KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ÇARPMA DEVRESİ SİMÜLASYONU

TASARIM PROJESİ

Reyhan HAYIR
Kübra AYDIN

2015-2016 GÜZ DÖNEMİ

KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ÇARPMA DEVRESİ SİMÜLASYONU

TASARIM PROJESİ

REYHAN HAYIR
KÜBRA AYDIN

Bu projenin teslim edilmesi ve sunulması tarafımca uygundur.

Danışman : PROF. DR. MURAT EKİNCİ .....

2015-2016 GÜZ DÖNEMİ

IEEE Etik Kuralları
IEEE Code of Ethics

Mesleğime karşı şahsi sorumluluğumu kabul ederek, hizmet ettiğim toplumlara ve üyelerine en yüksek etik ve mesleki davranışta bulunmaya söz verdiğimi ve aşağıdaki etik kurallarını kabul ettiğimi ifade ederim:

1. Kamu güvenliği, sağlığı ve refahı ile uyumlu kararlar vermenin sorumluluğunu kabul etmek ve kamu veya çevreyi tehdit edebilecek faktörleri derhal açıklamak;
2. Mümkün olabilecek çıkar çatışması, ister gerçekten var olması isterse sadece algı olması, durumlarından kaçınmak. Çıkar çatışması olması durumunda, etkilenen taraflara durumu bildirmek;
3. Mevcut verilere dayalı tahminlerde ve fikir beyan etmelerde gerçekçi ve dürüst olmak;
4. Her türlü rüşveti reddetmek;
5. Mütenasip uygulamalarını ve muhtemel sonuçlarını gözeterek teknoloji anlayışını geliştirmek;
6. Teknik yeterliliklerimizi sürdürmek ve geliştirmek, yeterli eğitim veya tecrübe olması veya işin zorluk sınırları ifade edilmesi durumunda ancak başkaları için teknolojik sorumlulukları üstlenmek;
7. Teknik bir çalışma hakkında yansız bir eleştiri için uğraşmak, eleştiriyi kabul etmek ve eleştiriyi yapmak; hatları kabul etmek ve düzeltmek; diğer katkı sunanların emeklerini ifade etmek;
8. Bütün kişilere adilane davranmak; ırk, din, cinsiyet, yaş, milliyet, cinsi tercih, cinsiyet kimliği, veya cinsiyet ifadesi üzerinden ayırımcılık yapma durumuna girişmemek;
9. Yanlış veya kötü amaçlı eylemler sonucu kimsenin yaralanması, mülklerinin zarar görmesi, itibarlarının veya istihdamlarının zedelenmesi durumlarının oluşmasından kaçınmak;
10. Meslektaşlara ve yardımcı personele mesleki gelişimlerinde yardımcı olmak ve onları desteklemek.

IEEE Yönetim Kurulu tarafından Ağustos 1990'da
onaylanmıştır.

ÖNSÖZ

"Çarpma Devresi Simülasyonu " adlı bu çalışma Karadeniz Teknik Üniversitesi Bilgisayar Mühendisliği Anabilim Dalında Tasarım Projesi olarak hazırlanmıştır. Bu proje ile bilgisayarın temel işlemlerinden biri olan çarpma işleminin mantıksal devresinin tasarımı yapılmıştır. Ayrıca kaydırmalı çarpma işleminin nasıl gerçekleştiği ve devrenin nasıl çalıştığı araştırılmıştır. Son olarak da Visual Studio programı ve C# dili kullanılarak bu devrenin simülasyonunun nasıl gerçekleşeceği konularında araştırmalarda bulunulmuştur.

Projemizin her aşamasında bizden yardımlarını esirgemeyen
Sayın Prof.Dr. MURAT EKİNCİ 'ye ve bütün Karadeniz Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü hocalarına teşekkürlerimizi sunarız.

REYHAN HAYIR
KÜBRA AYDIN
Trabzon 2015

İÇİNDEKİLER

ÖZET

Teknoloji, günümüzde son derece hızlı bir şekilde gelişmektedir. Teknoloji alanında meydana gelen bu gelişmeler, bizi bu teknolojinin temelde nasıl çalıştığını unutmaya ve öğrenmemeye itmektedir. Günümüzdeki karmaşık sistemlerin temelde basit dört işlem ile kontrol edildiği bilinmektedir. Mühendisler yaptıkları projelerde geliştirilmiş daha basit yapıları tercih ederek asıl mühendislikten uzaklaşıyorlar.

Bu projede Çarpma Devresinin Simülasyonu hazırlanarak Bilgisayarın donanımsal olarak bu işlemi nasıl gerçekleştirdiğinin anlatılması hedeflenmiştir. Bu simülasyonu gerçekleştirirken Visual Studio form uygulaması oluşturulmuştur. Uygulama üzerinde saat darbeleri, kayan bitler ve kaydedicilerin içinde meydana gelen değişimler gösterilmiştir.

1. GENEL BİLGİLER

1.1. Giriş

Projenin Konusu: Bilgisayar da gerçekleşen işlemlerin temelini oluşturan çarpma devresinin simülasyonunun oluşturulması.

Projenin Amacı: Proje, bilgisayar teknolojisi ile ilgili olan mühendis ve mühendis adaylarına yönelik olup, mantıksal çarpma devresinin işleyişinin gösterilmesi amacıyla gerçekleştirilmiştir.

Proje yazılımsal araçlar ile oluşturulmuştur. Devrenin içi form sayfasında sanal olarak gerçeklenmiştir. Buna ek olarak kaydırmalı çarpma işlemi de anlatılmıştır.

2. YAPILAN ÇALIŞMALAR

2.1. TOPLAMA DEVRESİ

Binary sayılarla toplama, çıkarma, çarpma ve bölme gibi işlemler yapılır. Sayısal bilgisayarlarda ve hesap makinalarında esas işlemler toplama ve çıkarma işlemleridir. Çarpma işlemi tekrarlanan toplama işlemleri ile yapılır. Onluk tabanda toplama işlemi normal toplama işlemi gibi yapılır.

ÖRNEK:

```
  1037483
  3849452
+_________
  4886935
```

İkili tabanda toplama işlemi ise aşağıdaki şekilde yapılır:

Kurallar:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10 (Sum = 0, carry = 1)

ÖRNEK:

$(1001)_2$
$(0110)_2$
+_____
$(1111)_2$

Lojik devrelerde kullanılan iki temel toplayıcı tipi vardır. İki bitin toplamasını yapan devreye Yarım Toplayıcı, üç bitin toplamasını yapan devreye de Tam Toplayıcı devresi adı verilir.

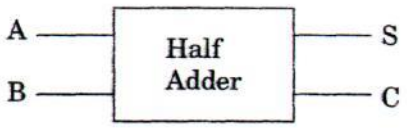
2.1.1. Yarı Toplayıcı Devreler

Bu devre giriş değişkenlerini toplar ve toplanan bitleri ve çıkış değişkenlerini ise toplam (SUM) ve elde (CARRY) oluşturur. Yarım toplayıcı devresi, en basit bir toplama devresidir. Burada yarım toplayıcı devresini tasarım yolu ile gerçekleştireceğiz.

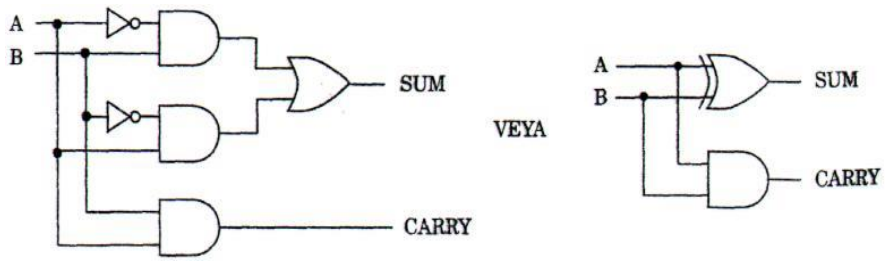
GİRİŞLER		ÇIKIŞLAR	
A	B	SUM (S)	CARRY (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

SUM (TOPLAM) = $\overline{A}B + A\overline{B} = A \oplus B$

CARRY (ELDE) = AB



Şekil 2.1. Yarım Toplayıcının Doğruluk Tablosu



Şekil 2.2. Yarım Toplayıcı Devre

2.1.2. Tam Toplayıcı Devreler

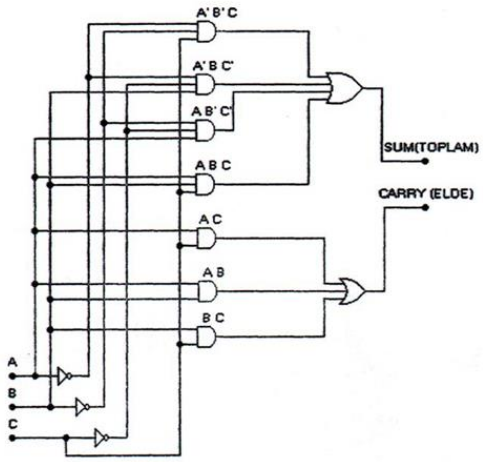
Girişindeki 3 bitin toplamını gerçekleştiren devredir. Bu devrenin 3 girişi ve 2 çıkışı olup, girişlerden ilk ikisi toplanacak iki değerlikli biti, son giriş ise bir önceki düşük değerlikli bitlerin toplamından gelen eldeyi gösterir. Yani girişlere A, B, C dersek, A ve B girişleri toplanacak iki biti gösterir. C giriş değişkeni ise, A ve B bitlerinin toplamındaki eldeyi (carry) varsa gösterir. Bir tam toplayıcı devresi iki yarım toplayıcının birleşiminden oluşur. Burada tam toplayıcı devresini iki türlü gerçekleştireceğiz. [5]

GİRİŞLER			ÇIKIŞLAR	
A	B	C	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\text{SUM (TOPLAM)} = \bar{A}\,\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\,\bar{C} + ABC$$

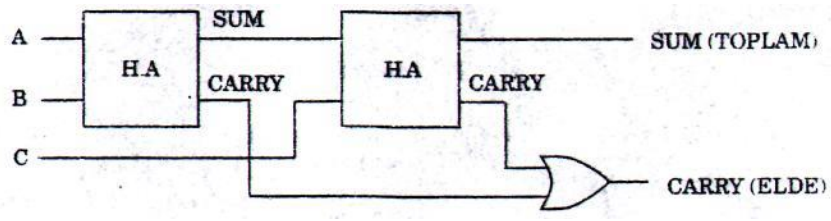
$$\text{CARRY (ELDE)} = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

Şekil 2.3. Tam Toplayıcı Devre1



Şekil 2.4. Tam Toplayıcı Devre2

2.1.3. İki Half Adder'ın Birleştirilmesiyle Full Adder'ın Elde Edilmesi



Şekil 2.5. Tam Toplayıcı Elde Edilmesi

2.2. ÇARPMA DEVRESİ

2.2.1. Decimal Sayıların Çarpılması:

Çarpan ve çarpılan değerlerinden oluşur. İki değer çarpılarak yine decimal bir sonuç elde edilir.

Örnek:

$$\begin{array}{r} 35 \\ \times\ 48 \\ \hline 280 \\ +\ 140 \\ \hline 1680 \end{array}$$

2.2.2. Binary Sayıların Çarpılması:

Çarpılan ve çarpan değerlerinden oluşur. Decimal'den farkı çarpan ve çarpılan değerlerimiz '0' ve '1' lerden oluşmasıdır. Çarpma işleminde n bitlik sayılar kullanılıyorsa sonuç en fazla 2n bit olur. Çarpma işleminin aşamaları aşağıdaki şekildedir:

- Decimal olan değer varsa binary'ye çevrilir.
- Çarpılan ve çarpan alt alta yazılır.
- Normal çarpma yapılır
  1*1=1
  1*0=0
  0*1=0
  0*0=0
- Her basamakta bir sola kaydırılır.
- Son olarak bulunan değerler alt alta toplanır.

Örnek:

```
  1011
x 1001
------
  1011
 0000
0000
+1011
------
1100011
```

```
  1011
x 1101
------
  1011
 0000
1011
1011
--------
10001111
```

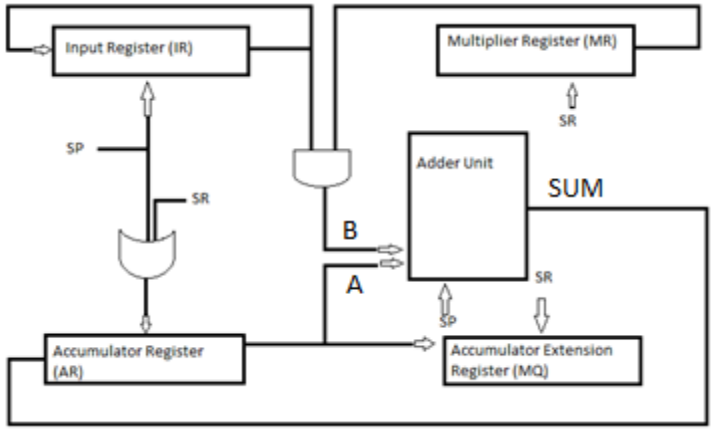
2.2.3. Kaydırmalı Çarpma:

Binary çarpmada bir diğer yöntem kaydırma yapılarak çarpma işleminin gerçekleştirilmesidir. Bu yöntemi aşağıdaki örnekle açıklayalım.

1.	1011	
2.	1001	
3.	1011	
4.	101	1
5.	0000	
6.	0101	1
7.	010	11
8.	0000	
9.	0010	11
10.	0001	011
11.	1011	
12.	1100	011
13.	0110	0011

- 1.satırdaki değer çarpılandır.
- 2.satırdaki değer çarpandır.
- Çarpanın en soldaki biti 1 olduğu için çarpılanın değeri 3.satıra aynen yazılır.
- Daha sonra bir sağa kaydırılır.
- Çarpanın soldan ikinci biti 0 olduğu için 5.satıra '0000' yazılır.
- 4.satırdaki kaydırılmış değer ile '0000' değeri toplanır ve 6.satıra yazılır.
- Her işlem sonrası kaydırılma yapıldığı için 6.satırdaki değer de bir sağa kaydırılarak 7.satıra yazılır.
- Çarpanın soldan üçüncü biti 0 olduğu için aynı işlemler tekrar yapılır ve 10.satıra kaydırılmış değer yazılır.
- Çarpanın en sağdaki biti 1 olduğu için çarpılanın değeri aynen 11.satıra yazılır.
- 10.satırdaki değer ile toplanır ve 12.satıra yazılır.
- Son olarak sağa kaydırılarak çarpma işlemi sonlandırılır.

2.2.4. Çarpma İşleminin Mantıksal Devresi



Şekil 2.6. Çarpma İşleminin Mantıksal Devresi

Input Register(IR) çarpılan değeri, Multiplier Register(MR) çarpan değeri tutar. Accumulator Register(AR) sonucun en anlamlı bitlerinin tutulduğu kaydedicidir. Accumulator Extension Register(MQ) ise sonucun en anlamsız bitlerinin tutulduğu kaydedicidir. Multiplier Register'ına bağlı olan SR'nin 1 saat darbesinde, Input Register'ına bağlı olan SP saat darbesi 4 kez tekrarlanır. Bu sayede çarpan sayının bir biti ile çarpılan sayının tüm bitleri AND kapısı yardımı ile çarpılır. Accumulator Register'ın SR ve SP saat darbelerinin herhangi birinde cevap vermesi OR kapısı yardımı ile sağlanmıştır. AND Kapısı çıkışı (B) ve AR' nin en sağdaki biti (A) Adder Unit'e gelerek eldeli toplama işlemini gerçekleştirir. Oluşan sonuç (SUM) AR'nin en soldaki bitine gelir ve AR'nin içeriği 1 sağa kaydırılır. MQ ise her SR saat derbesinde kaydırma işlemi gerçekleştirir. Bit sayısı kadar döngü olur. MR'deki bitler tamamlandığında çarpma işlemi sonucun AR ve MQ Register'ların da yazan değerlerdir.

2.3. MICROSOFT VISUAL STUDIO

Microsoft Visual Studio, Microsoft tarafından geliştirilen bir tümleşik geliştirme ortamıdır (IDE). *Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework* ve *Microsoft Silverlight* tarafından desteklenen tüm platformlar için yönetilen kod ile birlikte yerel kod ve Windows Forms uygulamaları, web siteleri, web uygulamaları ve web servisleri ile birlikte konsol ve grafiksel kullanıcı arayüzü uygulamaları geliştirmek için kullanılır.

Visual Studio IntelliSense'in yanı sıra "code refactoring" destekleyen bir kod editörü içerir. Entegre hata ayıklayıcı, hem kaynak-seviyesinde hem de makine-seviyesinde çalışır. Diğer yerleşik araçlar, GUI uygulamaları, web tasarımcısı, sınıf tasarımcısı ve veri tabanı şema tasarımcısı yaratabilmek için bir form tasarımcısı içerir.

Hemen hemen her işlevsellik düzeyinde dahil olmak üzere, kaynak kontrol sistemleri için destek (Subversion ve Visual SourceSafe gibi) sunan eklentileri kabul eder.

Visual Studio, değişik programlama dillerini destekler, bu da kod editörü ve hata ayıklayıcısının neredeyse tüm programlama dillerini desteklemesini sağlamaktadır. Dahili diller *C/C++* (Görsel yoluyla C++), *VB.NET* (Visual Basic .NET üzerinden), *C#* (Visual C# ile), ve *F#* (Visual Studio 2010 itibariyle) içermektedir.

Visual Studio'da görsellik ön plandadır. Genel olarak "sürükle bırak" mantığı ile çalışarak program yazmanızı kolaylaştırır. Bazı standart kodları bu şekilde sizin yerinize otomatik olarak yapar.

2.3.1. Araç Çubukları (Toolbars)

Visual Studio, menü komutlarını için görsel kısa yolları araç çubukları ile sunar. Benzer işlemler için kullanılan komutlar bir araç çubuğunda gruplanır. Örneğin; standart araç çubuğu, yeni dosya oluşturmak, bir dosyayı açmak – kaydetmek gibi genel dosya işlemleri için kullanılır.

Araç çubukları, varsayılan olarak menülerin altında bulunur. Ancak araç çubukları taşınarak yerlerini değiştirebilir veya kayan duruma getirilebilir. Ayrıca istenen çubuklar saklanılabilir veya gösterilebilir. Araç çubuklarını listesini görmek için View menüsünden Toolbars alt menüsüne işaret edin.

Visual Studio bize kendi araç çubuklarımızı oluşturma imkânı da verir. Farklı işlevlere sahip komutlar gruplanıp, kişisel araç çubuğu oluşturulabilir.

2.3.2. Menüler

Birçok çalışma ortamının yaptığı gibi Visual Studio da, benzer öğeler üzerinde işlevleri olan komutları menüler halinde gruplar. Araç çubuklarından farkı sabit olmaları ve özelleştirmeye açık olmamalarıdır. Menüler bu modülde detaylı olarak ele alınacaktır.

2.3.3. Paneller

Paneller, Visual Studio içindeki pencerelerdir. Çalışma ortamında birçok panel bulunmasıyla beraber, Solution Explorer, Toolbox, Object Browser, Properties, Watch, Output, Search Result, Task List gibi sıkça kullandığımız paneller vardır.

İPUCU: Görmek istenilen paneller View menüsünden seçilebilir.

Paneller, Visual Studio ortamı içersinde istenilen yere taşınabilir veya sabitlenebilir. Panellerin birkaç genel özelliği vardır:

Auto Hide (Otomatik Gizle):
Panelin, fare üzerindeyken gözükmesi ve fare çekildikten sonra gizlenmesidir.

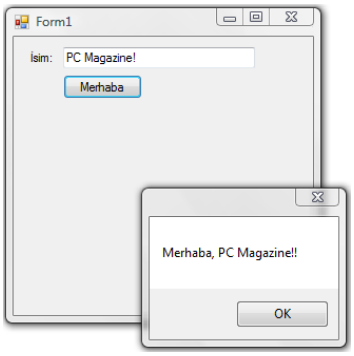
Dockable (Sabitlenebilir):
Panelin, Visual Studio ortamı içersinde bir yerde sabitlenebilme özelliğidir.

Floating (Kayan):
Kayan paneller herhangi bir yere sabitlenemez. Ancak her sayfanın üstünde durur ve böylece sürekli görünür. [1]

2.3.4. Windows Form Tasarımcısı (Windows Form Designer)

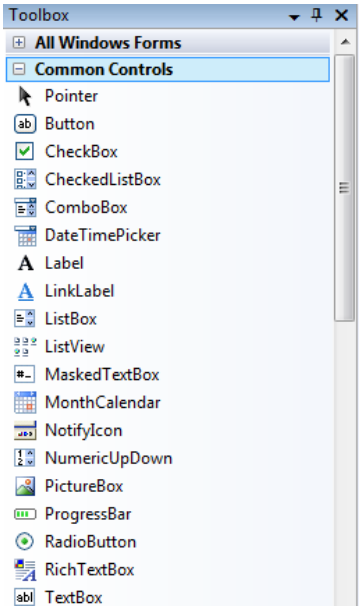
Windows Forms tasarımcısı Windows Forms kullanarak GUI uygulamaları oluşturmak için kullanılır. Plan, diğer konteynerler içerisinde kontrollerin barındırılarak ya da formun yan tarafında kilitlenerek kontrol edilebilir. Veri görüntüleyen kontroller (metin kutusu, liste kutusu, ızgara görünümü, vb gibi), veri tabanları veya sorgular gibi veri kaynaklarına bağlı olabilir. [2]

Aşağıdaki şekil Visual Studio'da oluşturulmuş bir form uygulamasıdır:



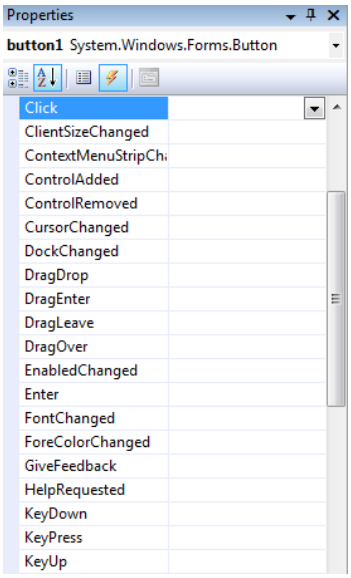
Şekil 2.7. Form Örneği

Aşağıda form uygulamasına ait Toolbox verilmiştir. Toolbox'ta form uygulamasında kullanılacak elemanlar bulunur. Kullanılacak eleman Toolbox'tan sürüklenerek form'da kullanılacak kısma bırakılır.



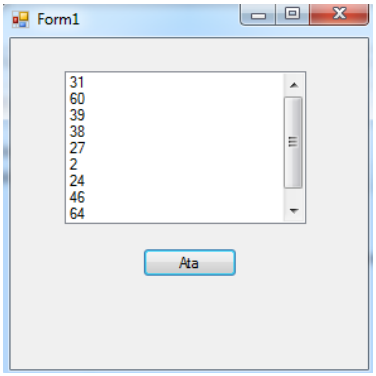
Şekil 2.8. ToolBox

Seçilen elemanın özelliklerinin gösterilmesi ve içeriğinin değiştirilmesi aşağıdaki bölümden yapılır.



Şekil 2.9. Properties

Aşağıda oluşturulmuş basit bir form örneği verilmiştir:



Şekil 2.10. Form Örneği

2.4. C# PROGRAMLAMA DİLİ

2.4.1. C# Nedir?

C#, yazılım sektörü içerisinde en sık kullanılan iki yazılım dili olan C ve C++ etkileşimi ile türetilmiştir. Ayrıca C#, ortak platformlarda taşınabilir bir (portable language) programlama dili olan Java ile pek çok açıdan benzerlik taşımaktadır . En büyük özelliği ise .Net Framework platformu için hazırlanmış tamamen nesne yönelimli bir yazılım dilidir. Yani nesneler önceden sınıflar halinde yazılıdır. Programcıya sadece o nesneyi sürüklemek ve sonrasında nesneyi amaca uygun çalıştıracak kod satırlarını yazmak kalır.

Microsoft tarafından geliştirilen C#, C++ ve Visual Basic dillerinde yer alan tutarsızlıkları kaldırmak için geliştirilmiş bir dil olmasına rağmen kısa süre içerisinde nesne yönelimli dillerin içinde en gelişmiş programlama dillerinden biri olmayı başarmıştır. Ayrıca gelişmiş derleyicisi (debugger) ile hata olasılığını ortadan kaldırmaktadır. Yazılan program çalıştırıldıktan sonra derleyici tarafından algılanan Sınıf (Class) ve söz dizimi (syntax) hataları yazılımcıya ayrı bir ekranda ayrıntısı ile gösterilir ve yazılımcı bu hata penceresinden hataları tespit ederek kolayca düzeltebilir. Ayrıca pek çok programcı tarafından kullanılan bir dil olmasından ötürü takıldığımız noktalarda uzman programcılardan yardım almak oldukça kolaydır.

2.4.2. .NET Framework Nedir?

C# ve .Net Framework bazı kişiler tarafından tek bir kavram olarak algılanmaktadır. Fakat bu iki kavram birbirlerinden tamamen farklı amaçlar için geliştirilmiştir. C#, nesne yönelimli bir programlama diliyken .Net Framework ise C# için geliştirilmiş bir çalıştırma

ortamıdır. Aslında C# dili, Microsoft tarafından .Net platformu için kod geliştirmek amaçlı tasarlanmış ve C# içerisindeki tüm kütüphaneler .Net platformu içinde tanımlanmış kütüphanelerdir. Java'dan önce, geliştirilen yazılımlar makine koduna çevrilerek çalıştırılırdı. Java ise program kodlarını önce byte sayı sistemine çevirir. Sonrasında Java Sanal Makinesi (JVM – Java Virtual Machine) bu kodları işletim sisteminin istediği koda çevirerek programın çalışmasına sağlar.

.Net Platformu da Java diline benzer bir çalışma mantığı izleyerek kodları çalışabilir hale getirmektedir. .Net platformunda kod ilk önce Microsoft Intermediate Language (Microsoft Ara Dili) olarak isimlendirilmiş dosya haline dönüştürülür bu dosya içerisinde derlenen kodların Microsoft'un standart haline getirdiği bir assembly dili haline dönüştürür. Bu ara dil de saklanan dosyalar çalıştırılmak istendiğinde ise CLR adı verilen sistem MSIL kodlarını çalıştırır.

Ortak dil çalışma zamanı (CLR – Common Language Runtime) sisteminin temel görevi ise C# dilini taşınabilir kılmak ve diğer diller ile güvenli bir şekilde çalışmayı sağlayan sistemdir. CLR, .Net platformuna ait bir kod çalıştığı zaman JIT (Just in Time – Tam Zamanında) derleyiciyi aktif hale getirir. Aktif hale gelen JIT derleyici, MSIL kodlarını yerel kod yapısına göre çalıştırarak ortak platform yapısı sağlanmış olur.

2.4.3. C# ile Neler Yapılabilir?

Bu kısımda ise sizlere C# ile proje geliştirme hakkında basit ipuçları ve giriş yapılacaktır. Aşağıda Visual C# kullanarak ne tür uygulamalar geliştirebileceğimizi anlatmaya çalıştım.

2.4.3.1. Konsol Uygulamaları (Console Applications)

Grafiksel kullanıcı arayüzünden çok, komut satırı penceresinde programcı tarafından yazılan uygulama çalışmalarına verilen isimdir. Konsol uygulamaları MS- DOS olarak ifade edilen nesne yönelimli programlamanın yaygınlaşmadığı dönemlerde kullanılan bir arayüz olmuştur. Bu uygulama modelini kullanıcının etkileşim kurması gerekmeyen uygulamalarda basit bir arayüz oluşturmak için kullanılabilir.

Şekil 2.11. Console Uygulaması

2.4.3.2. Windows Form Uygulamaları (Ado.Net)

Grafiksel kullanıcı arayüzü (GUI – Graphical User Interface) olarak tanımlanır. Windows form uygulamalarını, konsol uygulamalarından ayıran en önemli özelliği de budur. Windows Form uygulaması oluşturmak için Toolbox adı verilen araç kutusundan nesnel araçlar sürükle bırak yöntemi ile kolayca oluşturulabilir.

Windows Form uygulamaları**,** bilgisayar programları geliştirmek amacıyla kullanılmaktadır. Windows Forms Application seçeneği ile yeni bir windows uygulama penceresi açılarak program geliştirmeye başlanabilir. Ayrıca Windows Form uygulaması içinde grafiksel olarak daha gelişmiş çalışmalar yapmak da mümkün hale getirilmiştir. Daha güçlü grafiksel arayüzler için WPF ile uygulama geliştirmek gerekmektedir.
Windows form uygulamaları için veritabanı bağlantısı kurabilmek için kullanılan .Net kütüphanesinin adı Ado.Net olarak tanımlanmıştır. Bu konu üzerinde çalışmak isteyenlerin internet üzerinde Ado.Net olarak arama yapmaları gerekmektedir.

2.4.3.3. Web Form Uygulaması (Asp.Net)

Web form uygulamaları ise Ado.Net yerine **ASP.Net** sayfa yapısı ile çalışmaktadır. Web form uygulamaları temel olarak Windows Form Uygulaması gibi düşünülebilir. Fakat aralarında temel faklılıklar vardır. Web Form uygulamaları internet üzerinden erişilebilen form uygulamalarıdır. Bundan dolayı uygulamalar kişisel bilgisayar yerine web tarayıcısı ile çalışmaktadır.

ASP.Net kullanılarak oluşturulan bir web uygulaması içerisinde bir yada birden daha çok ASP.Net sayfası olabilir. Web formları, .Net kütüphanesinde özel etiketler içeren bir HTML sayfalarıdır. Web Formları, uzantısı .aspx olarak işlenir.

Ayrıca her web form uygulamasında Web.config ile belirtilen bir yapılandırma dosyası vardır. Bu dosya XML biçimindedir ve web uygulamasının güvenlik, önbellek yönetimi gibi ilgili işlerin bilgilerini içerir.

2.4.3.4. Mobil Programlama

Windows Phone işletim sistemleri için uygulama geliştirme kısmıdır. Eğer Windows Phone için uygulama geliştirmek istiyorsanız bu kısmı kullananız gerekmektedir. Mobil Programlamaya başlamak istiyorsanız daha önce hazırladığım Windows Phone uygulaması geliştirme adlı yazıma bakabilirsiniz.

2.4.3.5. Web Servisleri

Kısaca veri iletimi için kullanılan sistemlerdir diyebiliriz. Biraz daha açacak olursam ise Evrensel veri transer metodu sayılan XML ve http kuralları ile internet ağı ile dünyanın her hangi bir yerine veri taşıyan sistemlerdir. Web servisi işlemlerinde veriler bir başka kullanıcıya gönderildiği için güvenlik ve adresleme gibi konular oldukca önemlidir. Microsoft, C#'ta web servisleri kullanımını güvenli kılan pek çok işlemler yapmaktadır. Web servisleri ile ileri zamanlarda daha geniş bir yazı yazmayı planlamaktayım.[3]

2.5. SİMÜLASYON

Farklılık, simülasyon veya öğrence, teknik olmayan anlamda bir şeyin benzeri veya sahtesi anlamında kullanılır. Teknik anlamda gerçek bir dünya süreci veya sisteminin işletilmesinin zaman üzerinden taklit edilmesidir. Sistem nesneleri arasında tanımlanmış ilişkileri içeren sistem veya süreçlerin bir modelidir.

"Simülasyon" terimi, "benzer" anlamındaki *similis* kökünden gelen, bir şeyin benzerini (taklidini) yapmak demek olan ve 14. yüzyıldan beri Latince'de kullanılan *simulare* sözcüğünden türetilmiştir. Bu terim ancak 20. yüzyılda teknik bir anlam kazanmıştır. Günümüzde, Batı dillerinde teknik olan ve olmayan anlamları ile kullanılmakta ve yerine göre hangi anlama geldiği anlaşılmaktadır.

Bir araç olarak kullanılan benzetim, günümüzde mevcut olan ve ileride mevcut olabilecek işlemler hakkında objektif bilgiler sağlar. Taklit edilen gerçek bir olay, genelde bilgisayar yardımıyla yapılmaktadır.

Örneğin bir uçuş simülatörü, uçuşun bazı kurallarının bilgisayar üzerinde öğretilmesi amacıyla kullanılan bir benzetim modelidir. Pilotun kokpitte göreceği ekranın bir benzerini bilgisayar ekranında görmesi ve uçuşu kontrol etme işlemlerini gerçekten uçaktaymış gibi yapması bir benzetim olayıdır. Ayrıca sinyalizasyon sisteminde, trafik ışıklarının planlanmasında , hizmet ve üretim sektöründe kuyrukların ve birikimlerin planlamasında kullanılan bir matematik modelleme yöntemi olarak yerini almıştır.

Simülasyon kelimesinin günümüzde en çok kullanılan teknik anlamı ise, herhangi bir sürecin ya da sistemin işletilmesi için zamanlı olarak yapay bir ortam oluşturulması ya da düzenin taklit edilmesidir. Gerçek dünyada gerçekleşen süreçlerin ya da sistemlerin gerçeğe çok yakın bir şekilde taklit edilmesi insan emeğinden zaman tasarrufuna, ekonomik anlamda kazançtan kazanların önlenmesine birçok konuda büyük avantajlar sağlamaktadır.

Üretim maliyeti çok yüksek olan ve insan kaynaklı hatalar nedeniyle herhangi bir kaza gerçekleşmesi halinde hem insan canına mal olan hem de çok büyük bir maliyetin boşa gitmesine neden olan teknik donanımlarının uzmanlık eğitimi, bilgisayar ortamında kullanılan simülasyon programları sayesinde gerçek zamanlı ancak risksiz olarak yapılabilmektedir. Gerçek dünyadaki sistem ve süreçlerin programlama ile oluşturulan yapay ortama taşınması olarak da tanımlanan simülasyonlar, bilgisayar yazılımları sayesinde bire bir gerçeklikte herhangi bir olayı taklit edebilmektedir.

Endüstride kullanılan teknik ekipmanlardan uçaklara kadar birçok üretim maliyeti yüksek ve kaza anında büyük riskler oluşturma ihtimali kuvvetli olan cihazlar, bilgisayar ortamında oluşturulan simülasyonlar sayesinde insanların herhangi bir risk almadan kullanmayı öğrenebildiği ekipmanlar haline dönüşmüştür. Bilgisayarlı simülasyon kullanarak bir pilot gerçek hayatta bir uçağa binmese dahi, tıpkı gerçek hayatta olduğu gibi bir uçağı nasıl uçurması gerektiğini öğrenebilir.

Şüphesiz simülasyon ortamında uçağı düşürse dahi sistemin yeniden başlatılmasıyla tekrar deneme imkanı olan pilot adayının gerçek hayatta böyle bir imkanı yoktur ve bu durumda gerçek deneyimlerin insan üzerinde oluşturduğu stres etkisinin simülasyon ile tamamen ortadan kaldırılmasını sağlamaktadır. Stres ortamının tamamen ortadan kaldırılması ise kişinin daha verimli bir öğrenme süreci geçirmesi manasına geldiğinden simülasyon kullanımı dünya genelinde kabul görmüş bir uygulama haline gelmiştir.

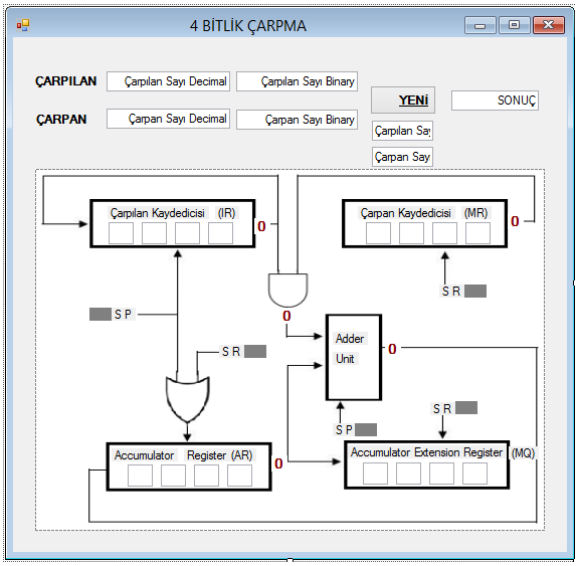
Gerçek olayların bilgisayar yazılımları ile desteklenen modellemeler sayesinde bire bir taklit edilebilmesi, insanların gerçek hayatta karşılaşma ihtimali çok düşük dahi olsa birçok olayı deneyimlemesi ve bu olayların gerçek hayatta gerçekleşmesi halinde kazandığı deneyimle doğru şekilde davranabilmesini sağlamaktadır. [4]

Şekil 2.12. Uçak Simülasyonu

2.6. PROJENİN GERÇEKLEŞTİRİLMESİ

Çarpma ve toplama devresi simülasyonunun oluşturulacağı form sayfası eklendi. Form sayfasının üzerine gerekli nesneler eklendi. (Çarpma devresi simülasyonunun gerçeklendiği form sayfası Şekil 2.1 de gösterilmiştir.) Ardından textBox'lara yazılan değerler arası dönüşümü gerçekleştirecek kod yazıldı. Aşağıda decimal olarak girilen sayıları binary sayıya dönüştüren kod verilmiştir.

```
static string ToBinary(int number)
        {
            int mod;
            string result = string.Empty;
            while (number > 0)
            {
                mod = number % 2;
                number /= 2;
                result = mod.ToString() + result;
            }
            return result.PadLeft(8, '0');
    }
```



Şekil 2.1. Çarpma Devresi Form Ekranı

Kaydediciler içinde tutulan bitlerin kaydırılması için gerekli kodlar yazıldı. Aşağıda Accumulator Extension Register (MQ)' ın kaydırılma işleminin gerçekleştiği kod bloğu yer almaktadır.

```
void mqsaatdarbesi()
        {
            textBox38.Text = textBox39.Text;
            textBox39.Text = textBox40.Text;
            textBox40.Text = textBox41.Text;
            textBox41.Text = label22.Text;
        }
```

Saat darbesinde kayma işleminin gerçekleşebilmesi için bu kodlar uygun yerlerde çağrılmıştır.

Çarpan ve çarpılan sayının tutulduğu kaydedicilerden gelen bitlerin çarpılması işleminin gerçekleştirildiği AND kapısı kodu yazıldı. Bu kod aşağıda verilmiştir.

```
  public int and(int x, int y)
        {
            int degerler = 1;
            int z;
                if (x == 1 & y == 1)
                {
                    z = 1;
                }
                else
                {
                    z = 0;
```

```
                }
                degerler= z;

            return degerler;
        }
```

Adder unit için eldeli toplama işlemini gerçekleştirecek kod yazıldı. Aşağıda bu kodun bir kısmı verilmiştir.

```
public int[] topla(int x, int y, int t)
        {
            int[] degerler = new int[2];
            int z;
            int tson;

            if (t == 0)
            {
                if (x == 0 & y == 1)
                {
                    z = 1;
                    tson = 0;
                }
                else if (x == 1 & y == 0)
                {
                    z = 1;
                    tson = 0;
                }
                else if (x == 1 & y == 1)
                {
                    z = 0;
                    tson = 1;
                }
                else
                {
                    z = 0;
                    tson = 0;
                }
            }
```

SR ve SP saat darbeleri için, her darbede ses çıkması sağlayacak kodlar yazıldı. Bu kodların kullanılabilmesi için media kütüphanesi (Media Kütüphanesi ekleme kodu: `using System.Media;`) eklendi. Yazılan kod saat darbelerinde referanslandı. Bu kodlardan "ting" isimli kod aşağıda verilmiştir.

```
      void ting()
        {
            SoundPlayer player = new SoundPlayer();
            player.SoundLocation = @"C:\Users\REYHAN\Documents\Visual Studio
2013\Projects\4\WindowsFormsApplication6\tiiiing-2013.wav";
            player.Play();
     }
```

Ayrıca saat darbelerini temsilen oluşturulan labellerın saat periyotlarında renk değiştirmesini sağlayacak kod yazılmıştır. Bu şekilde gelen saat darbelerinin gözlemci tarafından daha rahat izlenmesi hedeflenmiştir. Kullanılan kodun bir kısmı aşağıda verilmiştir.

```
      void darbe ()
        {
            label27.BackColor = Color.Red;
            label28.BackColor = Color.Blue;
            label29.BackColor = Color.Blue;
            label30.BackColor = Color.Red;
            label31.BackColor = Color.Blue;
    }
```

Kayan bitlerin hareketinin gözle rahatça gözlemlenebilmesi timer kullanımı ile sağlanmıştır. Labelın hareketinin koordinatları hesaplanarak ilgili timera yazıldı. Timerın interval özelliği ile kodun koşma süresi uygun değerlere setlenerek senkronizasyon sağlanmıştır. Örnek olarak aşağıda çarpılan sayı kaydedicisindeki son biti AND kapısına götüren timer kodu verilmiştir.

```
private void timer6_Tick(object sender, EventArgs e)
        {

            if (label19.Top == 180)
            {
                label19.Left += 1;
            }
            if (label19.Left == 255)
            {
                label19.Top += 1;
            }
            if (label19.Top == 222)
            {
                label19.Location = new Point(240, 180);
                timer6.Enabled = false;
            }
     }
```

Son olarak nesnelerin events larına gerekli kodlar yazılarak uygulamaya girilebilecek olası yanlış girişlerin engellenmesi hedeflenmiştir. Aşağıdaki kodda binary değerlerin alınacağı textboxa “1” ve “0” haricinde karakter yazılması engellenmiştir.

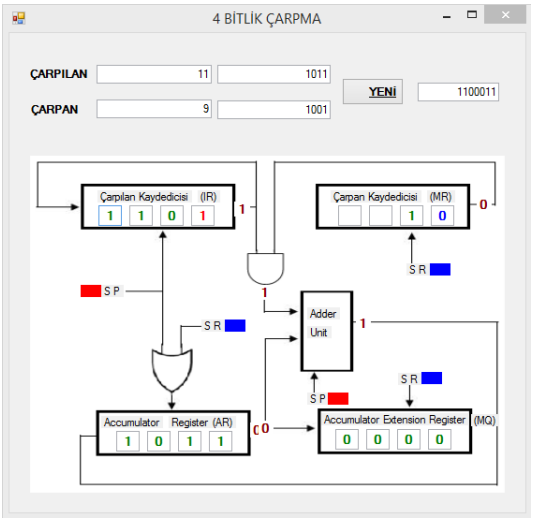
```
    private void textBox3_KeyPress(object sender, KeyPressEventArgs e)
        {
            if ((int)e.KeyChar >= 48 && (int)e.KeyChar <= 49)
            {
                e.Handled = false;//0 ya da 1 ise yazdır.
            }
            else if ((int)e.KeyChar == 8)
            {
                e.Handled = false;//eğer basılan tuş backspace ise yazdır.
            }
            else
            {
                e.Handled = true;//bunların dışında ise yazdırma!!!
            }
        }
```

Böylece hedeflenen simülasyon gerçeklenmiştir.

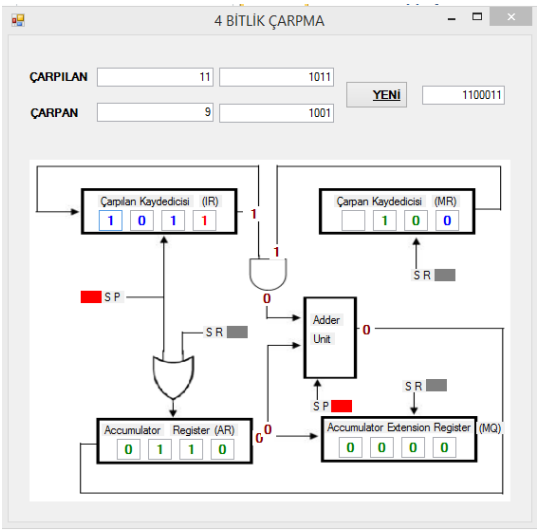
3. SONUÇLAR

Bu tasarım projesinde ALU'nun temel işlemlerinden olan toplama ve çarpma işlemlerinin simülasyonu gerçekleştirilmiştir. Saat darbelerinin daha iyi anlaşılabilmesi için ses ve görüntü efektleri kullanılmıştır. Projenin en önemli özelliği ise dışarıdan girilen 4 bitlik her sayının simülasyonunu gerçekleştirebilmesidir. Bu sayede kullanıcı etkileşimi sağlanmıştır.

Ayrıca mühendis adayları için toplama ve çarpma işlemlerinin adım adım nasıl gerçekleştiği de gösterilmiştir.



Şekil 3.1. Simülasyon 1



Şekil 3.2. Simülasyon 2

4. ÖNERİLER

Gerçekleştirilen projenin daha kapsamlı bir hale getirilebilmesi için çıkarma ve bölme işlemi simülasyonu da projeye eklenebilir.

5. KAYNAKLAR:

[1] http://www.chip.com.tr/forum/visual-studio-2010-nedir_t223469.html
[2] http://www.yigitkiran.net/blog/2010/02/c-dersleri-7-windows-forms-uygulamalari/
[3] http://www.teknokoliker.com/2011/11/c-nedir-c-temelleri-nelerdir.html
[4] http://simulasyon.nedir.com/
[5] sct.emu.edu.tr/courses/eet/elet313/userfiles/files/**CHAPTER**%203.doc