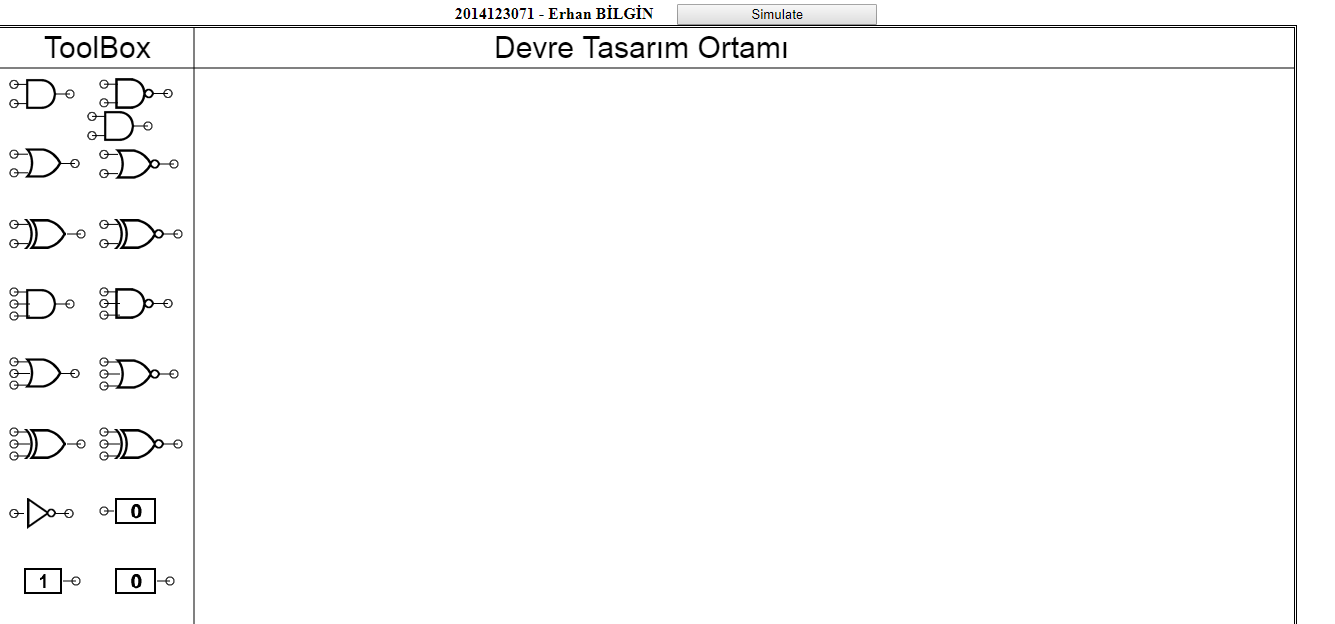
## SONUÇLAR

Projeyi çalıştırmak için size verilen klasör içerisindeki index.html dosyasını herhangi bir web tarayıcı uygulaması üzerinden açarak Şekil 1.3 deki gibi bir ekran karşılayacaktır sizi.

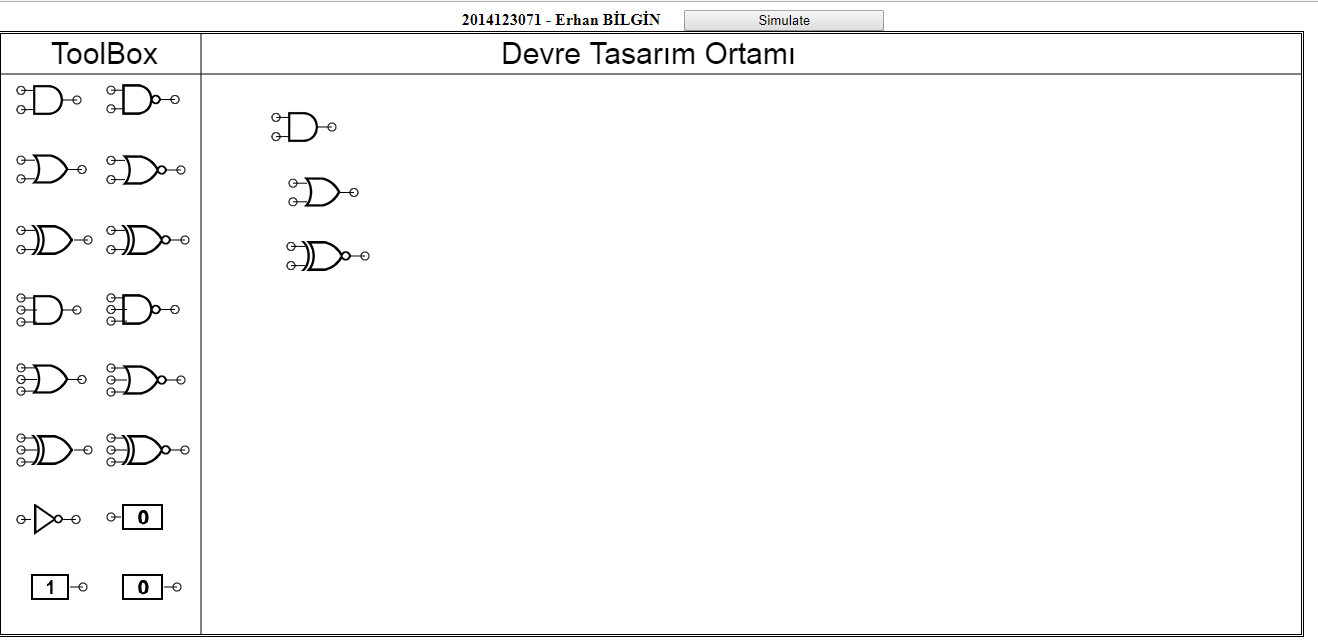


**Şekil 1.3 :** Uygulama Ekranı

Sol tarafta kullanabileceğiniz lojik kapılar bulunmaktadır bu kapıları mouse un sol tıkına basılı tutarak devre tasarım ortamı içerisine sürükleyerek bırakabilirsiniz. Silmek istediğiniz veya yanlış aldığınız lojik kapıları toolbox bölümü içerisine bırakarak silebilirsiniz. Şekil 1.4 de ve Şekil 1.5 de örnek bir sürükleme uygulaması gösterilmiştir.

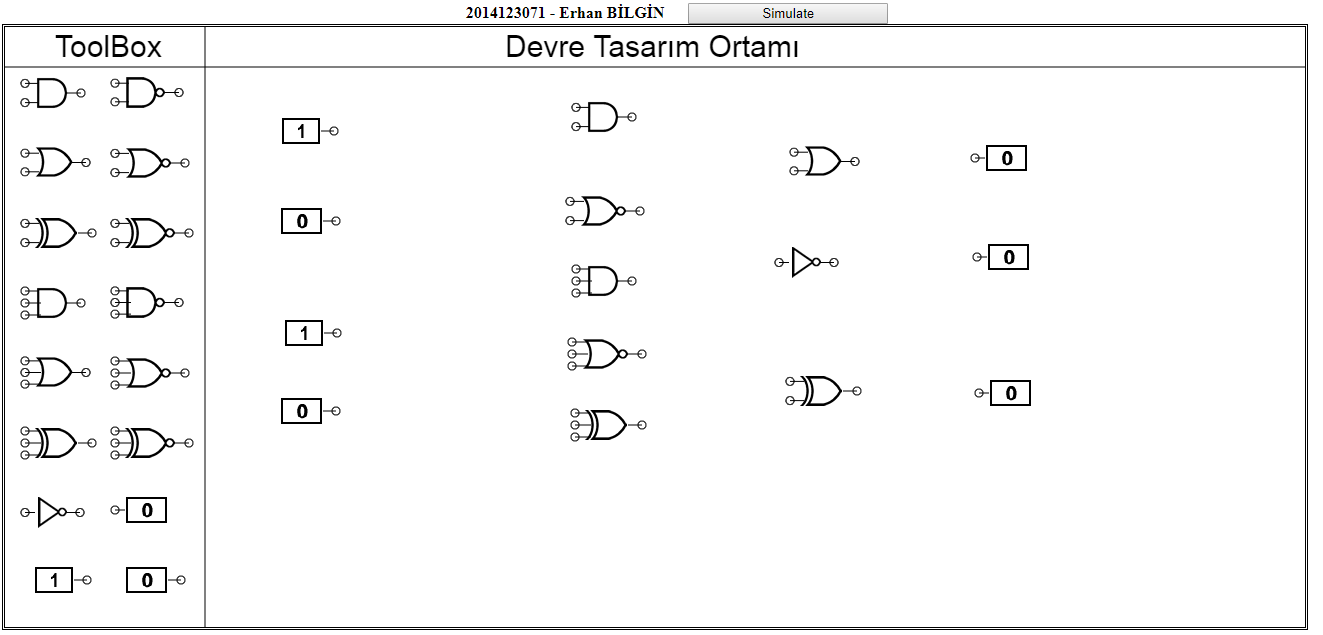


**Şekil 1.4 :** Logic Kapı sürükleme 1



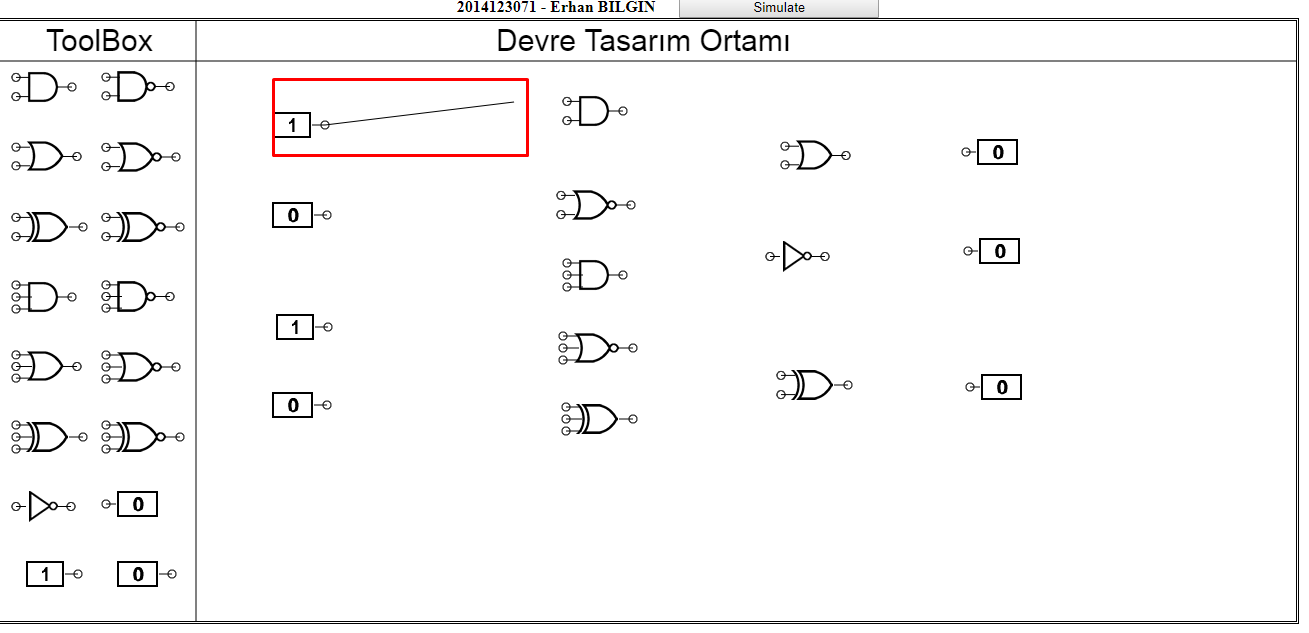
**Şekil 1.4 :** Logic Kapı sürükleme 2

Sürüklediğimiz lojik kapılara input ve output kapılarını birbirine bağlamak için önce input ve output nesnelerini devre tasarım ortamına sürükleyelim ve örnek bir uygulama Şekil 1.5 de gösterilmiştir.

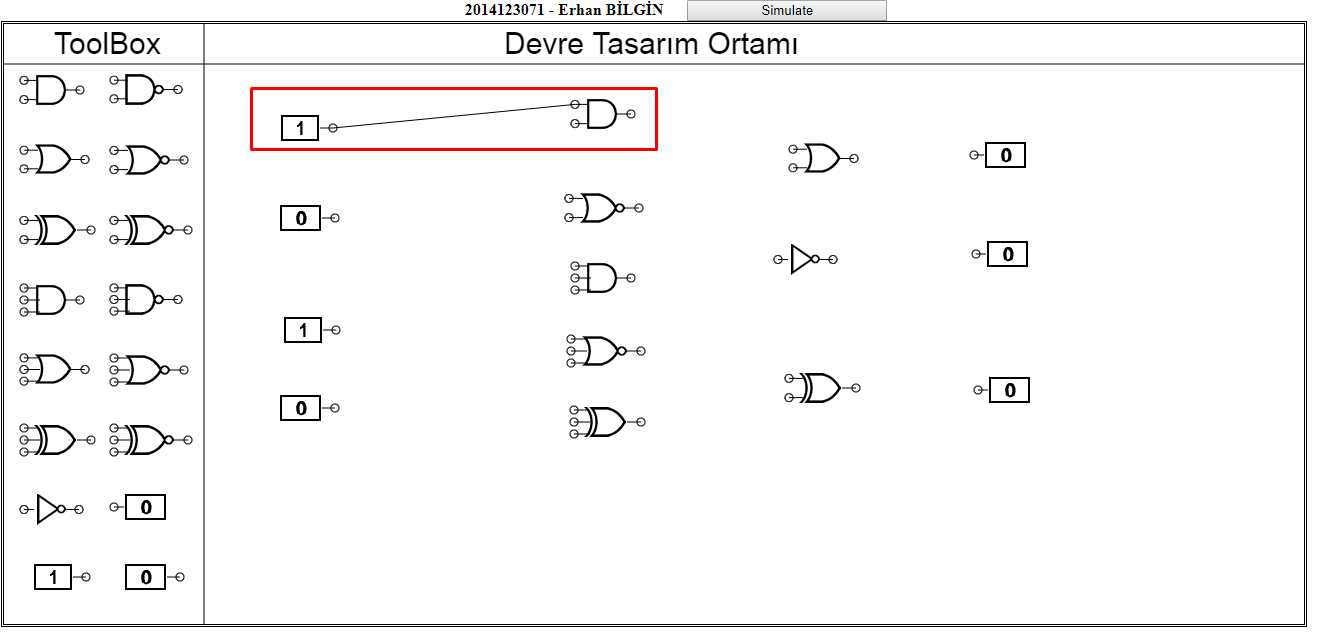


**Şekil 1.5 :** Simülasyon uygulaması

Sürüklediğimiz lojik kapılar şekil 1.5 de mevcuttur şimdi bu nesnelerimizi birbirine bağlayalım. Nesneleri birbirine bağlamak için bir nesnenin output yuvarlağına tıklama ile basılı tuttarak istediğimiz bir nesnenin input yuvarlağı kısmına getirerek bırakıyoruz . Örnek uygulama resmi Şekil 1.6 ve Şekil 1.7 de mevcuttur.

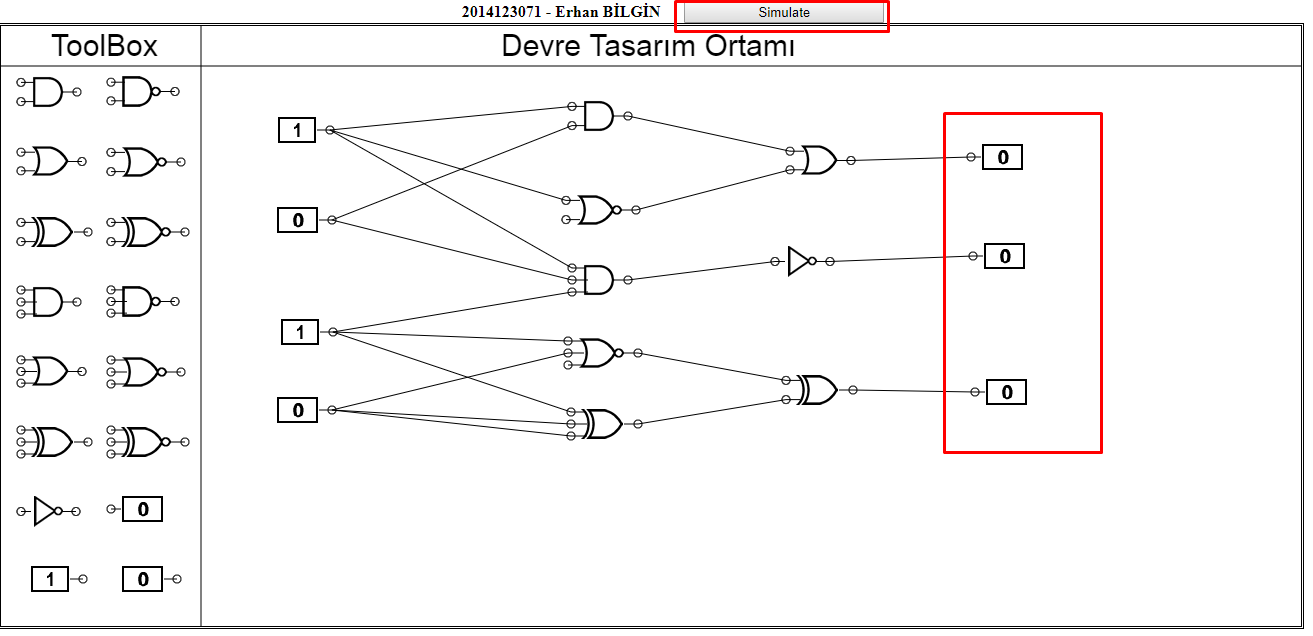


**Şekil 1.6 :** Nesneleri birbirine bağlama 1.1



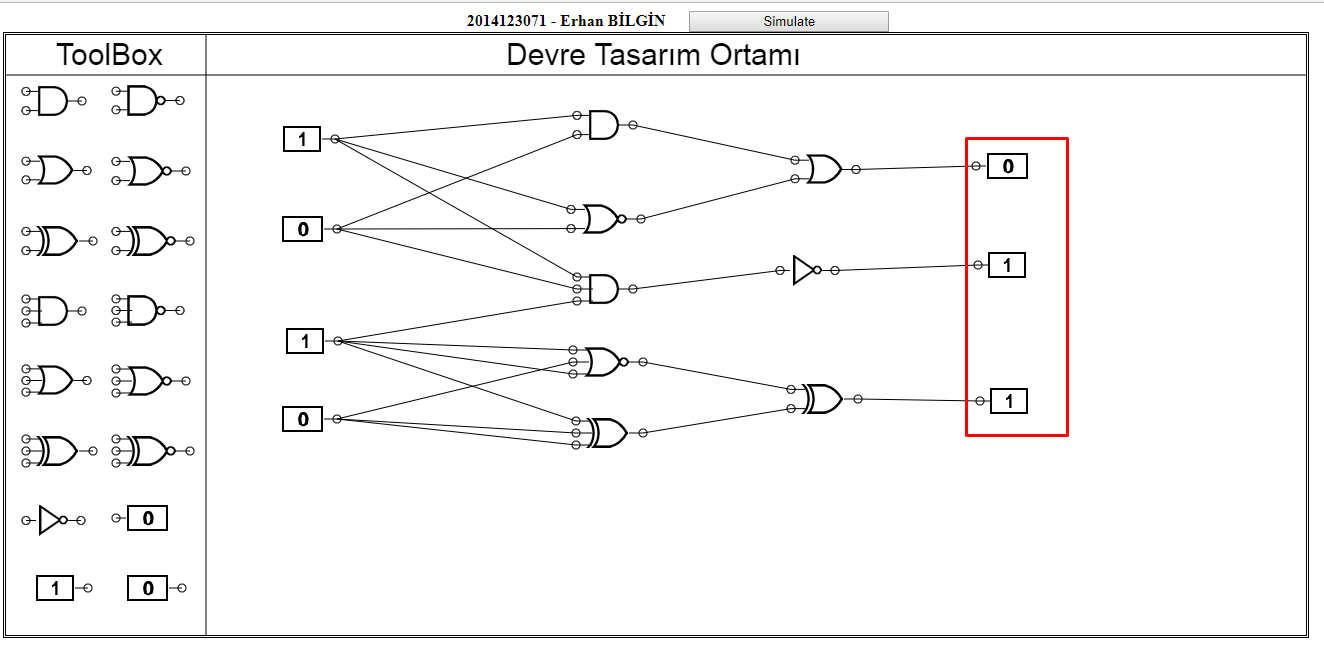
**Şekil 1.7 :** Nesneleri birbirine bağlama 1.2

Tüm lojik kapıları birbirine bağladıktan sonra simülasyonu çalıştırmak için sayfanın üst orta kısmında yer alan “Simulate” butonuna basılır. “Simulate” butonuna basmadan önce ve bastıktan sonraki aşamaları Şekil 1.8 ve Şekil 1.9 da görmek mümkündür.



**Şekil 1.8 :** Simulate Butonuna basmadan önce

Simulate butonuna basmadan önce görüldüğü üzere tüm output elemanları 0 değerini göstermektedir.



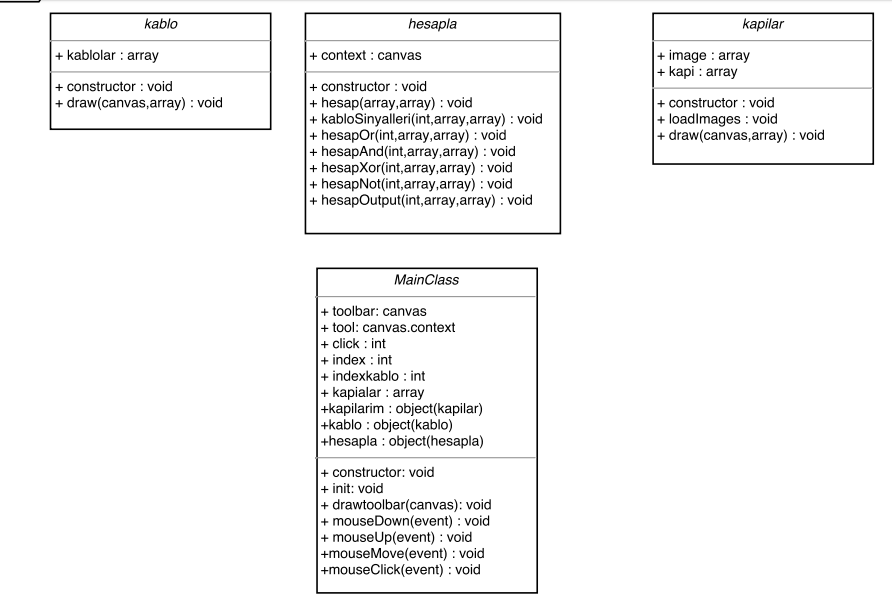
**Şekil 1.9 :** Simulate Butonuna basmadan sonra

Simulate butonuna basıldıktan sonra kapılara gelen 1 ve 0 sinyalleri ilgili olduğu kapının fonksiyonuna giderek hesaplamalar yaptıktan sonra sonucu output nesnelerine yansıtır.

Bu şekilde simülasyon bitmiş olmaktadır. Yeni bir devre oluşturmak için sayfayı yeniden başlattıktan sonra aynı adımlara uyarak kendi devrenizi tasarlayabilirsiniz.

Uygulamada toplamda 4 sınıf oluşturulurmuştur. Bu sınıflar birbirine kalıtım veya implemantasyonu yapılmamıştır sınıflardan nesneler türetilerek mainclass içerisinde kullanılmıştır.

Uygulamanın Uml diyagramı Şekil 2.0 da gösterilmiştir.



**Şekil 2.0 :** Uygulamanın UML diagramı

## KAYNAK KODLAR

Kaynak kodların github linki : <https://github.com/BilginErhan/MuhProje3>

index.html

\*------------------------------------------------------------------\*

<!DOCTYPE html>

<html>

<head>

<title>2014123071 - ERHAN BİLGİN</title>

</head>

<body>

<center><div><b>2014123071 - Erhan BİLGİN</b> &emsp; <button id ="simulate" style=" width: 200px;">Simulate</button></div></center>

<canvas id='canvas' width="1300" height="600" style="border-style: double;"></canvas>

<script src = "js/kapilar.js"></script>

<script src = "js/kablo.js"></script>

<script src = "js/hesapla.js"></script>

<script src = "js/main.js"></script>

</body>

</html>

\*------------------------------------------------------------------\*

kapilar.js

\*------------------------------------------------------------------\*

class kapilar {//kapılar sınıfı oluşturuldu

constructor() {//yapıcı method oluştuduldu

this.image = [];//resimler listesi

this.kapi = [];//kapılar listesi

this.loadImages(); //loadimage fonksiyonu

}

loadImages() {//bu fonksiyon resimler klasöründeki resimleri özelliklerine ve

//ilk çalıştırmada bulundukları konumlar girilir ve image listesine pushlanır

this.image.push({ image: new Image(), x: 30, y: 50, type: 'and', input: '2' });

this.image[0].image.src = "resimler/and.png";

this.image.push({ image: new Image(), x: 120, y: 50, type: 'nand', input: '2' });

this.image[1].image.src = "resimler/nand.png";

this.image.push({ image: new Image(), x: 30, y: 120, type: 'or', input: '2' });

this.image[2].image.src = "resimler/or.png";

this.image.push({ image: new Image(), x: 120, y: 120, type: 'nor', input: '2' });

this.image[3].image.src = "resimler/nor.png";

this.image.push({ image: new Image(), x: 30, y: 190, type: 'xor', input: '2' });

this.image[4].image.src = "resimler/xor.png";

this.image.push({ image: new Image(), x: 120, y: 190, type: 'xnor', input: '2' });

this.image[5].image.src = "resimler/xnor.png";

this.image.push({ image: new Image(), x: 30, y: 260, type: 'and', input: '3' });

this.image[6].image.src = "resimler/and.png";

this.image.push({ image: new Image(), x: 120, y: 260, type: 'nand', input: '3' });

this.image[7].image.src = "resimler/nand.png";

this.image.push({ image: new Image(), x: 30, y: 330, type: 'or', input: '3' });

this.image[8].image.src = "resimler/or.png";

this.image.push({ image: new Image(), x: 120, y: 330, type: 'nor', input: '3' });

this.image[9].image.src = "resimler/nor.png";

this.image.push({ image: new Image(), x: 30, y: 400, type: 'xor', input: '3' });

this.image[10].image.src = "resimler/xor.png";

this.image.push({ image: new Image(), x: 120, y: 400, type: 'xnor', input: '3' });

this.image[11].image.src = "resimler/xnor.png";

this.image.push({ image: new Image(), x: 30, y: 470, type: 'not', input: '1' });

this.image[12].image.src = "resimler/not.png";

this.image.push({ image: new Image(), x: 120, y: 470, type: 'output', input: '1' });

this.image[13].image.src = "resimler/output.png";

this.image.push({ image: new Image(), x: 30, y: 540, type: 'input', input: '1' });

this.image[14].image.src = "resimler/input.png";

this.image.push({ image: new Image(), x: 120, y: 540, type: 'input1', input: '1' });

this.image[15].image.src = "resimler/input1.png";

}

draw(context, event, nesne) {//draw fonksiyonu

//context(canvas nesnesi) event(gelen kapının ilk başlangıçtamı oluştuğu yoksa sonradan eklendiğimi)

//nesne ise kapı listesinin tutulduğu değişkendir.

for (var i = 0; i < nesne.length; i++) {//tüm nesneler üzerinde dönülür

if (nesne[i].input === '2') {//gelen nesne 2 inputluysa

//çizimlerde canvas nesnesi olan context kullanılmıştır

context.drawImage(nesne[i].image, nesne[i].x, nesne[i].y);//resimler yüklenir

//input1 çizimi

context.beginPath();//çizgi çizdirme

context.moveTo(nesne[i].x - 10, nesne[i].y + nesne[i].image.height / 5);

context.lineTo(nesne[i].x + 4, nesne[i].y + nesne[i].image.height / 5);

context.stroke();

//input2 çizimi

context.beginPath();//çizgi çizdirme

context.moveTo(nesne[i].x - 10, nesne[i].y + nesne[i].image.height / 5 \* 4);

context.lineTo(nesne[i].x + 4, nesne[i].y + nesne[i].image.height / 5 \* 4);

context.stroke();

//output1 çizimi

context.beginPath();//çizgi çizdirme

context.moveTo(nesne[i].x + nesne[i].image.width - 1, nesne[i].y + nesne[i].image.height / 2);

context.lineTo(nesne[i].x + nesne[i].image.width + 13, nesne[i].y + nesne[i].image.height / 2);

context.stroke();

//baglantilar array list

this.baglanti = [];//çizgiler baglanti listesine bulundukları konum ve özelliklerine göre bağlantı listesine pushlanır

this.baglanti[0] = { x: nesne[i].x - 10, y: nesne[i].y + nesne[i].image.height / 5, input: '1', output: '' };

this.baglanti[1] = { x: nesne[i].x - 10, y: nesne[i].y + nesne[i].image.height / 5 \* 4, input: '2', output: '' };

this.baglanti[2] = { x: nesne[i].x + nesne[i].image.width + 13, y: nesne[i].y + nesne[i].image.height / 2, input: '', output: '1' };

//input1 baglanti

context.beginPath();//çizgiler üzerine girişiler için yuvarlak daireler çizdirilir

context.arc(this.baglanti[0].x, this.baglanti[0].y, 4, 0, 2 \* Math.PI);

context.stroke();

//input2 baglanti

context.beginPath();//çizgiler üzerine girişiler için yuvarlak daireler çizdirilir

context.arc(this.baglanti[1].x, this.baglanti[1].y, 4, 0, 2 \* Math.PI);

context.stroke();

//output1 baglanti

context.beginPath();//çizgiler üzerine girişiler için yuvarlak daireler çizdirilir

context.arc(this.baglanti[2].x, this.baglanti[2].y, 4, 0, 2 \* Math.PI);

context.stroke();

//gelen nesneler event için değilse sayfa ilk açıldığındadır o yüzden kapi listesine pushlanır

if (event != 'event') {

this.kapi.push({ image: nesne[i].image, x: nesne[i].x, y: nesne[i].y, type: nesne[i].type, input: '2', baglanti: this.baglanti });

} else {//event için geldiyse nesneler

this.kapi[i] = { image: nesne[i].image, x: nesne[i].x, y: nesne[i].y, type: nesne[i].type, input: '2', baglanti: this.baglanti };

}

} else if (nesne[i].input === '3') {//gelen nesne 3 inputluysa

context.drawImage(nesne[i].image, nesne[i].x, nesne[i].y);

//input1 çizgi çizimi

context.beginPath();

context.moveTo(nesne[i].x - 10, nesne[i].y + nesne[i].image.height / 8);

context.lineTo(nesne[i].x + 4, nesne[i].y + nesne[i].image.height / 8);

context.stroke();

//input2 çizgi çizimi

context.beginPath();

context.moveTo(nesne[i].x - 10, nesne[i].y + nesne[i].image.height / 8 \* 4);

context.lineTo(nesne[i].x + 6, nesne[i].y + nesne[i].image.height / 8 \* 4);

context.stroke();

//input3 çizgi çizimi

context.beginPath();

context.moveTo(nesne[i].x - 10, nesne[i].y + nesne[i].image.height / 8 \* 7);

context.lineTo(nesne[i].x + 6, nesne[i].y + nesne[i].image.height / 8 \* 7);

context.stroke();

//output1 çizgi çizimi

context.beginPath();

context.moveTo(nesne[i].x + nesne[i].image.width - 1, nesne[i].y + nesne[i].image.height / 2);

context.lineTo(nesne[i].x + nesne[i].image.width + 13, nesne[i].y + nesne[i].image.height / 2);

context.stroke();

//baglantilar array list

this.baglanti = [];

this.baglanti[0] = { x: nesne[i].x - 10, y: nesne[i].y + nesne[i].image.height / 8, input: '1', output: '' };

this.baglanti[1] = { x: nesne[i].x - 10, y: nesne[i].y + nesne[i].image.height / 8 \* 4, input: '2', output: '' };

this.baglanti[2] = { x: nesne[i].x - 10, y: nesne[i].y + nesne[i].image.height / 8 \* 7, input: '3', output: '' };

this.baglanti[3] = { x: nesne[i].x + nesne[i].image.width + 13, y: nesne[i].y + nesne[i].image.height / 2, input: '', output: '1' };

//input1 baglanti çizimi

context.beginPath();

context.arc(this.baglanti[0].x, this.baglanti[0].y, 4, 0, 12 \* Math.PI);

context.stroke();

//input2 baglanti çizimi

context.beginPath();

context.arc(this.baglanti[1].x, this.baglanti[1].y, 4, 0, 2 \* Math.PI);

context.stroke();

//input3 baglanti çizimi

context.beginPath();

context.arc(this.baglanti[2].x, this.baglanti[2].y, 4, 0, 2 \* Math.PI);

context.stroke();

//output1 baglanti çizimi

context.beginPath();

context.arc(this.baglanti[3].x, this.baglanti[3].y, 4, 0, 2 \* Math.PI);

context.stroke();

if (event != 'event') {

this.kapi.push({ image: nesne[i].image, x: nesne[i].x, y: nesne[i].y, type: nesne[i].type, input: '3', baglanti: this.baglanti });

} else {

this.kapi[i] = { image: nesne[i].image, x: nesne[i].x, y: nesne[i].y, type: nesne[i].type, input: '3', baglanti: this.baglanti };

}

} else if (nesne[i].input === '1') {//gelen nesne 1 inputluysa

if (nesne[i].type === 'output' || nesne[i].type === 'not') {//nesne tipi output veya not nesnesiyse

//input1 çizimi yapılır

context.beginPath();

context.moveTo(nesne[i].x - 10, nesne[i].y + nesne[i].image.height / 2);

context.lineTo(nesne[i].x + 4, nesne[i].y + nesne[i].image.height / 2);

context.stroke();

}

if (nesne[i].type === 'input' || nesne[i].type === 'not' || nesne[i].type === 'input1') {

//gelen nesne input not veya input1(sıfır olan giriş) ise

//output1 çizgilieri çiz

context.beginPath();

context.moveTo(nesne[i].x + nesne[i].image.width - 1, nesne[i].y + nesne[i].image.height / 2);

context.lineTo(nesne[i].x + nesne[i].image.width + 13, nesne[i].y + nesne[i].image.height / 2);

context.stroke();

}

this.baglanti = [];//baglanti listesini oluştur

if (i > 15) {//15. elemandan sonraki nesneler için yani çizim ortamına yeni eklenen

//output nesnesinin görünümü sinyaline göre değiştirlir.

if (nesne[i].type === 'output' && typeof nesne[i].baglanti != 'undefined') {

if (nesne[i].baglanti[0].output === '1')

{

nesne[i].image = new Image();

nesne[i].image.src = "resimler/input.png";

}

if (nesne[i].baglanti[0].output === '0')

{

nesne[i].image = new Image();

nesne[i].image.src = "resimler/output.png";

}

}

}

context.drawImage(nesne[i].image, nesne[i].x, nesne[i].y);//nesnenin resmi çizdirilir

//baglantilar array list

if (nesne[i].type === 'input' || nesne[i].type === 'input1') {

this.baglanti[0] = { x: nesne[i].x + nesne[i].image.width + 13, y: nesne[i].y + nesne[i].image.height / 2, input: '', output: '1' };

}

if (nesne[i].type === 'output' || nesne[i].type === 'not') {

this.baglanti[0] = { x: nesne[i].x - 10, y: nesne[i].y + nesne[i].image.height / 2, input: '1', output: '' };

}

if (nesne[i].type === 'not') {

this.baglanti[1] = { x: nesne[i].x + nesne[i].image.width + 13, y: nesne[i].y + nesne[i].image.height / 2, input: '', output: '1' };

//input1 baglanti

context.beginPath();

context.arc(this.baglanti[1].x, this.baglanti[1].y, 4, 0, 2 \* Math.PI);

context.stroke();

}

//input1 baglanti

context.beginPath();

context.arc(this.baglanti[0].x, this.baglanti[0].y, 4, 0, 2 \* Math.PI);

context.stroke();

if (event != 'event') {

this.kapi.push({ image: nesne[i].image, x: nesne[i].x, y: nesne[i].y, type: nesne[i].type, input: '1', baglanti: this.baglanti });

} else {

this.kapi[i] = { image: nesne[i].image, x: nesne[i].x, y: nesne[i].y, type: nesne[i].type, input: '1', baglanti: this.baglanti };

}

}

}

}

}

\*------------------------------------------------------------------\*

kablo.js

\*------------------------------------------------------------------\*

class kablo{//kablo sınıfı

constructor() {

this.kablolar = []; //kablolar listemiz

}

draw(context,nesne) {//gelen canvas nesnesi ve kabloların tutulduğu liste

for(var i =0; i<nesne.length;i++){

context.beginPath();//tüm nesneler üzerinde dönerek kablo çizimleri yapılır.

context.moveTo(nesne[i].x1,nesne[i].y1);

context.lineTo(nesne[i].x2,nesne[i].y2);

context.stroke();

//sınıf içerisindeki kablo listesi güncellenir.

this.kablolar[i].x2 = nesne[i].x2;

this.kablolar[i].y2 = nesne[i].y2;

}

}

}

\*------------------------------------------------------------------\*

hesapla.js

\*------------------------------------------------------------------\*

class hesapla {//hesapla sınıfımız

constructor(context)

{//gelen canvas nesnesi sınfın bi değişkenine atılır

this.context = context;

}

hesap(kapilar, kablo) {//hesap fonksiyonu gelen kapı ve kablolar üzerinde hesaplama yapar

for (var i = 16; i < kapilar.kapi.length; i++) {//kapılar üzerinde gez

if (kapilar.kapi[i].type === "input") { //input kapıyı bul 1 sinyali

for (var j = 0; j < kablo.kablolar.length; j++) {//kablolar üzerinde döner

if (kablo.kablolar[j].kapi === i) { //bağlı olduğu kabloları bul ve kablo sinyali 1 yap

kablo.kablolar[j].sinyal = 1;//kablonun sinyali 1 yapılır

this.kabloSinyalleri(j, kapilar, kablo);//bağlı olduğu kapi ve kapılar ve kablololar yollanır

}

}

}else if(kapilar.kapi[i].type === "input1"){//input1 0 sinyal girişi

for (var j = 0; j < kablo.kablolar.length; j++) {//kablolar üzerinde döner

if (kablo.kablolar[j].kapi === i) { //bağlı olduğu kabloları bul ve kablo sinyali 1 yap

kablo.kablolar[j].sinyal = 0;//kablonun sinyali 0 yapılır

this.kabloSinyalleri(j, kapilar, kablo);//kablonun bağlı olduğu kapi indexi kapi ve kablolar

//fonksiyona yollanır

}

}

}

}

}

kabloSinyalleri(kabloindex, kapilar, kablo) { //kablonun bağlı olduğu kapı indexi kapilar ve kablolar yollanır

this.bkapiindex = kablo.kablolar[kabloindex].kapi1;//gelen kablo indexinden baplı olduğu kapı indexi bulunur

switch (kapilar.kapi[this.bkapiindex].type) {//switch case yapısında o kapı indexine bağlı

//olduğu kapilar ve kablolar yollanır

case "or"://or kapısı hesaplama fonksiyonu

this.hesapOr(this.bkapiindex, kapilar, kablo);

break;

case "nor"://nor kapısı hesaplama fonksiyonu

this.hesapOr(this.bkapiindex, kapilar, kablo);

break;

case "and"://and kapısı hesaplama fonksiyonu

this.hesapAnd(this.bkapiindex, kapilar, kablo);

break;

case "nand"://nand kapısı hesaplmama fonksiyonu

this.hesapAnd(this.bkapiindex, kapilar, kablo);

break;

case "xor"://xor kapısı hesapalama fonksiyonu

this.hesapXor(this.bkapiindex, kapilar, kablo);

break;

case "xnor"://xnor kapısı hesaplama fonksiyonu

this.hesapXor(this.bkapiindex, kapilar, kablo);

break;

case "not"://not kapısı hesaplama fonksiyonu

this.hesapNot(this.bkapiindex, kapilar, kablo);

break;

case "output"://output çıkışı hesaplama

this.hesapOutput(this.bkapiindex, kapilar, kablo);

break;

default:

}

}

hesapOr(kapiin, kapilar, kablo) { //or ve nor girişi hesaplaması

this.hesaplama = 0;

this.sonuc = 0;

for (var i = 0; i < kablo.kablolar.length; i++) {//kablolar üzerinde dönüşür

if (kablo.kablolar[i].kapi1 === kapiin) {//kapıya ait kablo üzerindeyse

if (kapilar.kapi[kapiin].input === '2') {

this.hesaplama += kablo.kablolar[i].sinyal;//toplama yapılır

}

if (kapilar.kapi[kapiin].input === '3') {

this.hesaplama += kablo.kablolar[i].sinyal;

}

}

}

if (this.hesaplama >= 1) {//eğer toplam 1 den büyük eşit ise

for (var i = 0; i < kablo.kablolar.length; i++) {

if (kablo.kablolar[i].kapi == kapiin) {

if (kapilar.kapi[kapiin].type === 'nor') {

kablo.kablolar[i].sinyal = 0;//nor kapısının çıkışındaki kabloya 0

this.kabloSinyalleri(i, kapilar, kablo);

} else if (kapilar.kapi[kapiin].type === 'or') {

kablo.kablolar[i].sinyal = 1;//or kapısının çıkışındaki kabloya 1 verilir

this.kabloSinyalleri(i, kapilar, kablo);

}

}

}

} else {//hesaplama 1 den küçük ise

for (var i = 0; i < kablo.kablolar.length; i++) {//kablo üstünde dön

if (kablo.kablolar[i].kapi == kapiin) {//kapı ya bak kabloya bağlımı ?

if (kapilar.kapi[kapiin].type === 'nor') {

kablo.kablolar[i].sinyal = 1;//nor girişinin çıkışındaki kablo 1

this.kabloSinyalleri(i, kapilar, kablo);

} else if (kapilar.kapi[kapiin].type === 'or') {

kablo.kablolar[i].sinyal = 0;//or girişinin çıkışındaki kablo 0

this.kabloSinyalleri(i, kapilar, kablo);

}

}

}

}

}

//kablonun bağlu olduğu kapiya ait output hesaplaması

hesapAnd(kapiin, kapilar, kablo) { //and ve nand girişi hesaplaması

this.hesaplama = 1;

this.sonuc = 0;

for (var i = 0; i < kablo.kablolar.length; i++) {

if (kablo.kablolar[i].kapi1 === kapiin) {

if (kapilar.kapi[kapiin].input === '2') {

this.hesaplama \*= kablo.kablolar[i].sinyal;//burda çaprma ile çözüm sonucu bulunur

}

if (kapilar.kapi[kapiin].input === '3') {

this.hesaplama \*= kablo.kablolar[i].sinyal;

}

}

}

if (this.hesaplama === 1) {//sonuc 1 ise

for (var i = 0; i < kablo.kablolar.length; i++) {

if (kablo.kablolar[i].kapi == kapiin) {

if (kapilar.kapi[kapiin].type === 'nand') {

kablo.kablolar[i].sinyal = 0; //nandın kablosuna 0 yollanır

this.kabloSinyalleri(i, kapilar, kablo);

} else if (kapilar.kapi[kapiin].type === 'and') {

kablo.kablolar[i].sinyal = 1;//andın kablosuna 1 yollanır

this.kabloSinyalleri(i, kapilar, kablo);

}

}

}

} else {//hesaplama 0 ise

for (var i = 0; i < kablo.kablolar.length; i++) {

if (kablo.kablolar[i].kapi == kapiin) {

if (kapilar.kapi[kapiin].type === 'nand') {

kablo.kablolar[i].sinyal = 1; //nande 1

this.kabloSinyalleri(i, kapilar, kablo);

} else if (kapilar.kapi[kapiin].type === 'and') {

kablo.kablolar[i].sinyal = 0; //and e 0

this.kabloSinyalleri(i, kapilar, kablo);

}

}

}

}

}

hesapXor(kapiin, kapilar, kablo) { //xor girişi hesaplaması

this.hesaplama = 0;

this.sonuc = 0;

for (var i = 0; i < kablo.kablolar.length; i++) {

if (kablo.kablolar[i].kapi1 === kapiin) {

if (kapilar.kapi[kapiin].input === '2') {

this.hesaplama += kablo.kablolar[i].sinyal;//sinyalleri toplama ile işleme

}

if (kapilar.kapi[kapiin].input === '3') {

this.hesaplama += kablo.kablolar[i].sinyal;

}

}

}

if (kapilar.kapi[kapiin].input == '2') {//xor veya xnor 2 girişli ise

if (this.hesaplama === 1) {//ve hesapşama 1 ,se

for (var i = 0; i < kablo.kablolar.length; i++) {

if (kablo.kablolar[i].kapi == kapiin) {

if (kapilar.kapi[kapiin].type === 'xnor') {

kablo.kablolar[i].sinyal = 0;//xnor çıkışı 0

this.kabloSinyalleri(i, kapilar, kablo);

} else if (kapilar.kapi[kapiin].type === 'xor') {

kablo.kablolar[i].sinyal = 1;//xor çıkışı 1

this.kabloSinyalleri(i, kapilar, kablo);

}

}

}

} else {//hesaplama 1 den farklı ise

for (var i = 0; i < kablo.kablolar.length; i++) {

if (kablo.kablolar[i].kapi == kapiin) {

if (kapilar.kapi[kapiin].type === 'xnor') {

kablo.kablolar[i].sinyal = 1;//xnor çıkışı 1

this.kabloSinyalleri(i, kapilar, kablo);

} else if (kapilar.kapi[kapiin].type === 'xor') {

kablo.kablolar[i].sinyal = 0;//xor çıkışı 0

this.kabloSinyalleri(i, kapilar, kablo);

}

}

}

}

}

if (kapilar.kapi[kapiin].input == '3') {//xor veya xnor 3 girişili ise

if (this.hesaplama === 1 || this.hesaplama === 3) {

for (var i = 0; i < kablo.kablolar.length; i++) {

if (kablo.kablolar[i].kapi == kapiin) {

if (kapilar.kapi[kapiin].type === 'xnor') {

kablo.kablolar[i].sinyal = 0;

this.kabloSinyalleri(i, kapilar, kablo);

} else if (kapilar.kapi[kapiin].type === 'xor') {

kablo.kablolar[i].sinyal = 1;

this.kabloSinyalleri(i, kapilar, kablo);

}

}

}

} else {

for (var i = 0; i < kablo.kablolar.length; i++) {

if (kablo.kablolar[i].kapi == kapiin) {

if (kapilar.kapi[kapiin].type === 'xnor') {

kablo.kablolar[i].sinyal = 1;

this.kabloSinyalleri(i, kapilar, kablo);

} else if (kapilar.kapi[kapiin].type === 'xor') {

kablo.kablolar[i].sinyal = 0;

this.kabloSinyalleri(i, kapilar, kablo);

}

}

}

}

}

}

hesapNot(kapiin, kapilar, kablo) { //not girişi hesaplamaso

this.hesaplama = 0;//not kapısı hesaplama

for (var i = 0; i < kablo.kablolar.length; i++) {

if (kablo.kablolar[i].kapi1 == kapiin)

{

if (kablo.kablolar[i].sinyal === 0)

{//sinyali tersine çevir

this.hesaplama = 1;

}else{

this.hesaplama = 0;

}

}

}

for(var i = 0; i<kablo.kablolar.length ; i++){

if(kablo.kablolar[i].kapi == kapiin){//kapı çıkışına yansıt

kablo.kablolar[i].sinyal = this.hesaplama;

this.kabloSinyalleri(i, kapilar, kablo);

}

}

}

hesapOutput(kapiin, kapilar, kablo) { //output girişi hesaplaması

for (var i = 0; i < kablo.kablolar.length; i++) {

if (kablo.kablolar[i].kapi1 == kapiin)

{//output çıkışı hesaplamnır

if (kablo.kablolar[i].sinyal === 1)

{//kablo sinyaline göre output değeri değiştirilir

kapilar.kapi[kapiin].baglanti[0].output = '1';

}else{

kapilar.kapi[kapiin].baglanti[0].output = '';

}

}

}

}

}

\*------------------------------------------------------------------\*

main.js

\*------------------------------------------------------------------\*

class MainClass {

constructor() { //sınıfdan nesne oluşunca ilk çalışacak fonksiyon

this.toolbar = document.getElementById('canvas');//canvas idli etiket alınır

this.tool = this.toolbar.getContext('2d');//tuval oluşturulur

document.addEventListener("mousedown", this.mouseDown.bind(this));//mouse tıklama eventi

document.addEventListener("mouseup", this.mouseUp.bind(this));//mouse tıklama bırakma eventi

document.addEventListener("mousemove", this.mouseMove.bind(this));//mouse hareketei eventi

document.addEventListener("click", this.mouseClick.bind(this));//mouse tıklayıp bırakma eventi

this.click = 0;

this.index = -1;//kapı index

this.indexkablo = -1;//kablo index

this.kapilar = [];//kapılar listesi

this.kapilarim = new kapilar();//kapılarım nesnesi

this.kablo = new kablo();//kablo nesnesi

this.hesapla = new hesapla(this.tool);//hesaplama nesnesi

}

init() { //başlangıç

this.drawtoolbar(this.tool);

}

drawtoolbar(context) { //canvas içerisi çizimi

context.fillStyle = 'White';

context.fillRect(0, 0, this.toolbar.width, this.toolbar.height);

context.font = "30px Arial";

context.fillStyle = 'Black';

context.fillText("ToolBox", 50, 30);

context.fillText("Devre Tasarım Ortamı", 500, 30);

context.beginPath();

context.moveTo(200, 0);//sınır çizgilerinin çizimi

context.lineTo(200, 850);

context.stroke();

context.beginPath();

context.moveTo(0, 40);

context.lineTo(1300, 40);

context.stroke();

if (this.kapilarim.kapi.length < 16) {//16 dan fazla nesne yoksa init dir

this.kapilarim.draw(context, 'init', this.kapilarim.image);

} else {//16 dan fazla nesne varsa event dir .

this.kapilarim.draw(context, 'event', this.kapilarim.kapi);

}//kapilarim sınıfının draw fonksiyonu yardımıyla kapılar çizdirilir

this.kablo.draw(context, this.kablo.kablolar);//kablo sınıfdanki draw fonksiyonu yardımıyla kablolar çizdirilir

}

mouseDown(e) { //kapi ve kabloların çizimi için ilk tıklama

this.bounds = e.target.getBoundingClientRect();

this.mouse = { x: e.pageX - this.bounds.left - scrollX, y: e.pageY - this.bounds.top - scrollY };

//tıklanınlan yerin tam x,y koordinatları için gerekli hesaplamalar

for (var i = 0; i < this.kapilarim.kapi.length; i++) {//tıklandıktan sonra kapılar üzerinde gezilir

if (e.pageX > this.kapilarim.kapi[i].x + 15 && e.pageX < this.kapilarim.kapi[i].x + 40 && e.pageY > this.kapilarim.kapi[i].y + 20 && e.pageY < this.kapilarim.kapi[i].y + 60) {

this.click = 1;//eğer tıklanınlan yer bir kapıyı kapsıyorsa click 1 yapılır

this.index = i;//kapı indexi alınır

if (e.pageX < 240 && e.pageX > 0) {

//ve kapı toolbox içerisindeyse listeye pushlanır

this.kapi = { image: this.kapilarim.kapi[i].image, x: this.kapilarim.kapi[i].x, y: this.kapilarim.kapi[i].y, type: this.kapilarim.kapi[i].type, input: this.kapilarim.kapi[i].input, baglanti: this.cbaglanti };

this.kapilarim.kapi.push(this.kapi);

this.index = this.kapilarim.kapi.length - 1;//kapı indexi tutulur

}

break;

}

for (var j = 0; j < this.kapilarim.kapi[i].baglanti.length; j++) { //kablo çizimi

if (this.mouse.x > this.kapilarim.kapi[i].baglanti[j].x && this.mouse.x < this.kapilarim.kapi[i].baglanti[j].x + 5 && this.mouse.y > this.kapilarim.kapi[i].baglanti[j].y && this.mouse.y < this.kapilarim.kapi[i].baglanti[j].y + 5) {

//tıklanan yer lojik kapı üzerindeki yuvarlak daireler ise

//kabloalr nesnesine pushlanır

this.kablo.kablolar.push({ x1: this.kapilarim.kapi[i].baglanti[j].x, y1: this.kapilarim.kapi[i].baglanti[j].y, x2: null, y2: null, kapi: i, baglanti: j, kapi1: null, baglanti1: null, sinyal: 0 });

this.indexkablo = this.kablo.kablolar.length - 1;

if (this.kapilarim.kapi[i].type === "input") {

this.kablo.kablolar[this.indexkablo].sinyal = 1;//input ise sinyali 1 yapılır

}

this.click = 2;//kablo clicki

}

}

}

}

mouseUp(e) { //kapi ve kablo tıklamalarını bırakılması

this.bounds = e.target.getBoundingClientRect();

this.mouse = { x: e.pageX - this.bounds.left - scrollX, y: e.pageY - this.bounds.top - scrollY };

if (this.click === 1) {

if (e.pageX < 240 && e.pageX > 0) {//eğer toolbox üzerindeyse diziden silinir.

this.kapilarim.kapi.splice(this.index, 1);

}

this.init();//init ile tüm nesneler tekrardan çizdirilir.

this.click = 0;//click sıfırlanır

}

if (this.click === 2) {//kablo çizme işlemi bittiğinde

this.sonuc = 1;

if (e.pageX < 160 && e.pageX > 0) {//toolbox üzerindeyse silinir.

this.kablo.kablolar.splice(this.indexkablo, 1);

}

for (var i = 16; i < this.kapilarim.kapi.length; i++) {//eğer bi kapının inputuna bırakmak isteniyorsa

for (var j = 0; j < this.kapilarim.kapi[i].baglanti.length; j++) {

if (this.mouse.x > this.kapilarim.kapi[i].baglanti[j].x - 5 && this.mouse.x < this.kapilarim.kapi[i].baglanti[j].x + 5 && this.mouse.y > this.kapilarim.kapi[i].baglanti[j].y - 5 && this.mouse.y < this.kapilarim.kapi[i].baglanti[j].y + 5 && this.kapilarim.kapi[i].baglanti[j].input != "") {

//bıraktığı yer kapının input yuvarlağısa x,y değişkenleri kabloya aktarılır

this.kablo.kablolar[this.indexkablo].x2 = this.kapilarim.kapi[i].baglanti[j].x;

this.kablo.kablolar[this.indexkablo].y2 = this.kapilarim.kapi[i].baglanti[j].y;

this.kablo.kablolar[this.indexkablo].kapi1 = i;//bağlandığı kapı ve hangi inputa bağlandığı bilgilerini alır

this.kablo.kablolar[this.indexkablo].baglanti1 = j;

this.sonuc = 1;

break;

} else {

this.sonuc = 0;

}

}

if (this.sonuc === 1) { break; }

}

if (this.sonuc === 0) {//sonuc 0 ise kablo silinir

this.kablo.kablolar.splice(this.indexkablo, 1);

}

this.init();//tüm nesneler çizdirilir.

this.click = 0;//click sıfırlanır

}

}

mouseMove(e) { //kapi ve kablo haraketleri

this.bounds = e.target.getBoundingClientRect();

this.mouse = { x: e.pageX - this.bounds.left - scrollX, y: e.pageY - this.bounds.top - scrollY };

if (this.click === 1) {//lojik kapı hareket halindeyse

this.kapilarim.kapi[this.index].x = e.pageX - 30;

this.kapilarim.kapi[this.index].y = e.pageY - 45;

//kapıyı oynatmada kapının koordinatları sürekli değişir o yüzden move eventinden gelen mouse x,y

//koordinatları kullanılır

}

if (this.click === 2) {//kablo hareket halindeyse

//aynı işlemler yapılır

this.kablo.kablolar[this.indexkablo].x2 = this.mouse.x;

this.kablo.kablolar[this.indexkablo].y2 = this.mouse.y;

}

this.init();//tüm nesneler tekrardan çizdirilir.

}

mouseClick(e) { //simulate butonu

if (e.path[0].nodeName === "BUTTON") {//tıklanılan buton simulate ise

this.hesapla.hesap(this.kapilarim, this.kablo);//hesapla sınıfındaki hesap fonksiyonu çalışır

this.init();//tüm nesneler tekrardan çizdirlir

}

}

}

const erhan = new MainClass();//MainClassdan erhan nesnesi oluşturulur

window.onload = () => erhan.init();//sayfa yüklendiğinde erhan nesnesinin init fonksiyonu çalışır.

\*------------------------------------------------------------------\*