

T.C.

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ



PC TABANLI SAYISAL ENTEGRE DEVRE
EMÜLATÖRÜ TASARIMI

Yalçın ATA

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

98287

TEZ DANIŞMANI

Doç.Dr. E. Murat ESİN

GEBZE

2000

TEZ YÜKSEK LİSANS ENSTİTÜSÜ
DOKÜMANLAMA MERKEZİ

Yalçın ATA'nın tez çalışması, G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 12.09.2000 tarih ve 2000/120 sayılı kararıyla oluşturulan jüri tarafından 07.12.2000 tarihinde Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE
(Tez Danışmanı)

: Doç. E. Murat ESİN

Murat Esin

ÜYE

: Yrd. Doç. İbrahim DOĞUK PINAR

Ibrahim Doguk Pinar

ÜYE

: Yrd. Doç. Ertuğrul KARAGÜNA

Ertugrul Karaguna

ONAY

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 07.12.2000 tarih ve 2000/129 sayılı kararı.



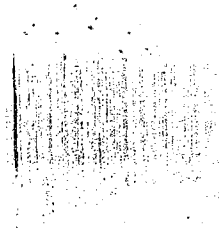


ÖZET

Bu çalışmada lojik entegre devrelerinin emülasyonunu sağlayacak PC tabanlı bir sistem tasarlanmıştır. Donanım ve Yazılımdan oluşan Entegre Emülatöründe; yazılım üzerinden herhangi bir sayısal entegrenin tanım bağıntısı ve giriş-çıkış ilişkisi tanımlanır. Donanım kısmı yazılım ile kontrol edilerek tanımlanan entegrenin görevini sorunsuz yerine getirir.

Bu sistem sayesinde istenilen durumlarda amaca uygun lojik işlemleri yapabilecek özel entegreler tanımlanabileceği gibi; piyasada bulunan fakat maliyeti yüksek entegreler de yazılım ile tanımlanarak kullanılabilir. Geliştirme aşamasında entegre fonksiyonlarını test etmek, söküp takma işlemleri gerektirdiği için tasarımcı açısından zaman ve bilgi kaybına neden olmaktadır. Emülatör kullanımında bu zorunluluk da ortadan kalkmakta ve zaman açısından büyük tasarruf sağlanmaktadır.

Tasarlanan sistem, en fazla 16 bacaklı bir dijital entegreyi emüle edebilmekte ve istenen performansı sağlamaktadır.



SUMMARY

As a result of rapid development in digital and microprocess circuit applications, expected features of needed integrated circuits have increased. Integrated circuits containing many logic processes are needed in design stages for different application at different times. It is difficult to sell ICs which are needed for all this applications. On the other hand, customized orders require huge costs.

This study aims to design a PC-based system to provide emulation for integrated logic circuits. Thanks to the system detailed here it will be possible to define custom integrates with the ability to perform certain desired processes; what is more, you can choose to define the commercially-available high-cost integrates with the aid of software. Due to the fact that it requires mounting and dismounting, testing integrates for their functions in the process of developing causes their designer to lose time and data. The use of emulators eliminates such requirement, providing great savings on time.

Emulator, consisting of hardware and software, carries out the function of any system with no trouble by controlling hardware and software.

TEŞEKKÜR

Bu tez çalışması süresince bana sürekli destek olan, bana güvenen ve beni sürekli yönlendiren, tecrübelerinden faydalandığım hocam Sayın Doç. E.Murat ESİN'e,

Desteklerini esirgemeyen mühendis arkadaşlarım Sn. Hayrettin BOZAN, Gökhan YÜRÜDOĞRU, İlker KÖSE, Ender DİVRİKLİOĞLU'na

Çalışmalarımı yürütürken bana sağladıkları imkan ve huzurlu ortam için Gebze Yüksek Teknoloji Enstitüsü'ne, İstanbul Ulaştırma A.Ş.'ye ve tüm çalışma arkadaşlarıma,

Öğrenim hayatım boyunca bana maddi ve manevi her konuda verdikleri desteklerinden, benim için yaptıkları fedakarlıktan dolayı aileme teşekkürlerimi borç bilirim.

Gebze 2000

Yalçın ATA

İÇİNDEKİLER DİZİNİ

VII

SAYFA

ÖZET	IV
SUMMARY	V
TEŞEKKÜR	VI
İÇİNDEKİLER DİZİNİ	VII
KISALTMALAR DİZİNİ	VIII
ŞEKİLLER DİZİNİ	IX
ÇİZELGELER DİZİNİ	X
1. GİRİŞ	1
2. EMÜLASYON UYGULAMALARI	6
3. SİSTEMDEN BEKLENTİLER	9
4. DONANIM	12
4.1 Sistemin Çalışma Prensibi	12
4.2 Kullanılan Elemanlar	16
4.3 Adres Çözücü Devre	17
4.3.1 Entegre Giriş Çıkışlarının Belirlenmesi İçin Kullanılan Adres Çözücü Devre	19
4.3.2 Okuma İşlemi İçin Kullanılan Adres Çözücü Devre	20
4.3.3 Yazma İşlemi İçin Kullanılan Adres Çözücü Devre	21
5. YAZILIM	22
6. UYGULAMA ÖRNEKLERİ	26
7. SONUÇ	28
KAYNAKLAR DİZİNİ	29
ÖZGEÇMİŞ	30
EKLER	31
EK-1 : Malzeme Yerleşimi	
EK-2 : Baskı Devre Üst yüzeyi	
EK-3 : Baskı Devre Alt Tüzeyi	
EK-4 : Devre Şeması	
EK-5 : Sistemde kullanılan yazılım	

SİMGELER ve KISALTMALAR DİZİNİ

ASIC	Application-Specific Integrated Circuits
CLB	Configurable Logic Block
CPU	Central Processing Unit
DOS	Disc Operating System
EISA	Enhanced Industry Standard Architecture
FPGA	Field-Programmable Gate Array
FPIC	Field-Programmable Integrated Circuits
HDL	Hardware Descriptive Languages
IBM	Intelligent Business Machine
ISA	Industry Standard Architecture
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MCI	Media Control Interface
PC	Personal Computer
PCI	Peripheral Component Interconnect Local Bus
RTL	Register Transfer Level
VLIW	Very Large Instruction Word
VLSI	Very Large Integration Systems



ŞEKİLLER DİZİNİ

Şekil	Sayfa
1.1 Tasarımların Test Edilme Yöntemleri	1
1.2 VEYA Kapısının Elektriksel Eşdeğeri	2
1.3 VEYA kapısının analitik ifadesi ve doğruluk tablosu	3
1.4 VEYA Kapısının Emüle Edilmesi	3
1.5 Herhangi Bir Sistemde Sıcaklık Kontrolü	3
1.6 Sıcaklık Kontrol Sisteminde Emülatör kullanımı	4
1.7 Altsistemlere Ayrılan Karmaşık Bir Sistemin Modeli	4
3.1 Tasarlanan Kartın Entegre Soketine Bağlantısının Blok Gösterimi	10
4.1 Emüle edilecek entegrenin giriş ve çıkışlarının belirlenmesi için kullanılan devrenin genel görünümü	12
4.2 Entegrenin girişlerinin okunması için kullanılan devrenin genel görünümü	13
4.3 Entegrenin çıkışlarının karta aktarılması için kullanılan devrenin genel görünümü	14
4.4 Sistemin Blok Şeması	15
4.5 Adres Çözücü Devrenin Bağlantısının Blok Gösterimi	16
4.6 Entegre Giriş Ve Çıkışlarının Belirlenmesi İçin Kullanılan Adres Çözücü Devre	19
4.7 Okuma İşlemi İçin Kullanılan Adres Çözücü Devre	20
4.8 Yazma İşlemi İçin Kullanılan Adres Çözücü Devre	21
5.1 Sistem yazılımının akış şeması	23
6.1 7408 Entegresinin Yapısı	26



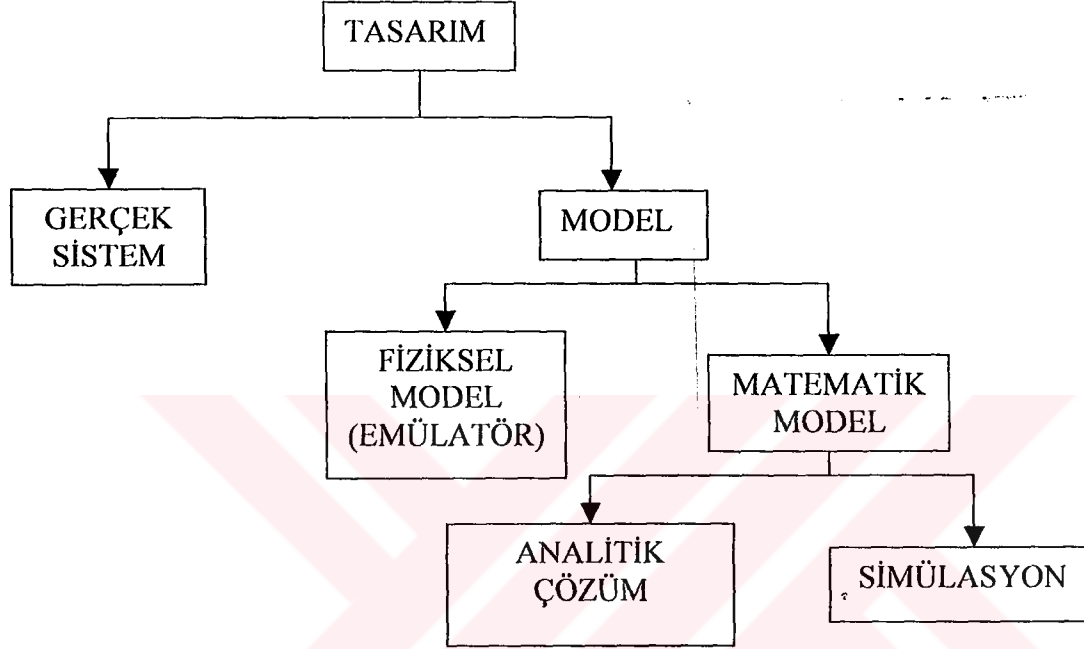
ÇİZELGELER DİZİNİ

Çizelge	Sayfa
6.1 7408 Entegresinin Tanımı İçin Girilen Değerler	26
6.2 7408 Entegresinin Testi Esnasında Okunan Ve Çıkışa Aktarılan Değerler	27



1.GİRİŞ

Günümüzde tasarımların karmaşıklığının sürekli artması, spesifik uygulamalı tümleşik devrelerin gerçekleştirilmesinde fonksiyonel testin yapılması sonucunu doğurdu. Yaygın olarak iki test tekniği kullanılmaktadır: Emülasyon ve Simülasyon. Tasarımlar gerçekleştirilirken birkaç yöntemle test edilebilir.



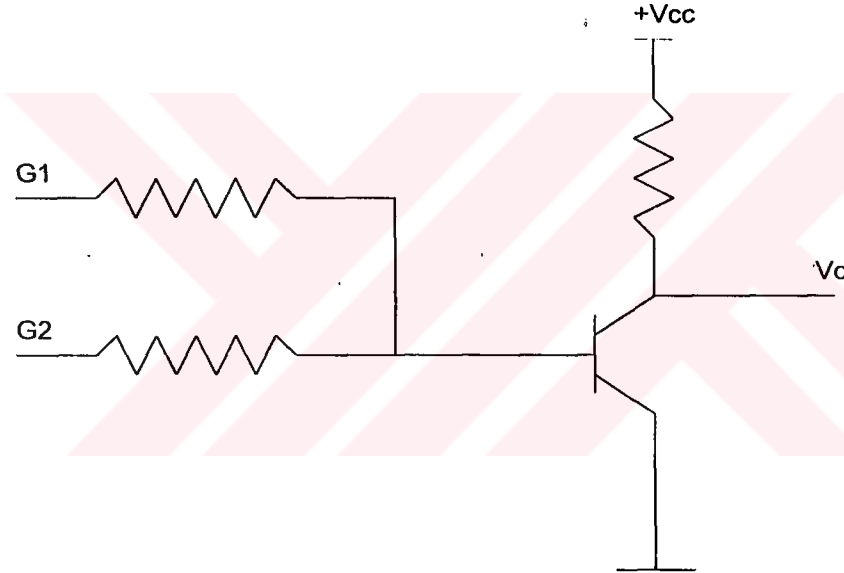
Şekil 1.1: Tasarım Yöntemleri

Gerçek sistemin kendisi oluşturularak test edilebilir. Ancak sistem karmaşık bir yapıya sahip, giriş/çıkış sayısı fazla ise sistemin modeli oluşturularak test edilebilir. Model oluşturularak test etmenin iki yolu vardır. Birincisinde; sisteme benzer ancak sistemden daha az karmaşık yapıya sahip bir fiziksel model oluşturulur ve test edilir. Bu fiziksel model emülatördür. Sistemin veya bir parçasının yerine kullanılır. İkincisinde ise; sistemin matematiksel modeli oluşturulur. Sistem ya analitik olarak çözümlenir, ya da simülasyon ile davranışı gözlenir. Simülasyonda sistemin girişi ile çıkışı arasındaki ilişki tanımlanarak, sistem gibi kullanılır ve sistem test edilir.

Emülatör, bir cihaz ya da sistemin yerine aynı görevi yerine getirebilecek şekilde kullanılabilen yazılım ve donanımların genel adıdır. Simülatör ise bir cihaz ya da sistemin davranışının test edilmesini sağlayan cihazdır. Diğer bir deyişle,

incelenen sistemdeki gerç k b y kl kler gibi, aynı tanım bağıntılarına uyan ancak daha ucuz ve kullanımı kolay bileşenler kullanılarak problemleri ç zen cihazdır. Em lat r ile sim lat r arasındaki temel fark şudur: em lat r bir sistemin yerine kullanılır, sim lat r ise sistemin modellenerek davranışını g zlemlemeyi saėlar. Yani sim lat r sistemin davranışını izah eder, em lat r sistem gibi davranır.  rneėin bir u aėın motorlarının arızalanması halinde nasıl davranacağına g zlenmesi iřlemi u ak hareket halinde iken yapılamayacağı i in, ancak sim lasyon ile yapılabilir. Ya da bir n kleer reakt r n bir kit'inin arızası halinde nasıl davranacağına izlenmesi sim lasyon ile yapılabilir. Fakat bir radarın anteninin arızalanması durumunda çalışmasının nasıl olacağı em lat r yardımıyla g zlenebilir.

řekil 1.2'de lojik VEYA kapısının elektriksel eřdeėeri g r lmektedir.



řekil 1.2: VEYA Kapısının Elektriksel Eřdeėeri

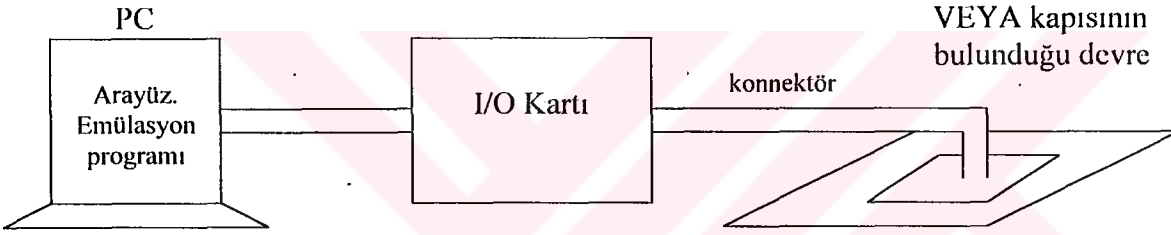
VEYA kapısının analitik ifadesi $S = G1 + G2$ şeklindedir. Bu analitik ifadenin elde edilmesi ve doėruluk tablosunun çıkarılması, buna baėlı olarak sistemin davranışının ifade edilmesi ile sistem sim le edilir (řekil 1.3).

G1	G2	S
0	0	1
0	1	0
1	0	0
1	1	0

$$S = G1 + G2$$

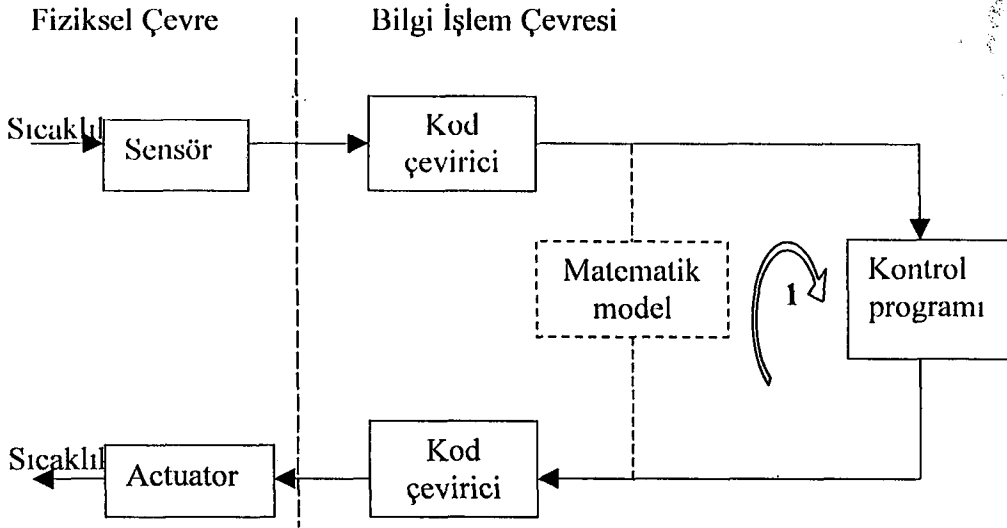
Şekil 1.3: VEYA Kapısının Analitik İfadesi Ve Doğruluk Tablosu

VEYA kapısının yerine bir soket takılıp, bilgisayarda tanımı yapılır ve bir I/O kartı üzerinden girişler okunup elde dilen değreler çıkışa aktarılırsa VEYA kapısı emüle edilmiş olur (Şekil 1.4).

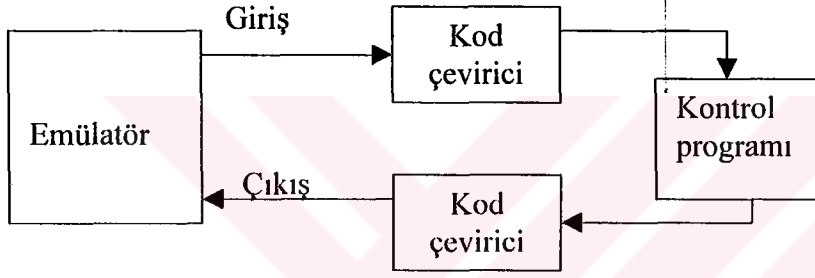


Şekil 1.4: VEYA Kapısının Emüle Edilmesi

Fiziksel ve Bilgi işlem çevresine sahip, sıcaklık kontrolü yapan herhangi bir sistemde (Şekil 1.5), (1) numaralı çevrimin matematiksel modeli çıkartılarak sistemin davranışı incelenirse simülasyon yapılır. Bu sistemde fiziksel çevrenin tümü çıkarılarak yerine emülatör kullanılabilir (Şekil 1.6).



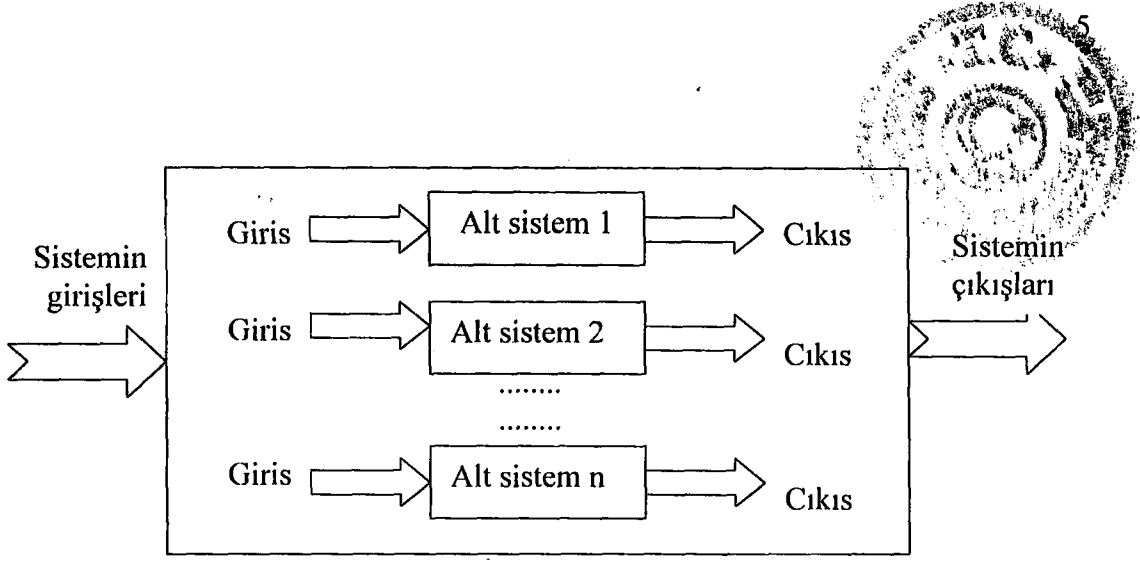
Şekil 1.5: Herhangi Bir Sistemde Sıcaklık Kontrolü



Şekil 1.6: Sıcaklık Kontrol Sisteminde Emülatör Kullanımı

Karmaşık sistemler tasarlanırken alt sistemlere (subsystem) ayrılırlar (Şekil 1.7). Bu sistemler test edilirken her bir alt sistem test edilir. Her bir alt sistemin doğru çalışması neticesinde bütün sistem doğru çalışır. Bir sistemin tasarımı gerçekleştirilirken, sistemin doğruluğunun test edilmesi, beklentilere uygunluğu her alt sistemin yerine emülatör kullanılarak yapılır. Bu şekilde üretime geçmeden önce tasarımın doğrulanması (verification) işlemi yapılır.

Lojik emülasyon sistemi, gerçekleştirilen tasarımın doğrulanması ve prototip oluşturulması için dinamik olarak yeniden programlanabilir sistemleri kullanan bir tekniktir. Tasarımcılar Emülatör kullanarak, yazılım konfigürasyon işlemi ile tasarımları gerçekleştirebilirler ve silisyum yonga üretiminden önce gerçek zaman uygulaması yapabilirler [Wen-Jon Fang et al, 1999]. Böylelikle yonga üretiminden önce gerçek zamanlı test yapılarak, üretilecek yonganın çalışması gözlenmiş olur.



Şekil 1.7: Alt sistemlere Ayrılan Karmaşık Bir Sistemin Modeli

Elektronik devre veya sistemlerin fiziksel emülasyonu için kullanılan bir sistem, kullanıcının devre veya sistem konfigürasyon' tanımını için giriş verilerini girdiği bir veri giriş istasyonu içerir. Veriler, zengin bir mimariye sahip programlanabilir kapı elemanlarının programlanması ile uygun bir forma dönüştürülür. İstenilen koşullar, kullanıcı devre, sistem parçaları veya VLSI (Very Large Integration Systems) cihazları tarafından üretilir [Maguire and Francis, 1992].

2.EMÜLASYON UYGULAMALARI

Günümüzde emülasyon sistemleri ile uygulamalar yaygın olarak kullanılmaktadır. Tasarımı yapılan bir sistemin donanım ve yazılımının önceden test edilebilmesi, çalışma fonksiyonlarının gözlenebilmesi, test işleminin gerçek zamanlı olması kullanımı daha da yaygınlaştırmaktadır.

Simülasyon yazılımı oldukça yavaştır ve gerçek yongaların eksikliklerini doğru olarak elde etmek zordur. Lojik emülasyon kullanıcılara yonga tasarımında hızlı bir şekilde donanım modeli oluşturmayı sağlar. Bu durumda özel bir emülasyon yazılımı kullanılır. Lojik emülasyon donanım-yazılım koverifikasyonu ve integrasyon sağlanarak elektronik tasarıma dönüştürülür. Tasarımcılar yongaları emüle ederler ve doğrularlar. Kullanıcılar hataları görüp hemen düzeltebilirler. Emülasyon tasarımların süresini azaltır, yeniden üretim masraflarını elimine ederek maliyeti düşürür.

Emülasyon kaçınılmazdır. Emülasyon, elektronik ürünlerin karmaşıklığını artırarak test etmek için bir yöntem sağlar. Son yıllarda entegre devrelerin karmaşıklığı hızlı bir şekilde arttı. Bugün tek bir yongayı yüzlerce, binlerce hatta milyonlarca lojik kapı ile donatmak anormal bir durum değildir.

Tasarımın karmaşıklığı arttıkça, sistem giriş çıkışlarını kontrol etmek zorlaşır. Bu durumda tasarımlar uygun şekilde alt sistemlere ayrılırlar. Bu alt sistemler ayrı ayrı test edilir. Emülatör yardımıyla yüksek performans sağlanır. Sadece lojik emülasyon gerçek sistemlerin yonga tasarımının doğrulanmasına izin verir. Tasarımcılar grafik yongası tarafından üretilen görüntüleri grafik ekranından, telekomünikasyon prosessöründen gelen veri paketlerini monitörden, gerçek uygulama programlarının çalıştığı yongaları izleyebilir.

Programlanabilme özelliklerinden dolayı; FPGA(Field-Programmable Gate Array)'lar hızlı sistem prototipi oluşturmak için en popüler ASIC (Application-Specific Integrated Circuits-Özel Uygulamalı Tümüleşik Devreler)'den biridir. Lojik emülasyon, tasarım oluşturma ve prototip işlemi için dinamik olarak yeniden programlanabilen sistemleri kullanan ilk tekniktir. Lojik emülatör içinde yüzlerce ve

binlerce FPGA veya FPIC(Field-Programmable Integrated Circuits) içeren bir sistemdir. Konfigüre edilen lojik emülatör tasarlanmış bir yongadır ve gerçek zamanlı tasarım, yazılım geliştirilmesi ve silisyum yonganın üretiminden önce prototip'inin oluşturulması için kullanılabilir. Son yıllarda Lojik emülasyon geniş olarak; çoğu kompleks merkezi işlem birimi (CPU), telecom, multimedia ve sistem tasarım projelerinde kullanılan metodik tasarım yardımcısıdır. Modül generatörü kullanılarak tek FPGA yongasına uygun olmayan büyük makrolar için çoklu FPGA makrosu üretilebilir. Tasarımcılar bu modül generatörünü kullanarak, tasarım geliştirme süresi ve fiyatını önemli ölçüde azaltarak kompleks makrolar üretebilmektedir. Elde edilen sonuçlar göstermektedir ki, orta büyüklükte bir endüstriyel tasarımda lojik emülasyon tasarım akışı modül üretim metodu ile birleştirildiğinde tasarım işleme süresi %33 oranında kısalmaktadır. Daha büyük tasarımlarda bu süre uygun olarak daha büyük oranda düşecektir. [Wen-Jon Fang et al, 1999].

Lojik tasarım doğrulamaları, göreneksel emülatörden ziyade bir hardware lojik emülatörü veya acceleratör (hızlandırma makinası) tarafından yerine getirilebilir. Simülasyondan farklı olarak, donanım emülatörü kapı (gate) evalüasyonu yapar. Donanım emülatörü veya acceleratör arıza simülasyonunun hızının artırılmasında da kullanılabilir. Emülatörler gerçek sistemde olmayan arıza senaryolarının oluşturulmasına da yardım ederler. Mevcut sistemin herhangi bir alt sisteminde bir arızanın olması durumunda çalışması gözlenebilmektedir. Arıza emülasyonunda asıl güçlük, arıza enjeksiyonudur. Orijinal bir devre arıza enjekte edebilen bir devreye dönüştürülebilir. Arızaların enjekte edilmesi devrede aynı-arıza (corresponding) enjeksiyon elemanları enable edilerek kolayca yapılabilir. Önceki metottan farklı olarak, arıza enjeksiyonu direkt olarak orijinal devreden yapılabilir; böylece simüle edilecek tüm arızalar arıza enjekte edebilen devrede tutulur [Hwang et al, 1998].

Tasarımın simülasyonu bir tasarımda değişkenlerin kontrol ve gözlenmesini sağlar; fakat üretilen tasarımdan 2 ile 10 kat arasında değişen büyüklükte daha yavaştır. Tasarım emülasyon ve üretimi, yüksek çalışma hızı sağlar fakat önemli derecede tasarım kontrol ve gözlem yeteneğini sınırlar. Elektronik devrelerin fiziksel emülasyonu için kullanılan yeniden konfigüre edilebilen modern sistemler, devre konfigürasyonunun sağlanması için kullanıcının veri girişi yapabildiği bir veri giriş

istasyonu içerir. Bu veri programlama için uygun bir forma dönüştürülür[Kirovski et al, 1999].

Büyük ardışıl devreler için arıza simülasyonu fazla zaman harcayan bir işlemdir. Artan FPGA performansı ve lojik emülasyon teknolojisi ile bir donanım arıza emülasyon sistemi sadece uygulanabilir olduğu için değil ayrıca var olan yazılım veya donanım temelli yaklaşımların karşılaştırılması gibi işlemleri de kapsar. Arıza sınıflandırılması için en önemli metot lojik emülasyon sistemi kullanmaktır. Emülasyon cihazı test vektörlerinin uygulanması, çıkış cevaplarının toplanması, önceden belirlenen doğru çıkış ile elde edilen çıkış arasındaki herhangi bir hatanın kontrolü işlemini yapar. Lojik emülatör uygun tasarım halinde 3.000.000 kapılı lojik tasarımın görevini yerine getirebilir ve 100K'dan birkaç milyona kadar test vektörünü emüle edebilir ki bu bilinen yazılım simülatörlerinden 10.000-1.000.000 kez daha hızlıdır [Cheng et al, 1999].

Emülasyon mekanizmalı bir bilgisayar sistemi, bir uygulama programını çalıştırmak için program çalıştırma birimi içerir. Uygulama programı, yazı komutu içerir. Emülasyon kontrol birimi, yazı komut cevaplarının bulunduğu buffer'larda, yazı komutları ile ilgili veri ve adresleri yazar. Önceden belirlenen koşullar olduğunda, Emülasyon kontrol birimi bir interrupt üretir ve program çalıştırma birimine gönderir. Interrupt cevabında, program çalıştırma birimi emülasyon programını çalıştırır ve emülasyon programının çalışması sırasında buffer'larda tutulan adres ve verileri emüle eder [Kishi, 1992].

Günümüzde yaygın olarak kullanılan mikroişlemci uygulamalı devrelerin testi emülasyon ile yapılabilmektedir. Devre içi emülasyon, donanım ve yazılım testinin bir başka yoludur. Temel düşünce şudur: Mikroprosessor sisteminin CPU(Central Processing Unit)'sunun yerine, CPU'yu komple emüle edebilecek bir cihaz yerleştirilir. Bu cihaz ayrıca program çalıştırıldığında ne olduğunu gözlemlemek için kullanılan bir "Kullanıcı Arayüzü" ne sahiptir. Emülatör donanımı gerçek CPU'nun yerine programı çalıştırır [Karsai And Driscoll, 1998].

3.SİSTEMDEN BEKLENTİLER

Bu tasarımın amacı var olan veya çalışması öngörülen herhangi bir entegre devreyi hatta daha da geliştirilerek daha büyük sistemleri emüle etmektir. Bu çalışmada bacak bağlantı şeması ve bacaklar arasındaki mantıksal ilişkileri bilinen ya da öngörülen bir entegre devre, kullanılacağı gerçek devreye bilgisayarla emüle edilecektir. Emüle edilecek entegre veya sistem en fazla 16 giriş ve çıkış'a sahip olacaktır. Tasarlanan kart herhangi bir PC'nin 16 bit'lik ISA yuvası üzerinden paralel olarak yazılım ile haberleştirilecektir. Bu sistemin istenilen şekilde çalışması, daha sonra giriş/çıkış sayısı fazla, daha karmaşık sistemlerin emülasyonuna da imkan sağlayacaktır. Sistemden beklentiler ikiye ayrılabilir:

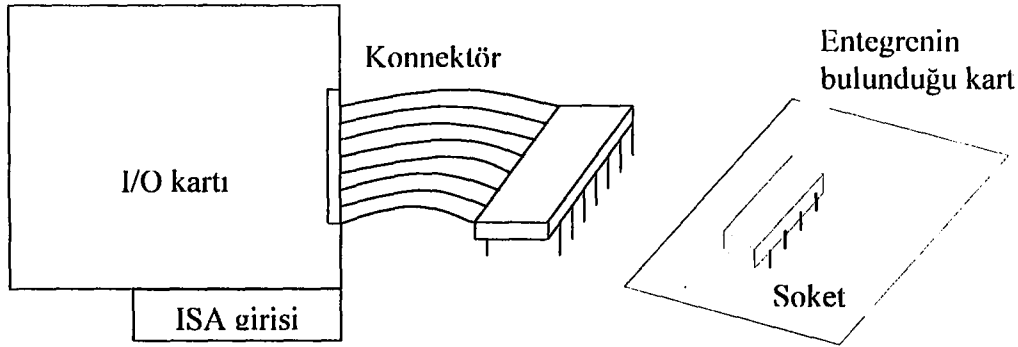
-Donanım : Standartlara uygun elemanlar kullanılarak, verilerin maksimum doğrulukta iletilebileceği, sistem kayıplarının minimum olduğu, yazılım ile kontrol edilebilecek bir donanım amaçlanmaktadır. Yazılım ile tanımlanan entegre giriş-çıkış ilişkileri donanım kısmına iletilir. Emüle edilecek entegrenin bulunduğu karta, donanım kısmından konnektör ile alınan soket takılır. Lojik işlemler gerçekleştirildikten sonra elde edilen çıkışlar, yine bu donanım üzerinden karta aktarılır.

-Yazılım : Emülasyon işlemi için, bütün dijital entegre ve sistemlerde çalışma fonksiyonlarının tanımlanabileceği esnekliğe ve donanım ile haberleşmenin en iyi şekilde sağlandığı ortama sahip bir program istenmektedir. Kart üzerinden okunan girişler yazılımda lojik işlemlere tabi tutulur. Yazılım kısmı karmaşık lojik işlemleri gerçekleştirebilecek esneklikte olmalıdır. Elde edilen çıkış değerleri yazılım tarafından karta aktarılır.

Entegre yerine emülatörü kullanıldığında:

1. Bacak bağlantılarına uygun, tanımlanan giriş/çıkışların programlama yoluyla gerçekleştirildiği bir karta bağlanan entegre devre konnektörü bulunur. Bu konnektör emüle edilecek entegrenin soketine uygun olacak şekildedir. I/O kartına yassı kablolar ile bağlanır. Sistemde emüle edilecek entegrenin giriş ve çıkışları emüle edilecek entegrenin bulunduğu karttan konnektör ile alınır. Sistem

girişleri entegrenin bulunduğu karttan okunur. Elde edilen çıkışlar tasarlanan kart üzerinden entegrenin bulunduğu karta aktarılır.



Şekil 3.1 : Tasarlanan Kartın Entegre Soketine Bağlantısının Blök Gösterimi

2. Elde edilen elektriksel sinyaller, entegrenin elektrik sinyalleri ile aynı karakteristiği göstermelidir. Emüle edilecek entegrenin giriş ve çıkışları dijitaldir. Dijital değerler 0-5V arasındadır ve 0V değeri Lojik-0, 5V değeri ise Lojik-1 olarak gösterilmektedir. Tasarlanan sistemde bu değerlerin sağlıklı elde edilmesi için, devrede kullanılan bütün entegrelerin besleme ucu ile toprak ucu arasına oluşabilecek gerilim değişimlerini önlemek için kondansatör bağlandı.
3. Bacaklar arasındaki lojik bağlantıların tanımlanması gerekir. Emüle edilecek entegrenin bacakları arasındaki ilişki yazılım üzerinden tanımlanmaktadır. Entegre bacaklarının giriş ya da çıkış olarak belirlenmesi işlemi lojik-1 ve lojik-0 değerleri ile yapılmaktadır. Girişler lojik-1, çıkışlar lojik-0 ile seçilir. Giriş ve çıkışlar belirlendikten sonra bacaklar arasındaki ilişkiler de yazılım üzerinden tanımlanır. Giriş/çıkış belirleme verileri, emüle edilen entegrenin girişleri ve çıkışları bilgisayarın başka cihazları tarafından kullanılmayan bellek adreslerinde ayrı ayrı tutulur.
4. Emülatör kullanılması durumunda zaman gecikmesi karakteristiği, emüle edilen entegrenin karakteristiğine benzer olmalıdır. Sistemde haberleşmede kullanılan ISA veriyolunun hızı 8 MHz dir. Dijital entegrelerin hızı yaklaşık olarak 2-2,5 MHz civarındadır. Sistemin hızı emüle edilecek herhangi bir entegrenin hızına göre yeterlidir.

5. Yazılım yoluyla emülasyon değişikliğinin yapılması gerekir. Entegre devre kılıf modeli üzerinde bacak giriş/çıkış özelliklerinin ve bacaklar arasındaki lojik bağıntıların tanımlandığı bir emülasyon programı bulunmaktadır. Bu program donanım ile haberleşmenin en iyi şekilde sağlayacak, giriş çıkış tanımlarını yapacak, kart üzerinden okuma işlemini kontrol edecek ve okunan girişlerin lojik işlemlerinden sonra elde edilen çıkışları karta gönderecektir. Emüle edilen entegrenin lojik ilişkileri yazılım üzerinden tanımlanır. Girişler emüle edilen entegrenin bulunduğu karttan okunur. Okunan bu girişler lojik işlemleri tabi tutulduktan sonra elde edilen çıkışlar yazılım tarafından emüle edilen entegrenin bulunduğu karta aktarılır. Emülasyon işlemi yazılım ile sonlandırılabilir. Bir entegrenin emülasyonundan çıkıp diğer bir entegrenin emülasyonuna geçilmesi de yazılım üzerinden yapılabilir.



4.DONANIM

4.1. Sistemin Çalışma Prensibi

Tasarlanan kart temelde iki ayrı sistem ile haberleşir. Bunlardan birincisi emüle edilecek entegrenin bulunduğu donanım kısmı, ikincisi ise emüle edilecek entegrenin tanımlandığı programın bulunduğu PC dir.

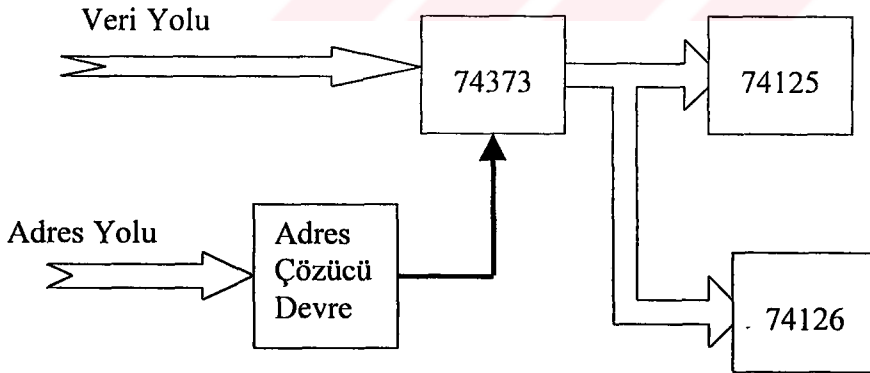
Emüle edilen entegre en fazla 16 bacaklı olacaktır. Tasarlanan kart 16 'lı bir I/O kartıdır.

Emülasyon işleminde; kart üzerindeki konnektöre, yassı kablo ile bağlı soket emüle edilecek entegrenin yerine takılır.

Sistemin çalışması üç aşamada gerçekleşir:

1. Emüle edilecek entegrenin giriş ve çıkışlarının belirlenmesi

PC TARAFI



Şekil 4.1: Emüle Edilecek Entegrenin Giriş Ve Çıkışlarının Belirlenmesi İçin Kullanılan Devrenin Genel Görünümü

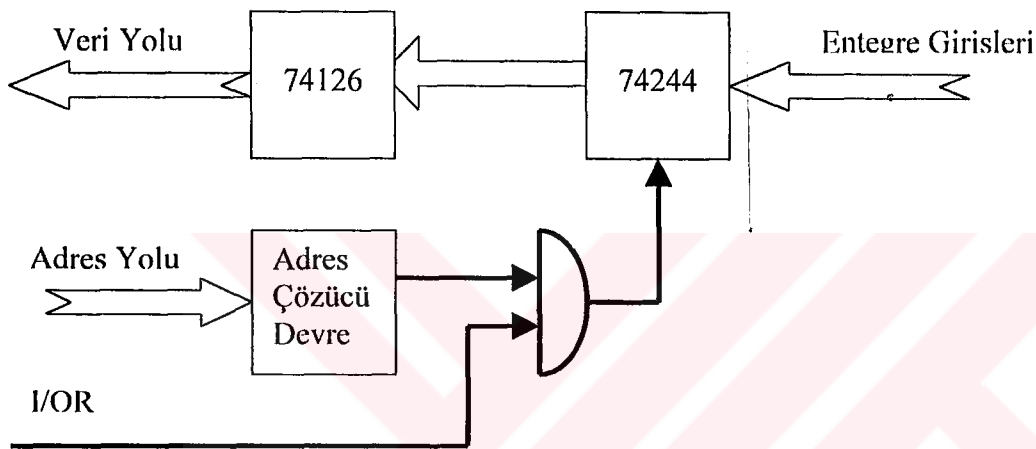
Emüle edilecek entegrenin bacak sayısı program üzerinden girilir. Bacak sayısı girildikten sonra emüle edilecek entegrenin giriş ve çıkışları belirtilir (Giriş/Çıkış belirleme). Sistem girişindeki latchler (74373) ISA üzerindeki adres yolundan gelen

0230h adresi ile enable edilerek emüle edilecek entegrenin hangi bacağına giriş ve çıkış olacağı bilgisi veri yolu üzerinden gönderilir ve bu veriler 0230h adresine kaydedilir. Entegre girişleri lojik-1 değeri ile 74126 entegresi içindeki tri-state buffer'lar enable edilerek, çıkışları ise lojik-0 değeri ile 74125 entegresi içindeki tri-state buffer'lar enable edilerek seçilir (Şekil 4.1).

2. Entegre Girişlerinin Okunması

PC TARAFI

ENTEGRE TARAFI



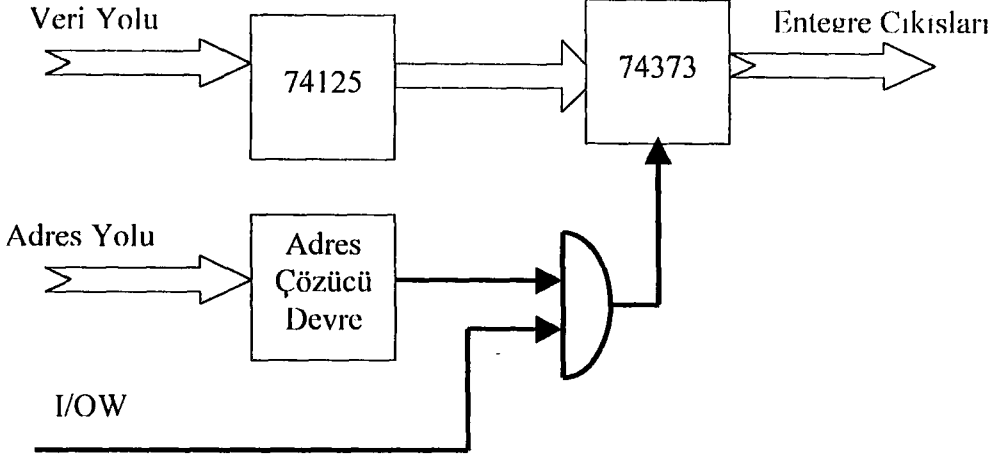
Şekil 4.2: Entegrenin Girişlerinin Okunması İçin Kullanılan Devrenin Genel Görünümü

Entegre giriş ve çıkışlarının belirlenmesinden sonra ISA üzerindeki I/OR ucu kullanılarak ($I/OR=0$) ve ISA üzerindeki adres yolundan gelen 0234h adres bilgisi ile lojik kombinasyona tabi tutularak sistem çıkışındaki 74244 entegreleri enable edilir (Şekil 4.2). Entegre girişleri 74244 ve 74126 entegreleri içindeki tri-state buffer'lar üzerinden veri yoluna aktarılır. Emüle edilecek olan entegrenin girişlerine gelen veriler eşzamanlı olarak okunarak 0234h adresine kaydedilir. Okunan bu veriler emüle edilecek entegrenin tanım bağıntısına uygun olarak programda lojik işleme tabi tutulurlar.

3. Elde edilen çıkışların entegrenin bulunduğu karta aktarılması

PC TARAFI

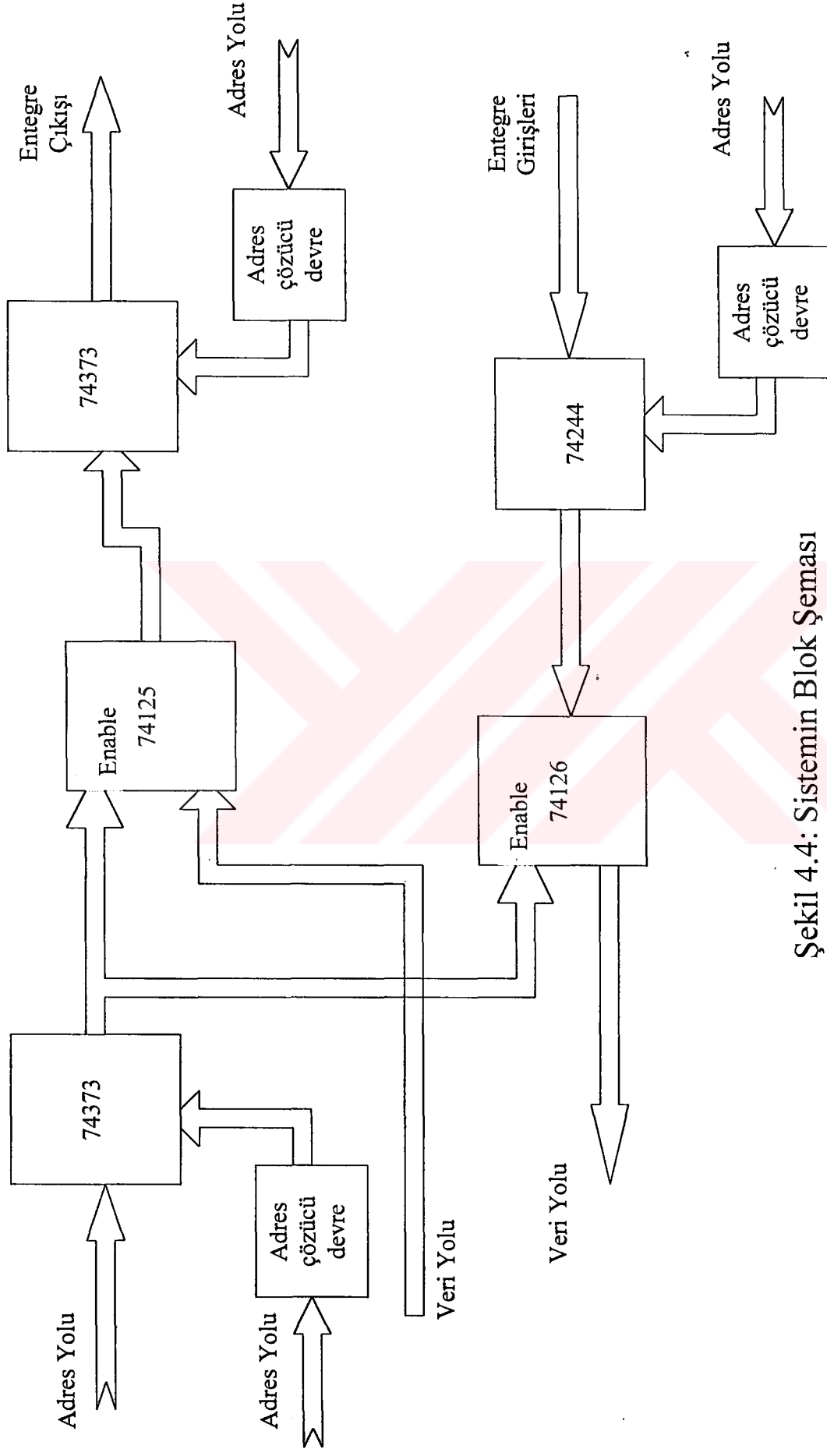
ENTEGRE TARAFI



Şekil 4.3: Entegrenin Çıkışlarının Karta Aktarılması İçin Kullanılan Devrenin Genel Görünümü

Entegre girişlerinin okunması işleminden sonra ISA üzerindeki I/O ucu kullanılarak (I/O=0) ve ISA üzerindeki adres yolundan gelen 0232h adres verileri lojik kombinasyona tabi tutularak sistem çıkışındaki latch'ler (74373) enable edilir. Veriler devrede bulunan 74125 entegresi içindeki tri-state buffer'ları ve devre çıkışındaki 74373 üzerinden emüle edilecek olan entegrenin bulunduğu sisteme aktarılır.

Şekil 4.4'te sistemin tümünün blok şeması verilmiştir. Bu emülasyon işlemi neticesinde emüle edilen entegre çalışır ve o entegrenin bulunduğu sistem de buna bağlı olarak çalışır. Bu işlemin en büyük avantajı piyasada bulunmayan bir entegrenin bile kullanıcının kendi isteği doğrultusunda tanımlanabilmesi, ayrıca piyasada çok fazla bulunmayan veya pahalı entegrelerin de emüle edilebilmesidir.



Şekil 4.4: Sistemin Blok Şeması

4.2. Kullanılan Elemanlar

Kullanılan elemanlar tercih edilirken, emülasyonda seçilen fonksiyonlara göre standartlara en uygun hazır devre ve devre elemanları, yazılımlar ve yazılım geliştirme ortamları araştırıldı.

Donanım ile yazılımın haberleşmesi ISA veriyolları üzerinden sağlandı. Haberleşme COM portları üzerinden seri olarak da sağlanabilir; fakat ISA veriyolu paralel iletim yaptığı için verilerin iletimi daha hızlı olmaktadır. Emülasyon işleminde nanosaniye düzeyinde veri akışı vardır; bu akışın seri haberleşme ile sağlanması paralel haberleşmeye göre dezavantajlıdır. Bu nedenle ISA veriyolu kullanıldı.

Veriyolları sistem elemanları ile bilgisayarın birbirine bağlanmasını sağlayan elemanlardır. Bilgisayar anakartı üzerinde ISA, VESA, PCI, MCI gibi veriyolları bulunmaktadır. Bu uygulamada günümüzde yaygın olarak kullanılan ISA veriyolu kullanıldı. ISA veriyolu IBM AT PC'lerle birlikte çıkmış olup Industry Standart Architecture ifadesinin kısaltmasıdır. 8-bit ya da 16-bit iletim yaparlar. 8-bit ISA kartı, PC'ler için imal edilen ilk kartların standartıdır. Bu kartlar MCI ve PCI yuvaları dışında tüm yuvalara girerler. 16-bit ISA kartları daha yeni olup daha hızlıdır. 4 kısımdan oluşur: A ve B kısmı 31'er slot, C ve D kısmı 18'er slot içerir. 8-bit yuvaya girse bile çalışmayabilir veya 8-bit olarak çalışır. PC üzerinde ISA'ya bağlı olarak geliştirilmiş EISA, VESA yuvaları da vardır.

Bu tasarımda ISA veriyolunun kullanılmasının nedenleri:

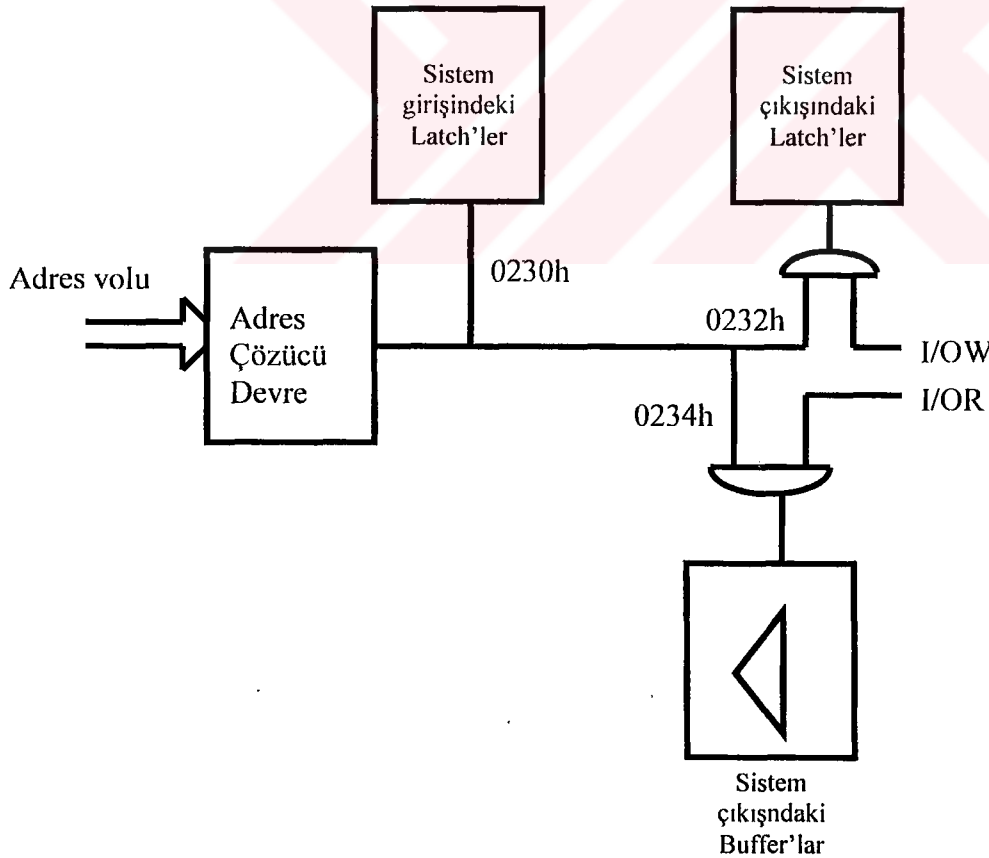
- ✓ Yaygın olarak kullanılması
- ✓ Hemen hemen tüm yuvalara girebilmesi
- ✓ Paralel ve 16-bit iletim yapması
- ✓ Hızının yüksek olması (8Mhz)

Sistemde, emüle edilen entegrenin giriş ve çıkışını belirlemek için gönderilen veriler sistem girişinde bulunan latch'lerde tutulur. Burada latch kullanılmasının nedeni emülasyon işlemi süresince emüle edilen entegrenin giriş ve çıkış bilgilerinin sabit bir şekilde tutulmasını sağlamaktır.

Ayrıca emüle edilen entegrenin tanım bağıntısına uygun olarak elde edilen çıkışlarda entegrenin bulunduğu karta aktarılırken sistem çıkışında bulunan latch'lerde tutulur. Bu çıkış değerleri bir sonraki adımda yeni çıkış değerleri elde edilmeye kadar sabit olarak tutulur.

Sistemde emüle edilen entegrenin girişleri tri-state buffer'lar üzerinden okunur ve elde edilen çıkışlar yine tri-state buffer'lar üzerinden entegrenin bulunduğu karta iletilir. Tri-state buffer'lar bir mantık sisteminde verilerin iletilmesinde kullanılırlar.

4.3 Adres Çözücü Devre



Şekil 4.5 : Adres Çözücü Devrenin Bağlantısının Blok Gösterimi

Sistem tasarımında PC'nin başka cihazlar tarafından kullanılmayan bellek alanı kullanıldı. 3 farklı adres alanı kullanıldı. Bu adres alanları 16 bit'lidir.

Tasarlanan bu sistemde üç farklı adres aralığı kullanılmıştır.

- Giriş/Çıkış belirleme için 0230h adresi
- Yazma işlemi için 0232h adresi
- Okuma işlemi için 0234h adresi

Adres çözücü devresi için AND kapıları kullanıldı. 4'lü And kapısı içeren 3 adet 7421, 3'lü AND kapısı içeren 1 adet 7411, 2'li AND içeren 1 adet 7408, inverter içeren 3 adet 7404 entegresi kullanıldı. "0" olan bitler için inverter devresi içeren 7405 entegreleri kullanıldı.

	A ₁₅	A ₁₄	A ₁₃	A ₁₂	A ₁₁	A ₁₀	A ₉	A ₈	A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀
0230h =	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0
0232h =	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0
0234h =	0	0	0	0	0	0	1	0	0	0	1	1	0	1	0	0

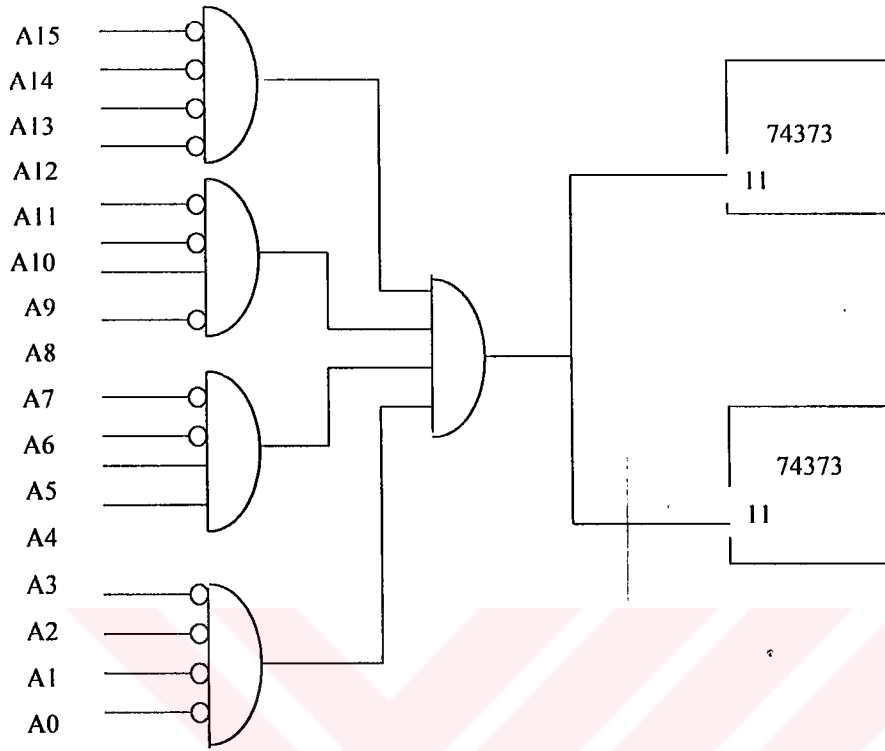
Emülatör'ün çalışması için önce emüle edilecek entegrenin hangi bacağına giriş hangisinin de çıkış olduğunu belirten veriler (1 ise giriş, 0 ise çıkış), veri yolu üzerinden program ile gönderilir. Bu veriler 0230h adresine yazılır.

Daha sonra emüle edilen entegrenin bulunduğu devre üzerinden entegrenin girişleri okunur. Okunan bu girişler 0234h adresine eşzamanlı olarak kaydedilir.

Programda tanımlamaya göre bu girişler lojik işleme tabi tutularak emüle edilen entegrenin bulunduğu devreye entegre çıkışı olarak aktarılır. Bu veriler 0232h adresine kaydedilir.

4.3.1 Entegre Giriş Çıkışlarının Belirlenmesi İçin Kullanılan Adres

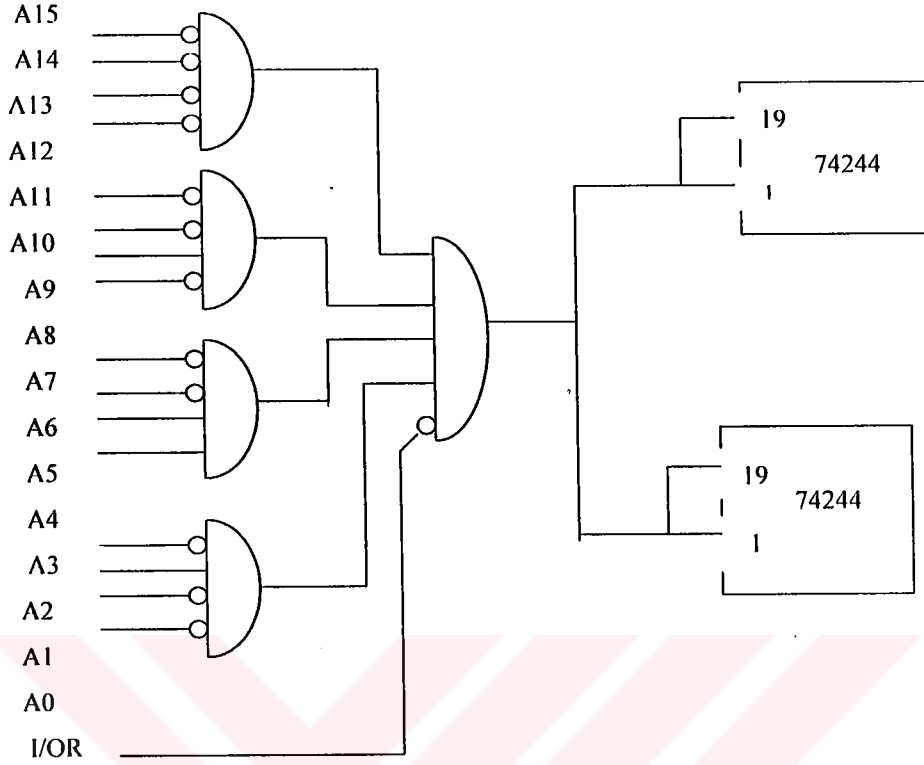
Çözücü Devre



Şekil 4.6 : Entegre Giriş Ve Çıkışlarının Belirlenmesi İçin Kullanılan Adres Çözücü Devre

Adres yolu üzerinden 0x0230 adres bilgisi gelirse sistem girişinde bulunan latch'lerin 11 numaralı Enable Latch ucu aktif olur ve Latch'ler enable olur. Daha sonra Veri yolu üzerinden Latch'lerin girişlerine 1 ve 0 gönderilerek emüle edilecek entegrenin hangi bacağına giriş, hangisinin çıkış olduğu belirtilir. 1 gönderilirse 74126 entegreleri içindeki tri-state buffer'lar enable olur ve pin giriş olarak seçilir. 0 seçilen pin'lerde ise 74125 entegresi içindeki tri-state buffer'lar enable olur ve çıkış olarak seçilmiş olur.

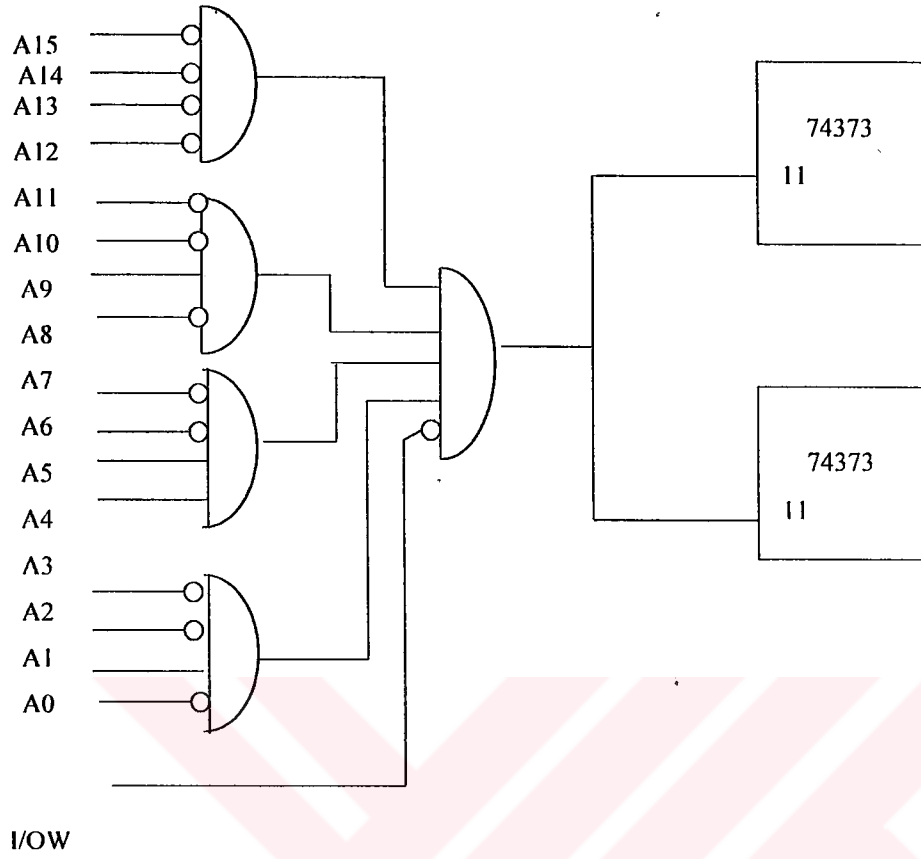
4.3.2 Okuma İşlemi İçin Kullanılan Adres Çözücü Devre



Şekil 4.7 : Okuma İşlemi İçin Kullanılan Adres Çözücü Devre

Adres yolu üzerinden 0x0234 bilgisi gönderilir ve I/OR komutu ile lojik işleme tabi tutulurak, sistem çıkışında bulunan 74244 tri-state buffer'ları enable edilerek emüle edilecek entegrenin girişleri entegrenin bulunduğu kart üzerinden okunur. Input olarak seçilen bit'ler 74244 entegreleri üzerinden 74126 entegrelerine geçer, oradan da veri yolu üzerinden okunur. Okunan veriler daha sonra yazılım üzerinde, emüle edilen entegrenin tanım bağıntısına göre lojik işlemlere tabi tutularak çıkışlar elde edilir.

4.3.3 Yazma İşlemi İçin Kullanılan Adres Çözücü Devre



Şekil 4.8 : Yazma İşlemi İçin Kullanılan Adres Çözücü Devre

Emüle edilecek entegrenin bacak giriş/çıkış tanımı yapıp, okuma işlemi yapıldıktan sonra okunan giriş ve çıkışlar entegrenin tanım bağıntısına uygun olarak lojik işleme tabi tutulurlar. Elde edilen değerlerin entegrenin bulunduğu karta yazılması için Adres yolu üzerinden 0x0232 adres bilgisi gönderilir. I/OW komutu ile birlikte lojik kombinasyona tabi tutulur. Bu durumda sistem çıkışında bulunan Latch'lerin 11 numaralı Enable Latch ucu aktif olur. Latch'lerin girişine gelen veriler çıkışa aktarılır. Çıkış olarak seçilen bit'ler 74373 üzerinden 74125 entegrelerine geçerek çıkışa aktarılırlar.

5.YAZILIM

Sistem; sistem verilerinin girildiği, tanımlamaların yapıldığı bir arayüze sahiptir. Elektronik devre veya sistemlerin fiziksel emülasyonu için kullanılan bir sistem, kullanıcının devre veya sistem konfigürasyon tanımı için giriş verilerini girdiği bir veri giriş istasyonu içerir. Veriler, zengin bir mimariye sahip programlanabilir kapı elemanlarının programlanması ile uygun bir forma dönüştürülür [Maguire and Francis, 1992].

Sistem yazılımında istenilen özellikler :

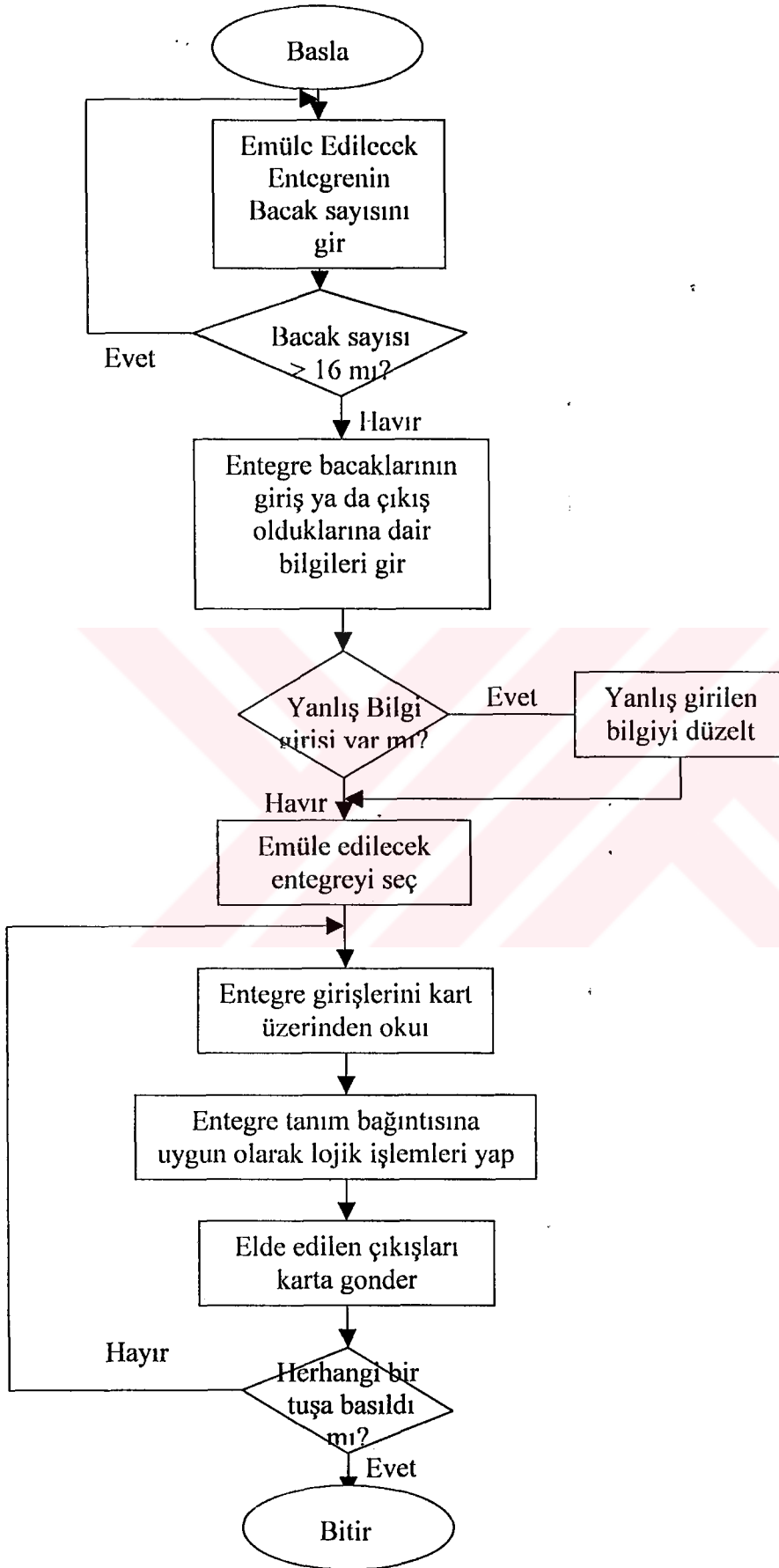
1. Yazılımın donanım ile haberleşmesinin mümkün olduğunca kolay olması ve doğrudan erişimi sağlaması
2. Donanımdan okuma ve donanıma veri gönderme işlemlerinin doğrudan ve kolay yapılması
3. Yazılımda kullanılan programlama dilinin karmaşık lojik işlemleri gerçekleştirebilecek operatörlere sahip olması
4. Programlama dilinin günümüzde yaygın olarak kullanılması

Kullanılan yazılım DOS altında çalışan Turbo C programlama dilidir. C programlama dili günümüzde en yaygın kullanılan dillerden biridir. Donanım ile haberleşme açısından programcıya büyük kolaylıklar sağlamaktadır.

Sistemin Yazılım kısmında ;

- İçinden herhangi bir Entegrenin seçilebileceği bir entegre kütüphanesi oluşturuldu.
- Emülasyon işleminde emüle edilecek entegre bu entegre kütüphanesinden seçilir.
- Seçilen entegrenin bacak bağlantı şeması, bacaklar arasındaki ilişki, doğruluk tablosu vb. gibi özellikleri gösterilir.
- Çevrimsel çalışma başlatılır. Bu çalışma sırasında:
 - ✓ Konnektörden entegrenin girişlerinin okunması işlemi yapılır.
 - ✓ Okunan giriş değerlerine bağlı olarak fonksiyon işlemleri yapılır.
 - ✓ Fonksiyon işlemleri yapıldıktan sonra emülasyon işlemine tabi tutulan entegrenin çıkışları elde edilir.
 - ✓ Elde edilen çıkışlar konnektöre gönderilir.

Sistem yazılımının akış şeması aşağıdaki gibidir:



Şekil 5.1: Sistem Yazılımının Akış Şeması

C dilinde donanımdan veri okuma işlemi **inport** komutuyla yapılmaktadır. Donanımdan byte(8-bit) okumak için **inportb**, word(16-bit) okumak için **inport** komutu kullanılır. Kullanımı

Okunan Değer = **inport** (Port Adresi); şeklindedir.

Donanıma veri gönderme işlemi ise **outport** komutu ile yapılır. Donanıma byte(8-bit) göndermek için **outportb**, word(16-bit) göndermek için **outport** kullanılır. Kullanımı

Outport (Port Adresi, Gonderilecek Değer); şeklindedir.

Oluşturulan entegre kütüphanesinden istenilen entegre seçilir. Seçilen entegrenin bacak bağlantı şeması, bacaklarının ilişkisi, doğruluk tablosu gibi özellikleri gösterilir. Arabirim kartına hangi bacağın giriş hangisinin de çıkış olduğunu belirten bacak maskeleri tanıtılır. Daha sonra çevrimsel çalışma içinde sırasıyla;

- Konnektör'den okuma işlemi

```
OkuDeg= inport(0x0234);
p=&I;
*p=OkuDeg;
```

- Fonksiyon işlemleri
- Çıkış bacak durumlarının belirlenmesi
- Konnektör'e çıkış verme

```
q=&Q;
YazDeg=*q;
outport(0x0232,YazDeg); işlemleri yapılır.
```

Program RUN edilirse önce emüle edilecek entegrenin bacak sayısını ister. Tasarlanan bu sistem için bacak sayısı maksimum 16 dır. Daha sonra tek tek hangi bacağın giriş hangisinin çıkış olduğunu; giriş için 1, çıkış için 0 vermek suretiyle girilmesini ister. Girilen bu değerleri Decimal sayıya çevirerek **Outport**

Outport(0x0230, Çıkışa Gönderilecek Değer)

komutuyla 0x0230 numaralı adrese gönderir. Bu durumda sistem girişindeki 74373 Latch'leri enable olur ve giriş çıkış belirlenmiş olur.

Daha sonra **inport** komutu kullanılarak

Okunan Değer = inport (0x0234)

karttan okuma işlemi yapılır ve okunan veriler 0x0234 adresine saklanır. 0x0234 adres bilgisi aynı zamanda sistem çıkışındaki 74244 buffer'larını enable eder ve okuma işlemi yapılır.

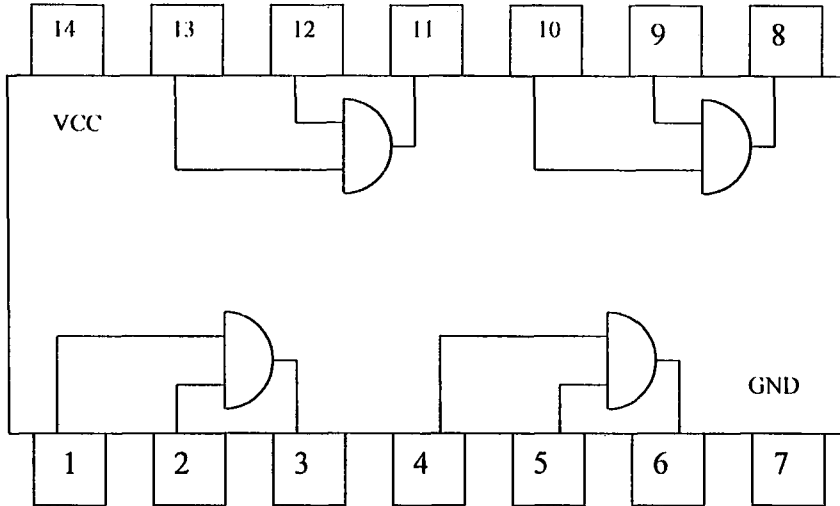
Seçilen entegrenin tanım bağlantısına uygun olarak okunan girişler lojik işlemlere tabi tutulur. Elde edilen çıkışlar Outport komutu ile

Outport(0x0232, Çıkışa Gönderilecek Değer)

0x0232 adresine gönderilir. Bu adres aynı zamanda sistem çıkışındaki 74373 Latch'lerini enable eder ve yazma işlemi yapılmış olur.

6. UYGULAMA ÖRNEKLERİ

Sistemin testi amacıyla 7408 entegresi emüle edildi. Bu entegrenin yapısı;



Şekil 6.1: 7408 Entegresinin Yapısı

Dijital bir kart üzerindeki 7408 entegresi sökülerek yerine tasarlanan karttan alınan 14'lü soket takıldı. Program Run edilip entegre giriş çıkış tanımları yapıldı (Giriş için 1, Çıkış için 0). GND ve VCC uçları giriş olarak seçildi ama herhangi bir lojik işleme tabi tutulmadığı için sadece bu uçların okunması işlemi yapıldı. Bacak giriş çıkış tanımları için Tablo 6.1'deki değerleri girildi.

Tablo 6.1: 7408 Entegresinin Tanımı İçin Girilen Değerler

Entegre Bacak Numarası	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Giriş	1	1		1	1		1		1	1		1	1	1
Çıkış			0			0		0			0			

Daha sonra sistem girişleri okundu. Okunan bu değerler 7408'in tanım bağıntısına bağlı olarak lojik işlemlere tabi tutulduktan sonra elde edilen değerler

çıkışa aktarıldı. Herhangi bir anda Tablo 6.2'deki değerleri okundu ve lojik işlemlerle elde edilen değerler çıkışa aktarıldı.

Tablo 6.2: 7408 Entegresinin Testi Esnasında Okunan Ve Çıkışa Aktarılan Değerler

Entegre Bacak Numarası	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Okunan Değerler	1	1		1	1		0		1	0		1	0	1
Yazılan Değerler			1			1		0			0			

Böylece sistemin doğruluğu test edilmiş oldu.





7.SONUÇ

Bu çalışma neticesinde:

- Donanımı süren yazılımların hazırlanması
- Gerçek zaman kontrolü
- Arabirim devrelerinin tasarlanması
- Yazılım ile donanımın haberleştirilmesi

konularında üretime yönelik önemli deneyimler elde edilmiştir.

Bazı değişiklikler ile daha karmaşık ve kritik sistemlerin emülatörleri yapılabilir. Bu yolla eğitim ve askeri amaçlı simülatörlerin daha gerçekçi kullanılması ile gelişmesine katkıda bulunulabilir. Sistem tasarımı aşamasında sistem simülatörü olarak çalıştırılabilir.

Tasarlanan bu sistem herhangi bir dijital Entegre'nin, Eprom'un, piyasada olmayan fakat uygulamalarda kullanılması gereken giriş/çıkış sistemlerinin veya geliştirilerek daha büyük sistemlerin emülasyonunda kullanılabilir. Bunun için emüle edilecek entegre veya sistemin giriş/çıkış tanımlarının yapılması ve bu giriş/çıkış'lar arasındaki lojik ilişkinin yazılım üzerinden tanımlanması yeterli olacaktır. Bu işlemler yapıldıktan sonra Emülatör'den konnektör ile alınacak giriş/çıkış'lar emüle edilecek entegre veya sistemin yerine takılarak emülasyon işlemi yapılacaktır.

Ayrıca bu sistem arıza arama ve sistem sorgulama cihazı olarak kullanılabilir. Sistem herhangi bir elektronik karttaki, herhangi bir entegre devresinin yerine kullanılabilecek şekilde tasarlanmıştır. Bu şekilde arızalı bir kartta, arızanın düşünülen entegreden kaynaklanıp kaynaklanmadığına karar verilebilir. Entegrenin yerine takılacak soketin pinlerinin entegre ile aynı şekilde programlanmasından sonra entegre bağımsız olarak enerjilendiğinde gerekli ölçümler yapılarak arızalı entegrenin tespiti daha da kolaylaşmaktadır.

Bu sistem daha da geliştirilerek, giriş/çıkış sayısı çok fazla sistemlerin emülasyonunda da kullanılabilir.

KAYNAKLAR DİZİNİ

1. Cheng, Kwang-Ting, Huang, Shi-Yu, and Dai, Wei-Jin, Fault Emulation : A New Methodology For Fault Grading, IEEE Transactions on Computer-aided design of integratedcircuits and Systems, Vol. 18, No.10, October 1999
2. HITACHI, "Electronic Data Book"
3. Hwang, Shih-Arn, Hong, Jin-Hua, and Wu, Cheng-Wen, Sequential Circuit Fault Simulation Using Logic Emulation, IEEE Transactions on Computer-aided Design of Integrated Circuits And Systems, Vol. 17, No. 8, August 1998
4. Karsai, G., and Driscoll, J., Lab # 2: In-circuit Emulation, <http://www.vanderbilt.edu>, 1998
5. Kirovski, Darko, Potkonjak, Miodrag, and Guerra, Lisa M., Improving the Observability and Controllability of datapaths for Emulation-Based Debugging, IEEE Transactions on Computer-aided Design of integrated circuits and Systems, Vol. 18, No. 11, November 1999
6. Kishi, Minoru, Computer System With Emulation Mechanism, Kabushiki Kaisha Toshiba, October 1992
7. Maguire,Jr., and Francis, J., Apparatus For Emulation Of Electronic Hardware System, Quickturn Systems, Incorporated, April 1992
8. Wen-Jon Fang, Allen C.-H. Wu, and Duan Ping Chan, A Module Generator For Logic Emulation Applications, IEEE Transactions on Very Large Scale Integration (VLSI) Systems , Vol. 7, No.4, December 1999

ÖZGEÇMİŞ

1977 yılında Elazığ'da doğdu. İlk, Orta ve Lise tahsilini Malatya'da tamamladıktan sonra 1998 yılında Karadeniz Teknik Üniversitesin'den Elektronik Mühendisi ünvanı ve bölüm 2.ligi derecesiyle ile mezun oldu. Lisans bitime çalışmasını Radyo İletişim Sistemleri üzerine yaptı. 1998 yılında Gebze Yüksek Teknoloji Enstitüsü Bilgisayar Mühendisliği Bölümü'nde yüksek lisans öğrenimine başladı.



EKLER

Ek-1 : Malzeme Yerleşimi

Ek-2 : Baskı Devre Üst Yüzeyi

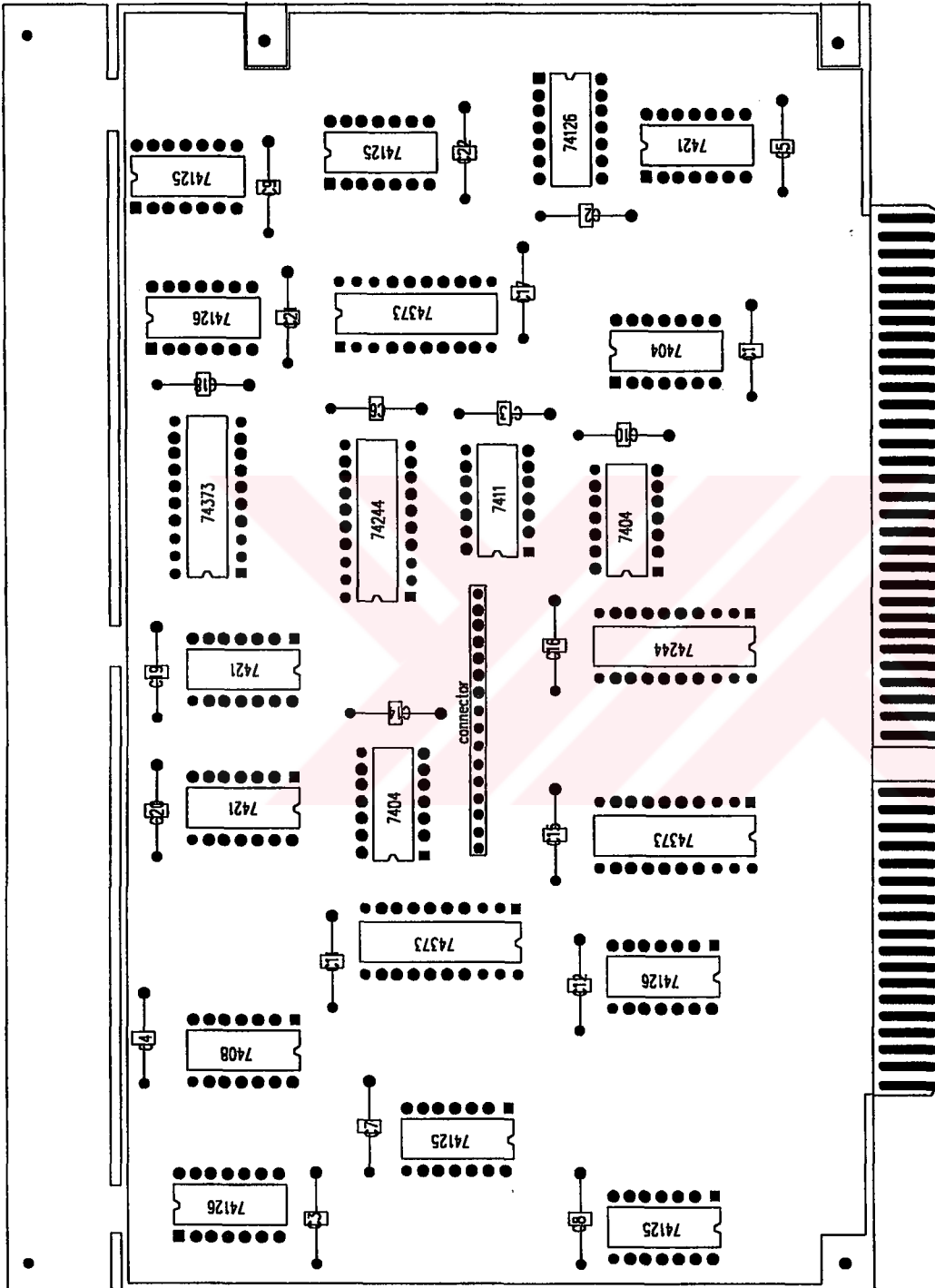
Ek-3 : Baskı Devre Alt Yüzeyi

Ek-4 : Devre Şeması

Ek-5 : Sistemde kullanılan yazılım



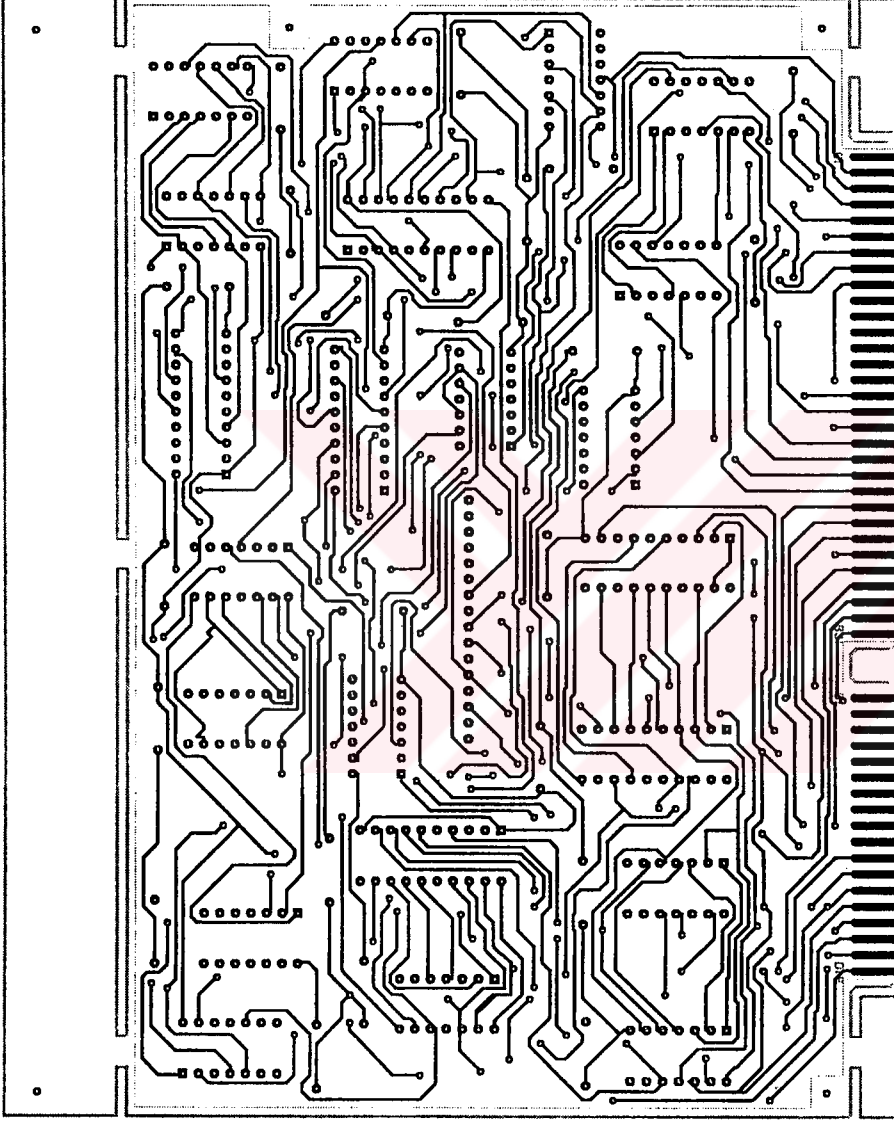
Ek-1 : Malzeme yerleşimi



MALZEME YERLESİMİ

Ek-2 : Baskı Devre Üst Yüzeyi

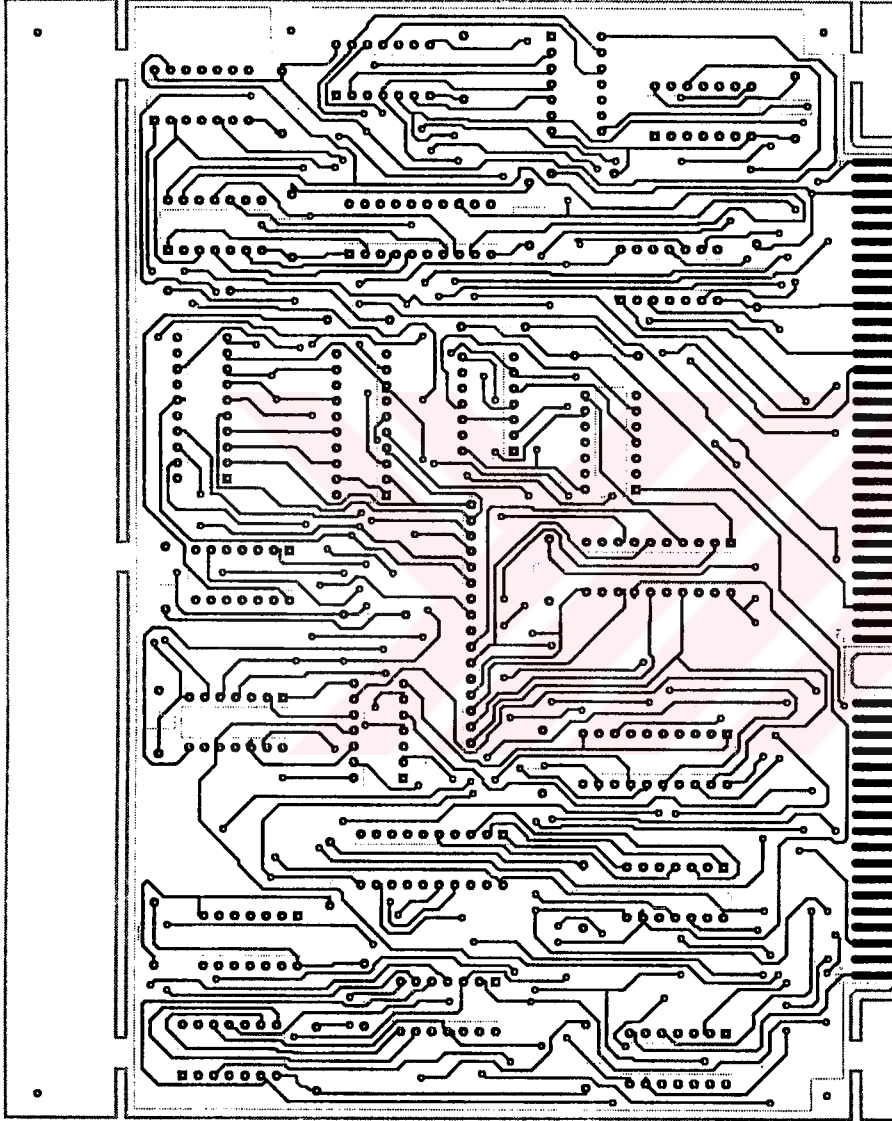
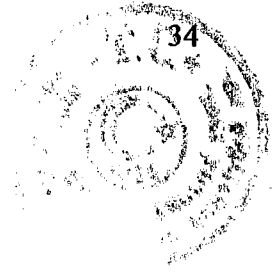
33



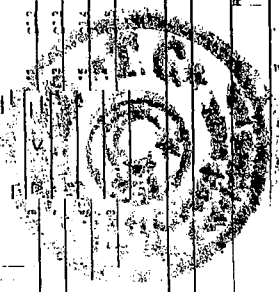
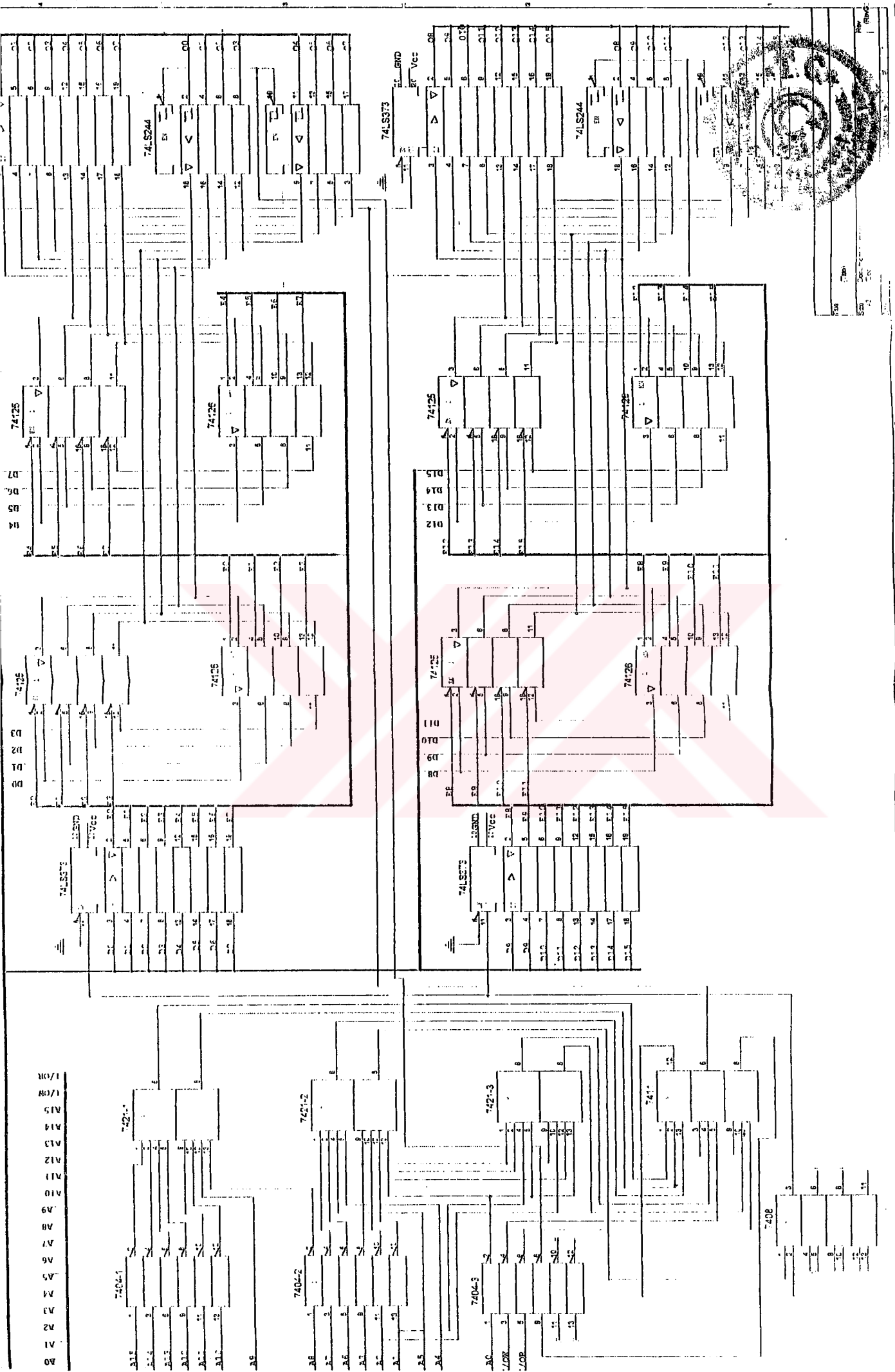
BASKI DEVRE ÜST YÜZEYİ

.GTL

Ek-3 : Baskı Devre Alt Yüzeyi



BASKI DEVRE ALT YÜZEYİ



Rev. 1.0
Date: 10/10/00
By: [Signature]
Check: [Signature]
Drawn: [Signature]

Ek-5 : Sistemde kullanılan yazılım

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<dos.h>
#include<bios.h>
struct giris{
    unsigned    int  I0:1;
    unsigned    int  I1:1;
    unsigned    int  I2:1;
    unsigned    int  I3:1;
    unsigned    int  I4:1;
    unsigned    int  I5:1;
    unsigned    int  I6:1;
    unsigned    int  I7:1;
    unsigned    int  I8:1;
    unsigned    int  I9:1;
    unsigned    int  I10:1;
    unsigned    int  I11:1;
    unsigned    int  I12:1;
    unsigned    int  I13:1;
    unsigned    int  I14:1;
    unsigned    int  I15:1;
    }I;

struct cikis{
    unsigned    int  Q0:1;
    unsigned    int  Q1:1;
    unsigned    int  Q2:1;
    unsigned    int  Q3:1;
    unsigned    int  Q4:1;
    unsigned    int  Q5:1;
```

```

    unsigned    int Q6:1;
    unsigned    int Q7:1;
    unsigned    int Q8:1;
    unsigned    int Q9:1;
    unsigned    int Q10:1;
    unsigned    int Q11:1;
    unsigned    int Q12:1;
    unsigned    int Q13:1;
    unsigned    int Q14:1;
    unsigned    int Q15:1;
    }Q;

```

```

long int *p;
long int *q;
unsigned int OkuDeg,YazDeg;

```

```

char kr;
int entegre;
unsigned int n=0;
int i=0,a,b;
int bacak[16];
unsigned int GirisDec;

```

```

void main()

```

```

{

    clrscr();
    dongu:
    printf("\nEmule edilecek Entegrenin Bacak Sayısını
Giriniz.\nN<16 olmalı\nN=");
    scanf("%d",&n);
    if(n>16)
    {

```

```
printf("\a yanlis girdiniz");
goto dongu;
}
else
    printf("\nN=%d olarak girdiniz\n\n",n);
printf("Bacaklarin giris ya da cikis olduguna dair
bilgiyi giriniz.\n");
printf("Giris icin 1, cikis icin 0 degerini
giriniz.\n");

for(i=0;i<n;i++)
{
    printf("%d. bacak: ",i+1);
    scanf("%d",&bacak[i]);
    if(bacak[i]==1)
    {
        printf("                giris\n");
    }
    else if(bacak[i]==0)
    {
        printf("                cikis\n");
    }

    else
    {
        printf("yanlis bir deger girdiniz\n");
        i--;
    }
}

printf("Yanlis bilgi girdiginiz entegre bacagi var mi?
E/H:");
```

```
kr=getch();
```

```
    if((kr=='E')||(kr=='e'))
    {
        printf("\nlutfen bacak numarasini ve degerini
giriniz:\n" );
        scanf("%d",&a);
        scanf("%d",&b);
        bacak[a-1]=b;

        if(b==1)
            printf("%d. bacak giris olarak duzeltildi.
\nLutfen bir tusa basınız.. ",a);

        if(b==0)
            printf("%d. bacak cikis olarak duzeltildi.
\nLutfen bir tusa basınız.. ",a);
    }

    else
        printf("\nbilgilerinizin dogrulugu onaylandi.
\nLutfen bir tusa basınız..");
```

```
    getch();
```

```
clrscr();
```

```
GirisDec=0;
```

```
for(i=0;i<n;i++)
```

```
{
```

```
    printf("%d. bacak: %d\n",i+1,bacak[i]);
```

```
    GirisDec= GirisDec+bacak[i]*pow(2,i);
```

```
}

printf("Girdiginiz degerlerin decimal karsiligi %d
dir\n",GirisDec);
outport(0x0230,GirisDec);
```

```
/* Entegrelerimiz .....*/
```

```
printf("\n1-7405");
printf("\n2-7408");
printf("\n3-7420");
printf("\n4-7432");
```

```
printf("\nentegrenizi seciniz");
scanf("%d",&entegre);
```

```
if(entegre==1){
    clrscr();
    puts("Emulasyondan cikmak icin her hangi bir
tusa basin");
```

```
while(!kbhit()) {
    OkuDeg= inport(0x0234);
    p=&I;
    *p=OkuDeg;
```

```
Q.Q1=!(I.I0);
Q.Q3=!(I.I2);
Q.Q5=!(I.I4);
Q.Q7=!(I.I8);
Q.Q9=!(I.I10);
Q.Q11=!(I.I12);
```

```
q=&Q;
YazDeg=*q;
outport(0x0232,YazDeg);
}
```

```
else if(entegre==2){
    clrscr();
    puts("Emulasyondan cikmak icin her hangi bir
tusa basin");
    while(!kbhit()) {
```

```
        OkuDeg= inport(0x0234);
```

```
        p=&I;
```

```
        *p=OkuDeg;
```

```
        Q.Q2=I.I0&I.I1;
```

```
        Q.Q5=I.I3&I.I4;
```

```
        Q.Q7=I.I8&I.I9;
```

```
        Q.Q10=I.I11&I.I12;
```

```
q=&Q;
```

```
YazDeg=*q;
```

```
outport(0x0232,YazDeg);
```

```
}
```

```
}
```

```
else if(entegre==3){
    clrscr();
    puts("Emulasyondan cikmak icin her hangi bir
tusa basin");
```



```
while(!kbhit()) {
    OkuDeg= inport(0x0234);
    p=&I;
    *p=OkuDeg;

    Q.Q5=!(I.I0&I.I1&I.I3&I.I4);
    Q.Q7=!(I.I8&I.I9&I.I11&I.I12);

    q=&Q;
    YazDeg=*q;
    outport(0x0232,YazDeg);
}

else if(entegre==4){
    clrscr();
    puts("Emulasyondan cikmak icin her hangi bir
tusa basin");
    while(!kbhit()) {
        OkuDeg= inport(0x0234);
        p=&I;
        *p=OkuDeg;
        Q.Q2=I.I0||I.I1;
        Q.Q5=I.I3||I.I4;
        Q.Q7=I.I8||I.I9;
        Q.Q10=I.I11||I.I12
        q=&Q;
        YazDeg=*q;
        outport(0x0232,YazDeg);
    }
}
```