YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR KONTROLLU TTL ENTEGRE TEST DEVRESİ

Elektronik ve Hab. Müh. Türkay DUMAN

F.B.E. Elektronik ve Haberleşme Mühendisliği Anabilim Dalı Elektronik Programında hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı: Prof. Şefik Sarıkayalar

İSTANBUL, 1994

İÇİNDEKİLER

TEŞEKKÜR

Bu çalışma ile gün geçdikçe önemi artan sayısal entegre devreler hakkında bilgi verilmiş ve bu entegre devreleri test edebilecek bir sistem geliştirilmiştir.

Beni bu konuya teşfik eden ve yardımlarını esirgemeyen başta, danışmanım Sayın Prof. Şefik Sarıkayalar'a, Sayın Yrd. Doç. Dr. Tuncay Uzun'a, Sayın Arş. Gör. Savaş Barış'a ve Sayın Arş. Gör. Ekrem Özorbeyi'ye teşekkürlerimi bir borç bilirim.

Mümkün olduğu kadar kullanışlı yapmaya çalıştığım bu sistemin üniversite laboratuvarında değerlendirilmesini ve geliştirilmesini dilerim.

Türkay Duman

26.9.1994

ÖZET

Bu çalışmada, aşağıda fonksiyonları verilen Bilgisayar Kontrollu TTL Entegre Test Devresinin teorik ve uygulamalı olarak tasarımı yapılmıştır.

Sistemin başlıca fonksiyonları şunlardır:

1. TTL ve CMOS lojik entegre devrelerin fonksiyonlarını kontrol ederek arızalı olup olmadığını test eder. Diğer bir ifade ile entegre devrenin bir devre üzerinde kullanıp kullanılmayacağını belirler.
2. Fonksiyonu bilinmeyen sağlam entegre devrelerin fonksiyonunu belirler.
3. Tanımlanmamış lojik entegre devre fonksiyonları kullanıcı tarafından rahatlıkla tanımlanabilir.

Sistem iki ayrı bölümden oluşur:

1. Donanım,
2. Yazılım.

Donanım üç bölümden oluşur:

1. Kontrol ünitesi,
2. Paralel haberleşme devresi,
3. Bağlantı devresi.

Sistemin kontrolunu 386DX-40 tipi bir kişisel bilgisayar sağlamıştır.

Bilgisayar ile bağlantı devresinin haberleşmesini, toplam 72 bit giriş/çıkışa olanak veren 3 adet 8255 PPI'dan oluşan paralel haberleşme port devresi sağlamıştır.

Bağlantı devresi, bilgisayardan gönderilen giriş/çıkış bilgileri için, test edilecek entegre devre bacaklarına uygun elektriksel bağlantıyı sağlayan devredir.

Yazılım PC'de Turbo Pascal V5.0 ile yapılmıştır. Yazılımın başlıca alt bölümleri şunlardır:

1.Self test,
2.Arıza testi,
3.Tanıma testi,
4.Listeleme,
5.Tanımlama,
6.Açıklama.

Bu çalışmanın ikinci bölümünde entegre devreler incelenmiş ve ilerleyen bölümlerde bu bilgiler kullanılarak bilgisayar kontrollu TTL entegre test devresi tasarlanmıştır.

ABSTRACT

In this working, given below its functions The Computer Controlled TTL Test Circuit is designed theoretically and practically.

Main functions of the system are as follows:

1.It tests if TTL and CMOS logic circuits has failure controlling its functions. In other words, it determines that an integrated circuit can be used in a circuit or not.

2.It determines the name of the undefined healthy integrated circuit.

3.The function of the undefined logic integrated circuits are defined by the user.

System consist of two parts :

1.Hardware,

2.Software.

The hardware has three parts:

1.Control Unit,

2.Parallel Communication Circuit,

3.Connection Circuit.

The control of system is provided by a 386DX-40 personal computer.

The communication beetwen the computer and the connection circuit is provided by the parallel communication port that has three 8255 PPI(Programmable peripheral interface) integrated circuits with 72 bit input/output.

The connection circuit provides the proper electrical connection to testing integrated circuit pins.

Software is written on Turbo Pascal V5.0.

Software consist of 6 parts:

1.Self test,

2.Failure test,

3.Determining test,

4.Menu,

5.Function defining,

6.Help.

In the second section of this study, integreted circuits are examined and in following sections building of the Controlled Computer TTL Test Circuit has been planned by using this knowledges.

1. GİRİŞ

Tüm teknik konularda olduğu gibi elektronikte de ayrıntılardan bütüne ulaşma çabası söz konusudur. Bu durum özellikle transistörün bulunmasından sonra, elektronik devrelerin yarı iletken yonga üzerinde tümleştirilip hazır eleman haline gelmesi ile hızlanmıştır. Bu hazır devreler sayesinde tasarımcılar daha geniş çaplı projelere zaman ayırma olanağına kavuşmuştur.

Bu entegre (tümleşik) devrelerin ortaya çıkmasından sonra yeni arıza olayları da söz konusu olmuştur. Entegre devrelerin testi, çok çeşitli fonksiyonlara sahip olduklarından normal şartlar altında hızlı ve kolay bir şekilde yapılamamaktadır. Bu testler için ayrıca düzenek kurmak gerekebilir. Bu nedenle bu testleri daha kolay ve hızlı yapabilmek için entegre test cihazları geliştirilmiştir.

Bu çalışmada, bahsedilen türde bir test cihazı tasarlanmıştır. TTL ve CMOS yapıya sahip lojik ya da sayısal entegreler test edilebilmektedir. Test gerilimi olarak 5-0 V seçilmiştir. Aynı zamanda bilinmeyen sağlam bir entegreyi tanıyabilme yeteneğine sahiptir. (Tanınacak entegrenin daha önce sisteme tanımlanmış olması gerekir.) Entegreler kullanıcı tarafından rahatlıkla tanımlanabilmektedir. Bu sayede sisteme esneklik ve kullanışlılık getirilmiştir.

Bu cihaz kişisel bilgisayar kontrollu olarak tasarlanmıştır. Yazılımı Turbo Pascal V.5 üzerinde yapılmıştır. Sistem, karmaşık fonksiyonların ve tanımların kullanılması nedeniyle bilgisayar kontrollu olarak tasarlanmıştır. Entegre tanımları için oldukça geniş hafıza gerektirdiğinden bilgisayar olarak hard diske sahip bir kişisel bilgisayar tercih edilmiştir.

Tezin konu akışı şu şekildedir:

2. bölümde ön bilgi olması açısından sayısal entegre devreler ve TTL,CMOS yapılar hakkında bilgi verilmiştir.

3. bölümde, geliştirilen TTL entegre test devresinin çalışma mantığı ve özelliklerinden bahsedilmiştir. Ayrıca sistemin kullanımı hakkında bilgi verilmiş, entegre tanımlarının nasıl yapıldığı örneklerle açıklanmıştır.

4. bölümde TTL entegre test devresinin donanım yapısından bahsedilmiştir. Bilgisayar ile habarleşen 72 bit paralel haberleşme portu ve test edilecek entegre devre ile bağlantıyı sağlayan bağlantı devresi açıklanmıştır. Bağlantı devresi test edecek entegre devrenin bacak durumuna göre giriş, çıkış ve besleme uçlarını düzenleyen bir devredir.

4. bölümde TTL entegre test devresinin yazılımı ana akış diyagramları ile anlatılmış ve program açıklanmıştır.Yazılım Turbo Pascal V.5 ile yazılmıştır. Ana hatları ile programlar şunlardır:

1.Self test,

2.Arıza testi,

3.Tanıma testi,

4.Liste,

5.Tanımlama,

6.Açıklama.

2. SAYISAL ENTEGRE DEVRELER

Entegre (tümleşik) devreler yarı iletken yonga üzerinde hazırlanarak eleman haline getirilmiş devrelerdir. Bu devreler tasarımda hazır şekilde kullanılmakta ve mühedislere kolaylık sağlamaktadır.

Entegre devreler analog ve sayısal tümleşik devreler olmak üzere iki çeşitdir. Anolog devreler gerilim değeri zamanla sürekli değişen entegre devreler, sayısal devreler ise gerilim değeri zamanla belli değerler arasında değişen tümleşik devrelerdir. Anolog devrelerde anolog işaretler, sayısal devrelerde sayısal işaretler kullanılır.

Bu tezde sayısal entegre devrelerden bahsedilmektedir.

İşlevsel olarak, elektronik mantıksal değişimlerin ve buna bağlı kararların uygulanarak sonuca ulaşılması amacıyla kullanılan devrelere sayısal elktronik devreler denir.

2.1. Sayısal Mantık

Sayısal elektronik devrelerde sayısal mantık kullanılır. Sayısal mantığın anlaşılabilmesi için iki kavramın açıklanması gerekir: Karar uygulayıcı özellikler ve hafıza özellikleri.

Karar uygulayıcı özellik, sistemin dışarıdan verilen görev doğrultusunda işlem yapmasıdır. Bu özellliğe sahip elemanlara karar uygulayıcı elemanlar denir.

Hafıza özelliği ise geçmişte aldığı bir mantık seviye değerini istenildiği sürece saklayabilme ve yine istenildiğinde bu değeri değiştirip yeni değeri saklamaya devam etmesidir.Bu özellliğe sahip elemanlara hafıza elemanları ya da saklayıcılar denir.

Sayısal mantık, 1 ve 0 sayılarını kullanan ikili sayı sistemi üzerine kuruludur. Bu nedenle sayısal devrelerde 1 ve 0 sayıları ile simgelenen iki gerilim seviyesi kullanılır.

2.2. Karar Uygulayıcı Elemanlar

Sayısal mantıkta 3 ana eleman kullanılır: AND, OR, NOT.

2.2.1. AND Kapısı

Sadece tüm girişlerin "1" mantık seviyesinde olduğu anda çıkışın "1" ve diğer tüm halllerde çıkışın "0" olduğu bir elemandır.

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Tablo 2.1. AND Doğruluk tablosu

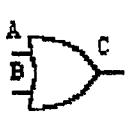
Şekil 2.1. AND Kapısı

2.2.2. OR Kapısı

Sadece, girişlerinden herhangi birinin ya da ikisinin "1" mantık seviyesi olduğu anda çıkışı da "1" olan elemandır.

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Tablo 2.2. OR Doğruluk tablosu

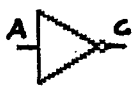


Şekil 2.2. OR Kapısı

2.2.3. NOT Kapısı

Girişinde bulunan mantık seviyesinin tersini çıkışına verir.

A	C
0	1
1	0



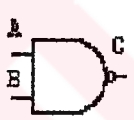
Tablo 2.3. NOT Doğruluk tablosu Şekil 2.3. NOT Kapısı

Bu ana elemanlar dışında kullanışlı 3 eleman daha vardır: NAND, NOR, EXOR.

2.2.4. NAND Kapısı

Çıkışında çevirici olan AND kapısıdır.

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

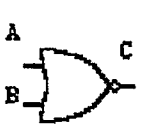


Tablo 2.4. NAND Doğruluk tablosu Şekil 2.4.NAND Kapısı

2.2.5. NOR kapısı

Çıkışında çevirici olan OR kapısıdır.

A	B	C
0	0	1
0	1	0
1	0	0
1	1	0



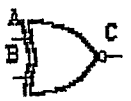
Tablo 2.5. NOR Doğruluk tablosu Şekil 2.5. NOR Kapısı

2.2.6. EXOR kapısı

Çıkışı, girişlerinin ikisininde "1" mantık seviyesi veya ikisinin de "0" mantık seviyesinde olması durumunda "0"dır.

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Tablo 2.6. EXOR Doğruluk tablosu

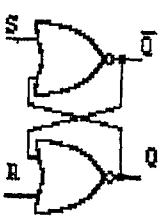


Şekil 2.6. EXOR Kapısı

2.3. Temel Hafıza Elemanları

2.3.1. RS Mandalı (RS Latch)

Normalde bir hafıza elemanı, veriyi belli bir süre saklaması ve sonra bırakması için kullanılır. Bu tip hafıza elemanı iki NOR kapısı gerektirir.(Şekil 2.7.)



Şekil 2.7. RS Mandalı

RS mandalı ya da RS flip-flop, flip-flop olarak adlandırılan devre sınıfının en temel şeklidir.

İlk olarak set girişi S'yi 1, reset girişi R'yi 0 durumunda tutalım. (NOR doğruluk tablosunda olduğu gibi.)Bu durumda $\overline{Q}$ çıkışı, "0" olur. Ancak $\overline{Q}$'daki "0", alttaki kapıya bağlanmıştır ve bu kapının

iki "0" girişi vardır. Q çıkışı, "1" olur. Q'daki "1" gerisin geriye yukarıdaki kapının girişine bağlanmıştır ve şimdi bu kapının iki girişi de "1"dir. NOR kapısı olduğundan, bu kapının çıkışını "0" durumunda tutabilmek için sadece bir tane "1"e ihtiyacı vardir.Bu yüzden S'teki "1", "0"a dönüşebilir ve $\overline{Q}$ çıkışı aynı kalır .

Devreyi reset etmek için R'yi "1" yapalım. Yukarıdaki işlemleri tekrarlayarak, bu sefer $\overline{Q}$'nun "0" ve Q'in "1" olması dışında, aynı sonuçları elde edebiliriz. Devre reset edilmiş ve hafızası silinmiştir. RS mandalı S girişi "1" yapıldığında Q çıkışını Q=1 durumuna "mandallamak"(sabit tutmak) ve R girişi "1" yapıldığında Q çıkışını sıfırlamak özelliğine sahiptir. Eğer değişimli olarak girişleri "1" yaparsak Q ve $\overline{Q}$ "1" ve "0" arasında değişir.

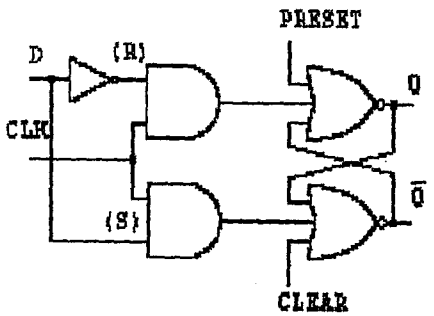
Düşünülmesi gereken son bir durum vardır: R ve S girişlerinin her ikisininde "1" olması. NOR kapılarına yapılan herhangi bir "1" girişi, çıkışı "0" yaptığından, bu şartlar altında her iki çıkış da "0" olacaktır.Bu kaçınılması gereken bir durumdur.Çünkü çıkışı belirsiz kılar. Ancak devrenin temel fonksiyonunu değiştirmez.

2.3.2. D Mandalı (D Latch)

R girişini üretmek için bir çevirici kullanıldığından NOR kapılarının girişleri birbirlerinin eşleniği olacaktır yani biri "1" olduğunda diğeri mutlaka "0" olacaktır. Böylelikle belirsizlik durumundan kaçınılmış olur. Bu yüzden D-mandalı, RS-mandalındaki belirsizlik sorununa basit bir çözüm getirir ve çok yaygın olarak kullanılır. (Şekil 2.8)

D mandalının bir data girişi olduğunu farzedelim. Mandal saat sinyali "1" yapmakla set (Q=1), "0" yapmakla reset (Q=0) edilir. D mandalını doğru bir şekilde kullanmak için önce saat sinyali "1" yapılır, data hattına istenilen giriş uygulanır ve sonra data

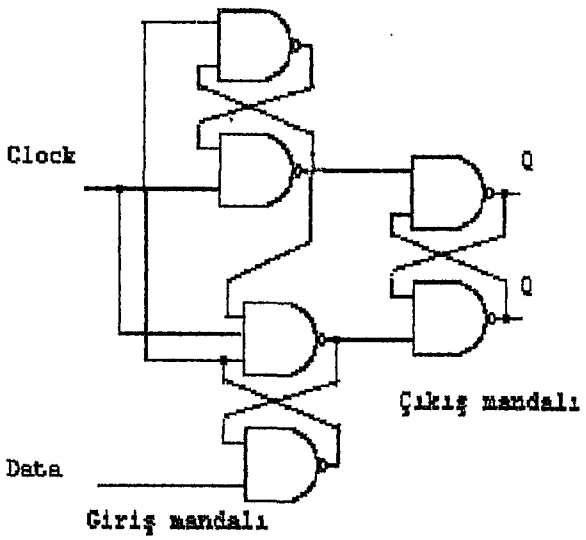
hattındaki giriş sinyalini değiştirmeden önce saat sinyali çıkartılır. Saat sinyali çıkartılır çıkartılmaz devre "mandallanır" ve data girişi, çıkışı etkilemeden değişebilir.



Şekil 2.8. D Mandalı

2.3.3. D Flip-flop

D flip-flop'u ile D mandalı arasındaki fark data alırken saat sinyalinin farklı bir şekilde kullanılmasıdır. Devreyi analiz edersek, çıkış mandalının durumunun sadece saat sinyalinin "0"dan "1"e dönüştüğü anda değiştiğini görürüz. Diğer bir deyişle bir flip-flop'un ana özelliği girişteki data'yı sadece saat vuruşunun yükselen kenarıyla örneklemesidir. Bu olay kenar tetiklemesi olarak bilinir ve D,JK flip-flop'ları gibi pek çok flip-flop'un karakteristiğidir. (Şekil 2.9)



Şekil 2.9. D Mandalı

2.3.4. JK Flip-flop

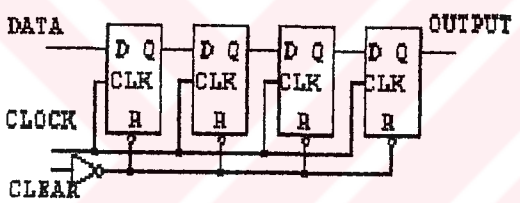
RS mandalı gibi 2 data girişi vardır, ancak RS mandalının noksanlarının hiçbirisi bu flip-flop'ta yoktur. Belirsiz bir çıkış vermez. D-flip-flop'u gibi saatli ve daima kenar tetiklemelidir, ancak JK flip-flop'unun birçok çeşidi saat sinyalinin yükselen kenarlarıyla (leading edge) değil,takip eden (düşen,trailing edge) kenarlarıyla kontrol edilir.

2.4. Temel Devreler

Karar uygulayıcı ve hafıza elemanları kullanılarak çeşitli mantık devreleri tasarlanmıştır. Başlıcaları şunlardır:

2.4.1. Kayan Saklayıcılar (Shift Registers):

Kayan saklayıcı, şekil 2.10'da da görüldüğü gibi birinin çıkışı diğerinin girişine bağlanmış bir dizi flip-flop'dan oluşur. Kayan saklayıcıda bütün flip-flop'lar ortak bir saat sinyaline bağlıdır ve senkronize olarak aynı anda set ve reset durumundadırlar. Her clock darbesinde her flip-flop çıkışındaki bilgi bir sonraki flip-flop'un girişinden alınır.



Şekil 2.10 Kayan Saklayıcı

2.4.2.Sayıcılar (Counters):

Kayan saklayıcılara benzeyen sayıcı devrelerinin amacı, veriyi özel bir şekilde çıkartmak veya belli sayıdaki flip-floplarla elde edilebilecek farklı durum sayısını azami hale getirmektir. Sayıcılar aritmatik işlem, frekans bölme ve ölçme, zaman aralığı ölçümü ve diğer bir çok amaç için kullanılırlar.

Sayıcı devreler JK, D, RS flip-floplarından yapılırlar. Asenkron ve senkron olmak üzere iki ana tipe ayrılırlar. Senkron sayıcılarda bütün flip-flop'lar aynı anda değiştirilir. Asenkron flip-flop'u tetikler. Diğerleri de aynı şekilde birbirini tetikler.

2.4.3. Diğer Devreler

-Kod Çözücüler (Decoders):

Kod çözücüler ikili, BCD veya başka bir kodu kodlanmamış hale çevirirler.

-Kodlayıcılar (Encoders):

Kodlayıcılar kodlanmamış bir veya daha fazla sinayali ikili,BCD veya başka bir koda çevirirler.

-Çoklayıcılar (Multiplexers) :

Bir çok giriş hattındaki veriyi yeniden düzenleyip bir çıkış hattından veren devrelere çoklayıcı denir.

-Çoklama Çözücüler (Demultiplexers):

Çoklama çözücü, çoklayıcının tersine, bir kaynaktan gelen veriyi belli bir kalıba bağlı kalarak bir çok çıkış hattına dağıtır.

2.5. Lojik Aileler

Lojik devrelerin gerçekleştrildiği temel elektronik devre yapılarına lojik aileler ya da lojik yapılar denir. Bipolar ve unipolar olmak üzere iki gruba ayrılırlar.

2.5.1. Bipolar Lojik Aileler

DCTL:Doğrudan bağlamalı (direct-coupled) tranzistor lojiği(artık kullanılmamaktır).

RTL:Direnç-transistor lojiği (artık kullanılmamaktadır).

DTL:Diyot-transistor lojiği.

TTL:Transistor-transistor lojiği (T2L).

CTL:Eşlenik (complementary) transistor lojiği.

ECL:Emetör bağlamalı (emitter-coupled) lojik (CML=Current mode logic=Akım modu lojiği).

I2L:Akım enjeksiyonu lojiği (Integrated injection logic).

Özel gruplar:

TTL ailesi:L serisi (Low-power=Düşük güçlü).

H serisi (High-speed=Yüksek hızlı),yerini S serisi (Schottky kenetleme diyotlu seri) almıştır.

LS serisi (düşük güçlü, Schottky kenetleme diyotlu seri)

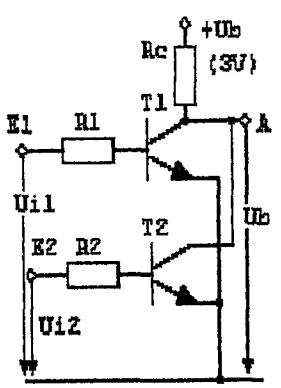
DTLZ(Z-diyotlu diyot-transistor lojiği)

HLL (High-level logic=Yüksek gürültü marjlı lojik)

LSL (yavaş ve gürültü bağışıklığı büyük lojik)

2.5.1.1. RTL (direnç-transistor lojiği)

Direnç-transistor lojiğinin temel devresi Şekil 2.11.'de gösterilmiştir. Burada -pozitif lojik kabulü altında- bir NOR bağlantısı söz konusudur.Ne kadar çok sayıda giriş (E1,E2...) H seviyesine getirilirse, transistör de o derecede daha fazla doymaya sürülür ve buna bağlı olarak iletim konumundan tıkama konumuna geçiş süresi o oranda daha uzun olur.

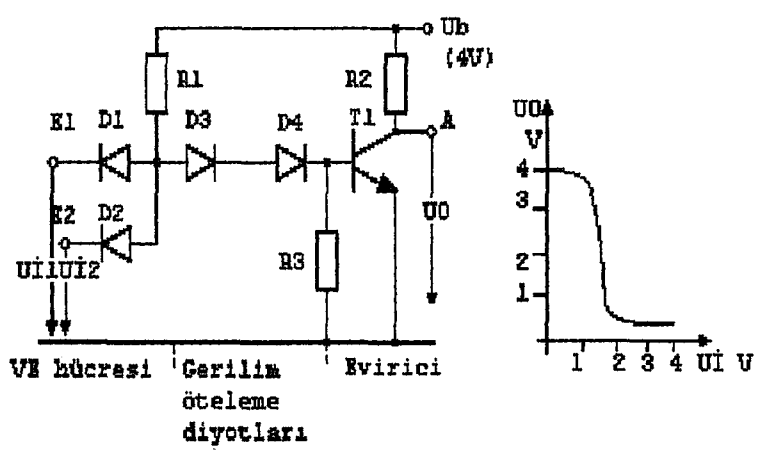


Şekil 2.11. RTL NOR bağlantısı

2.5.1.2 DTL (diyot-transistor lojiği)

Devrenin girişindeki diyotlar bir AND bağlantısı oluştururlar (Şekil 2.12). Bunların ardına bağlanan transistorla (evirici,inverter) birlikte bir NAND hücresi elde edilir. T1 transistorunun kesime girmesini sağlayabilmek üzere, D3 ve D4 diyotlarının kullanılması gerekli olmaktadır.

Örneğin, 1 girişine L seviyesi uygulanırsa, -D3 ve D4 diyotlarının bulunmaması halinde- T1 transistorunun bazı D1 diyodunun uçlarında düşen geçirme yönü geriliminde (yaklaşık olarak 0,6 V) olur. Buna karşılık, D3 ve D4 diyotlarının da bulunmaları halinde, Ui1<1,2 V durumunda T1 transistoru tıkanır; yani, T1'in iletime geçmemesi için, Ui1 geriliminin değeri en fazla 1V olabilir. L konumunda U0 çıkış gerilimi yaklaşık olarak 100...200mV değerleri arasında bulunur; uygulamada DTL kapıların ard arda bağlanacağı göz önünde tutuluursa, bu şekilde 0,9V'luk bir SL gürültü marjı elde edilmektedir. R3 direnci,devrenin konum değiştirmesi sırasında baz yükünün boşaltılması için gereklidir.



Şekil 2.12. DTL NAND kapısı ve geçiş karakteristiği

2.5.1.3 TTL (transistor-transistor lojiği)

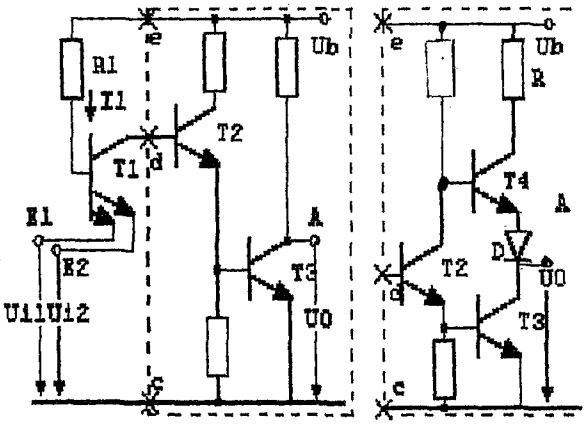
TTL ailesinin temel devresi NAND hücresi olarak çalışır (Şekil 2.13). DTL ailesine göre en önemli fark, DTL'deki AND bağlantısını sağlayan diyotların yerlerini burada çok emetörlü bir transistora bırakmış olmalarıdır.

En az bir girişin L potansiyelinde olması halinde, T1 transistorunun buna ilişkin baz-emetör yolu iletime geçer ve T1 doymaya girer. Küçük kolektör akımlarında bu transistorun kollektör potansiyeli, giriş geriliminden sadece küçük değerli kolektör-emetör doyma gerilimi (yaklaşık olarak 0,1V) kadar daha yüksektir. Bu nedenle T2'nin baz gerilimi, T2 transistorunu ve bununla da T3 transistorunu kesime sokacak kadar düşük değerli olur.

Buna karşılık, bütün girişlerin H potansiyelinde bulunmaları halinde, T1 "ters transistor" olarak çalışır; yani,baz-emetör yolları tıkanır, buna karşılık baz-kolektör yolu iletime geçer. I1 akımı T1'in baz-kolektör yolu üzerinden T2'nin bazına akar. Bu durumda her iki tranzistör de -T2 ve T3- iletime geçerler (doyma durumu).

Şekil 2.13-a çıkış devresinin alışılagelen biçimini göstermektedir: İki transistorlu çıkış (puşpul). D diyodu öteleme olarak çalışır, T2 ve T3 transistorları iletken durumda iken T4 transistorunun tamamen kesimde olmasını sağlar. Böylece, T3 transistorunun kollektör akımının tümü çıkıştan akar; dolayısıyla çıkışa on taneye kadar bu tür TTL kapı girişi bağlanarak sürülebilir.

Şekil 2.13-b'deki devre açık kollektör TTL hücresi olarak bağlanmış AND fonksiyonlarının gerçeklenmesi için kullanılabilir.

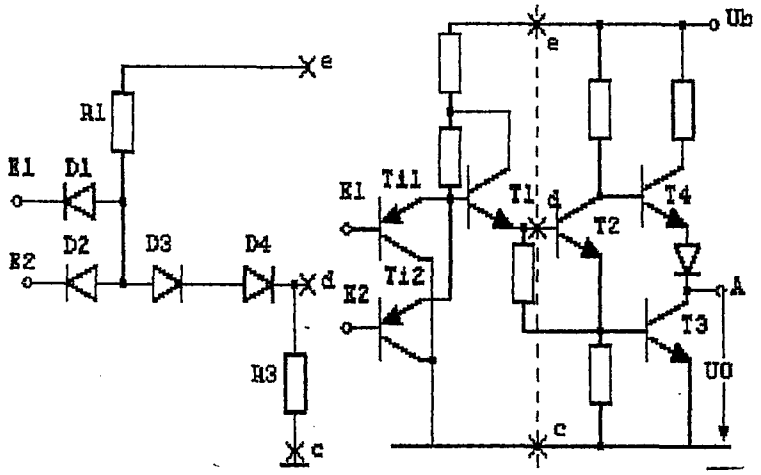


a) b)

Şekil 2.13. TTL NAND kapısı a)AND fonksiyonuna uygun çıkış devresi b)Alışagelen puşpul çıkış katı.

2.5.1.4 CTL (eşlenik transistor lojiği)

DTL ailesinin AND bağlantı diyotları, burada yerlerini PNP tipi Ti1 ve Ti2 transistorlarına bırakmışlardır. DTL devresinden CTL devresinin nasıl oluşturulduğu Şekil 2.14'de gösterilmiştir. DTL ailesinde diyotlarla gerçekleştirilen VE hücresi, burada şekil 2.14-b deki gibi iki PNP tranzistör yardımıyla oluşturulmuştur. Gerilim öteleme diyotları yerine Ti tranzistörü kullanılmıştır. Bu eleman T2'yi sürmek üzere kuvvetlendirilmiş olan bir akım üretir. RB baz direnci R3'ten daha küçük olabilir. Şekil 2.14-b'deki devrenin sol taraftaki bölümünün ardına iki transistorlu bir çıkış katı bağlanmıştır.



a) b)

Şekil 2.14. CTL NAND kapısı a) Bir DTL kapısının giriş devresi b) Çıkışında iki transistorlu puşpul devre bulunan CTL kapısı.

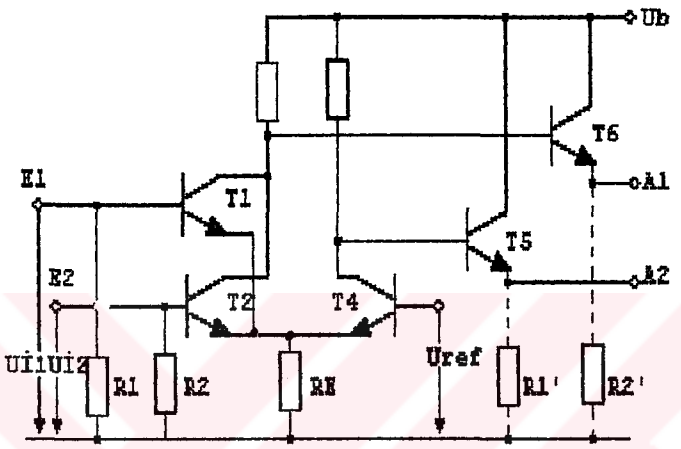
2.5.1.5 ECL (emetör bağlamalı lojik;CML=Current mode logic=Akım modu lojiği)

ECL ailesi için temel devre yapısı Şekil 2.15'de görülmektedir. RE emetör direnci, bu lojik ailesinin temel özelliklerini belirleyen karakteristik bir elemandır. Bunun yardımıyla transistorların doymaya kadar sürülmemeleri sağlanır. Doyma durumunda olduğu gibi baz bölgesinde ek yük taşıyıcıları birikmedikleri için, çok kısa iletime geçme ve kesime gitme süreleri elde edilir; yani birikme süresi ortadan kalkar.

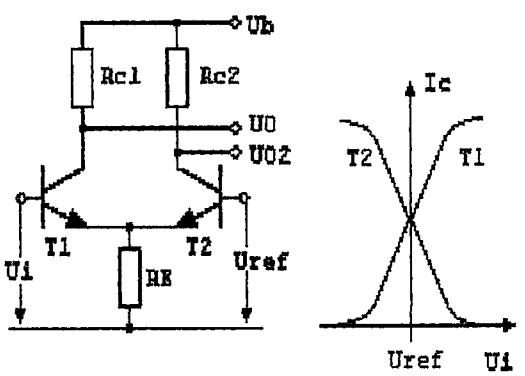
Sadece tek bir giriş tranzistörü göz önüne alınırsa, devrenin çalışması daha kolay izlenebilir (Şekil 2.16-a). Burada bir fark kuvvetlendiricisinin söz konusu olduğu görülmektedir. Şekil 2.16-b de,

Ui geriliminin referans geriliminden daha büyük veya daha küçük olması halinde kollektör akımlarının nasıl değişecekleri gösterilmiştir. Ui>Uref ise, T1 transistoru iletir; Uo1 bu durumda L seviyesinde ve Uo2 de H seviyesinde olur. Ui<Uref ise, T2 tranzistörü iletir; bu durumda ise Uo1 çıkışı H ve Uo2 çıkışı L seviyesine gelir.

Şekil 2.15'deki devre A1 çıkışı dikkate alınırsa bir NOR hücresi, A2 çıkışına göre de OR hücresi olarak çalışmaktadır.



Şekil 2.15. ECL hücresinin temel devresi. A1,A2 çıkışlarındaki yük dirençleri, bir sonraki kapıların R1've R2' giriş dirençleridir.



a) b)

Şekil 2.16. a) ECL bağlatısı için prensip şeması b)Kollektör akımlarının Ui/Uref oranına bağımlılığı.

2.5.1.6 I^2L (akım enjeksiyonu lojiği)

Akım enjeksiyonu lojiği bir dizi yarar sağlamaktadır:Küçük güç harcaması,işaret gecikmesinin küçük olması,düşük besleme gerilimi ve küçük kristal yüzeyi. Söz konusu özellikleri nedeniyle, bu teknik bazı noktalarda CMOS tekniği ile yarışabilmektedir.

Şekil 2.17 bu ailenin yapı prensibini göstermektedir. Az sayıda difüzyon işlemi ile çeşitli tabakalar ve bölgeler oluşturulmaktadır: Bir difüzyonla aynı kırmık (chip) üzerindeki tranzistörlerin karşılıklı yalıtımı, bir P-difüzyonu ile baz bölgesi ve enjeksiyon bölgesi, bir N+ difüzyonu ile de kollektör bölgesi oluşturulur.

Alışılagelen transistorlardakinin tersine,burada N+ bölgesi kollektör ve epitaksiyel N tabakası emetör olarak çalıştırılmaktadır. Bu difüzyon işlemleriyle iki tranzistörün oluştuğu fark edilebilir:Bir boyuna NPN transistor (1,2,3 bölgeleri) ve 5,3,2 bölgelerinden oluşan bir enine (lateral) PNP tranzistor.

Çalışma İlkesi:

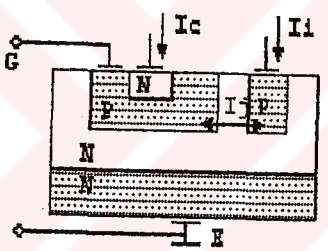
5 ile gösterilen P bölgesinden NPN transistorun bazına Ij enjeksiyon akımı akar. Bazın G ucu üzerinden E emetör ucuna kısa devre edilmesi halinde, Ij dışarıya doğru akar; NPN transistor kesime gider ve Ic kollektör akımı sıfır olur. Buna karşılık G girişi açıksa, yani Ig=0 ise, bazdan (2) emetöre (3) bir delik (hole) akımı akmak zorundadır. Böylece, PN jonksiyonu pozitif yönde kutuplanır ve emetörden (3) gelen elektronlar baz (2) üzerinden kollektöre (1) akabilirler. NPN transistor (3,2,1) iletken olur.

Buradan şu sonuç çıkmaktadır:G girişinden akım akmazsa C çıkışı akım iletmekte yahut tersi olmaktadır (G'den akım akarsa C akım iletmemektedir). Dolayısıyla,söz konusu düzen evirici (inverter) olarak çalışır.

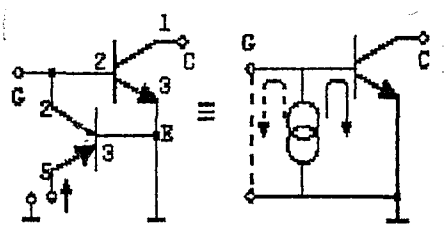
Eşdeğer Devre:

Yukarıda anlatılan çalışma ilkesinden yararlanılarak bir eşdeğer devre türetilmiştir (Şekil 2.18). Bu eşdeğer devrenin, NPN transistorun nasıl sürüldüğünün daha belirgin olarak fark edilmesini sağlayacağı açıktır. G ucu açık ise, Ij akımı NPN tranzistörün bazına doğru akar ve bunu iletken hale getirir. Tersine G ucu referans noktasına veya emetör ucuna kısa devre edilirse, NPN tranzistor tıkanır.

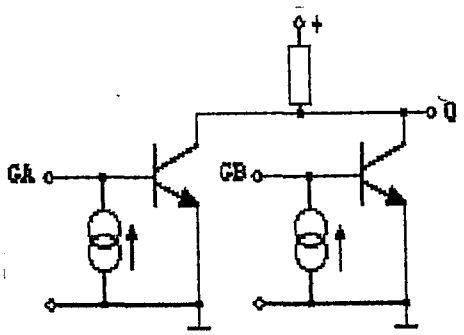
Bu eşdeğer devre, böyle bir evirici yardımıyla temel kapı devrelerinin nasıl kurulacağını da göstermektedir. Buna ilişkin bir örnek Şekil 2.19'da verilmiştir. Örnekte, iki eviriciden bir NOR kapısı yapılmıştır, bunun için her iki kollektör birbirine bağlanmıştır.



Şekil 2.17. I^2L yapısı.



Şekil 2.18. I^2L için eşdeğer devre.



GA	GB	Q
L	L	H
L	H	L
H	L	L
H	H	L

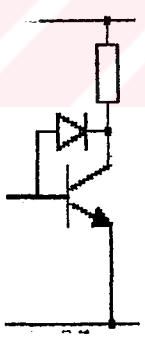
Şekil 2.19. İki I^2L için kurulan NOR devresi.

2.5.1.7 L-TTL ("Low-Power-TTL":Düşük güçlü TTL)

Bu seride,daha yüksek çalışma dirençleri ve bunlarla sağlanan daha küçük akımlar yardımıyla harcanan güç azaltılmıştır. (Tablo 2.7.)

2.5.1.8 S-TTL (Schottky-TTL)

Alışılagelen yarı iletken diyot yolları kesime sürülürken, birikme ve düşme süreleri nedeniyle dikkati çekecek ölçüde gecikmeler ortaya çıkar. Bu sakınca S-TTL serisinde önemli oranda giderilmiştir. S-TTL serisinde her bir transistorun kollektörü ce bazı arasında bir Schotty diyodu bağlanmıştır (Şekil 2.20).Bu diyot baz ve kollektör arasındaki gerilim farkını küçük tutar, yani aşırı doymaya girmeyi ve böylece PN jonksiyonunda yüksek bir azınlık taşıyıcıları yoğunluğu oluşmasını önler.



Şekil 2.20. Baz ve kollektör arasına bir schottky diyodu yerleştirilmesi.

2.5.1.9 LS-TTL ("Low-Power-Schottky-TTL":Düşük güçlü Schottky-TTL)

Tablo 2.7'den görüldüğü gibi TTL'den L-TTL ailesine geçildiğinde işaret gecikmeleri artmakta,yani devrenin hızı azalmaktadır.Schottky diyodu tekniği uygulanırsa, hem küçük güç harcıyan, hem de yüksek hızlı olan bir seri elde edilir.

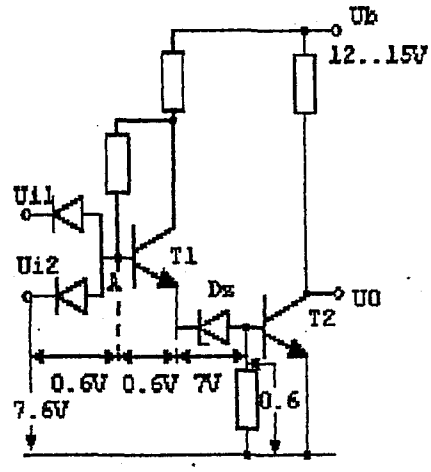
	Standart TTL	L-TTL	S-TTL	LS-TTL
İşaret vakış süresi (Gecikme süresi) [ns]	10	30	5..3	9..10
Çekilen güç [mw]	10	1	10..20	2

Tablo 2.7.Farklı TTL serilerinde işaret akış süresi ve çekilen güç.

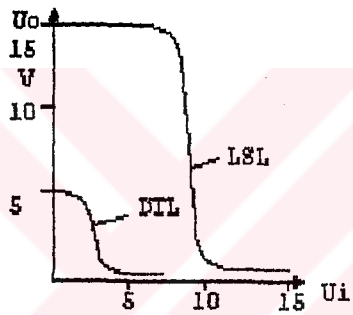
2.5.1.10 LSL (yavaş ve gürültü bağışıklığı büyük lojik; DTLZ=Z-diyotlu diyot-tranzistör lojiği)

Bu devrelerde AND bağlantısı ile çıkış devresi arasına bir Z diyodu yerleştirilmiştir (Şekil 2.21) Bu şekilde elde edilen etkiye (eşik geriliminin artırılması) daha önceki bölümde değinilmişti. Ancak, Z diyodunun yapıya katılması, besleme geriliminin artırılmasını zorunlu kılar. İşaret gecikmesi DTL lojiğine göre daha büyüktür.

Şekil 2.22'de gösterildiği gibi 7 V'luk bir Z-diyodunun kullanılması halinde, T2 transistorunun iletime geçmesi için her iki girişe de 7,6 V'tan daha büyük bir gerilim uygulanmalıdır. Yani eşik gerilimi yaklaşık 7,6 V'tur. Geçiş karakteristikleri açısından DTL ve LSL devrelerinin karşılaştırılması Şekil 2.22'de görülmektedir.



Şekil 2.21. LSL NAND kapısı



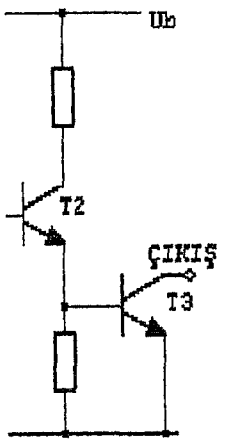
Şekil 2.22. DTL ve LSL ailelerinin geçiş karekteristiklerinin karşılaştırılması.

2.5.1.11 Bağlanmış Fonksiyonlar, Açık kollektör (wired-AND, open collector)

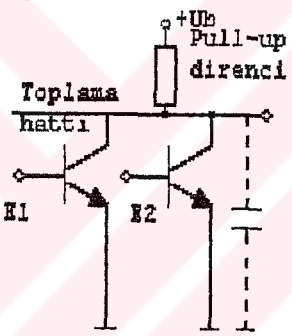
Şekil 2.23'de bir kapı çıkış devresi görülmektedir. Bu şekil 2.13-a'daki çerçevelenmiş devre parçasına karşı düşmektedir; sadece T3 transistorunun kollektörü bir kollektör direnci üzerinden besleme hattına bağlanmamıştır. Bu nedenle,böyle düzenler açık kollektörlü devreler olarak anılırlar.

Bu türden çok sayıda kapı çıkış taraflarından paralel bağlanabilir ve böylece ortak bir kolektör direnci üzerinden

beslenebilir.Bunun yapılmasıyla her iki kapının çıkışının birlikte bir AND fonksiyonu sağlayacak şekilde bağlanması gerçekleştirilmiş olur.



Şekil 2.23.Bağlanmış (açık kollektör) AND çıkışı



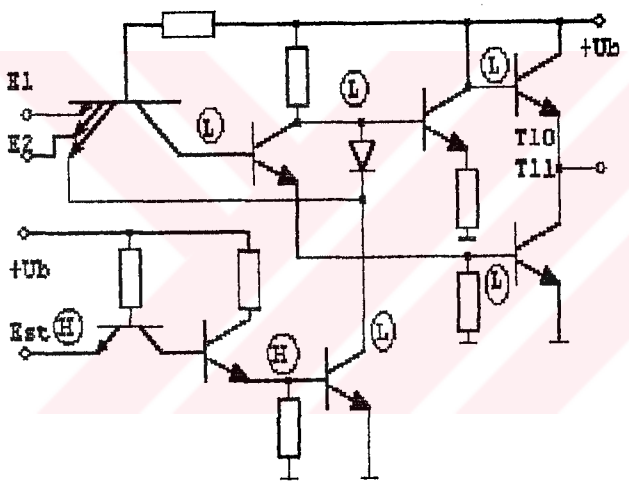
Şekil 2.24.Açık kollektörlü çok sayıda çıkışın birbirine bağlanması

2.5.1.12 Üç konumlu devre

Şekil 2.24'te gösterilen devrenin şu sakıncası bulunmaktadır:Bütün tranzsistorların tıkanması durumunda, A ucunun potansiyelinin besleme geriliminin tam değerine ulaşması, yani hat kapasitesinin "pull-up" (yukarıya çekme) direnci üzerinden dolması için uzunca bir süre geçer. Transistorların yüklenme durumu da göz önüne alındığı için, bu direnç çok küçük seçilememektedir.

Bu nedenle,üç konumlu ("Tristate") devre geliştirilmiştir. Devre üç konuma sahiptir, çıkıştan alışılagelen biçimde H ve L seviyeleri alınabilir, ayrıca her iki çıkış transistoru da aynı anda kesime sürülebilmektedir. Her defasında sadece bir kapının çıkışı toplama hattına bağlanırsa diğer kapıların farklı işaret seviyesine sahip olmaları halinde oluşabilecek kısa devrelerin önüne geçilmiş olur.

Şekil 2.25'in üst kısmında bir TTL NAND kapısı görülmektedir. Alt taraftaki ek devre üzerinden puşpul çıkış transistorların her ikisi de kesime sürülebilmektedir.



Şekil 2.25. Üç konumlu lojik ile NAND kapısı

2.5.2 Unipolar lojik aileleri

Unipolar lojik ailelerini şu alt gruplara ayırmak gerekir : P kanallı MOS lojiği, N-kanallı MOS lojiği ve CMOS lojiği (eşlenik MOS lojiği).

MOS teknolojisi daha kolay gerçeklenmektedir:Difüzyon işlemlerinin sayısı daha az olmakta ve baz tabakasına gereksinme duyulmamaktadır (az ve kritik kalınlık). Ayrıca, enerji harcaması daha azdır ve bir kırmık üzerine tümleştirmede gerekli olan yüzey daha küçüktür.

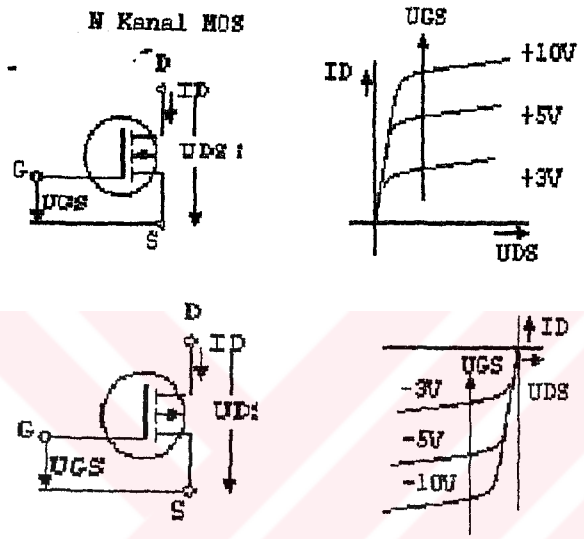
MOS transistorları çok yüksek bir giriş direncine sahiptirler,giriş ve çıkış arasında galvanik bağlaşma bulunmamaktadır.

Akım iletimi için sadece çoğunluk taşıyıcıları (elektronlar ve delikler) görev yaparlar.Kontrol,yarıiletken bölgedeki (bu bölge kanal olarak isimlendirilir) çoğunluk taşıyıcılarının sayısını etkiliyen bir elektriksel alan üzerinden sağlanmaktadır.MOS-FET'dekontrol elektrodu ve kanal,bir metal oksidi izolasyonu ile birbirinden ayrılmışlardır. LSI devrelerde, MOS tekniği tercih edilerek kullanılmaktadır.

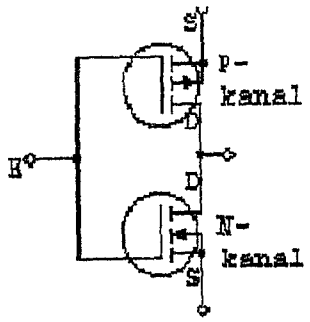
Dijital devrelerde daha çok kanal oluşturmalı tipten MOSFET'ler ("enhancement-FET",Şekil 2.26) kullanılırlar. Bu tipte,kanaldan bir akım (Id savak akımı)akabilmesi için, Ugs geçit-kaynak gerilimi belirli bir değeri (Ut eşik gerilimi:"Threhold-voltage")aşmak zorundadır.

CMOS lojik ailesinde (CMOS="Complementary symmetrical MOS") P ve N-kanallı elemanlar karışık olarak bir arada kullanılırlar.Bu yol izlenerek,her lojik eleman başka devre elemanları (örneğin;dirençler,diyotlar)kullanılmaksızın gerçekleştirilebilir. Örnek olarak alınan bir evirici devresi (NOT fonksiyonu),Şekil2.27de gösterilmiştir.

Şekil2.27 de verilen CMOS eviricide,belirli bir giriş konumunda transistorlardan biri daima tıkalı, diğeri ise iletimdedir. Yani göze çarpacak derecede bir güç harcaması sadece bir lojik konumdan diğerine geçerken ortaya çıkmaktadır.Statik çalışmada güç harcaması yaklaşık olarak 10nW'tır. Frekansın artırılması halinde,kapasitelerin dolup boşalmaları sırasındaki kayıplar nedeniyle bu güç artar.



Şekil 2.26. Bir P kanallı ve bir N kanallı MOS transistötü için devere sembolü ve çıkış karakteristiği.



Şekil 2.27.Bir CMOS eviricinin yapısı

CMOS lojiğinin yararları giriş direncinin yüksek olması (10^{10} ile 10^{10} Ohm değerleri arasında), düşük akım çekmesi,besleme

geriliminin değişim bölgesinin büyük olması, gürültü marjının büyük olması ve besleme geriliminin değerinin artırılmasıyla büyümesi,çalışma sıcaklığı bölgesinin geniş olması ve devrelerin büyük bir çıkış yelpazesine ("fan-out") sahip bulunmalarıdır. Sakıncalı yanları ise,iletime geçme ve kesime gitme sürelerinin uzun olmaları ve diğer ailelere göre daha yüksek besleme gerilimlerine gereksinme göstermeleridir.

2.5.3. Lojik Ailelerinin Karşılaştırılması

Taşınabilir aygıtlarda,çekilen güç önemli bir rol oynar.Bu tür düzenlerde,iletime geçme-kesime gitme süreleri için özel şartlar ileri sürülmemişse,L-TTL veya CMOS aileleri tercih edilirler. Aksi halde ise S-TTL ailesi kullanılır.

Endüstriyel uygulamalarda genellikle parazitik etkilerin de hesaba katılmaları gerekir. Bunun için LSL ailesi geliştirilmiştir. Ancak,TTL ve CMOS devre elemanları da çok geniş bir uygulama alanı içinde kullanılmaktadır.

Hesaplayıcılarda, bilgisayarlarda çoğunlukla saat veya anahtarlama frekansı,yani devrenin hızı baskın olur.Bu şartlar altında,ECL tekniği ile birlikte yaklaşık olarak 3ns değerindeki iletime geçme-kesime gitme süresiyle S-TTL ailesi de ön planda gelmektedir.

Besleme düzeninin basit tutulması isteniyorsa,devre kalitesini kaybetmeksizin besleme gerilimi geniş sınırlar içerisinde değişebildiğinden, CMOS devre elemanları kullanılmalıdır.

2.5.4. Önemli Kavramlar

2.5.4.1 Fan-out (çıkış yelpazesi,çıkış yüklenme faktörü,çıkış dallanması)

Bu deyim,lojik düzenin çalışması tehlikeye sokulmaksızın, bir çıkış tarafından aynı anda sürülebilecek olan girişlerin maksimum sayısını verir.Burada,normal olarak en uygun olmayan çalışma durumunun bulunduğu (en kötü hal),örneğin bağlanan girişlerin akımlarının üst tolerans sınırına karşı düşen değerlerde oldukları kabul edilir.

2.5.4.2 Gürültü marjı

Örnek olarak TTL serisi için geçerli olan şu değerler bulunmaktadır:

1. Çıkışta H seviyesinin bulunabilmesi için Ui<0,8V olmalıdır.
2. Çıkışta L seviyesinin bulunabilmesi için Ui>2V olmalıdır.
3. Normal bir TTL kapısında,L seviyesinde çıkışta bulunan gerilim (tam yük altında) 0,4V'tan daha küçüktür.
4. Benzer şekilde, H seviyesinde çıkışta bulunan gerilim 2,4V'tan daha büyüktür.

Böylece -bir sonraki giriş katına göre- güvenilirlik marjı(gürültü marjı):

SL (çıkışta L seviyesi varken) sınır durumda: 0.4V'dan 0.8V'a=0.4V (pratikte yaklaşık olarak 0.2V'dan 0.8V'a=0.6V);

SH (çıkışta H seviyesi varken 2.4V'dan 2.0V'a=0.4V (pratikte yaklaşık olarak 2.8V'dan 2.0V'a=0.8V)

3. ENTEGRE TEST DEVRESİ VE KULLANIMI

TTL entegre test devresinin amacı, öncelikle TTL mantık ailesine sahip entegre devreler olmak üzere CMOS entegre devreler de dahil tüm 0-5V arası çalışabilen lojik entegre devreleri test etmektir. Bu test fonksiyonel bir testdir. Test edilecek entegre devrenin TTL ortamda gerekli fonksiyonları doğru olarak yerine getirip getirmediğine bakılır. Entegre devrenin çıkışı 74LS365 TTL Buffer devreler ile alınıp porta ve bilgisayara gönderilir. Bu nedenle test edilecek entegre devrenin TTL entegre devreleri sürebilecek yetenekte olması gerekir. Yine test edilecek entegre devre, girişine "1" seviyesi için 5V, "0" seviyesi için 0V uygulandığından bu girişleri kabul edebilecek yetenekte olması gerekir.

-Sistemle ilgili seviyeler:

Sistem tarafından, test edilecek entegre devrenin girişine uygulanan seviyeler:

"1" seviyesi için 4,8V

"0" seviyesi için 0,2V

Test edilecek entegre devrenin kabul edilebilir çıkış seviyeleri:

"1" seviyesi için 2.4V...5V

"0" seviyesi için 0V...0.4

-Test edilecek TTL entegre devre ile ilgili seviyeler:

Test edilecek TTL entegrenin kabul edebileceği giriş seviyeleri:

"1" seviyesi için 2V...5V

"0" seviyesi için 0V...0,8V

Test edilecek TTL entegrenin çıkış seviyeleri:

"1" seviyesi için 2,4V...5V

"0" seviyesi için 0V...0,4V

-Test edilecek CMOS entegre devre ile ilgili seviyeler:

Test edilecek CMOS entegrenin kabul edebileceği giriş seviyeleri (0..5V çalışma aralığı için):

"1" seviyesi için 3,5V..5V

"0" seviyesi için 0V..1,5V

Test edilecek CMOS entegrenin çıkış seviyeleri (0..5V çalışma aralığı için):

"1" seviyesi için 4,4V..5V

"0" seviyesi için 0V..0,8V

3.1. Sistemin Ana Çalışma Mantığı:

Test edilecek entegre devre ile ilgili değerler ve giriş çıkış fonksiyonları daha önce tanımlanmış ve dosyalanmıştır. Bu entegre devre test soketine yerleştirildikten ve ismi girildikten sonra bigisayar tarafından, dosyadan tanımlanmış besleme ve bağlantı kontrol bilgileri alınır ve giriş çıkış bağlantıları yapılır. Yine bilgisayar tarafından gerekli giriş bilgisi entegre devreye gönderilir ve kısa bir süre sonra entegre devrenin çıkış bilgisi bilgisayar tarafından okunup kendi dosyasındaki datalar ile karşılaştırır. Doğru ise diğer giriş/çıkış bilgilerinin testine geçilir. Hepsi doğru ise "entegre sağlam" mesajı verilerek test sona erer. Herhangi bir yerde karşılaştırmalarda farklılık çıkarsa "entegre hatalı" mesajı verilir ve test sona erer.

3.2. Sistemin Özellikleri

Sistemin özellikleri aşağıda sıralanmıştır:

1. Program açıldığında sistem kendi kendini test eder, hata varsa mesaj vererek uyarır.

2. İsmi bilinen entegre devre test edildiği gibi ismi bilinmeyen entegre devreler de test edilebilmektedir. Entegre devre daha önce tanımlanmış ve sağlam ise ismi ekranda gösterilmektedir. Aksi halde ekranda "YOK" mesajı çıkar.

3. Sadece TTL değil 0-5V seviyelerinde çalışabilen tüm entegre devrelerin testi yapılabilir.

4. Daha önce tanımlanmamış entegre devrelerin tanımı kullanıcı tarafından kolaylıkla yapılabilmektedir. Bu özellik sisteme esneklik vermiştir.

Sistemin standart özellikleri aşağıda sıralanmıştır:

1. 36 fonksiyona sahip 138 adet TTL ve CMOS entegre devrenin testi yapılabilir.(Tanımlanan entegre devrelerin pek çoğu ortak fonksiyona sahiptir.)

2.500 fonksiyon ve 10000 entegre devreye kadar yeni tanım yapılabilir. Her bir fonksiyon için 50 giriş/çıkış bilgisi (bağlantı bilgisi dahil) tanımlanbilir.

3.Test 0V-5V gerilim değerleri arasında yapılır.

3.3. Sistemin Kullanımı

Sistem açılmadan önce bilgisayar ve devrenin bağlantı kabloları kontrol edilir. Bilgisayar ve test cihazı açılır. "TEST" yazılarak program çalıştırılır. Program çalıştırılır çalıştırılmaz sistem kendi kendini test eder. Test işlemi olumsuz ise program kırılır. Test işlemi olumlu ise ekrana aşağıda gösterildiği gibi Ana Menü çıkar:

ANA MENÜ

1.DOĞRULUK TESTİ

2.TANIMA TESTİ

3.LİSTE

4.TANIMLAMA

5.AÇIKLAMA

6.ÇIKIŞ

-1 nolu Doğruluk Testi seçildiğinde ekrana "İsim" gelir. İsim girilmeden önce entegre sokete yerleştirilir. Sonra isim girilip "enter" tuşuna basılır. Test başlar. Kısa bir süre sonra gönderilen alınan datalar ve testin sonucu ekrana çıkar. Tekrar aynı cins entegre devre test edilecek ise entegre yerleştirildikten sonra "enter" tuşuna basmak yeterlidir. Farklı entegre test edilecek ise escape tuşu ile çıkıp yeni entegrenin ismi yazılmalı ve sokete yerleştirildikten sonra enter tuşuna basılmalıdır. Çıkış için önce escape tuşuna sonra q ve enter tuşlarına basılmalıdır.

-2 nolu Tanıma Testi seçilmeden önce entegre sokete yerleştirilir. 2'ye basıldıktan sonra entegre devrenin olabilecek isimleri görünür. Eğer ekranda "YOK" görünür ise entegre tanımlanmamıştır veya arızalıdır. Tekrar test için enter "tuşuna", çıkış için "escape" tuşuna basılmalıdır.

-3 nolu Liste seçildiğinde daha önce tanımlanmış entegre devre isimleri görünür. "Page Up" tuşu ile listenin ilerisi, "Page Down" tuşu ile listenin gerisi görüntülenir. Çıkış için "Escape" tuşuna basılır.

-4 nolu Tanımlama seçildiğinde tanımlanmamış bir entegre devrenin tanımı yapılır. Ekrana Tanımlama Menu'sü gelir:

TANIMLAMA

1.İNCELEME VE DÜZELTME

2.YENİ TANIMLAMA

3.ANA MENÜYE DÖNÜŞ

"2" yazılarak "YENİ TANIMLAMA" seçilirse yeni entegre devre tanımı yapılabilr. Tanım bitdikten sonra kaydedilir. Tanım şu şekilde yapılır:

Sırayla "İsim","Bacak Sayısı","Besleme Vcc" ve "Gnd" girilir. 20 adet isim girilir. Bacak sayısı 14,16,20,24 olabilir. Besleme Vcc girişinin kaçıncı bacakta olduğu girilir. Gnd girişinin kaçıncı bacakta olduğu girilir. Sırasıyla bağlantı kotrol bilgileri girilir. Giriş olanlar için "1", Çıkış olanlar için "0"dır. Sonraki adımda Giriş/Çıkış bilgileri girilir. Herhangi bir anda çıkmak istenildiğinde "Q" yazılarak çıkılır. Heradım için alt satırda açıklamalar yapılır. Tanımlama yapılırken ekranın görüntüsü aşağıdaki örnekte olduğu gibidir:

İsim: 7400 74LS00 74S00 74HC00 74HCT00 7403
74LS03 74S03 74HC03 74HCT03 7426 7437
74LS37 7438 74LS38 74S38 74LS132 74F00

Bacak Sayısı:

14

Besleme Vcc:

14

GND:

7

BACAK DURUMU

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
BACAK GİRİŞ	BACAK GİRİŞ	BACAK ÇIKIŞ	BACAK GİRİŞ	BACAK GİRİŞ	BACAK ÇIKIŞ	BACAK	BACAK ÇIKIŞ	BACAK GİRİŞ	BACAK GİRİŞ
1	1	0	1	1	0	GND	0	1	1

11.	12.	13.	14.
BACAK ÇIKIŞ	BACAK GİRİŞ	BACAK GİRİŞ	BACAK
0	1	1	Vcc

"1" yazılarak "İNCELEME VE DÜZELTME" seçilirse tanımlanmış bir entegre devrenin tanım parametreleri ve giriş/çıkış bilgileri incelenebilir veya üzerinde değişiklik yapılabilr. İncelenecek entegre devrenin ismi girilir. Çıkmak isteniyorsa "Q" girilir. "Page Up" tuşu diğer entegre devre tanımlarına ulaşmak, "Page Down" geri dönmek için 6 kullanılır. "Yukarı Ok" tuşu giriş/çıkış bilgilerine ulaşmak, "Aşağı Ok" tuşu geri dönmek için kullanılır. Düzeltme yapmak için "F1" tuşuna basılır ve yukarıda anlatıldığı gibi tanımlar girilir. "F2" entegre devre tanımının tamamını silmek için kullanılır. "ESC" çıkmak için kullanılır.

3.3.1. Entegre Devre Tanımlama Yöntemleri:

Bazı entegre devreler farklı elktronik yapıda olmalarına rağmen aynı fonksiyona sahiptir. Bu entegre devreler için aynı fonksiyon tanımları yapılır. Birden fazla entegre ismi verilir.

Multivibratör gibi ek devre gerektiren entegre devreler tanımlanmamalıdır. 24 adetden fazla bacağa sahip ve besleme yapıları farklı entegre devreler için tanım yapılamaz.

Örnek olarak bazı entegre fonksiyonlarının tanımları verilmiştir:

Örnek 1. 7400 2 Girişli NAND Kapısı:

Aynı fonksiyona sahip 4 kapıdan oluşur.

Bağlantı şeması şu şekildedir:

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
A1	B1	Y1	A2	B2	Y2	GND	Y3	A3	B3	Y4	A4	B4	Vcc
G	G	Ç	G	G	Ç		Ç	G	G	Ç	G	G	

(G:Giriş,Ç:Çıkış)

Doğruluk tablosu katalogda şu şekilde verilmiştir:

Girişler		Çıkışlar
A	B	Y
L	L	H
L	H	H
H	L	H
H	H	L

Sisteme girilen değerler ise şu şekildedir:

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
Bağlantı Durumu: (1 girişi, 0 çıkışı temsill eder.)													
1	1	0	1	1	0	GND	0	1	1	0	1	1	Vcc
1. Giriş/Çıkış Durumu:													
0	0	1	0	0	1	GND	1	0	0	1	0	0	Vcc
2. Giriş/Çıkış Durumu:													
0	1	1	0	1	1	GND	1	0	1	1	0	1	Vcc
3. Giriş/Çıkış Durumu:													
1	0	1	1	0	1	GND	1	1	0	1	1	0	Vcc
4. Giriş/Çıkış Durumu:													
1	1	0	1	1	0	GND	0	1	1	0	1	1	Vcc

Örnek 2. 7430 8 Girişli NAND Kapısı:

8 girişli bu kapının giriş kombinasyonunu bulalım.

$GK=2^n=2^8=256$

Buna bağlı olarak 256 adet giriş/çıkış bilgisi girilmelidir.Bu ve benzeri çok girişli kapıların giriş kombinasyonu çok yüksek olduğu için sadece hassas giriş bilgisi girilmesi y NAND kapısı için, girişlerin 1 olması durumunda çıkışın 0 olması nedeniyle girişleri 1 ağırlıklı olan giriş/çıkış bilgileri tanımlanmalıdır. Çünkü girişlerden biri 0, diğerleri 1 ise sadece 0'ın 1 olarak değişmesi sonucu değiştirmeye yetecektir.

7430 Bağlantı Şeması:

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
A	B	C	D	E	F	GND	Y	NC	NC	G	H	NC	Vcc
G	G	G	G	G	G		Ç			G	G		

(G:Giriş,Ç:Çıkış,NC:kullanılmıyor)

7430 Doğruluk Tablosu:

GİRİŞLER	ÇIKIŞLAR
A'dan H'ya	Y
Tüm girşler H	L
1 ya da daha fazla giriş L	H

Sisteme Girilen Değerler:

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
Bağlantı Durumu: (1 girişi, 0 çıkışı temsil eder.)													
1	1	1	1	1	1	GND	0	1	1	1	1	1	Vcc
1. Giriş/Çıkış Durumu:													
0	0	0	0	0	0	GND	1	0	0	0	0	0	Vcc
2. Giriş/Çıkış Durumu:													
1	1	1	1	1	1	GND	1	1	1	1	0	1	Vcc
3. Giriş/Çıkış Durumu:													
1	1	1	1	1	1	GND	1	1	1	0	1	1	Vcc
4. Giriş/Çıkış Durumu:													
1	1	1	1	1	0	GND	1	1	1	1	1	1	Vcc

5. Giriş/Çıkış Durumu:

1	1	1	1	0	1	GND	1	1	1	1	1	1	Vcc

6. Giriş/Çıkış Durumu:

1	1	1	0	1	1	GND	1	1	1	1	1	1	Vcc

7. Giriş/Çıkış Durumu:

1	1	0	1	1	1	GND	1	1	1	1	1	1	Vcc

8. Giriş/Çıkış Durumu:

1	0	1	1	1	1	GND	1	1	1	1	1	1	Vcc

9. Giriş/Çıkış Durumu:

0	1	1	1	1	1	GND	1	1	1	1	1	1	Vcc

10. Giriş/Çıkış Durumu:

1	1	1	1	1	1	GND	0	1	1	1	1	1	Vcc

3.Örnek: 74164 Seri Giriş/Paralel Çıkış Shift Register:

74164'de olduğu gibi clock girişi içeren entegrelerin tanımı, clock işareti peşpeşe "1" ve "0" arasında değiştirilerek yapılır.

Bağlantı Şeması:

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
A	B	Q_A	Q_B	Q_C	Q_D	GND	CLO.	CLE.	Q_E	Q_F	Q_G	Q_H	Vcc
G	G	Ç	Ç	Ç	Ç		G	G	Ç	Ç	Ç	Ç	

(G:Giriş,Ç:Çıkış)

Doğruluk Tablosu:

GİRİŞLER				ÇIKIŞLAR							
CLEAR	CLOCK	A	B	Q_A	Q_B	Q_C	Q_D	Q_E	Q_F	Q_G	Q_H
L	X	X	X	L	L		...				L
H	L	X	X	Q_{A0}	Q_{B0}		...				Q_{H0}
H	l	H	H	H	Q_{An}		...				Q_{Gn}
H	l	L	X	L	Q_{An}		...				Q_{Gn}
H	l	X	L	L	Q_{An}		...				Q_{Gn}

Sisteme Girilen Değerler:

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
Bacak Durumu: (1 girişi, 0 çıkışı temsil eder.)													
1	1	0	0	0	0	GND	1	1	0	0	0	0	0
1. Giriş/Çıkış Durumu:													
0	0	0	0	0	0	GND	0	0	0	0	0	0	Vcc
2. Giriş/Çıkış Durumu:													
1	1	0	0	0	0	GND	0	0	0	0	0	0	Vcc
3. Giriş/Çıkış Durumu:													
0	0	0	0	0	0	GND	1	0	0	0	0	0	Vcc
4. Giriş/Çıkış Durumu:													
1	1	0	0	0	0	GND	1	0	0	0	0	0	Vcc
5. Giriş/Çıkış Durumu:													
1	1	1	0	0	0	GND	1	1	0	0	0	0	Vcc
6. Giriş/Çıkış Durumu:													
1	1	1	0	0	0	GND	0	1	0	0	0	0	Vcc
7. Giriş/Çıkış Durumu:													
0	1	1	0	0	0	GND	0	1	0	0	0	0	Vcc
8. Giriş/Çıkış Durumu:													
0	1	0	1	0	0	GND	1	1	0	0	0	0	Vcc
9. Giriş/Çıkış Durumu:													
0	1	0	1	0	0	GND	0	1	0	0	0	0	Vcc
10. Giriş/Çıkış Durumu:													
1	0	0	1	0	0	GND	0	1	0	0	0	0	Vcc
11. Giriş/Çıkış Durumu:													
1	0	0	0	1	0	GND	1	1	0	0	0	0	Vcc
12. Giriş/Çıkış Durumu:													
1	0	0	0	1	0	GND	0	1	0	0	0	0	Vcc
13. Giriş/Çıkış Durumu:													
0	0	0	0	1	0	GND	0	1	0	0	0	0	Vcc
14. Giriş/Çıkış Durumu:													
0	0	0	0	0	1	GND	1	1	0	0	0	0	Vcc
15. Giriş/Çıkış Durumu:													
0	0	0	0	0	1	GND	0	1	0	0	0	0	Vcc
16. Giriş/Çıkış Durumu:													
0	0	0	0	0	1	GND	0	1	0	0	0	0	Vcc
17. Giriş/Çıkış Durumu:													
0	0	0	0	0	0	GND	1	1	1	0	0	0	Vcc
18. Giriş/Çıkış Durumu:													
0	0	0	0	0	0	GND	0	1	1	0	0	0	Vcc
19. Giriş/Çıkış Durumu:													
0	0	0	0	0	0	GND	0	1	1	0	0	0	Vcc
20. Giriş/Çıkış Durumu:													
0	0	0	0	0	0	GND	1	1	0	1	0	0	Vcc

21. Giriş/Çıkış Durumu:

0	0	0	0	0	0	GND	0	1	0	1	0	0	Vcc

22. Giriş/Çıkış Durumu:

0	0	0	0	0	0	GND	0	1	0	1	0	0	Vcc

22. Giriş/Çıkış Durumu:

0	0	0	0	0	0	GND	1	1	0	0	1	0	Vcc

23. Giriş/Çıkış Durumu:

0	0	0	0	0	0	GND	0	1	0	0	1	0	Vcc

24. Giriş/Çıkış Durumu:

0	0	0	0	0	0	GND	0	1	0	0	1	0	Vcc

25. Giriş/Çıkış Durumu:

0	0	0	0	0	0	GND	1	1	0	0	0	1	Vcc

26. Giriş/Çıkış Durumu:

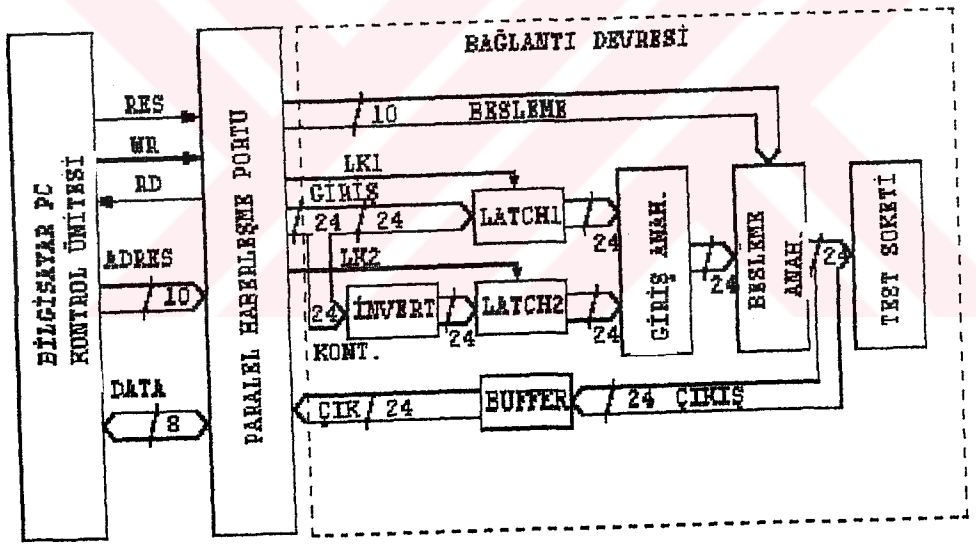
0	0	0	0	0	0	GND	0	1	0	0	0	1	Vcc

4. TTL ENTEGRE TEST DEVRESİ DONANIMI

Sistem donanımı genel olarak 3 üniteden oluşmuştur:

1. Kontrol ünitesi,
2. Paralel haberleşme devresi,
3. Bağlantı devresi.

Sistemin genel yapısı şekil 4.1'de gösterilmiştir.



Şekil 4.1. TTL entegre test devresinin genel yapısı

4.1. Kontrol Ünitesi:

Yazılımın ve entegre devre tanım kütüphanesinin (veri dosyasının) oluşturulduğu, sistemin tüm kontrolunu sağlayan kontrol ünitesi 386DX-40 modeline sahip bir kişisel bilgisayardır.

Bu bilgisayarın özellikleri şunlardır:

İşlemci:386DX-40

Hafıza:2MB RAM

Hard Disk:130MB

Monitör:SVGA Renkli monitör

Sistemin çalışabileceği bilgisayarın optimum özellikleri:

İşlemci:8086

Hafıza:2MB

Hard Disk:10MB

4.2. Paralel Haberleşme Devresi:

Paralel haberleşme devresi, bilgisayar ile bağlantı devresi arasındaki veri iletişimini sağlayan bir devredir. Bu devre üç adet 8255 PPI entegre devresinden oluşmuştur ve 72 bitlik bir kapasiteye sahiptir. (4.2.2'de ayrıntılı bir şekilde açıklanmıştır.)

4.2.1. 8255 Paralel Giriş/Çıkış Elemanı:

8255 paralel giriş/çıkış ünitesi herbiri 8 adet hatda sahip 3 giriş/çıkış kapısı içermektedir.

Bu üç kapı iki gruba ayrılmıştır. Birinci kapı grubu A kapısının tüm hatlarını (PA0..PA7) ve C kapısının PC0,PC1,PC2,PC3 hatlarını kapsar. B kapısının tüm hatları (PB0..PB7) ile C kapısının PC0,PC1,PC2,PC3 hatları da ikinci kapı grubunu oluşturur.

8255 entegre devresi şu fonksiyon bloklarından oluşmaktadır: Veri ara belleği, okuma/yazma kontrol mantığı, kontrol durum saklayıcıları, A ve B kapı gruplarının kontrolu ve her kapının sürücüsü.

8255 entegre devresi kapı hatlarından başka veriyolu bağlantısı (D0..D7), eleman seçme ucu (CS), resetleme ucu (reset), adres hatları (A0,A1), okuma/yazma (RD/WR) kontrol uçları bulunmaktadır.

Reset bilgisinin alınmasından sonra, 8255 entegre devresinin bütün uçları giriş olarak tanımlanır. 8255 4 byte'lık bir bellek bölgesi kapsar.

Eleman içerisinde iki saklayıcı bulunmaktadır. Bunlar kontrol saklayıcısı ile durum saklayıcısıdır.

1. ve 2. kapı gruplarının çalışabileceği üç farklı çalışma türü bulunmaktadır. A ve B kapılarının çalışma türleri biribirlerinden farklı olabilirken, C kapısının yarıları ait oldukları kapı grubunun çalışma türüne uyarlar. Kontrol saklayıcıları 1. ve 2. modlar türlerinde veri iletişiminin durumunu verir.

Mod 0 (Standart giriş/çıkış): Kullanıcıya üç adet 8 bitlik giriş/çıkış kanalı verilir. Her kanal 8 bitlik çıkış veya 8 bitlik giriş olarak programlanabilir. C kapısı ise 4 bitlik iki parçaya ayrılabilir ve bunlar birbirinden bağımsız olarak giriş veya çıkış olarak kullanılabilir.

Mod 1 (El sıkışmalı giriş/çıkış): El sıkışmalı arabirimli cihazlarla bağlantı için iki kapı grubu bulunmaktadır. 1. kapı grubuna ait olan PC4..PC7 hatları A kapısının el sıkışma hatları olarak çalışırlar. Benzer şekilde PC0..PC3 de B kapısının el sıkışma hatlarıdır. Bir kapı grubu üzerinden ya sadece giriş, ya da sadece çıkış yapılabilir.

Mod 2 (İki yönlü yol): 1. kapı grubu iki yönlü ve el sıkışmalı arabirim olarak çalıştırılabilir. Aynı anda 2. kanal grubu da, 0. veya 1. çalışma türünde işletilebilir.

Kontrol saklayıcısında tutulan kontrol sözcüğünün yapısı tablo 4.1'de gösterilmiştir.

BİT	ANLAMI			
D0	1.Grup	C kapısı(alt yarı)	1:Giriş	0:Çıkış
D1		B kapısı	1:Giriş	0:Çıkış
D2		Mod türü	1:MOD 1	0:MOD 0
D3	2.Grup	C kapısı(üst yarı)	1:Giriş	0:Çıkış
D4		A kapısı	1:Giriş	0:Çıkış
D5 D6		Mod tür	00:MOD0 , 01:MOD1 , 1X:MOD2	
D7		Kontrol Formatı	1:Aktif	

Tablo 4.1.Kontrol Sözcük Yapısı

4.2.2. 72 Bit Paralel Haberleşme Devresi

Paralel haberleşme devresi 3 adet 8255 PPI entegre devresinden oluşmuştur. Her bir 8255 entegre devresinin giriş/çıkış uçları port ile temsil edilir. Buna göre Şekil 4.2'de gösterilen devre üç adet portdan oluşur.

Bilgisayar programı çalıştırılmadan önce portların tamamı giriş durumundadır. Bu paralel haberleşme devresi ve bağlantı devresine zarar vermez. Portların giriş/çıkış ayarları, programın başlangıç bölümünde yapılır.

1.port, test edilecek entegre devrenin çıkış bilgisini okuması için ayrılmıştır. Port 1 kapıları giriş olarak progamramlanır.

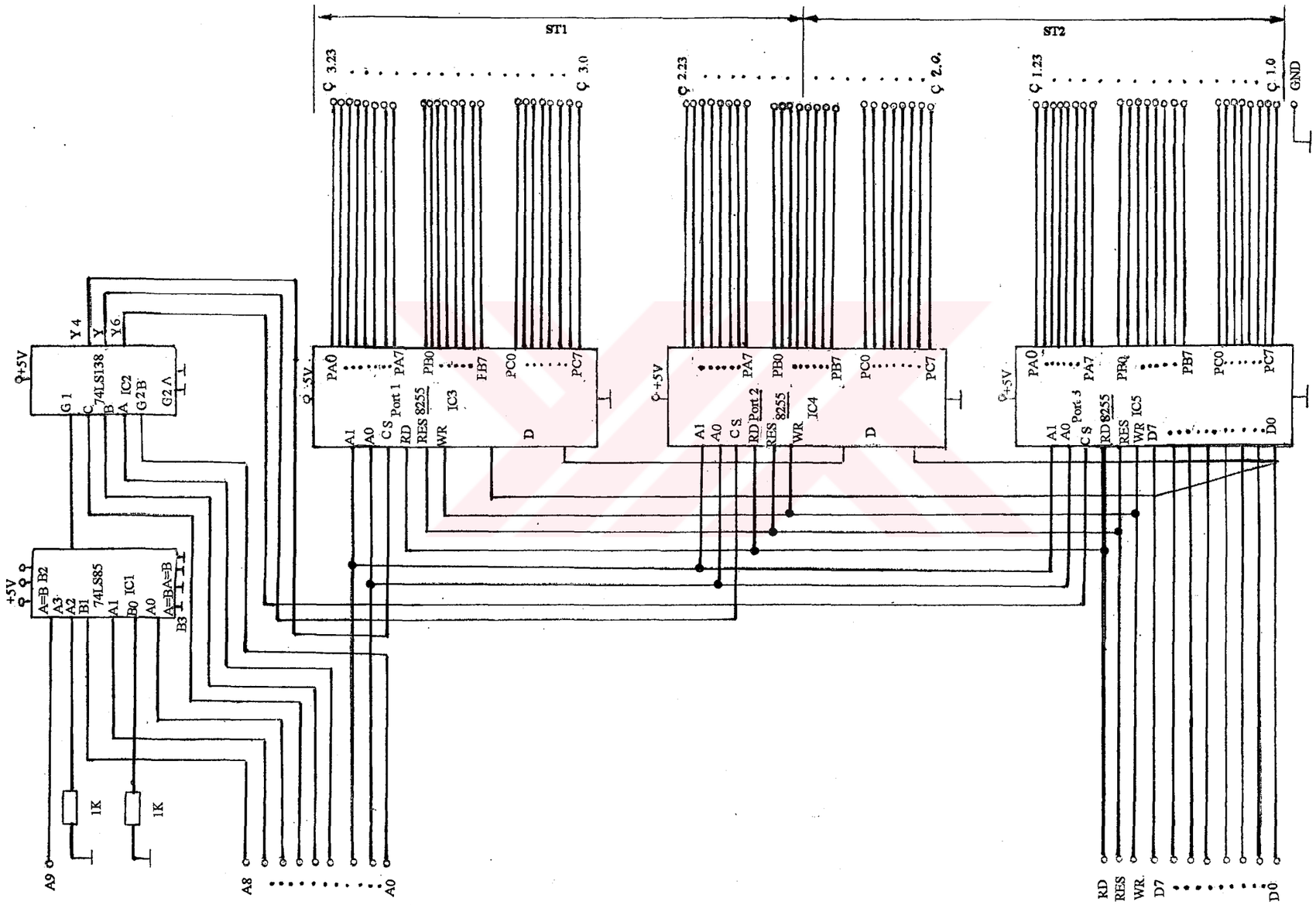
2.port, entegre devrenin bacaklarının giriş/çıkış bağlantısını belirleyen bağlantı kontrol bilgisi ve giriş bilgisi için ayrılmıştır. Port 2 kapıları çıkış olarak programlanır.

3.port, entegre devrenin besleme ve latch kontrolları için ayrılmıştır. Port 3 kapıları çıkış olarak programlanır.

Buna göre kapı adresleri ,kumanda sözcük bilgileri ve kapı kullanım alanları Tablo 4.2'de gösterilmiştir.

ADRESLER	PORT VE KAPI İSMİ	KULLANIM ALANI
288D	Port1 A Kapısı	Çıkış Bilgisi (23'den 16'ya Test Çıkışı)
290D	Port1 B Kapısı	Çıkış Bilgisi (15'den 8'e Test Çıkışı)
292D	Port1 C Kapısı	Çıkış Bilgisi (7'den 0'a Test Çıkışı)
294D	Port1 Kontrol Saklayıcısı	155D (Girişe programlandı.)
296D	Port2 A Kapısı	Bağlantı ve Giriş Bilgisi (23'den 16'ya Test Girişi)
298D	Port2 B Kapısı	Bağlantı ve Giriş Bilgisi (15'den 8'e Test Girişi)
300D	Port2 C Kapısı	Bağlantı ve Giriş Bilgisi (7'den 0'a Test Girişi)
302D	Port2 Kontrol Saklayıcısı	128D(Çıkışa programlandı.)
304D	Port3 A Kapısı	Latch Kontrol 6.bit:Bağlantı Bilgi Seç. (LK2) 7.bit:Giriş Bilgisi Seçimi(LK1)
306D	Port3 B Kapısı	Besleme Kontrolu
308D	Port3 C Kapısı	Besleme Kontrolu
310D	Port3 Kontrol Saklayıcısı	128D(Çıkışa programlandı.)

Tablo 4.2. Paralel haberleşme portunun adres ve program yapısı



Şekil 4.2. Paralel haberleşme devresi

4.3. Bağlantı Devresi

Şekil 4.1'de bağlantı devresi ve tüm sistemin genel yapısı gösterilmiştir.

Bağlantı devresi, bilgisayar tarafından hazırlanan ,saklanan ve paralel haberleşme portu üzerinden gönderilen bilgileri değerlendirerek test edilecek entegre devrenin besleme ve giriş/çıkış bağlantısını sağlayan devredir.

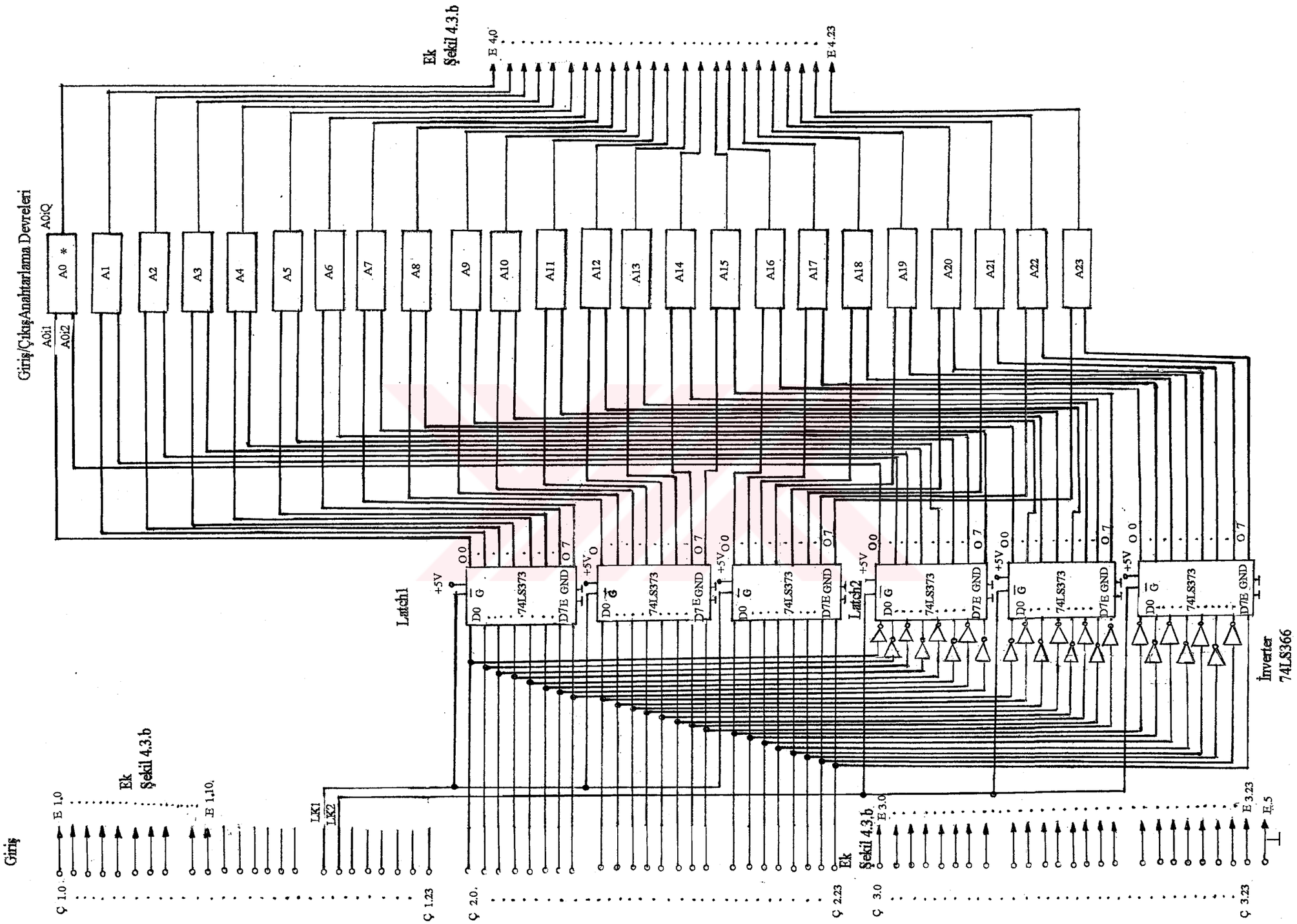
Bir çok entegre devrede farklılık gösteren besleme yapıları daha önceden diğer bilgilerle birlikte bilgisayara tanımlanmıştır. Bu bilgiler tanımlama anında bilgisayarda besleme kontrol bilgisi olarak düzenlenmiş ve veri dosyasında saklanmıştır. Test başlangıcında bu bilgi port üzerinden besleme anahtarlama düzeneğine gönderilir. Test entegresinin +5V ve GND besleme bağlantıları bu düzenek ile yapılır. Besleme anahtarlama devresi 10 adet 5V röle ve röleyi kontrol eden BC517 tipi transistorlardan oluşur. Herhangi bir transistorun girişi "0" durumunda iken röle "OFF" konumundadır ve bağlı bulunduğu entegre test bacağına test giriş/çıkış izni verir. Transistorun girişi "1" durumunda iken röle "ON" konumundadır ve bağlı bulunduğu entegre test bacağını +5V'a veya GND'ye bağlar. Şekil 4.3'de devre yapısı ve şekil 4.4'de test soketi besleme bağlantıları gösterilmiştir.

Entegre devrelerde diğer farklılık gösteren yapı entegre giriş/çıkış yapısıdır. Entegre devrenin giriş/çıkış bağlantı durumunu belirleyen bilgi "Bağlantı Durum Bilgisi" olarak girilir. Bilgisayar tarafından düzenlenip veri dosyasında saklanır. Test başladıktan ve besleme bağlantıları tamamlandıktan sonra 3. portun A kapısındaki 6.bit aktif edilir yani bu kapıdan 64D bilgisi gönderilir. Bu Latch2'yi aktif, Latch1'i pasif yapar. Sonraki aşamada bilgisayar 2. portdan 24 bitlik bağlantı durum bilgisini gönderir. Aktif durumdaki Latch2 bağlantı durum bilgisini inverti alındıktan sonra alır ve

saklar. Test bitene kadar Latch2'de saklı kalır.Latch'lerin çıkışı giriş/çıkış anahtarlama devrelerine bağlıdır. Her entegre test bacağı için bir giriş/çıkış anahtarlama devresi mevcuttur. Giriş/çıkış anahtarlama devresi, latch2'nin çıkışlarından herhangi biri "0" olduğunda bağlı olduğu entegre test bacağını "Z" yüksek empedans konumuna getirir. Bu test yapılan entegre devrenin girişini keser. Bu konum entegrenin çıkış bacakları için kullanılır. Giriş/çıkış anahtarlama devresi Latch2'nin çıkışlarından herhangi biri "1" olduğunda bağlı olduğu entegre test bacağını giriş'e izin verir. Her iki konumda da entegre test çıkışı okunabilir.

Entegre devrenin giriş/çıkış bilgileri daha önce tanımlanmış bilgisayar tarafından düzenlenip dosyalanmıştır. Besleme ve bağlantı durum bilgileri gönderildikten sonra 3. portun A kapısındaki 7.bit aktif edilir yani bu kapıdan 128D bilgisi gönderilir. Bu Latch1'i aktif, latch2'yi pasif yapar. Sonraki aşamada bilgisayar 2. portdan 24 bitlik giriş/çıkış bilgisini gönderir. Aktif durumda olan latch1 alır ve saklar. Latch'lerin çıkışında bulunan giriş/çıkış anahtarlama devresi daha önce giriş olarak set edilen entegre test bacaklarına latch1'deki bilginin ulaşmasını sağlar. Çıkış olarak set edilen bacaklara latch1'deki bilgi ulaşmaz.

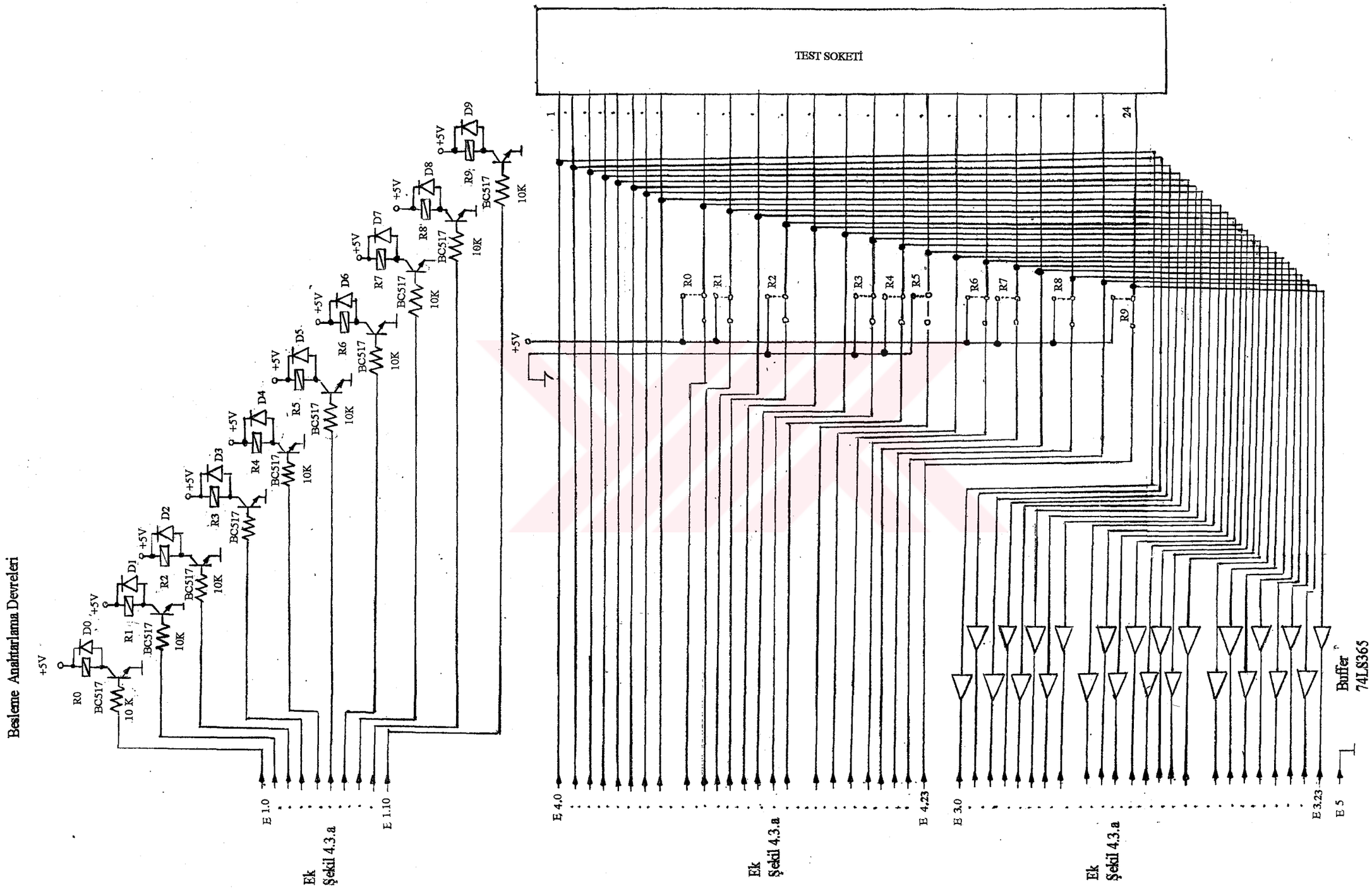
Entegre test bacaklarındaki tüm bilgiler Buffer tarafından okunup Port1 yoluyla bilgisayara ulaşır. Bilgisyarda saklı olan ve gönderilen giriş/çıkış bilgisi ile karşılaştırılır. Doğru ise diğer giriş/çıkış bilgisi gönderilir. Entegre test bacakları okunup karşılaştırılır. Bu tüm giriş/çıkış bilgileri sona erene kadar devam eder. Tüm bilgilerin karşılaştırması olumlu çıktığı takdirde "Entegre Sağlam" sonucuna varılır.



Şekil 4.3.a. Bağlantı devresi

---

* A0..A23 ile simgelenen devreler bölüm 4.3.2.'de açıklanmıştır.



Şekil 4.3.b. Bağlantı devresinin devamı

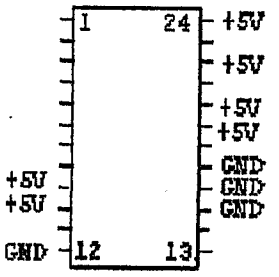
4.3.1. TTL Entegre Devrelerin Besleme Yapıları

TTL entegre devrelerin bacak yapılarında besleme uçları farklılık göstermektedir. Tüm TTL entegrelerin bacak sayısı ve besleme bağlantıları Tablo 4.3'de gösterilmiştir.

BACAK SAYISI	Vcc	GND
14	14.bacak	7. bacak
14	4. bacak	11.bacak
14	5. bacak	10.bacak
16	16.bacak	8. bacak
16	16.bacak	13.bacak
16	5. bacak	12.bacak
20	20.bacak	10.bacak
24	24.bacak	12.bacak

Tablo 4.3. TTL entegrelerin bacak

Bu tabloya bağlı olarak hazırlanan test soketinin besleme bağlantıları şekil 4.4'de gösterilmiştir. Test edilecek entegre devre, test soketine altdan itibaren (12. ve 13. bacaklardan itibaren) yerleştirilir.



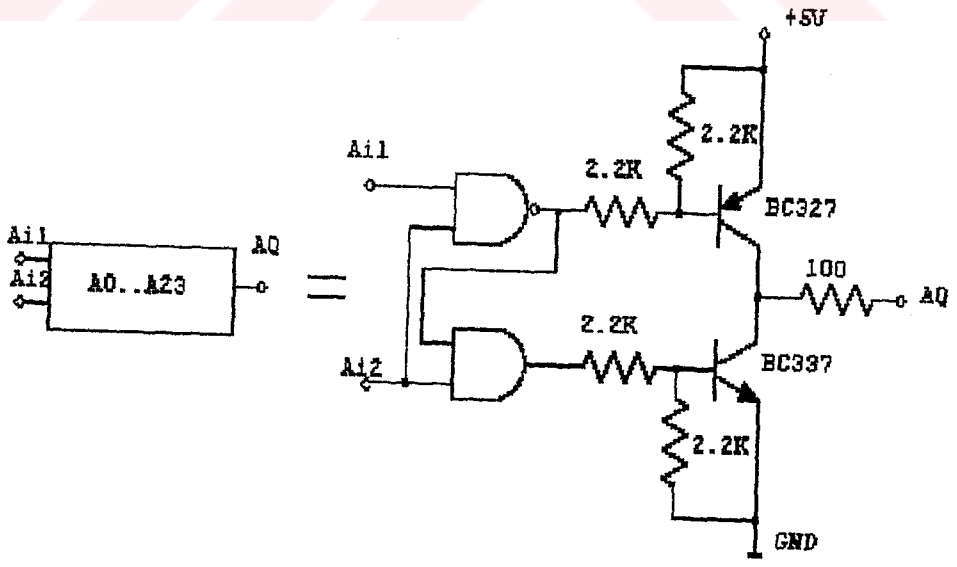
Şekil 4.4. Test soketi besleme bağlantıları.

4.3.2. Giriş/Çıkış Anahtarlama Devreleri

Latch1 ve Latch2 çıkışlarına yerleştirilen 24 adet Giriş/Çıkış Anahtarlama Devresi uygun giriş/çıkış bağlantılarını ve giriş/çıkış bilgi akışını sağlar. Üç konumlu çıkışa sahiptir. Doğruluk tablosu Tablo 4.4'de ve devre şeması Şekil 4.5'de görüldüğü gibi 3 konumlu fonksiyonu sağlayabilmek için pnp ve npn tipi transistorlar kullanılmıştır. Kısa devre ihtimali göz önüne alınarak çıkışta 100Ω'luk direnç kullanılmıştır.

A GİRİŞ BİLGİSİ	B BAĞLANTI BİLGİSİ	C TEST ENTEGRE GİRİŞİ
0	0	Z(girişe izin verilmez)
0	1	0(giriş 0)
1	0	Z(girişe izin verilmez)
1	1	1(giriş 1)

Tablo 4.4. Giriş/anahtarlama devresi doğruluk tablosu



Şekil 4.5. Giriş/çıkış anahtarlama devresi

5. TTL ENTEGRE TEST DEVRESİ YAZILIMI

Program kişisel bilgisayarda Turbo Pascal V5.0 ile yazılmıştır. Programın dosya ismi "Test"dir. Entegre devreler ile ilgili bilgilerin saklandığı veri dosyasına "Entegre Tanım Kütüphanesi" denir. Bu veri dosyasının ismi "Entegre.dat"dır.

Program şu kısımlardan oluşur:

-Self test,

-Arıza testi,

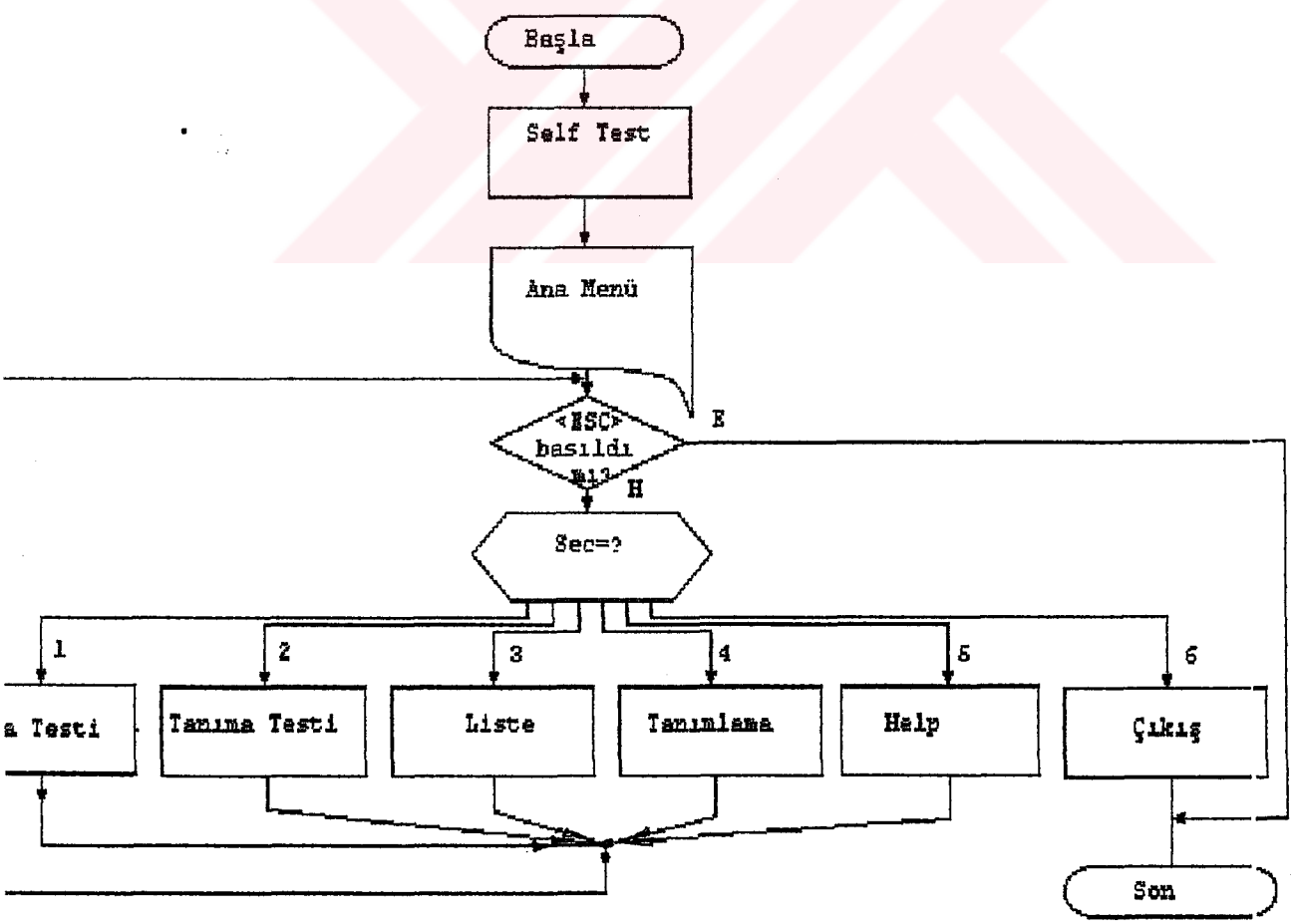
-Tanıma testi,

-Listeleme,

-Tanımlama,

-Açıklama

Programın ana akış diyagramı şekil 5.1'de verilmiştir.



Şekil 5.1. Ana akış diyagramı

5.1. Port Tanımları ve Port Giriş/Çıkışı:

Programın özü, hem donanım hemde yazılımı ilgilendiren port tanımlama ve port giriş/çıkışının kullanılmasıdır. Bilgisayar,devre ve ya program kapalı iken portlar giriş konumunda ve soket uçları yüksek empadans konumundadır.

Self test, arıza test ve tanıma test programlarında kullanılan bu yöntem şu aşamalardan oluşur:

1. Aşama:Port tanımını yap.

Program ilk çalıştığında portların tanımı kontrol saklayıcıları üzerinde yapılır.

Port 1 Kontrol saklayıcısı Port[294]:=155 Giriş
Port 2 Kontrol saklayıcısı Port[302]:=128 Çıkış
Port 3 Kontrol saklayıcısı Port[310]:=128 Çıkış
Tablo 5.1. Port tanım komutları

2. Aşama:Uygun besleme rölelerini aktif,LK2'yi"1",lk1'i"0" yap.

+Vcc ve GND bağlantılarını sağlamak için port1 ve port2'de uygun değerler aktif edilir. Aynı anda "Bağlantı Saklayıcısı" olan "Latch2"yi aktif etmek için port3 6. bacak olan LK2 "1" yapılır. Latch1 pasif, LK1 "0"dır.Port3'e 64 gönderilir.

Port 3 A kapısı Port[304]:=64 LK2 aktif
Port 3 B kapısı Port[306]:=XX Besleme Vcc ve GND
Port 3 C kapısı Port[308]:=XX Besleme Vcc ve GND
Tablo 5.2. Besleme ve latch kontrol komutları

3. Asama: Bağlantı kontrol bilgisini gönder.

Entegrenin bacakları uygun giriş/çıkış durumuna getirilir. Gönderilen bilgiler aktif durumda olan Latch 2'de saklanır. "1" çıkış, "0" girişdir.

Port 2 A kapısı Port[296]:=XX 23...16. bacaklar
Port 2 B kapısı Port[298]:=XX 15...8. bacaklar
Port 2 C kapısı Port[300]:=XX 7...0. bacaklar
Tablo 5.3. Bağlantı kontrol komutları

4.Aşama: LK1'i "1" lk2'yi "0" yap.

Bağlantı kontrol saklayıcısıı Latch 2 pasif, giriş/çıkış bilgisi saklayıcısı Latch 1 aktif yapılır.

Port 3 A kapısı Port[304]:=128 LK1 aktif
Tablo 5.4. Latch kontrol komutu

5. Aşama: Giriş/çıkış bilgisini entegre soketine gönder.

Giriş/çıkış bilgisi Latch1'e gönderilir ve saklanır. Çıkış bilgisi entegre soketine ulaşmaz. Sadece giriş bilgisi ulaşır. Çünkü soket çıkış bacakları ile Latch1 arası yüksek empedans durumundadır.

Port 2 A kapısı Port[296]:=XX 23...16. bacaklar
Port 2 B kapısı Port[298]:=XX 15...8. bacaklar
Port 2 C kapısı Port[300]:=XX 7...0. bacaklar
Tablo 5.5. Giriş(entegreye giriş) komutları

6. Aşama: Entegreden giriş/çıkış bilgisini al.

Entegrenin uçlarındaki tüm bilgiler port tarafından okunur.

Port 1 A kapısı L1:=Port[288] 23...16. bacaklar

Port 1 B kapısı L2:=Port[290] 15...8. bacaklar

Port 1 C kapısı L3:=Port[292] 7...0. bacaklar

Tablo 5.6. Çıkış(entegre çıkışı) okuma komutları

7. Aşama: Karşılaştır.

Okunan L1,l2 ve L3 dosyada yüklenmiş datalar ile karşılaştırır.

8. Aşama: Karar ver.

Yanlış ise hata mesajı verir. Doğru ise diğer bilgileri gönderir, alır ve karşılaştırır. Son bilgi de doğru ise olumlu mesaj verilir.

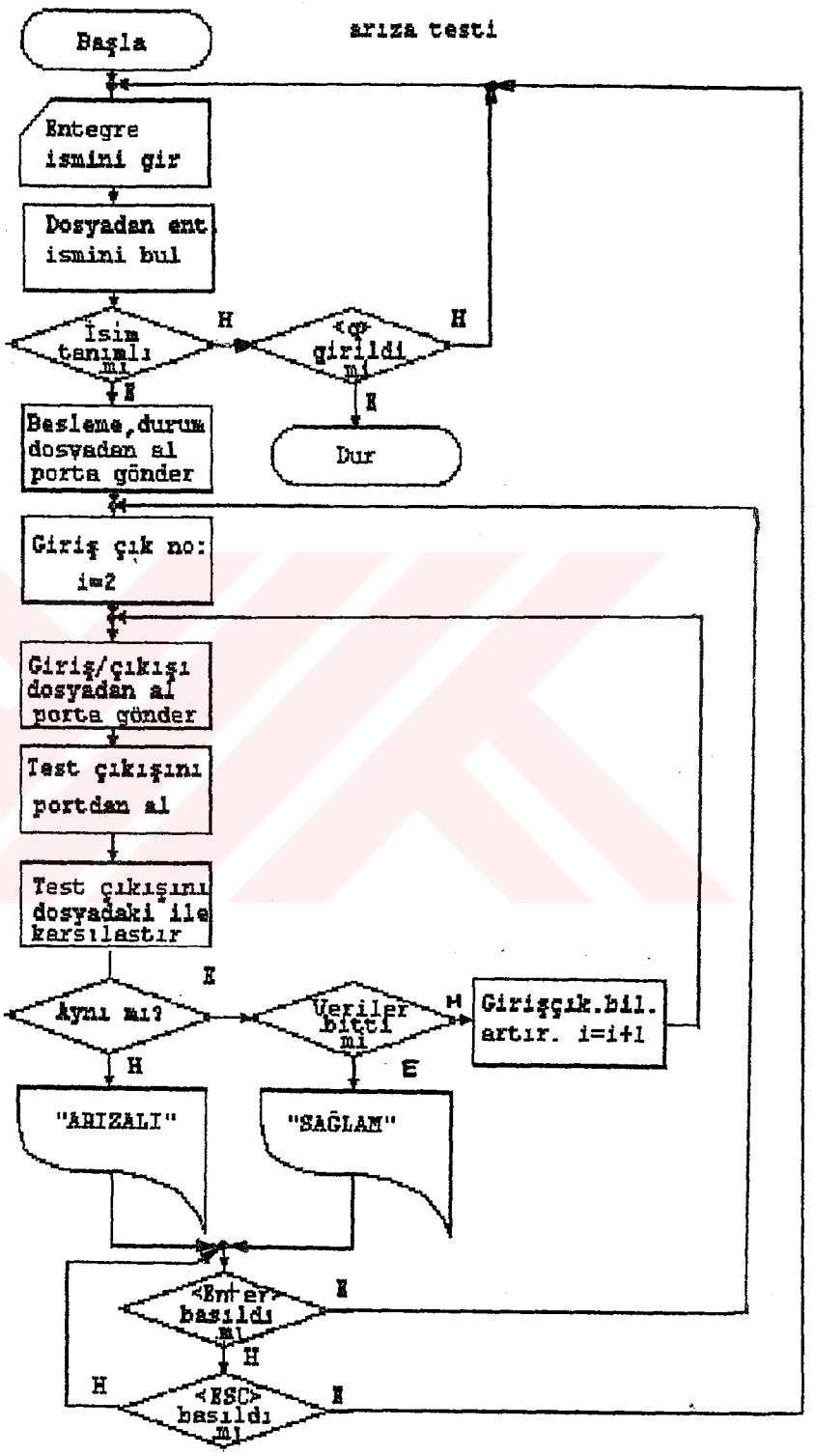
5.2. Self Test

Program ilk çalıştırıldığında tüm portlar tanımlanır. Ardından sokete paternler gönderilir ve okunur. Gönderilen paternler okunanlardan farklı ise hata mesajı verir ve program kapanır. Doğru ise sistem teste hazırdır ve ekrana ana menü çıkar.

5.3. Arıza Testi

Ana menüden "1" seçilirse ekrana "isim" çıkar. Test edileceek entegrenin isimi yazılır ve isim tanımlı ise besleme ve durum (bağlantı) bilgileri porta gönderilir. Ardından giriş/çıkış bilgileri gönderilir ve test çıkışı okunur. Aynı değil ise "arızalı" mesajı

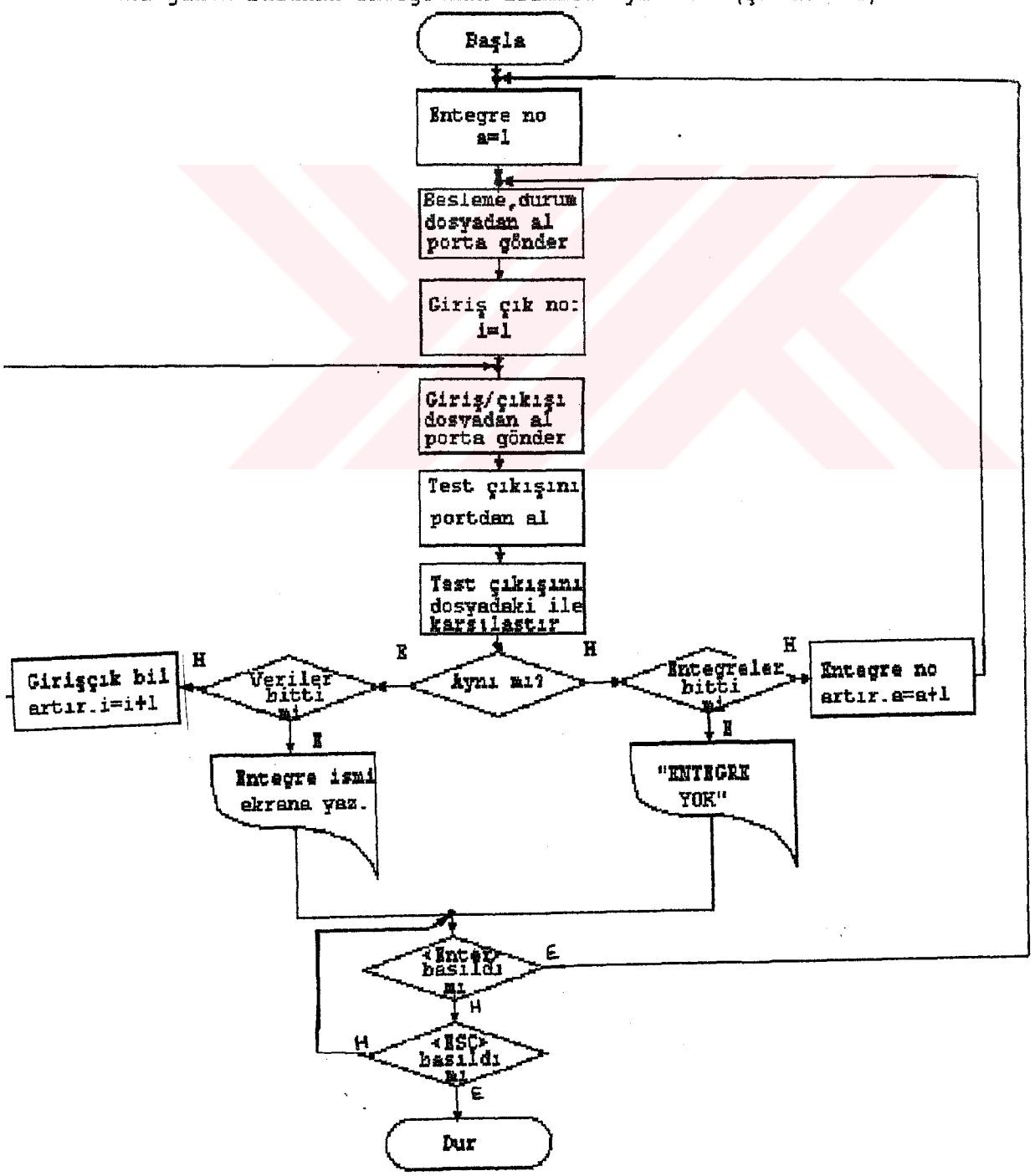
çıkar. Aynı ise diğer bilgiler gönderilir. Bilgiler sona erdiğinde ekranda "sağlam" mesajı çıkar (Şekil 5.1).



Şekil 5.1. Arıza testi akış diyagramı

5.4. Tanıma Testi

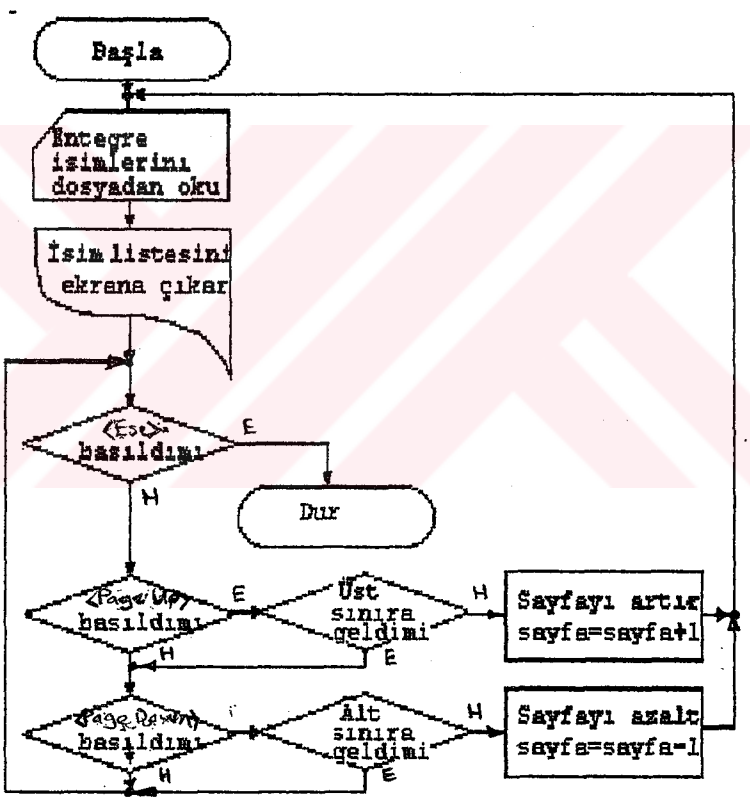
Ana menüden 2 seçilirse ilk olarak ilk entegre fonksiyonunun besleme ve durum (bağlantı) bilgileri gönderilir. Giriş/çıkış bilgileri gönderilir ve çıkış okunur. Okunan bilgiler kütüphanedeki ile karşılaştırılır. Farklı olması durumunda kütüphanedeki diğer entegre fonksiyonları taranır. Doğru fonksiyon bulunamadığı taktirde "entegre yok" mesajı çıkar. Bu fonksiyonun tüm bigileri doğru olduğunda bulunan entegrenin isimleri yazılır.(Şekil 5.2)



Şekil 5.2. Tanıma testi

5.5. Listeleme

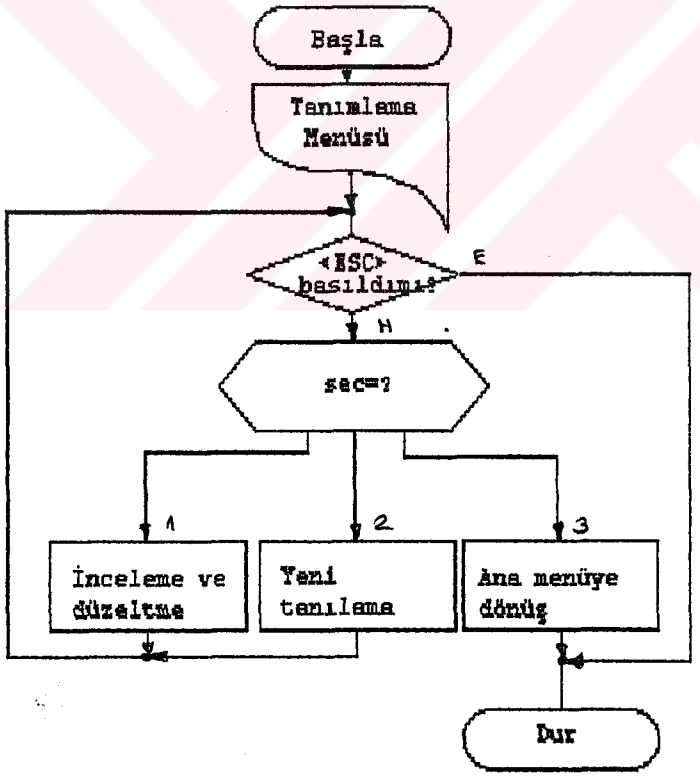
Ana menüden 3 seçilirse tanımlı entegreler dosyadan okunur ve isim listesi olarak ekrana gelir. Sayfa atlatmak için "page up" ve "page down" tuşları kullanılır.(Şekil 5.3)



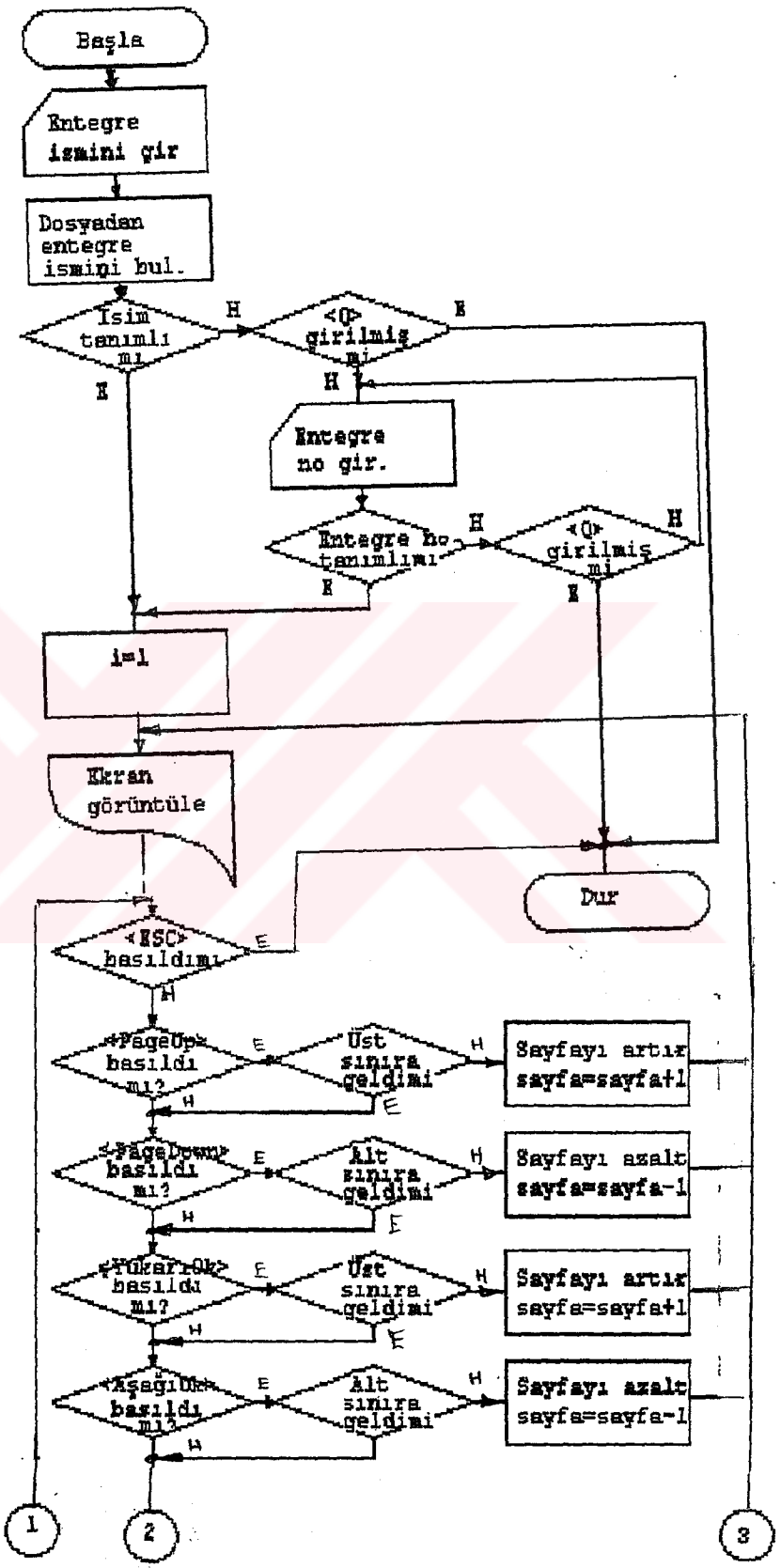
Şekil 5.3. Listeleme akış diyagramı

5.6. Tanımlama

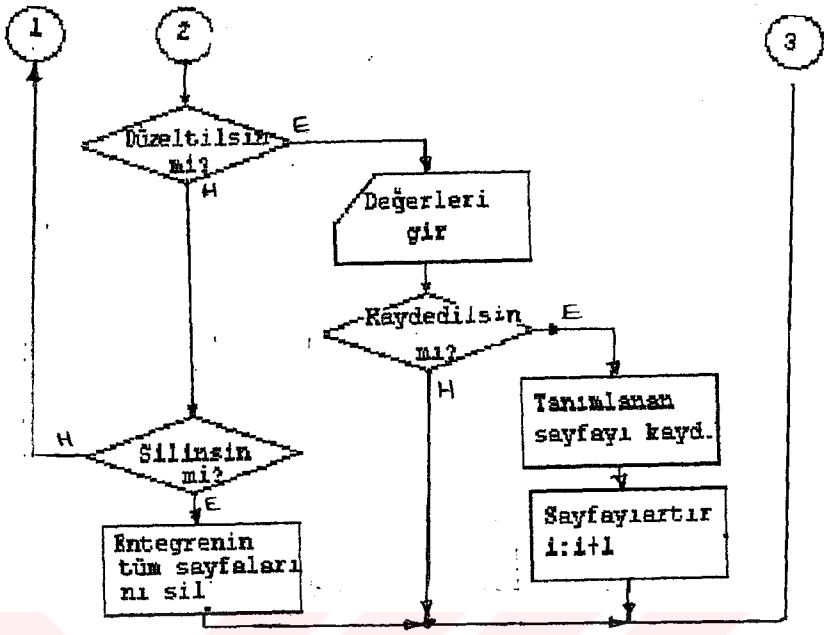
Yeni entegre tanımlamak için kullanılan "4" basıldığında "Tanımlama Menusu" ekrana gelir. "İnceleme ve düzeltme" seçildiğinde ekrana "isim" gelir. Doğru isim girildiğinde ekrana ilk tanımlar olan besleme ve bağlantı bigileri çıkar. Değişiklik için F1'e silme için F2'ye basılır. Her tanımlanan bilgi için kayıt sorulur. Olumlu yanıt alındığında kaydedilir. "Page up" ve "page down" diğer entegre fonksiyonlarına, "yukarı ok" ve "aşağı ok" ise farklı bilgilere erişebilmek için kullanılır. "Yeni tanımlama" seçildiğinde ilk tanımlama için boş sayfa açar. Diğer tüm işlemeler "İnceleme ve düzelme" programında olduğu gibidir. (Şekil 5.4,5.5)



Şekil 5.4. Tanımlama ana akış diyagramı



Şekil 5.5.a. İnceleme ve düzeltme akış diyagramı



Şekil 5.5.b. İnceleme ve düzeltme akış diyagramının devamı

5.7. Açıklama

Ana menüde "6"ya basıldığında ekrana kullanım ile ilgili yazı gelir. "ESC"ye basıldığında ana menüye döner.

5.8. Veri Dosyası (Entegre Tanım Kütüphanesi)

Tanımalanan entegre devre fonksiyonlarının tamamı "Entegre.Dat" isimli veri dosyasında saklıdır. Yaklaşık 1,6MB'lik hafızaya sahip bu dosya üzerinde 50'er giriş/çıkış ve bağlantı bilgisini içeren 500 ayrı fonksiyona sahip 10000 adet entegre tanımalanabilir. Şidiye kadar tanımlanan entegreler aşağıda verilmiştir:

1. NAND kapıları: 7400, 74LS00, 74L00, 74S00, 74HC00, 74HCT00, 7403, 74LS03, 74HC03, 74HCT03, 7426, 74LS26, 7437, 7438, 74LS38, 74S38, 74LS132,74F00

2. NOR kapıları: 7402, 74LS02, 74L02, 74S02, 74HC02, 7433, 74LS33, 7428.

3. Inverter kapıları: 7404, 74LLS04, 74L04, 74S04, 74HC04, 74HCT04, 74HCU04, 7405, 74LS05, 74S05, 74HC05, 74HCT05, 7406, 7416, 7414, 74LS14, 74HC14.

4. AND kapıları: 7408, 74LS08, 74S08, 74HC08, 74HCT08, 7409, 74LS09, 74S09.

5. 3 girişli NAND kapıları: 74LS10, 74L10, 74S10, 74HC10.

6. 3 girişli AND kapısı: 7411, 74LS11, 74HC11.

7. Farklı bağlantıya sahip 2 girişli NAND kapıları: 7401,7439

8. Bufferlar: 7407,7417.

9. 4 girişli NAND kapıları: 7413, 74LS13, 7420, 74LS20, 74S20, 74HC20, 7440,, 74LS40, 74S40.

10. 4 girişli AND kapıları: 7421, 74LS21.

11. 4 girişli NOR kapısı:7425.

12. 3 girişli NOR kapıları: 7427, 74S27, 74HC27.

13. 8 girişli NAND kapıları: 7430, 74LS30, 74HC30.

14: İki girişli OR kapıları: 7432, 74LS32, 74S32, 74HC32, 74HCT32.

15. 2 grup, 2 girişli AND-OR-INVERT kapıları: 7451, 74S51.

16. 3,2 girişli AND-OR-INVERTER kapıları: 74LS54

17. 4,2,3,2 girişli AND-OR-INVERTER kapıları: 74S64

18. 2 grup, 2girişli AND-OR-INVERT kapıları:74LS51

19. Kenar tetiklemeli D Flipflop: 7474, 74LS74, 74S74.

20. 2 girişli EXOR: 7486, 74LS86, 74S86, 74LS136.

21: JK Flipflop: 74107, 74LS107, 74S107.

22. JK Flipflop: 74LS113, 74S113.

23. Üç konumlu buffer: 74125, 74LS125.

24. Üç konumlu buffer: 74126, 74LS126.

25. 4 bit shift register: 7495, 74LS95.

26. 8 bit seri giriş paralel çıkış: 74164, 74LS164.

27. 9 bit parity genarator: 74180.

28. 5 girişli NOR kapısı: 74LS260 74S260.

29. EX-NOR kapısı: 74LS266

30. CMOS 3 girişli NOR-INVERT kapısı: 4000

31. CMOS NAND kapısı: 4011

32. CMOS NOR kapısı: 4001

33. CMOS 4 girişli NOR kapısı: 4002

34. CMOS 4 girişli NAND kapısı: 4012

35. JK Flipflop: 7473

36. JK Flipflop: 7476,74LS76

SONUÇLAR VE ÖNERİLER

Bu çalışma, teorik ve uygulamalı olarak gerçekleştirilmiştir. Ön bilgi olarak sayısal entegre devreler ve lojik aileler hakkında bilgi verilmiştir. Tasarım sırasında, bilgisayarda "Smart" baskılı devre çizim programı ve "Turbo Paskal V5.0" programlama dili kullanılmıştır. Bilgisayarda paralel haberleşme ve 8255 PPI entegresi hakkında bilgi edinilmiş ve bu bilgiler kullanılmıştır.

Kontrol ünitesi, paralel haberleşme devresi ve bağlantı devresinden oluşan TTL entegre test devresi mümkün olduğu kadar kullanışlı hale getirilmiştir. Sistemin avantajları şunlardır:

1. Esnektir. Kullanıcı tarafından rahatlıkla entegre tanım kütüphanesi oluşturulabilir. Kütüphane hafızası oldukça geniştir (10000 adet entegre, 500 adet fonksiyon, 50 adet giriş/çıkış ve bağlantı bilgisi).

2. Sadece TTL değil,diğer 0..5V arası çalışan lojik aileleri de (CMOS gibi) test edebilir.

3. Sistem, tanıma testi özelliği sayesinde fonksiyonu bilinmeyen entegrelerin fonksiyonunu belirler.

4. Program başlangıcında sistem kendini test eder. Sistemde herhangi bir hata olduğunda program durur.

5. Tanımlanmış tüm entegre devrelerin ismi listelenebilir.

Sistemin dezavantajları ise, multivibrator gibi ek bağlantılar gerektiren entegre devreleri test edememesi ve entegre test geriliminin 0..5V ile sınırlı olmasıdır.

Her ne kadar bu çalışmayı kapsamasa da, ileriki çalışmalarda sisteme ilave olarak önerilecek özellikler şunlar olabilir:

1. Entegre tanımlama özelliği daha kullanışlı hale getirilebilir.

2. Otomatik tanımlama özelliği getirilebilir. Otomatik tanımlamada sistem, sağlam bir entegre yerleştirildikten sonra giriş kombinasyonlarını uygular, çıkışları okur ve dosyalar.

3. Arızanın hangi bacakta olduğu görüntülenebilir.

4. Test gerilimi kademeli olarak uygulananilir.

5. Fanout sayısı ve elktriksel özellikler test edilebilir.

6. Test soketi 24 bacaktan 40 bacağa çıkarılabilir. Mikroişlemciler ve özel entegre devreler test edilebilir.

7. Aynı devre üzerinde küçük değişikliklerle, ek olarak EPROM programlama devresi olarak da kullanılabilir.

KAYNAKLAR

1. Bayburan, B., Şubat 1988. Turbo Pascal. İstanbul.

2. CMOS Logic Databook, 1987. National Semiconductor Corporation, U.S.A.

3. LS/S/TTL Logic Databook, 1988. National Semiconductor Corporation, U.S.A.

4. Microprocessors and Peripherals Databook, 1990. OKI, Japan.

5. Practical Digital Electronics, 1986. Hewlett Packart.

6. TTL Logic Databook, 1989. Philips, Germany.

EK A:TTL ENTEGRE TEST DEVRESİ PROGRAMI

```
uses crt;
type
 test=record
  no:integer;
  isim:packed array[1..20,1..10] of char;
  bs:integer;
  bes:integer;
  gnd:integer;
  b:array[1..50,1..24] of integer;
  port4:longint;
  port5:longint;
  port1:array[1..50] of longint;
  port2:array[1..50] of longint;
  port3:array[1..50] of longint;
 end;


var
  entegre:test;
  entegredos:file of test;

w1,w2,w3,q1,q2,d,w,l,s,ki,dur,acik1,l1,l2,n,bsa,j,i,t,a,a2,q,bes,besa,
bes1,gnd,gnda,gnd1,
  giris,giris1,giris2,deger,kdeger,kdeger1,kdeger2,sayi,sayi1,sayi2,
  role,role1,role2,bdurum,bdurum1:longint;
  bb:array[1..24] of integer;
  ba:array[1..50,1..24] of integer;
  bt:array[1..50,1..24] of integer;
  etik,isim:string;
  bek,bek2,kar1,kar2,kayit,sor,cev,devam,sec,sec3,cev1:char;
  son,noa,b,no2,buft1,buft2,e,buf,k,isima,no1,h,s1,l3:integer;
  isim1:packed array[1..10] of char;

label
  g8,g40,g35,sec2,sec1,g1,g2,g3,g4,g6,g9,g20,g21,g30,g31,git1,git2;

procedure self;
 begin
  clrscr;
  gotoxy(20,8);
  write('LÜTFEN BEKLEYİN. SİSTEM TEST EDİLİYOR.');
  port[294]:=155;                                {KONTROL GİRİŞLERİ}
  port[302]:=128;
  port[310]:=128;
  delay(100);
```

```
port[308]:=0;port[306]:=0;port[304]:=64;
delay(100) ;
port[300]:=0;port[298]:=0;port[296]:=0;
delay(100);
port[308]:=0;port[306]:=0;port[304]:=128;
delay(100) ;
port[300]:=170;port[298]:=170;port[296]:=170;
l3:=port[292] ;
l2:=port[290];
l1:=port[288];
if (l3=170) and (l2=170) and (l1=170) then
  begin
    port[300]:=85;port[298]:=85;port[296]:=85;
    l3:=port[292] ;
    l2:=port[290];
    l1:=port[288];
    if (l3=85) and (l2=85) and (l1=85) then
      begin
        delay(100);
        port[308]:=0;port[306]:=0;port[304]:=64;
        delay(100) ;
        port[300]:=255;port[298]:=255;port[296]:=255;
        l3:=port[292] ;
        l2:=port[290];
        l1:=port[288];
        if (l3=255) and (l2=255) and (l1=255) then
          begin
            gotoxy(30,10);
            write('SYSTEM TAMAM');
            DELAY(1000);
          end
          else
          begin
            gotoxy(30,10);
            write('SYSTEM HATALI 1') ;DELAY(1000);halt;
          end;
      end
      else
      begin
        gotoxy(30,10);
        write('SYSTEM HATALI 2');DELAY(1000);halt;
     end;
  end
  else
  begin
    gotoxy(30,10);
    write('SYSTEM HATALI 3');DELAY(1000);halt;
```

```
    end;
 end;

procedure anamenu;
 begin;
    assign(entegredos,'entegre.dat');
  clrscr;
  gotoxy(30,7);write('DIKKAT!    ');
  gotoxy(25,8);write('SISTEM YUKLENIYOR...    ');
    reset(entegredos);
  a:=0;
  i:=0;
  repeat
    i:=i+1;
    seek(entegredos,i-1);
    read(entegredos,entegre);
  until entegre.no=0;
  son:=i;
  textbackground(0);                              {ANA MENÜ}
  clrscr;
  textbackground(4);
  textcolor(14);
  gotoxy(30,7);write('ANA MENÜ');
  textbackground(1);
  gotoxy(30,10);write('1.DOªRULUK TESTÿ');
  gotoxy(30,12);write('2.TANIMA TESTÿ');
  gotoxy(30,14);write('3.LÿSTE');
  gotoxy(30,16);write('4.TANIMLAMA');
  gotoxy(30,18);write('5.HELP');
  gotoxy(30,20);write('6.ÇIKIP');
  sec:=readkey;
end;

procedure gir;

  begin
  with entegre do begin
    for l:=1 to  bs do
      begin
        writeln (l,'.  BACAK',etik,':');
        repeat;
          read (b[j,l]);
        until (b[j,l]=0) or (b[j,l]=1);
    end;
    a:=0;q:=1;
    for l:=bs downto 1 do
      begin
```

```
        a:=a+(b[j,l]*q);q:=q*2;
      end;
      end;
  end;

procedure liste;

label gitll;
begin;
  clrscr;
  gotoxy(30,1);write('LÝSTE');write(filesize(entegredos));
  gotoxy(1,3);write('NO');
  gotoxy(4,3);write('ISIM1');
  gotoxy(11,3);write('ISIM2');
  gotoxy(18,3);write('ISIM3');
  gotoxy(25,3);write('ISIM4');
  gotoxy(32,3);write('ISIM5');
  gotoxy(39,3);write('ISIM6');
  gotoxy(46,3);write('ISIM7');
  gotoxy(53,3);write('ISIM8');
  gotoxy(60,3);write('ISIM9');
  gotoxy(67,3);write('ISIM10');
  gotoxy(4,4);write('ISIM11');
  gotoxy(11,4);write('ISIM12');
  gotoxy(18,4);write('ISIM13');
  gotoxy(25,4);write('ISIM14');
  gotoxy(32,4);write('ISIM15');
  gotoxy(39,4);write('ISIM16');
  gotoxy(46,4);write('ISIM17');
  gotoxy(53,4);write('ISIM18');
  gotoxy(60,4);write('ISIM19');
  gotoxy(67,4);write('ISIM20');
  gotoxy(4,5);write('-------------------------------------------------
--------------------');
  gotoxy(30,24);write('CIKIS:ESC');
  a:=1;
  b:=0;
   repeat;
   for i:=1 to 5 do
    begin
    a:=5*b+i;
seek(entegredos,a-1);
read(entegredos,entegre);
    gotoxy(40,20);clreol;write(filepos(entegredos));
    gotoxy(1,6+(i*2));clreol;
    gotoxy(1,5+(i*2));clreol;
    if a<filesize(entegredos)+1 then
```

```
    begin
     writeln(a,'.');
     for h:=1 to 10 do
     for l:=1 to 10 do begin gotoxy(3+((h-1)*7)+l,5+(i*2));
                  write(entegre.isim[h,l]);end;
                  write(entegre.no);
    for h:=11 to 20 do
     for l:=1 to 10 do begin gotoxy(3+((h-11)*7)+l,6+(i*2));
                  write(entegre.isim[h,l]);end;
    end;
    end;
gitll:bek:=readkey;
    if keypressed  then
    begin
     bek2:=readkey;
     if (ord(bek)=0) and (ord(bek2)=73) and (b<filesize(entegredos)/5-
1) then b:=b+1;
     if (ord(bek)=0) and (ord(bek2)=81) and (b>0) then b:=b-1;
    end;
    if (bek=#27) or (bek=#0)  then  else goto gitll;
   until bek=#27;
end;

procedure tanimlama;

begin
clrscr;
  i:=0;
  repeat
   i:=i+1;
   seek(entegredos,i-1);
   read(entegredos,entegre);
  until entegre.no=0;
  son:=i;
textbackground(1);                                    {TANIMLAMA
MENUSU}
  clrscr;
  a:=1;
  textbackground(4);
  gotoxy(30,7);write('TANIMLAMA    ');
  textbackground(1);
  gotoxy(30,10);write('1.ÖNCELEME VE DÜZELTME   ');
  gotoxy(30,12);write('2.YENÝ TANIMLAMA');
  gotoxy(30,14);write('3.ANA MENÜYE DÖNÜP');
  sec3:=readkey;
end;
```

```
procedure inceleme;

label gitil;
begin
clrscr;
    j:=1;
    gotoxy(30,7);write('ŸSŸM    ');
    l:=0;
    repeat
     l:=l+1;
      repeat
       gotoxy(36+l,7);
       read(isiml[l]);
      until (isiml[l]<>' ') and ((l<>1) or ((isiml[l]<>#13) and
(isiml[l]<>#10)))  ;
    until (l=10) or (isiml[l]=#13) or (isiml[l]='Q');
    if upcase(isiml[l])='Q' then  halt;

 {ISIM KONTROL}
 clrscr;
  gotoxy(30,7);write('DIKKAT!    ');
  gotoxy(25,8);write('SISTEM YUKLENIYOR...    ');
  i:=0;
  repeat
   i:=i+1;
   seek(entegredos,i-1);
   read(entegredos,entegre);
  until entegre.no=0;
  son:=i;
 a:=0;
 repeat
  a:=a+1;
  seek(entegredos,a-1);
  read(entegredos,entegre);
  h:=0;
   repeat
    h:=h+1;
    l:=0;
    d:=0;
     repeat
      l:=l+1;
      if upcase(entegre.isim[h,l])=upcase(isiml[l]) then d:=d+1;
     until (l=10) or (entegre.isim[h,(l+1)]=' ') ;
   until    (h=21)
          or ((l=d) and (upcase(entegre.isim[h,l])=upcase(isiml[l]))
and (isiml[l+1]=#13));
```

```
 if (h<>21)  then begin j:=1;goto gitil end;
 until (a>son-2) ;

 {  NO GÝRME}
 clrscr;
 repeat
    Gotoxy(30,10);write(a,'  ',filesize(entegredos));
    write('LÝSTE NO    ');read(no1);
    if ioresult=106 then halt;
 until (ioresult=0) and (no1<son);
 a:=no1;
  j:=1;
gitil:
end;

procedure dec;

begin
 with entegre do begin
     i:=0;q:=1;
     for k:=bs downto 1 do
      begin
        i:=i+(b[j,k]*q);q:=q*2;
      end;
        for k:=1 to bs do
      write(b[j,k]);
      writeln;
      end;
end;

procedure kaydec;

begin
      bdurum:=i;
      giris:=bdurum;
      deger:=giris;
    with entegre do begin
    t:=(24-bs) div 2;
    kdeger:=deger shl t;
    writeln(deger,'  ',t,'   ',kdeger);
      kdeger1:=kdeger;
      if j=1 then bdurum1:=16777216+not(kdeger1)
            else begin
                  i:=0;q:=1;
                  for k:=bs downto 1 do
                  begin
```

```
                    i:=i+q;
                    q:=q*2;
                  end;
                  bdurum:=i;
                  giris:=bdurum;
                  deger:=giris;
                  t:=(24-bs) div 2;
    kdeger:=deger shl t;
    writeln(deger,'  ',t,'   ',kdeger);
                  kdeger2:=16777216+not(kdeger);
                  bdurum1:=kdeger2 or kdeger1;
                end;
      writeln(bdurum1);
      end;
end;

procedure portcev;

begin
 with entegre do begin
  w1:=bdurum1 and 255;
  w2:=(bdurum1 shr 8) and 255;
  w3:=(bdurum1 shr 16) and 255;
  port1[j]:=w1;
  port2[j]:=w2;
  port3[j]:=w3;
  end;
end;

procedure binary;
begin
 with entegre do begin
  q1:=8388608;
  a2:=bdurum1;
  for 1:=1 to 24 do
    begin
      bb[1]:=trunc(a2 div q1) ;
      a2:=a2-bb[1]*q1;
      q1:=q1 div 2;
      write(bb[1]);
    end;
    writeln;
      deger:=8388607;
      t:=(24-bs) div 2;
      kdeger:=deger shl t;
      writeln(deger,'  ',t,'   ',kdeger);
      t:=(24-bs) div 2;
```

```
      kdeger:=deger shl t;
      writeln(deger,'  ',t,'   ',kdeger);
      sayi:=(not((16777215 shl ((24-bs) div 2)) and (16777215 shr
((24-bs) div 2))
      and 16777215))+16777216;
      giris1:=giris or sayi;
      writeln('giris=','   kaymıf giris=',giris,'   giris1=',giris1);
end;
end;

procedure ekran;
  begin
    clrscr;
    with entegre do begin
    gotoxy(1,1);Writeln('ÿSÿM');
    gotoxy(30,5);write('ENTEGRE NO:',a,' GÿRÿP:',j,' TANIM
SAYISI:',i);
    for s1:=1 to 7 do
    for s:=1 to 10 do begin gotoxy((8*s1)+s,1);write(isim[s1,s]);end;
    for s1:=8 to 14 do
    for s:=1 to 10 do begin gotoxy((8*(s1-
7))+s,2);write(isim[s1,s]);end;
    for s1:=15 to 20 do
    for s:=1 to 10 do begin gotoxy((8*(s1-
14))+s,3);write(isim[s1,s]);end;
    gotoxy(1,4); writeln ('BACAK SAYISI:');
    gotoxy(1,5); writeln(bs);
    gotoxy(1,6); writeln('BESLEME Vcc:');
    gotoxy(1,7); writeln(bes);
    gotoxy(1,8); writeln('GND');
    gotoxy(1,9); writeln(gnd);
                   if j=1 then begin
                              gotoxy(25,8);  writeln('BACAK DURUMU');
                         end
                         else begin
                              gotoxy(25,8); writeln('GÿRÿP ÇIKIP
BÿLGÿSÿ  ',j-1);
                              end;


 {}
  for ki:=1 to bs do
    begin
      if ki<14 then begin l1:=ki*6-5;l2:=10;end
              else begin l1:=(ki-13)*6-5;l2:=14;end;
      gotoxy(l1,l2);
```

```
      if ki=bes then  begin
write(ki,'.');gotoxy(l1,l2+1);write('BACAK');
                    gotoxy(l1,l2+3);writeln('Vcc');
                    end
                    else begin
                    if ki=gnd then  begin write(ki,'.');
                                 gotoxy(l1,l2+1);write('BACAK');
                                  gotoxy(l1,l2+3);writeln('GND');

                                  end
                                  else  begin write(ki,'.');
                                  gotoxy(l1,l2+1);
                                  write('BACAK');
                                    if b[1,ki]=1 then etik:='GÝRÝP';
                                    if b[1,ki]=0 then etik:='ÇIKIP';

                                  gotoxy(l1,l2+2);write(etik);

                     if (b[j,ki]=1) or (b[j,ki]=0) then begin
                                       gotoxy(l1,l2+3);writeln
(b[j,ki]);
                                       end;
                    end;
                    end;

    end;
    end;


  {}
  end;



{PROGRAM BAPLANGICI}

begin
  (*$I-*)
  (*I-*)

  self;

  goto git1;
  g4:g3:
  close(entegredos);
```

```
  git1:
  anamenu;

  if sec='1' then goto sec1;
  if sec='2' then goto sec2;
  if sec='3' then liste;
 git2:  if sec='4' then begin tanimlama;
  if sec3='1' then begin buf:=1;inceleme;goto g21; end;
  if sec3='2' then begin buf:=3;goto g8;end;
  if (sec3='3') or (sec3=#27) then goto g4 else goto git2;
  end;
  if sec='5' then begin clrscr; gotoxy(30,7); write('ACIKLAMA');
                  sor:=readkey;if sor=#27 then goto git1;end;
  if (sec='6') or (ord(sec)=27) then goto g9 else goto g3;

  g8:
clrscr;
if buf=3 then a:=son;
seek(entegredos,a-1);
read(entegredos,entegre);
 entegre.no:=a;
  J:=1;
g20:  with entegre do begin
  ekran;


if (buf=3) or (buf=1) then
begin
entegre.no:=a;
h:=0;
repeat
h:=h+1;
  l:=0;
  repeat
l:=l+1;
repeat
 if h<8 then gotoxy((8*h+1),1);
 if 7<h then if h<15 then gotoxy((8*(h-7)+1),2);
 if 14<h then gotoxy((8*(h-14)+1),3);
 read(isim[h,l]);
until (isim[h,l]<>' ') and ((l<>1) or ((isim[h,l]<>#13) and
(isim[h,l]<>#10)));
until (l=10) or (isim[h,l]=#13) or (upcase(isim[h,l])='Q');
for s:=l to 10 do isim[h,s]:=' ';
             if (upcase(isim[h,1])='Q') then begin
                  gotoxy(20,24);write('ÇIKMAK ÜÇÜN EMİN MİSİNİZ
(E/H)');
```

```
                if upcase(readkey)='E'then goto git2
                ;end;


isima:=1;
gotoxy (20,24);write('BASKA ISIM GIRILECEK MI (E/H)?');
cev1:=readkey;
gotoxy(1,24);
clreol;
until (upcase(cev1)<>'E') or (h=20);
if h<20 then for s:=h+1 to 20 do for 1:=1 to 10 do isim[s,1]:=' ';

 {BS GRŸPŸ}
    ekran;
    repeat;
    gotoxy(20,24);write('14,16,20 VEYA 24 GŸRŸN');
    gotoxy(1,5);
    read (bs);
     if ioresult=106 then begin

               gotoxy(20,24);write('ÇIKMAK ŸÇŸN EMŸN MŸSŸNŸZ (E/H)');
               if upcase(readkey)='E'then goto git2;end;
                {}
    until (bs in[14,16,20,24]) or (ioresult<>0) ;
    clreol;
    ekran;
    bsa:=1;

 {BES GŸRŸPŸ}

g1: repeat;
    ekran;
    gotoxy(20,24);write('4,5,14,16,20 VEYA 24 GŸRŸN');
    gotoxy(1,7);
    read (bes);
{}
     if ioresult=106 then begin
               gotoxy(20,24);write('ÇIKMAK ŸÇŸN EMŸN MŸSŸNŸZ (E/H)');
               if upcase(readkey)='E'then goto git2;end;
{}
    until  (ioresult=0);
    clreol;

                        {Role çikiflarinin düzenlenmesi     }
  besa:=1;
  ekran;
  role2:=0;
```

```
  case bes of
   14:begin if bs=14 then begin bes1:=32;      role:=2;     end
                     else goto g1;end;
    4:begin if bs=14 then begin bes1:=32768;   role:=128    end
                     else goto g1;end;
    5:begin if bs=14 then begin bes1:=16384;   role:=64;    end;
            if bs=16 then begin bes1:=32768;   role:=128;   end
                     else goto g1;end;
   16:begin if bs=16 then begin bes1:=16;      role:=1;     end
                     else goto g1;end;

   20:begin if bs=20 then begin bes1:=4;       role2:=128;  end
                     else goto g1;end;
   24:begin if bs=24 then begin bes1:=1;       role2:=64;   end
                     else goto g1;end;
  else goto g1;
  end;
g2:repeat
 gotoxy(20,24);write('7,11,10,8 VEYA 12 GÝRÝN');
  gotoxy(1,9);
  readln(gnd);
{}
   if ioresult=106 then begin
              gotoxy(20,24);write('ÇIKMAK ÝÇÝN EMÝN MÝSÝNÝZ (E/H)');
              if upcase(readkey)='E'then goto git2;end;
{}
  until (ioresult=0);
  clreol;
  gnda:=1;
  ekran;
  case gnd of
    7:begin if bs=14 then begin gnd1:=8388608+not(4096);
role1:=role+32;  end
                          else goto g2;end;
   11:begin if bs=14 then begin gnd1:=8388608+not(256);
role1:=role+8;   end
                          else goto g2;end;
   10:begin if bs=14 then begin gnd1:=8388608+not(512);
role1:=role+16;  end;
            if bs=20 then begin gnd1:=8388608+not(4096); role1:=32;
end
                          else goto g2;end;
    8:begin if bs=16 then begin gnd1:=8388608+not(4096);
role1:=role+32;  end
                          else goto g2;end;
    13:begin if bs=16 then begin gnd1:=8388608+not(4096);
role1:=role+4;  end                          else goto g2;end;
```

```
   12:begin if bs=16 then begin gnd1:=8388608+not(256);
role1:=role+8;   end;
            if bs=24 then begin gnd1:=8388608+not(4096); role1:=32;
end
                            else goto g2;end;
  else goto g2;
  end;
  port5:=role1;
  port4:=role2;

  acik1:=1;
  ekran;

    {BA³LANTI DURUMUNU  GÝRME VE KAYDIRMA}

g40:    dur:=0;
    for k:=1 to  bs do
      begin

g35:     ekran;
        if k=gnd then  if j=1 then b[j,k]:=1 else b[j,k]:=0;
        if k<>gnd then
        begin
        if k=bes then  if j=1 then b[j,k]:=0 else b[j,k]:=1
        else begin
        repeat;
            if k<14 then begin l1:=k*6-5;l2:=10;end
             else begin l1:=(k-13)*6-5;l2:=14;end;
          gotoxy(20,24);write('GÝRÝP ÝÇÝN 1 ÇIKIP ÝÇÝN 0 GÝRÝN');
          gotoxy(l1,l2+3);
          readln (b[j,k]);



                if ioresult<>0 then
           begin

          gotoxy(20,24);clreol;write('ÇIKMAK ÝÇÝN EMÝN MÝSÝNÝZ
(E/H)');
             if upcase(readkey)='E'then goto git2 else goto g35;
           end;

         ba[j,k]:=1;
        until (b[j,k]=0) or (b[j,k]=1);
        clreol;
        end;
```

```
        end;
        end;
{GÝRÝLEN DEªERLERÝN DEC DEªERÝ}
dec;

{GÝRÝLEN DEªERLERÝN KAYMIP DEC DEªERÝ  }
kaydec;

{PORT LAR}
portcev;

writeln;
writeln('PORT1 ',w1);writeln('PORT2 ',w2);writeln('PORT3 ',w3);

{Binary'e  çevirme}
binary;


{KAPATMA}
G6:
  clrscr;
  gotoxy(30,7);
  writeln('KAYDEDILSIN MI?(E/H)');
  kayit:=readkey;
  if upcase(kayit)='E' then begin
  clrscr;

  gotoxy(30,7);
  WRITE ('DIKKAT!');
  gotoxy(25,8);
  write('DOSYA KAYDOLUYOR...');
  seek(entegredos,a-1);
  write(entegredos,entegre);

  end;
end;
end;

{G21}

g21:if (buf=1) or (buf=3) then begin
if  (kar1=#0) and (kar2=#59) and (j<51) then j:=j+1;
kar1:=' ';
kar2:=' ';
repeat
seek(entegredos,a-1);
read(entegredos,entegre);
```

```
ekran;
gotoxy(30,18);write('DÜZELTME: F1');
gotoxy(30,20);write('SİLME: F2');
gotoxy(30,22);write('ÇIKIP: ESC');
delay(100);

{TUSLARLA HARAKET}

kar1:=READKEY;
     if keypressed  then begin
         kar2:=readkey;
                    if (ord(kar1)=0) and (ord(kar2)=73) and
(a<filesize(entegredos)) then a:=a+1;
          if (ord(kar1)=0) and (ord(kar2)=81) and (a>1) then a:=a-1;
                    if (ord(kar1)=0) and (ord(kar2)=72) and (j<>50)
then j:=j+1;
          if (ord(kar1)=0) and (ord(kar2)=80) and (j<>1) then j:=j-1;
end;
until (ord(kar1)=27) or ((ord(kar1)=0) and (ord(kar2)=59)) or
                        ((ord(kar1)=0) and (ord(kar2)=60));
          if (ord(kar1)=0) and (ord(kar2)=59)  then begin
               if (j=1) and (buf=3) then  j:=j+1;
               if (j=1) and (buf=1) then goto g20;
              goto g40;
              end;
          if (ord(kar1)=0) and (ord(kar2)=60)  then
           begin
            entegre.no:=0;
            entegre.bs:=0;
            entegre.bes:=0;
            entegre.gnd:=0;
            for h:=1 to 20 do for l:=1 to 10 do
              begin
                 entegre.isim[h,l]:=' ';
                 for j:=1 to 50 do
                 begin
                  for k:=1 to 24 do entegre.b[j,k]:=0;
                  entegre.port1[j]:=0;
                  entegre.port2[j]:=0;
                  entegre.port3[j]:=0;
                 end;
                entegre.port4:=0;
                entegre.port5:=0;
             end;
          seek(entegredos,a-1);
          write(entegredos,entegre);
          j:=1;
```

```
          goto g21;
          end;
          end;
  goto g4;


  {SECÝM1}
sec1:
clrscr;
a:=0;

{ÝSÝM GÝRME}
  repeat
  clrscr;
  gotoxy(30,7);write('ÝSÝM    ');
  l:=0;
  repeat
   l:=l+1;
   repeat
   gotoxy(36+l,7);
   read(isim1[l]);
   until (isim1[l]<>' ') and ((l<>1) or ((isim1[l]<>#13) and
(isim1[l]<>#10)) and (isim1[l]<>#0))  ;
   until (l=10) or (isim1[l]=#13) or (ioresult=106) or
(upcase(isim1[l])='Q') ;
   if upcase(isim1[l])='Q'  then goto g4;
   a:=0;
repeat
   a:=a+1;
   seek(entegredos,a-1);
   read(entegredos,entegre);
   h:=0;
   repeat
   h:=h+1;
   l:=0;
   d:=0;
      repeat
     l:=l+1;
     if entegre.isim[h,l]=isim1[l] then d:=d+1;
     until (l=10) or (entegre.isim[h,(l+1)]=' ') ;
     until    (h=21)
           or ((l=d) and (entegre.isim[h,l]=isim1[l])
           and (isim1[l+1]=#13));
     until (a=son)
           or ((l=d) and (entegre.isim[h,l]=isim1[l])
                     and (isim1[l+1]=#13));
  until (a<>son) and (h<>21) ;
```

```
g30:seek(entegredos,a-1);
read(entegredos,entegre);
  clrscr;
 j:=1;
  port[308]:=entegre.port5 ;  port[306]:=entegre.port4
;port[304]:=64   ;{port3.c port3.b port3.a}
  delay(1);                          {KONTROL}
  port[300]:=entegre.port3[j]  ;
  port[298]:=entegre.port2[j]  ;
  port[296]:=entegre.port1[j]  ;{port2.c port2.b port2.a}
  writeln('entegre giris    ',entegre.port3[j],' ',entegre.port2[j],'
'
  ,entegre.port1[j],' ',entegre.port5,' ',entegre.port4);
  port[304]:=128   ;                                  {port3.a}

  repeat
 j:=j+1;
DELAY(10);                                                {KONTROL}
  port[300]:=Entegre.port3[j]  ;
  port[298]:=entegre.port2[j]  ;
  port[296]:=entegre.port1[j]  ;

DELAY(1);
  l3:=port[292] ;
  l2:=port[290] ;
  l1:=port[288] ;
DELAY(1);                {port1.c port1.b port1.a}
  write( 'entegre giris ',i,'  ',entegre.port3[j],'
',entegre.port2[j],' ',entegre.port1[j]);
    gotoxy(40,wherey);
  q1:=128;
  a2:=entegre.port3[j];
  for 1:=1 to 8 do
    begin
      bb[1]:=trunc(a2 div q1) ;
      a2:=a2-bb[1]*q1;
      q1:=q1 div 2;
      write(bb[1]);
    end;
    write(' ');

   q1:=128;
  a2:=entegre.port2[j];
  for 1:=1 to 8 do
    begin
      bb[1]:=trunc(a2 div q1) ;
```

```
      a2:=a2-bb[1]*q1;
      q1:=q1 div 2;
      write(bb[1]);
    end;
    write(' ');

  q1:=128;
  a2:=entegre.port1[j];
  for 1:=1 to 8 do
    begin
      bb[1]:=trunc(a2 div q1) ;
      a2:=a2-bb[1]*q1;
      q1:=q1 div 2;
      write(bb[1]);
    end;
    writeln;

  write ('entegre cikis    ',13,' ',12,' ',11);
{GIRIS}
    gotoxy(40,wherey);
  q1:=128;
  a2:=13;
  for 1:=1 to 8 do
    begin
      bb[1]:=trunc(a2 div q1) ;
      a2:=a2-bb[1]*q1;
      q1:=q1 div 2;
      write(bb[1]);
    end;
    write(' ');


  q1:=128;
  a2:=12;
  for 1:=1 to 8 do
    begin
      bb[1]:=trunc(a2 div q1) ;
      a2:=a2-bb[1]*q1;
      q1:=q1 div 2;
      write(bb[1]);
    end;
    write(' ');

  q1:=128;
  a2:=11;
  for 1:=1 to 8 do
    begin
```

```
      bb[l]:=trunc(a2 div q1) ;
      a2:=a2-bb[l]*q1;
      q1:=q1 div 2;
      write(bb[l]);
    end;
    writeln;


  until ((entegre.port3[j+1]=0) and (entegre.port2[j+1]=0) and
(entegre.port1[j+1]=0))
       or ((l3<>entegre.port3[j]) or (l2<>entegre.port2[j]) or
(l1<>entegre.port1[j]));
   if ((l3<>entegre.port3[j]) or (l2<>entegre.port2[j]) or
(l1<>entegre.port1[j]))
       then write('HATALI') else write('DOGRU');
       port[308]:=0;port[306]:=0;port[304]:=64;
        for e:=1 to 1000 do write ;
        port[300]:=255;port[298]:=255;port[296]:=255;


   repeat
   cev:=readkey;
   until (cev=#27) or (cev=#13);
   if cev=#27 then goto sec1;
   if cev=#13 then goto g30 ;



  {SECÝM2 TANIMA TESTI}
sec2:
clrscr;

g31:  a:=0;
  buft1:=0;
  buft2:=0;
  repeat
  a:=a+1;
  seek(entegredos,a-1);
  read(entegredos,entegre);

  clrscr;
 j:=1;
  port[308]:=entegre.port5  ;  port[306]:=entegre.port4
;port[304]:=64   ;{port3.c port3.b port3.a}
 delay(10) ;                          {KONTROL}
  port[300]:=entegre.port3[j]  ;
  port[298]:=entegre.port2[j]  ;
```

```
  port[296]:=entegre.port1[j]  ;{port2.c port2.b port2.a}
  writeln('entegre giris kont ',entegre.port3[j],'
',entegre.port2[j],' '
  ,entegre.port1[j],' ',entegre.port5,' ',entegre.port4);
  port[304]:=128   ;                                    {port3.a}
DELAY(1);
repeat
 j:=j+1;                                                {KONTROL}
  port[300]:=Entegre.port3[j]  ;
  port[298]:=entegre.port2[j]  ;
  port[296]:=entegre.port1[j]  ;
  {port2.c port2.b port2.a}

DELAY(1);

  l3:=port[292] ;
  l2:=port[290];
  l1:=port[288];


              {port1.c port1.b port1.a}
  write( 'entegre giris j:',j,'  ',entegre.port3[j],'
',entegre.port2[j],' ',entegre.port1[j]);

  gotoxy(40,wherey);
  q1:=128;
  a2:=entegre.port3[j];
  for l:=1 to 8 do
    begin
      bb[l]:=trunc(a2 div q1) ;
      a2:=a2-bb[l]*q1;
      q1:=q1 div 2;
      write(bb[l]);
    end;
    write(' ');
   q1:=128;
  a2:=entegre.port2[j];
  for l:=1 to 8 do
    begin
      bb[l]:=trunc(a2 div q1) ;
      a2:=a2-bb[l]*q1;
      q1:=q1 div 2;
      write(bb[l]);
    end;
    write(' ');

   q1:=128;
```

```
  a2:=entegre.port1[j];
  for l:=1 to 8 do
    begin
      bb[l]:=trunc(a2 div q1) ;
      a2:=a2-bb[l]*q1;
      q1:=q1 div 2;
      write(bb[l]);
    end;
    writeln;

  write ('entegre cikis j:',j,'  ',l3,' ',l2,' ',l1);
{GIRIS}
    gotoxy(40,wherey);
    q1:=128;
  a2:=l3;
  for l:=1 to 8 do
    begin
      bb[l]:=trunc(a2 div q1) ;
      a2:=a2-bb[l]*q1;
      q1:=q1 div 2;
      write(bb[l]);
    end;
    write(' ');

      q1:=128;
  a2:=l2;
  for l:=1 to 8 do
    begin
      bb[l]:=trunc(a2 div q1) ;
      a2:=a2-bb[l]*q1;
      q1:=q1 div 2;
      write(bb[l]);
    end;
    write(' ');

    q1:=128;
  a2:=l1;
  for l:=1 to 8 do
    begin
      bb[l]:=trunc(a2 div q1) ;
      a2:=a2-bb[l]*q1;
      q1:=q1 div 2;
      write(bb[l]);
    end;

   writeln;
```

```
  until (entegre.port3[j+1]=0) and (entegre.port2[j+1]=0) and
(entegre.port1[j+1]=0)
        or ((l3<>entegre.port3[j]) or (l2<>entegre.port2[j]) or
(l1<>entegre.port1[j]));
   if ((l3<>entegre.port3[j]) or (l2<>entegre.port2[j]) or
(l1<>entegre.port1[j]))
        then
        {writeln('HATALI',i,' ',j)}
        else begin {writeln('DOGRU','  ',j);}
        buft1:=1;end;
   if ((entegre.port3[1]=0) and (entegre.port2[1]=0) and
(entegre.port1[1]=0))
        then buft2:=1;
   until (buft1=1) or  (a=son-1);
  if buft1=1 then begin
    for s1:=1 to 20 do
    for s:=1 to 10 do begin  {   2gotoxy((10*s1)+s,15);
}write(entegre.isim[s1,s]);end;
       end
       else begin;writeln('YOK');end;
      port[308]:=0;port[306]:=0;port[304]:=64;
      for e:=1 to 1000 do write ;
      port[300]:=255;port[298]:=255;port[296]:=255;


   repeat
   cev:=readkey;
   until (cev=#27) or (cev=#13);
   if cev=#27 then goto g4;
   if cev=#13 then goto g31;


g9:
end.


{SON}
```

ÖZGEÇMİŞ

Doğum Tarihi	: 26 Aralık 1970
Doğum Yeri	: Beykoz
İlkokul	: Ortaçeşme Hacı Numan İlkokulu 1976-1981
Ortaokul	: Beykoz Ziya Ünsel Ortaokulu 1981-1984
Lise	: Paşabahçe Ferit İnal Lisesi 1984-1987
Üniversite	: Yıldız Üniversitesi ,Mühendislik Fakültesi Elektronik ve Haberleşme Mühendisliği 1987-1991
Yüksek Lisans	: Yıldız Teknik Ünivversitesi, Fen Bilimleri Enstitüsü, Elektronik Programı 1991-