www.ieeeturkiye.wordpress.com adına yüklenmiş olup toplumları geliştiren bilginin herhangi bir şekilde ulaşılaşılamaz olmasını kabullenemeyen kişi veya kişiler tarafından upload edilmiştir.saygılarımzla...

Bilgisayar ve Elektronik Mühendisliğinde

Lojik Devre Tasarımı

Dr. Taner ARSAN ve Dr. Rifat ÇÖLKESEN

PAPATYA YAYINCILIK
İstanbul, Ankara, İzmir, Adana

© PAPATYA YAYINCILIK EĞİTİM
Bilgisayar Sis. San. ve Tic. A.Ş.
Ankara Caddesi, Prof. Fahreddin Kerim Gökay Vakfı İşhanı Girişi,
No: 11/3, Cağaloğlu (Fatih)/İstanbul

Tel : (0 212) 527 52 96 - (0 532) 311 31 10
Faks : (0 212) 527 52 97
e-Posta : admin@papatya.gen.tr
Web : http://www.papatya.gen.tr

Lojik Devre Tasarımı - Taner ARSAN ve Rifat ÇÖLKESEN

3. Basım Ocak 2013

Yayın Danışmanı : Dr. Cengiz UĞURKAYA (Post-Edu Institute)
Türkçe Dil Uzmanı : M. Çetin ALKIŞ
Üretim : Olcay KAYA
Öğrenci Gözüyle : Batuhan AVCI (Bilgisayar Mühendisliği)
Satış : Mustafa DEMİR
Sayfa Düzenleme : Papatya - Kelebek Tasarım
Kapak Tasarım : Papatya - Kelebek Tasarım
Basım ve Ciltleme : Altan Basım San. ve Tic. Ltd. Şti. (Sertifika No:11968)
Matbaacılar Sit. 222/A Bağcılar/İstanbul (0 212 629 03 74)

© Bu kitabın her türlü yayın hakkı Papatya Yayıncılık Eğitim A.Ş.'ye aittir. Yayınevinden yazılı izin alınmaksızın alıntı yapılamaz, kısmen veya tamamen hiçbir şekil ve teknikle ÇOĞALTILAMAZ, BASILAMAZ, YAYIMLANAMAZ. Kitabın, tamamı veya bir kısmının fotokopi makinesi, ofset gibi teknikle çoğaltılması, hem çoğaltan hem de bulunduranlar için yasadışı bir davranıştır.

Lütfen kitabımızın fotokopi yöntemiyle çoğaltılmasına engel olunuz. Fotokopi hırsızlıktır.

Arsan, Taner ve Çölkesen, Rifat.
Lojik Devre Tasarımı / Arsan, Taner - Çölkesen, Rifat – İstanbul: Papatya Yayıncılık Eğitim, 2013.
xii, 400 s. ; 24 cm.
Kaynakça ve dizin var.
Sertifika No: 11218
ISBN 978-975-6797-07-X
1. Dijital Devre Tasarımı. 2. Sayısal Devre Tasarımı. 3. Bilgisayar Donanım. 4. Sayı Sistemleri.
I.Title.

Üniversite öğrencilerimizin ve mühendislerimizin lojik devre tasarım konusundaki bilgilerinin gelişmesine yararlı olacağına inandığımız bu kitabımızı; Çalışma titizliği ve kişilikleriyle bizleri derinden etkileyen doktora hocalarımız

Prof. Dr. Atilla BİR

ve

Prof.Dr. Bülent ÖRENCİK'e

ithaf ediyoruz.

Teşekkür

Kitabın hazırlanması sırasında daha iyi olması için görüşlerini söyleyen ve eleştiren hocalarımız ve meslektaşlarımız;

Sayın,

Prof.Dr. Atilla BİR'e,

Prof.Dr. Bekir KARAOĞLU'na

Prof.Dr. Bülent ÖRENCİK'e,

Prof.Dr. Gökhan UZGÖREN'e

Prof.Dr. İbrahim EKSİN'e,

Doç.Dr. Metin GÖKAŞAN'a,

Doç.Dr. Müjde GÜZELKAYA'ya

Deneyler kısmının ve benzetim ortamında lojik devre tasarım bölümünün hazırlanmasında emeği geçen ve birkaç tane örnekle destekleyen;

Sayın,

Murat BAŞKAN'a.

Tüm bölümleri öğrenci gözüyle okuyan ve görüşlerini söyleyen;

Sevgili,

Ayşenur KARAYAĞIZ'a,

Olcay KAYA'ya.

Benzetim ortamı için gerekli yazılımı sağlayan

ERA firmasına,

teşekkür ederiz. Hoşgörü ve sevgi her zaman sizinle olsun; başarı ve mutluluk arkasından gelir.

Taner ARSAN

Rifat ÇÖLKESEN

İçindekiler

Önsöz 11

Bölüm 1 **Lojik Devre Tasarım Dünyası** 13

Analog İşaret & Sayısal Veri 14
Kodlama & Kod Tabloları 14
Kombinezonsal Devre & Ardışıl Devre 15
Kanonik Model & Algoritmik Model 16
Tasarım Adımları ve Üretim 16
Transistör & Kapılar 17
Saklayıcı & Bellek 18
Teknoloji: TTL ve CMOS 19
Tasarım ve Benzetim Ortamları 19
Maliyet 20
Kart & ASIC 20
Kenar & Düzey Tetiklemesi 20
bit, *nipple* ve Byte (birli, dörtlü, sekizli) 21
Analog Sayısal Dönüşüm (ADC, DAC) 21
Donanım Tanımlama Dili (HDL) 21
Sorular 22

Bölüm 2 **İşaretler ve Analog/Sayısal Dönüşüm** 23

2.1. İşaretlerin Sınıflandırılması 23
Sayısal İşaret 25
2.2. Analog/Sayısal Dönüşüm 27
2.2.1. ADC'nin Mimarisi 28
Örnekleme ve Tutma İşlemleri 28
Kuantalama 29
Kodlama 30
2.2.2. DAC'ın Mimarisi 32
2.3. Özet 33
2.4. Sorular 34

Bölüm 3 **Sayı Sistemleri** 35

3.1. Çeşitli Sayı Sistemleri 36
İkili Sayı Sistemi 37
İkili Sayılar Üzerinde Aritmetik İşlemler 38
Tümleyen Aritmetiği 38
Sekizli Sayı Sistemi (Octal) 39
Onaltılı Sayı Sistemi (Hexadecimal) 39

BCD Sayı 39
Sayıların Bilgisayarda Gösterilimi 40
Bir Sayının Genel İfadesi 42
3.2. Sayı Sistemleri Arasında Dönüşüm Kuralları 42
3.2.1. *n* Tabanından 10 Tabanına Dönüşüm 42
3.2.2. 10 Tabanından *n* Tabanına Dönüşüm 45
3.2.3. *m* Tabanından *n* Tabanına Dönüşüm 50
3.2.3.1. 2 Tabanından 8/16 Tabanına Dönüşüm 52
3.3. İşaretli Sayıların Gösterilimi 53
3.3.1. İşaretli Sayılarda Tümleme Aritmetiği 53
3.3.1.1. Taban'a Göre Tümleme 53
3.3.1.2. (Taban-1)'e Göre Tümleme 55
3.3.2. Çıkarma İşleminin Tümleyen Aritmetiğiyle Yapılması 56
3.3.3. Çıkarmanın (Taban-1)'e Tümleyen Aritmetiğiyle Yapılması 58
3.4. Özet 60
3.5. Sorular 60

***Bölüm 4* Kodlama 63**

4.1. Kodlamanın Tanımı 64
4.2. Sayısal Kodlama 66
4.2.1. İkili Kodlanmış Ondalık (BCD) Gösterim 67
BCD Sayılar Üzerinde Aritmetik İşlemler 68
BCD Toplama 68
BCD Çıkarma 69
4.2.2. Üç Fazlalık Kodu (Excess-3) 70
Üç Fazlalık Kodunda Aritmetik İşlemler 70
4.2.3. *Aiken* Kodu 71
Aiken Kodunda Aritmetik İşlemler 72
4.2.4. Bitişik Kodlar ve *Gray* Kodu 73
Hamming Uzaklığı 74
Gray Kodu 75
4.3. Hata Sezme ve Onarma 76
4.3.1. Eşlik Biti Ekleme 76
4.3.2. Boyuna Fazlalık Sınaması - LRC 77
4.3.3. Çevrimli Fazlalık Sınaması - CRC 79
4.3.4. *Hamming* Kodlaması 81
4.4. Alfanumerik Kodlar 82
ASCII 83
Unikod (Unicode) 84
Türkçe Karakter Tablosu 84
4.5. Özet 85
4.6. Sorular 86

***Bölüm 5* Lojik Devre Temelleri 87**

5.1. Lojik İşlemlerin Temeli 88
5.1.1. VE İşlemi (AND) 88
5.1.2. VEYA İşlemi (OR) 90
5.1.3. TÜMLEME İşlemi (NOT) 91
5.2. Türetilen İşlemler 93
5.2.1. TÜMLEYEN-VE (TVE) İşlemi (NAND) 93
5.2.2. TÜMLEYEN-VEYA (TVEYA) İşlemi (NOR) 94
5.2.3. YA DA İşlemi (XOR) 95
5.2.4. EŞDEĞERLİK (TYA DA) İşlemi (XNOR) 96
5.3. Temel Lojik İşlemlerinin Kümeler Cebri İle Gösterilişi 97
5.3.1. Küme Kavramı 97
5.3.2. Kümeler Cebrinde Kesişim 99
5.3.3. Kümeler Cebrinde Birleşim 99
5.3.4. Kümeler Cebrinde Tümleme 99
5.3.5. Kümeler Cebrindeki Diğer Kavramlar 100
5.4. *Boole* Cebrinin Aksiyom ve Teoremleri 100
5.4.1. *Boole* Cebri Aksiyomları 101
5.4.2. *Boole* Cebri Teoremleri 101
5.5. *Boole* Cebri Fonksiyonları ve Standart Biçimler 107
Minimum ve Maksimum Terimler 107
5.5.1. Minimum Terimler Kanonik Biçimi 108
5.5.2. Maksimum Terimler Kanonik Biçimi 110
5.6. Kanonik Biçimler Arasındaki Dönüşüm 112
5.7. Doğruluk Tablosundan Kanonik Biçimlerin Bulunması 113
5.8. Kanonik Biçimlerin Lojik Kapılarla Gerçeklenmesi 116
5.9. Özet 118
5.10. Sorular 118

***Bölüm 6* Lojik Fonksiyonların İndirgenmesi 121**

6.1. Görüşe Dayanarak İndirgeme 122
6.2. *Karnaugh* Diyagramı Yöntemi 124
Karnaugh Yönteminde Komşuluk Kavramı 129
6.3. *Quine-Mc Cluskey* Yöntemi 139
6.4. Eksik *Boole* İşlevleri 144
6.5. Türetilmiş Kapılarla Temel Lojik Kapıların Yapılması 146
TÜMELEME Kapısı 146
VE Kapısı 146
VEYA Kapısı 147
6.6. İndirgenmiş İfadelerin Aynı Tür Kapılarla Gerçekleştirilmesi 148
Çarpımlar Toplamı - TVE Tasarımı 148
Çarpımlar Toplamı - TVEYA Tasarımı 149
Toplamlar Çarpımı - TVE Tasarımı 151
Toplamlar Çarpımı - TVEYA Tasarımı 152
6.7. Özet 153
6.8. Sorular 154

Bölüm 7 **Kombinezonsal Devreler** 159

7.1. Kombinezonsal Devreler 160
7.2. Kombinezonsal Devre Elemanları 161
7.3. Kombinezonsal Devre Tasarımı 162
7.4. Kombinezonsal Tümleşik Devreler 167
7.4.1. Aritmetik Toplama/Çıkarma Devreleri 167
Yarı Toplayıcı (Half Adder) 168
Tam Toplayıcı (Full Adder) 168
Yarı Çıkarıcı (Half Subtractor) 170
Tam Çıkarıcı (Full Subtractor) 171
Toplama/Çıkarma Devresi (Adder and Subtractor) 173
7.4.2. Seçiciler (Multiplexer-MUX) 174
7.4.3. Dağıtıcılar (Demultiplexer-DEMUX) 179
7.4.4. Kod Çözücüler (Decoder) 182
7.4.5. Kodlayıcı (Encoder) 184
7.4.6. ALU Tasarımı (Arithmetic Logic Unit) 186
7.4.7. 7-Parçalı Gösterge (7-Segment Display) 188
7.5. Çıkış Bağlantı Şekilleri 191
7.6. Özet 193
7.7. Sorular 193

Bölüm 8 **Maliyet Faktörü ve Karmaşıklık Hesabı** 201

8.1. Maliyet Hesabı 202
8.2. Karmaşıklık Hesabı 204
Zaman Karmaşıklığı/Çalışma Hızı 205
Eleman Karmaşıklığı 208
8.3. Büyük O Notasyonu ve Karmaşıklık 210
8.4. Özet 211
8.5. Sorular 211

Bölüm 9 **Programlanabilir Kombinezonsal Devreler** 213

9.1. Programlanabilir Kombinezonsal Devreler-PLD 214
9.2. Programlanabilir Kombinezonsal Devre Elemanları 215
9.2.1. PROM'un İç Yapısı 217
9.2.2. PAL'in İç Yapısı 219
9.2.3. PLA'nın İç Yapısı 223
9.3. Özet 225
9.4. Sorular 226

Bölüm 10 **Ardışıl Devre Temelleri** 229

10.1. Ardışıl Devrelerin Genel Yapısı 230
10.1.1. Mealy ve Moore Makinaları 231
10.1.2. Asenkron & Senkron Ardışıl Devreler 232
10.1.3. Durumlar ve Durum Diyagramları 234

Durum Tablosu, Durum Diyagramı 235
Sonlu Durum Makinaları 238
Durum İndirgeme ve Durum Atama 241
Eşdeğerlik Tablosu ile Durum İndirgeme 241
10.1.4. Standart Tasarım Birimleri 245
10.1.5. Kenar ve Düzey Tetikleme 248
10.1.6. Saat İşareti ve Senkron Hücrelerin Zaman Davranışı 250
Senkron Ardışıl Devrelerde Zamanlama Parametreleri 251
Senkron Hücrenin Zaman Diyagramı 252
10.2. Flip-Flop'lar 253
Ana-Uydu Flip-Flop 257
10.3. Tutucular (Latches) 258
10.4. Ardışıl Lojik Devre Analiz Yöntemi 260
10.5. Ardışıl Devre Tasarım Yöntemi 266
10.6. Özet 274
10.7. Sorular 274

Bölüm 11 **Saklayıcı, Sayıcı ve Bellek Elemanları** 277

11.1. Saklayıcılar 278
Ötelemeli Saklayıcı (Shift Register) 279
PIPO, PISO, SIPO, SISO ve Üniversal Saklayıcı 280
11.2. Sayıcılar 282
İkili, Ondalık, *Gray*-Kod, Halka, *Modulo*, *Ripple* 284
11.3. Bellekler 287
RAM , ROM , PROM , EPROM , E2ROM CAM 289
11.4. Programlanabilir Ardışıl Devreler 298
11.5. Özet 300
11.6. Sorular 301

Bölüm 12 **Ardışıl Devre Tasarım Yöntemleri** 303

12.1. Kanonik Yaklaşım/Tasarım Yöntemi 304
12.2. Modüler Yaklaşım/Tasarım Yöntemi 307
12.3. Algoritmik Yaklaşım/Tasarım Yöntemi 314
12.4. Özet 323
12.5. Sorular 323

Bölüm 13 **Benzetim Ortamında Lojik Devre Tasarımı** 327

13.1. multiSIM Programının Özellikleri 328
multiSIM Kullanıcı Arayüzü 329
multiSIM Tasarım Araç Çubuğu 331
multiSIM Kullanıcı Arayüzünü Özelleştirme 332
multiSIM Kullanıcı Tercihlerinin Tanımlanması 334
multiSIM Benzetim Ortamı Arayüzü 337
13.2. Lojik Devrelerin Benzetimi 339
13.3. Örnek Çalışmalar 341

Lojik Düzey Algılayıcısı 342
Lojik Probe 342
4-bit BCD Sayıcı ve 7-parçalı gösterge 344
4-bitlik İkili Yukarı Sayıcı 344
13.4. Lojik Devre Şemasından Baskılı Devreye Geçiş 345
13.5. Özet 347
13.6. Sorular 347

Ek A **Lojik Devre Katalog Bilgileri** **349**

Kombinezonsal Devreler 351
Tümleşik Kombinezonsal Devreler 351
Ardışıl Devreler 353
Bazı Tümdevrelerin Şematik Gösterimleri 354

Ek B **TTL ve CMOS Tümdevre Özellikleri** **357**

TTL Tümdevre Özellikleri 357
Propagasyon Gecikmesi 358
CMOS Tümdevreler 359
Propagasyon Gecikmesi 360
TTL ve CMOS Karşılaştırılması 360

Ek C **Lojik Devreler Lab. Deney Önerileri** **361**

Deney 1. Lojik Kapılar ve Fonksiyon İndirgeme 362
Deney 2. TTL ve CMOS Özelliklerinin Çıkarılması 364
Deney 3. Kombinezonsal Devre Tasarımı 367
Deney 4. Aritmetik İşlem Devreleri 369
Deney 5. PLD Uygulamaları 371
Deney 6. İşaret Üreteçleri 374
Deney 7. Üç-Durumlu Çıkış ve Ortak Yol Kullanımı 376
Deney 8. Saklayıcı ve Sayıcı Uygulamaları 378
Deney 9. Ardışıl Devre Uygulaması 380
Deney 10. ADC Uygulamaları 382

Kaynakça **387**

Yazarlarımız **389**

Dizin **395**

Önsöz

Eskilerin mantık olarak adlandırdıkları lojik, kesin usavurum ilkelerini konu edinen ve temelleri antik felsefeye dayanan çok eski bir bilim dalıdır. Çağdaş bilim ve teknolojinin rasyonel ilkeleri bu geleneksel batı felsefesinde gelişerek bugünkü tartışılmaz konumuna ulaşmıştır. Çağdaş teknolojide karmaşık sistemlerin tümü, bir dizi mantıksal işlevleri gerçekleştiren işlem ya da aygıtlardan oluşur. Bu mantıksal işlemleri temsil eden yapı elemanları, her ne kadar artık bundan yirmi sene önceki toplu parametreli elektronik devrelerle gerçekleştirilmese de, hala daha lojik devreler olarak adlandırılır. Lojik devreler, günümüz teknolojisinde sayısal bir ortamda modellenir ve değerlendirilir. Bu amaca yönelik olarak, çağdaş mühendisler fiziksel olaylar arasındaki etkileşimleri tanımlayıp yeterince gerçeğe uygun modelleyebilmeli, temel bağlantılar arasındaki ilişkileri kurabilmeli ve şüphesiz ki gün geçtikçe karmaşıklaşan sistemleri rasyonel ve güvenilebilir bir şekilde tasarlayabilmelidir. Sürekli gelişen ve her gün biraz daha ucuzlayan kişisel bilgisayarlar (PC), programlanabilir lojik kontrolörler (PLC) ve bunlara yüklenen gelişmiş yazılımlar yukarıda belirtilen tüm işlem ve tasarımların, daha kolay ve hassas bir şekilde gerçekleştirilebilmesini sağlar.

Lojik devre tasarımını konu edinen bu kitap, yukarıda ifade edilen şekliyle öğrenmek ve bildiklerini uygulamak isteyen Elektrik, Elektronik, Bilgisayar, Endüstri ve Makina Mühendisleri ile, bu mühendislik dallarında lisans seviyesinde eğitim gören, tüm mühendis adayları ve Meslek Yüksekokul Öğrencilerine yöneliktir. Yazarların, sıkıcı akademik bir yaklaşımı benimsemek yerine, uygun yerlere çok sayıda özgün tasarım ve uygulama örnekleri ekleyerek konuyu uygulamalı akademik yaklaşımla ilginç kıldıkları görülmektedir. Bu yaklaşım, ülkemizde lojik devre tasarımında bir ilki temsil etmektedir. Bu kolay okunabilir eseri Öğrenci, Mühendis ve Araştırmacıların hizmetine sunmuş ve bu konudaki büyük bir eksikliği gidermiş olmalarından dolayı yazarlarını tebrik ederim.

Prof. Dr. Atilla Bir*

* Öğretim Üyesi, İTÜ Elektrik Elektronik Fakültesi Kontrol ve Kumanda Sistemleri AnaBilim Dalı Başkanı.

1.

Lojik Devre Tasarım Dünyası

Lojik devre tasarımı sayısal devre tasarımı konusuna girer; sayısal sistemlerde tüm işlemler sonlu sayıda ayrık değerler üzerinde gerçekleştirilir; aynı bir merdivenin basamakları gibi... Yukarı çıkabilmek veya aşağıya inebilmek için merdiven basamakları düzeyinde adım atılması gerekir; aksi durumda aynı basamakta kalınır. Doğada oluşan hemen hemen tüm fiziksel değişiklikler için ise "analog"tur denilebilir. Analog sistemlerde giriş ve çıkış değerlerinde kesintisiz bir süreklilik vardır. Bu bir karayoluna benzetilebilir; yol dolambaçlı olabilir, ancak ani sıçramalar yoktur ve süreklilik gösterir. Fiziksel değişikliklerin, doğada analog olmasına karşın, günümüzde bilgisayar dahil kamera, müzik sistemleri, saat, endüstriyel cihazlar, ev cihazları gibi hemen herşey sayısal tabanlı tasarlanıp üretilmektedir. Çünkü sayısal tabanlı sistem geliştirmenin birçok olumlu yanı, esnekliği ve kolaylığı vardır.

Sayısal sistemler, analog sistemlere göre birçok açıdan avantajlıdır. Herşeyden önce hata oluşma olasılığı daha azdır denilebilir ve hatalarda düzeltme yapılması kolaydır; ve, bilgi sayısal hale dönüştürüldükten sonra üzerinde her türlü işlem, olabildiğince hızlı ve kesin sonuçla bitirilecek şekilde gerçekleştirilebilir:

- Hata olasılığı zayıf; olursa da sezilmesi ve düzeltilmesi kolay.
- Daha hassas hesaplamalar yapılabilir.
- Sayısallaştırılmış bilgi (veri) kolay işlenir.
- Veri her şekilde ve uygulamada kullanılabilir.
- Verinin saklanması, büyük miktarlarda bile olsa daha kolaydır.
- Verinin sistemleri arası iletişimi kolaylaşır ve esnekleşir.
- Mikroelektronik teknolojisindeki gelişmeler her geçen gün daha performanslı ve ekonomik tasarımlar yapılmasına olanak vermektedir.

Lojik devre tasarımı, herhangi bir işi sayısal olarak kotaracak sistemlerin tasarımıyla ilgilenir; burada söz konusu olan 2 tabanında sayı sistemidir ki buna kısaca ikili sayı sistemi denir. İkili sayı sisteminde bir durumu ifade edebilmek için, en

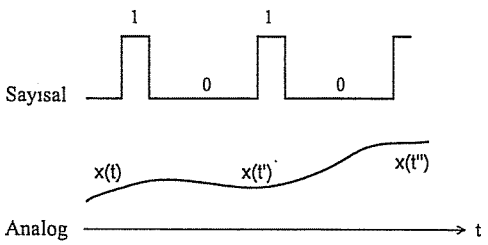
temelde, "var" ve "yok" ifadeleri kullanılır; ilki lojik 1, ikincisi lojik 0 değerine karşı düşer. Herşey bu iki ifadenin kombinezonuyla elde edilir.

Lojik devre tasarımı basitten karmaşığa birçok sayısal devrenin tasarımını içerir. En basitinden iki nesnenin varlığını tespit eden bir lojik VE kapısı da bir lojik devredir; bir kapıdan içeri girenleri sayan devre, sayısal saat veya bilgisayarın içerisindeki işlemci de birer lojik devredir.

Lojik devre tasarımında çeşitli yaklaşım yöntemleri ve sınıflamalar vardır. Tasarlanacak sistemin yalınlığına veya karmaşıklığına göre en uygun yaklaşım yöntemi seçilmelidir. Böylece, en optimum tasarım yapılmış olunur. Ters yönde bir yaklaşım seçilmesi hem tasarım maliyetini hem de üretim maliyetini oldukça arttırabilir.

Analog İşaret & Sayısal Veri

Analog işaret, bulunduğu aralık içerisinde her değeri alabilen ve süreklilik özelliği olan fiziksel işaretlerdir; doğadaki fiziksel olayların hemen hepsi analog işaret şeklinde ortaya çıkmaktadır denilebilir: rüzgarın esmesi, bir ortamın ısı değişimi, bir cismin hareket şekli, bir canlının yaşam evreleri gibi herşey analog işaret şeklindedir. Bir kalemi kağıt üzerine koyup soldan sağa veya sağdan sola kesintisiz olarak birşeyler çizildiğinde tipik bir analog işaret elde edilmiş olunur. Sayısal işaret ise, belirli bir aralık içerisinde atlamalı değerler alabilen işaretlerdir.



En çok bilenen sayısal işaret ikili (*binary*) olanıdır; ikili işarette yalnızca iki farklı değer vardır. Bu iki değer, genel olarak, 1/0, darbe/boşluk, H/L (*High/Low*), açık/kapalı, ver/yok vs. şeklinde isimlendirilmektedir. Bir kapının açık veya kapalı olduğu gösteren işaret sayısal iken, kapının açılması veya kapatılması durumunu gösteren işaret analog'tur. Yukarıda, sayısal ve analog işaretlerin grafiksel şekli gösterilmiştir. Görüleceği gibi sayısal işarette yalnızca 0 ve 1 düzeyleri var iken, analog işarette her düzey vardır.

Kodlama & Kod Tabloları

Kodlama, bilgisayar veya benzeri sayısal ortamlarda bilgiyi/veriyi tutmak için kullanılan bir simge atama işlemidir. Bu amaçla kod tablosu kullanılır; bu tablolarda herbir simgeye karşılık gelecek ikili bir sayı karşı düşürülür. Çünkü sayısal sistemlerde simgeler değil ikili tabanda sayılar saklanabilmektedir. Böylece ikili tabanda sayı olmayan A, B, C gibi alfabe harfleri ve @, Ω, π, τ, ☺ ve √ gibi alfabe

dışı karakterler sayısal sistemlerde saklanabilir ve işlemlerde kullanılabilir. Örneğin, ASCII olarak adlandırılan kod tablosunda 128 tane karakter vardır[1] ve bu tablo aracılığıyla herhangi bir yazı ve simge bilgisayar ortamında saklanabilir. Örneğin, ASCII'ye göre 65 sayısı A, 66 sayısı B harflerine ve 97, 98, 33 sayıları da sırasıyla a, b ve ! harf/karakterlerine karşılıktır. Bu durumunda *Baba!* sözcüğünü sayısal ortamda saklamak için 65, 97, 98, 97 ve 33 sayılarının saklanması yeterlidir.

Kod	10 tabanında	2 tabanında	Baba! sözcüğünün saklanması
...	...	...	
!	33	010 0001	
A	65	100 0001	Baba! ⇨ 65 97 98 97 33 ⇨
B	66	100 0010	
C	67	100 0011	1000001 1100011 1100010 10 0011 010 0001
...	...	...	B a b a !
a	97	110 0001	
b	98	110 0010	
...	...	...	

Eğer kodlara verilecek ikili sayıların mutlak değerleri doğrudan kodlarla aynı ise kod tablosu kullanılmadan doğrudan kodlama yapılır. Örneğin 7 sayısına ait kod doğrudan 7 sayısının ikili karşılığı olan 0111 veya 65535 sayısına ait kod doğrudan bu sayının ikili karşılığı olan 1111111111111111 ise kod tablosu kullanılmadan doğrudan ikili karşılıklarında kodlama yapılır. Bilgisayar sistemlerinde sayılar doğrudan kodlama, harfler ve özel karakterlerde kod tablosu üzerinden kodlama yapılarak tutulurlar.

Kombinezonsal Devre & Ardışıl Devre

Bu bir sınıflamadır; sayısal sistemler, genel olarak, biri kombinezonsal diğeri ardışıl devre olmak üzere iki sınıf altında toplanırlar. Kombinezonsal devrelerde t anındaki çıkışlar yine t anındaki girişlere bağlıdır; çıkış ile giriş arasında yalnızca küçük bir zaman dilimi gecikmesi (τ_g) vardır.

$z(t+\tau_g)=G(x(t))$

► z: çıkış, G: geçiş, x giriş fonksiyonlarıdır.

[1] ASCII, ilk tanımlamasında 7 bitliktir; dolayısıyla 2^7=128'den 128 tane karakter vardır. Bu karakterlerden ilk 33 tanesi kontrol karakterleri (veya yazılamayan karakterler) olarak adlandırılır. Diğerleri ise, alfabedeki harfler ve özel karakterlerdir. Daha sonra ASCII'nin 8 bitlik uyarlaması tanımlanmıştır; bu genişletilmiş ASCII olarak bilinir ve 2^8=256'den 256 tane karakter içerir. Genişletilmiş ASCII'de ikinci yarıdaki 128 karakter genelde grafiksel karakterler ve İngiliz alfabesinde olmayan, ancak başka dillerde olan Ç, ş, Ü gibi karakterlerdir.

Ardışıl devrelerde ise, t anındaki çıkışlar t anındaki girişlere ek olarak t anından önceki girişlerin etkisine de bağlıdır; bu tür devrelerde geçmiş bilgisini tutmak için bellek veya saklayıcı gibi bilgi tutan birimlere ihtiyaç vardır.

$z(t)=G_1(x(t), d(t))$

$d(t+1)=G_2(x(t), d(0..t))$

▶ d: durum, G_1: çıkış geçiş, G_2: durum geçiş fonksiyonlarıdır; $d(t)$: t anındaki durumu, $d(0..t)$: tüm geçmiş durumları göstermektedir.

Kanonik Model & Algoritmik Model

Bunlar birer yaklaşım şeklidir; sayısal sistemler, temelde, biri kanonik model diğeri algoritmik model olmak üzere iki yaklaşım yönteminden biri kullanılarak tasarlanır. Kanonik model, daha çok, basit uygulamalar için tutulan yol olurken, algoritmik model karmaşık sayısal sistemlerin tasarımında kullanılır.

Algoritmik modele dayanan tasarımlar daha karmaşıktır; model, temel işleri yapan donanımsal yapı üzerine program benzeri bir süreç ekleyip daha karmaşık işlerin temel işlemlerle kotarılmasına dayanır. Örneğin, toplama, çıkarma, tümleme ve öteleme gibi basit işlemlerle oldukça karmaşık işlemler yapılabilir. Bu dört temel işlem bir algoritmik ifade altında yürütülürse karekök alma, standart sapma vs. gibi hesaplamalar gerçekleştirilebilir. Algoritmik modele dayalı tasarımlarda kullanılan temel birimler şunlardır:

- Saklama/bellek birimleri
- Operatör anlamında işlem birimleri
- Denetim işareti üreteçleri

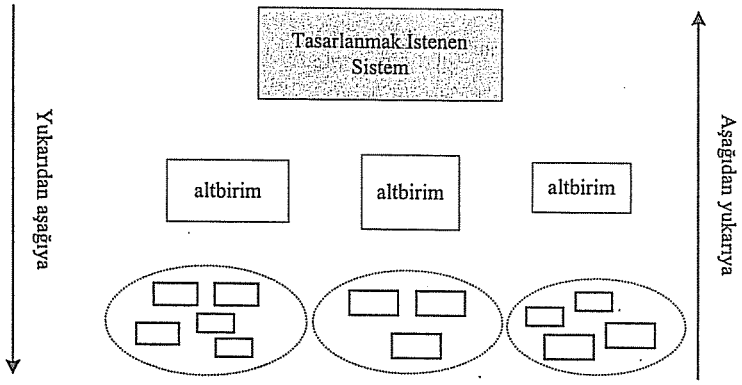
Kanonik modele göre tasarım, bu işin başlangıcıdır denilebilir. Giriş değerleri ve bunlara karşı üretilen çıkış değerleri vardır; bunu sağlayan devrede algoritmik bir ifade olmayıp doğrudan *Boole* cebri ifadeleri kullanılır. Kanonik modele göre tasarımı yapılmış devreler algoritmik tasarımında birer alt devre olarak kullanılabilir.

Tasarım Adımları ve Üretim

Herşeyden önce,

1. Tasarlanacak devrenin fonksiyonu açık ve net olarak tanımlanmalıdır.
2. Daha sonra tasarımı yapılır.
3. Analiz edilir; analiz sonuçları başlangıçta verilen tanıma uyuyorsa,
4. gerçekleştirilir ve
5. test aşamasına geçilir.

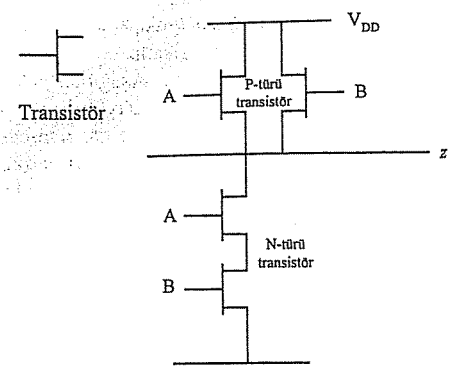
Tasarımda, biri yukarıdan aşağıya (*top-down*), diğeri aşağıdan yukarıya (*down-up*) doğru adlandırılan iki yöntemden biri kullanılır. İlkinde sistem alt parçalara/birimlere ayrılır ve tasarımı kolay birimler haline dönüştürülür. Aşağıdan yukarıya yaklaşımdaysa, en küçük tasarım biriminden başlanarak daha büyük birimler oluşturulur; onlardan da daha büyük birimler ve en son hiyerarşide de tasarlanmak istenen sistem ortaya çıkarılır. Heriki yönteminde kolaylıkları ve zorlukları vardır. İstenirse, heriki yaklaşım birleştirilerek kullanılabilir; İşe yukarıdan aşağıya yaklaşımla başlanır; belirli bir aşamaya gelindiğinde aşağıdan yukarı yöntemi kullanılır ve orta bir yerde buluşulur...



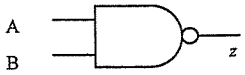
Transistör & Kapılar

Transistör, aktif bir elektronik elemandır ve sayısal tümdevrelerin en temel birimidir; bir tümdevrenin büyüklüğü, çoğu zaman, içerdiği transistör sayısıyla ifade edilir. Örneğin bir tümdevre bin, 10 bin, 500 bin, 1 milyon, 10 milyon transistör içerebilir; mikroelektronik teknolojilerindeki gelişmeler her geçen gün bir tümdevre içerisine yerleştirilebilecek transistör sayısının artmasını sağlamaktadır. Böylece, daha fonksiyonel ve daha performanslı sayısal tümdevreler tasarlanmasına imkan oluşmaktadır.

Bilgisayarların içerisindeki işlemciler onbinler mertebesinde transistörden oluşmaktadır. Transistörler, elektronik devrelerde bolca kullanılan yarı iletken elemanlardır.



TVE (NAND) Kapısının iç yapısı



TVE (NAND) kapısının Simgesi

(Tümleyen VE - NAND)

Kapılar ise, mantıksal işlemleri yapmak için kullanılan en temel lojik elemanlardır. Örneğin VE, TVE, VEYA gibi mantıksal işlemler, temelde, kapılar ile gerçekleştirilir. Kapılar da transistörlerin uygun şekilde bağlanmasıyla elde edilirler. Yukarıdaki şekilde TVE (Tümleyen VE - NAND) kapısının transistör düzeyindeki iç yapısı görülmektedir. Görüldüğü gibi bir TVE kapısı en azından dört tane transistörden oluşmaktadır.

Lojik devreler, Elektronik Mühendisliğinde transistörlerden, Bilgisayar Mühendisliğinde kapılardan başlar...

Saklayıcı & Bellek

Saklayıcı ve bellek birimleri lojik devre tasarımında önemli bir yer tutar; bilindiği gibi ardışıl devrelerin *t* anındaki çıkışları, yine o andaki girişlere ve önceki durumlara bağlı olarak üretilir. Durum bilgilerinin tutulması için saklayıcı, bellek gibi saklama birimleri kullanılır. Eğer tutulması gereken durum bilgisi az sayıda ise saklayıcı, çok fazlaysa bellek birimleri kullanılır. Saklayıcılar bit düzeyinde birimlerle ifade edilirken, bellek birimleri sözcük, sekizli veya bit düzeyde birimlerle ifade edilirler. Örneğin, saklayıcılardan bahsedilirken 4 bitlik, 8 bitlik, 1 bitlik saklayıcı denilir; belleklerden bahsedilirken de 1024xSözcük, 512xSekizli, 256x4 gibi ifade edilir. İlk rakam bellek biriminde kaç tane göz olduğunu, ikincisi de bir bellek gözünde kaç bitlik veri saklanabileceğini gösterir. 256x4, herbir gözü 4 bit olan 256 gözlü bellek anlamına gelir; 4 bit ile 0 ile 15 arasındaki sayılar ifade edilebileceği düşünülürse, bu bellek biriminde 0 ile 15 arasında 256 tane sayı tutulabilir. Sekizli, 8 bitlik, Sözcük de yerine göre 16, 32 veya 64 bitlik veri anlamında kullanılır.

Teknoloji: TTL & CMOS

Lojik tümdevreler RTL, TTL, ECL, NMOS, CMOS olarak adlandırılan birçok teknolojiyle üretilmektedir; ancak küçük ve orta ölçekli üretimlerde TTL ve CMOS teknolojileri endüstri standardı olmuşlardır. Bu teknolojilerle üretilmiş çeşitli tümdevreler lojik devre tasarımcısının tüm ihtiyacını karşılayabilecek şekilde çeşitlidir; kapı, kodlayıcı/kod çözücü, aritmetik lojik işlem (ALU) yapan; bellek, sayıcı/saklayıcı gibi birçok birim ayrı birer tümdevre olarak üretilmiştir ve satılmaktadır.

Tümdevrelerden bahsedilirken TTL veya CMOS olduğu da belirtilir; çünkü teknolojilerin elektriksel ve zaman değerleri farklıdır. Örneğin (standart) CMOS tümdevreler daha az güç harcarlar ancak TTL'den daha yavaştır denilebilir. Bir TTL kapının gecikmesi tipik olarak 10-20 ns iken bir CMOS kapının gecikmesi tipik olarak 25-100 ns arasındadır (yüksek hızlı CMOS'larda bu değer 8-10 ns mertebesindedir!). Bir başka açıdan TTL ve CMOS arasındaki fark şöyledir: lojik 1 ve lojik 0 değerleri heriki teknolojide farklıdır. Lojik 0 değeri giriş için TTL'de 0-0,8 V arası (besleme 5 V için); CMOS'da 0-3 V arasındadır (besleme 10 V için). Yani, lojik 0 ve 1 değerlerinin kaç Volt olduğu teknolojiye ve giriş çıkış olmasına göre değişmektedir. Teorik işlemlerde lojik 1 ve lojik 0 değerleri sırasıyla 5 V ve 0 V'a karşılık düşürülerek gösterilmesine karşın, uygulamada tümdevrenin üretim teknolojisine ve giriş/çıkış olmasına bağlıdır.

Tasarım ve Benzetim Ortamları

Lojik devre tasarımının bilgisayar ortamında yapılması için yardımcı araç niteliğinde birçok yazılım vardır. Bu yazılımlar aracılığıyla lojik devre tasarımı ve benzetimi yapılabilir. Tümdevre olarak üretilen lojik devrelerin tasarımları, halihazırda, böyle yardımcı araçlarla yapılmaktadır. Kart şeklinde tasarım için de tasarım ve benzetim ortamları vardır; üstelik tasarlanan kartın baskılı devre çizimi de yaptırılabilmektedir. Bu tür yazılımlara örnek olarak Verilog[2], Multisim[3] ve SOLO[4] verilebilir.

Bu tür ortamlar sunan yardımcı araçlara genel olarak CAD (*Computer Aided Design*) araçları denilmektedir; birçok karmaşık sistem bu araçlarla kolayca tasarlanmakta ve davranışları sınanabilmektedir.

---

[2] Verilog ailesi, CADENCE firması tarafından geliştirilmiş ve ASIC tasarımında kullanılan oldukça güçlü bir yardımcı araçtır. Yazarlarımızdan Dr. ÇÖLKESEN, doktora çalışmasının bir kısmını ETA Vakfı (İstanbul) Lab.da bu ailenin yazılımını kullanarak yapmıştır; laboratuvar ortamını kullanma imkanını sunan sayın Prof.Dr. Duran LEBLEBİCİ'ye teşekkür ederiz.

[3] Multisimm, Electronic Benchmark firması tarafından geliştirilmiş ve lojik devre tasarımı ve benzetimi yapılabilecek esnek bir yardımcı araçtır. Yazarlarımızdan Dr. ARSAN birçok tasarımını bu ortamda yapmaktadır.

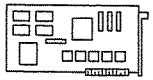
[4] SOLO, ES2 (European Silicon Structures) firması tarafından geliştirilmiş ve ASIC tasarımında kullanılan oldukça güçlü bir yardımcı araç/tümdevre tasarım paketidir. Yazarlarımızdan Dr. ÇÖLKESEN, ICTP (Trieste-İtalya)'de düzenlenen birçok eğitimde *ICTP-Microprocessor Lab.*da birçok tasarım yapmıştır.

Maliyet

Maliyet, tasarlanan lojik devrenin transistör veya kapı sayısıyla ifade edilir. Aynı fonksiyonu yerine getirecek daha az sayıda transistör veya kapı içeren tasarım, çalışma hızında bir kayıp yoksa, daha iyi bir tasarımdır denilebilir. Genel olarak, transistör veya kapı sayısı, çalışma hızı ile ters orantılı olur; çok hızlı devreler tasarlanması çok fazla transistör veya kapı kullanılmasını gerektirir. Ancak, çoğu zaman, hızdan kayıp vermeksizin transistör veya kapı sayısı azaltılabilir / indirgenebilir. Bunun için de doğru tasarım yöntemi seçilmesi ve bilinçli bir tasarım yapılması gerekir.

Kart & ASIC

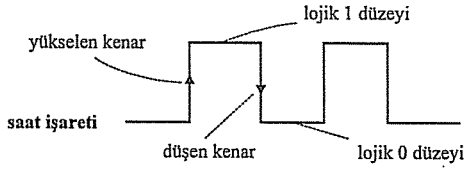
Lojik devre tasarlandıktan sonra fonksiyonel davranışı sınanmışsa iki türlü gerçekleştirilebilir. Biri, doğrudan ilgili birimlere ait tümdevreler kullanılarak kart şeklinde, diğeri yalnız başına bir tümdevre olarak. Tasarımın bir tümdevre olarak gerçekleştirilmesi genel olarak ASIC (*Application Specific Integrated Circuit*) tasarımı ve üretimi olarak adlandırılır; yani, uygulamaya dönük özel tümdevre anlamına gelir. Kart şeklinde gerçekleştirilecekse, tasarımda, halihazırda var olan ve kolayca satın alınabilecek tüm-devreler seçilmelidir; devre önce deney seti üzerinde veya bir CAD yardımcı araç ortamında sınanır ve ardından özel baskılı devre hazırlanarak üstüne yerleştirilir. Yüksek frekanslar için baskılı devre üzerindeki bağlantılı yollarının şekli ve geçtiği yerler önemlidir; yollardan geçen işaretler birbirini olumsuz etkilememelidir.


Kart

ASIC tasarımlarda tüm tasarım ve sınama, bir CAD yardımcı aracın sunduğu ortamda yapılır; tasarım bittikten sonra lojik devre, sürücü dosya (*driver file*) olarak adlandırılan ve olabilecek tüm giriş durumlarını içeren giriş bilgileriyle sınanır ve davranışı gözlenir. İstenen davranış şekli elde edilmişse, tasarım bilgileri ve verileri tümdevre üretimi yapılan fabrikalara gönderilir ve örnek üretim yaptırılır. Üretilen tümdevrenin, fiziksel olarak ta; elektriksel ve fonksiyonel davranışı gerçek ortamda sınanmalıdır.

Kenar & Düzey Tetiklemesi

Tetikleme, bir ardışıl devrenin o andaki durumunu değiştirip bir sonraki duruma geçmesi için dışarıdan uygulanan uyarma işaretidir. Bu amaçla, lojik devrelerde saat işareti olarak adlandırılan ve genelde kare dalga biçiminde olan bir elektriksel işaret kullanılır. Bir ardışıl devre, saat işaretinin yükselen kenarında, düşen kenarında ya da belirli bir düzeye gelmesi anında durumu değiştirerek bir sonraki duruma geçebilir; ilki, yükselen kenarda, ikincisi düşen kenarda ve üçüncüsü de düzey tetikleme olarak adlandırılır.



bit, nibble ve Byte (birli, dörtlü ve sekizli)

bit en küçük bellek birimidir ve **bi**nary digi**t** sözcüklerinden türetilmiştir. Bu ikili tabanda sayı sisteminde 1 hanelik bir sayı anlamındadır; dolayısıyla 1 bitlik bilgi ya 0 ya da 1 olabilir. Çünkü ikili tabanda 0 ve 1 sayıları vardır ve 1 bitlik/hanelik sayı ile de ya 0 ya da 1 saklanabilir. *nibble* 4 bitlik veri paketidir denilebilir; ikili tabanda 4 haneli bir sayı anlamındadır. 1 nibble'da 0_{10}-15_{10} arasıda bir sayı saklanabilir; bu sayılar $\underline{0000}_2$, $\underline{0001}_2$, $\underline{0010}_2$...$\underline{1110}_2$, $\underline{1111}_2$ şeklinde toplam 16 tanedir. *byte*, sekiz bitlik veri paketi olarak değerlendirilir; ikili tabanda 8 haneli sayı anlamına gelir. 1 byte ile 0_{10} ile 255_{10} arasında bir sayı tutulabilir; bu sayılar $\underline{0000\ 0000}_2$, $\underline{0000\ 0001}_2$... $\underline{1111\ 1110}_2$, $\underline{1111\ 1111}_2$ şeklinde 256 tanedir. Dolayısıyla bir Byte'lık alanda hangi büyüklükte bir sayı saklanabilir sorusu geldiğinde 0_{10} ile 255_{10} arasında denilmesi gerekir.

Analog ⇔ Sayısal Dönüşüm (ADC, DAC)

Doğada işaretler, genel olarak analog biçimdedir; ancak, sayısal sistemler ikili tabanda sayı sisteminde işlem yaparlar. Bu nedenle, özellikle dış fiziksel büyüklerin giriş veya çıkış olarak kullanıldığı sayısal sistemlerde dönüştürücüler kullanılır; analog işareti sayısal veriye veya tersi olarak sayısal veriyi analog işaret dönüştüren birimlere sırasıyla ADC (*Analog-to-Digital Converter*) ve DAC (*Digital-to-Analog Converter*) denir. Bu tür dönüştürücü iki temel özellik önemlidir; biri kaç bitlik olduğu diğeri de dönüştürme hızıdır. Kaç bitlik olduğu, doğrudan dönüştürme duyarlılığını belirler. Örneğin giriş değeri 0 ile 16 Volt arasında ve dönüştürücü 4 bitlik ise, duyarlılık $16/2^4$= 16/16= 1'den 1 Volt olur. Aynı giriş gerilimi için dönüştürücü 8 bitlik ise, $16/2^8$= 16/256= 0,0625 V'dan 0,0625 V veya 62,5 mV olduğu görülür. Kısaca dönüştürücünün bit sayısı artarsa duyarlılığı artar. Dönüştürme hızı ise, dönüştürmede kullanılan yönteme göre değişir; değişik hızlarda dönüştürme yapan yöntemler vardır. Uygulama, ihtiyaca cevap verebilecek kabul edilebilen bir hız seçilmelidir.

Donanım Tanımlama Dili (HDL)

Donanım Tanımlama Dilleri (*Hardware Description Languages*), tasarımı yapılmak istenen lojik devrelerin, fiziksel olarak gerçekleştirmeden bilgisayar ortamında

benzetiminin yapılmasına olanak veren özel amaçlı dillerdir. Özellikle ASIC tasarımlarında, devrenin yapısal ve davranışsal modeline sınamak için kullanılır. Bir HDL ile VE, VEYA gibi kapılar, saklayıcı, sayıcı gibi elemanlar tanımlanabildiği gibi tüm devrenin yapısı da tanımlanabilir ve davranışı sınanabilir. Genel olarak, bir lojik tasarım, tümdevre olarak gerçekleştirilecekse bir ASIC yardımcı aracı/yazılımı kullanılır; burada, devre içerisindeki kapı, MUX, sayıcı gibi birimler ya çizilerek, ya hazır kütüphane elemanları kullanılarak ya da HDL ile tanımlanarak elde edilirler. Nesneye dayalı programlama dilleri de, örneğin C++, donanım tanımlama dili olarak kullanılabilirler; öyle ki, bu dillerle tüm tasarımın benzetimi yapılabilir. [ÇÖLKESEN ve ark.-2001]

Sorular

1. Çevrenizde gördüğünüz olayların hangisi analog; hangisi sayısal işaret biçimine uygundur? Bir kuşun uçması analog biçime uygunsa, kuşun neler yapması sayısal biçime uygun düşer?
2. ASCII bir kod tablosudur ve bilgisayar sistemlerinde en yaygın kullanılanlardan birisidir. Başka hangi kod tabloları vardır ve özellikleri nelerdir?
3. Bir kavşağın denetimi için lojik devre tasarlanacaktır; devre kombinezonal mı veya ardışıl mı olmalıdır? Nedeniyle beraber açıklayınız.
4. Lojik devre tasarımında en temel birim kapılardır, Aslında, kapıların da temel yapı taşı olan transistörlerdir. Bir kapı kaç tane transistör içermektedir?
5. Elimizde 2Kx2'lik 8 tane bellek elamanı vardır. Bunları kullanarak hangi boyutta bellek elemanları oluşturabiliriz? Göstererek açıklayınız.
6. Güç harcaması az olması istenen bir lojik devre kart şeklinde tasarlanacaktır. Hangi teknoloji, TTL mi CMOS mu seçilmelidir? Cevabınızın tersi seçilmiş olsaydı, yani daha fazla güç harcasaydı, bu seçimin hangi özelliği daha iyi olurdu?
7. Bir lojik devre kart olarak üretilecekse nelere dikkat edilmelidir? ASIC tasarımı yapılmış olsaydı aynı noktalara dikkat edilmesi gerekir miydi? Açıklayınız.
8. Lojik devre tasarımında maliyet unsurları nelerdir? Açıklayınız.
9. Ardışıl lojik devrelerde tetikleme için saat işareti kullanılır; tetikleme, işaretin nerelerinde yapılabilmektedir? Sizce herbirinin olumlu ve olumsuz yanları nelerdir?
10. Bir ADC tümdevresi 16 bitliktir. Sayısala çevrilmesi istenen analog işaret ise 65 Volt'tur. ADC'nin duyarlılığı yaklaşık kaç mV olur? ADC 8 bitlik olsaydı duyarlılık kaç mV olurdu? Hesaplayarak karşılaştırınız.
11. Lojik 1 ve lojik 0 değerleri, giriş için, TTL'de 0-0,8 V arası; CMOS'da 0-2,6 V arasındadır; bu değerler, çıkış için, TTL ve CMOS'da kaç Volt arasındadır? Lojik 1 veya 0 olduğu anlaşılmayan belirsiz bölge var mıdır?
12. En küçük bellek birimi *bit*'tir. Daha sonra sırayla *nibble* ve *Byte* gelir; *Byte* bellek biriminin üst katları nasıl adlandırılır?

2.

İşaretler ve Analog/Sayısal Dönüşüm

İşaret, fiziksel bir olayın elektriksel olarak gösterilmesidir. Örneğin bir odadaki ısı değişimi veya bir elektrik devresinden çekilen akımın değişimi işaretle gösterilebilir. Doğadaki fiziksel olayların hemen hemen tümü işaretlerle gösterilebilir ve tanımlanabilir. Lojik devre tasarımı konusunun temeli de belirli özellikte işaretleri giriş olarak alıp belirli bir işi yerine getirecek biçimde çıkış işareti üretmektir. Fiziksel işaretlerin doğal hali analog yapıdadır; ancak lojik devre uygulamalarında sayısal (veya ikili) işaretler kullanılır. Bu bölümde analog ve sayısal işaretlerin şekli ve analog işaretten sayısal işarete dönüşüm veya tersinin nasıl yapılabileceği aşağıdaki başlıklar altında ele alınmıştır:

2.1. İşaretlerin Sınıflandırılması
Sayısal İşaret
2.2. Analog / Sayısal Dönüşümü
Örnekleme ve Tutma İşlemleri
Kuantalama, Kodlama
2.3. Özet
2.4. Sorular

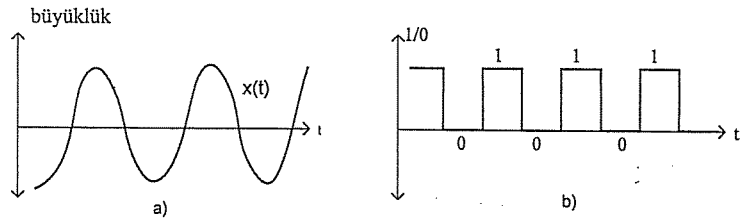
Lojik devreler, kendilerine gelen sayısal işaretleri giriş olarak alıp üzerlerinde işlem yaparlar ve yine sayısal olan sonuç işaretleri üretirler. Ancak, lojik devreler ile analog özelliklere sahip sistemler de kontrol edilebilir. Bu durumda analog özelliğe sahip giriş bilgileri önce sayısala dönüştürülmelidir; tersi olarak ta, sayısal olarak üretilen sonuçların analog'a çevrilmesi gerekebilir. Bu dönüştürme işlemleri ADC/DAC (*Analog Digital Converter/Digital Analog Converter*) olarak adlandırılan birimler ile gerçekleştirilir.

2.1. İşaretlerin Sınıflandırılması

İşaretler çeşitli şekillerde sınıflanmaktadır; lojik devre tasarımcısı, bunlardan analog/sayısal sınıflamayla ilgilenir. Analog işaretlerde bir süreklilik vardır; çünkü

fiziksel büyüklüklerin çoğu kesintisiz sürekli değişen işaretlerdir. Sayısal işaretler ise, bilginin veya analog işaretin belirli aralıklarla örneklenmesi/kodlanmasıyla elde edilir. Dolayısıyla sayısal işaretlere ayrık işaretler de denir; analog'da olduğu gibi bir süreklilik yoktur, geçişler arasında merdiven basamağı gibi atlamalar vardır.

Aşağıda, Şekil-2.1 de analog ve sayısal işaret örnekleri görülmektedir; dikey eksen bağımlı, yatay eksen ise bağımsız değişkeni temsil eder. Şekil-2.1.a da sinüs benzeri bir analog işaret görülmektedir; işaretin dikey ekseni büyüklüğü yatay ekseni ise zamanı göstermektedir. Analog işaretlerde büyüklük çoğu zaman gerilim veya akım değerleridir. Şekil-2.1.b de ise sayısal işaretin bir örneği olabilecek bir ikili işaret görülmektedir. İkili işaretlerde önemli olan büyüklüğün değeri değil, o andaki durumun, yani durumun var olup olmamasıdır; lojik tasarımcılar işaretin var olmasını lojik 1, olmamasını da lojik 0 değerleri olarak adlandırırlar.



Şekil-2.1. İşaret örnekleri a) analog b) ikili işaret.

İşaretin şekli belirli zaman aralıklarında kendini tekrarlayan biçimdeyse, bu tür işaretler periyodik özelliğe sahiptirler. Şekil-2.1 de gösterilen işaretler periyodik özellikte çizilmiştir. Periyodik işaretlerin biri periyot zamanı, diğeri frekans olmak üzere iki önemli parametresi vardır. Periyot (T), işaretin tekrarını içermeyen en küçük zaman dilimidir; birimi saniye (s) cinsindendir. Frekans ise, birim zamanda işaretin kaç kere tekrarlandığını gösterir; birimi Hertz (Hz) cinsindendir. Periyot ile frekans arasında f=1/T şeklinde bir bağıntı vardır. Örneğin periyodu 1 *ns* olan bir periyodik işaretin frekansı, $f = 1/T = 1/10^{-6} = 10^{6}$ Hz = 1 MHz'dir.

Fiziksel büyüklüklerin çoğu sürekli değişen işaretlerdir, ancak bilgisayar sistemlerinde ve sayısal kontrol sistemlerinde zamanda ayrık işaretler olan sayısal işaretler kullanılır.

Analog ve sayısal işaretlerdeki kavramı göstermek kapı örneği ele alınabilir. Bir kapının açıklığının az veya çok olması, yani ne kadar açık olduğunun gösterilmesi analog işarettir; ancak kapının aralık olmasından çok kapının açık veya kapalı olduğu bir ikili işarettir. Kapı açık ise lojik 1; kapalı ise lojik 0'dır denilebilir.

1/0 veya açık/kapalı gibi 2 değerli büyüklüklerin gösterilebilmesi için "ikili işaret" kullanılır ve ikili işaret lojik 1 veya lojik 0 değerlerini alabilir. Bir ikili işaret bir

hanelik 2 tabanında sayı ile temsil edilir. Bir hanelik 2 tabanında sayının alabileceği değerse, 1 veya 0'dır.

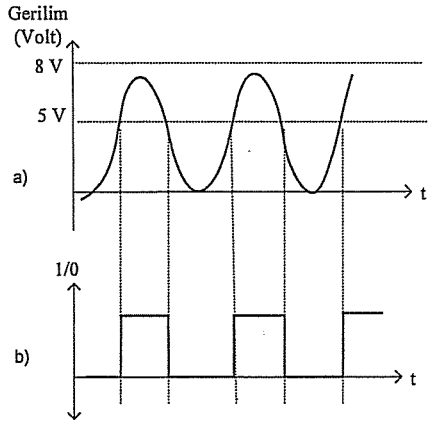
bit, sayısal sistemlerde en küçük saklama birimidir ve 2 tabanında hane anlamına gelir; 1 bitlik bir sayı, 1 hanelik 2 tabanında sayı demektir. İkili işaretler bir bitlik sayılar ile gösterilirler.

Sayısal İşaret

Sayısal işaret, en temelde bir ikili işarettir. Yani, 1 ya da 0 durumu vardır; işaret, herhangi bir anda ya 1 veya 0 değerinde bulunabilir. Örneğin bir lambanın yanması 1, sönük olması 0 ile temsil edilebilir; bu aşamada, lambanın ne kadar akım çektiği veya lambaya uygulanan gerilim büyüklüğü önemli değildir. Önemli olan lambanın yanması veya sönük olmasıdır. Çünkü lambanın birisi, yanarken 100 mA akım çekerken, bir başkası 500 mA çekebilir. Dolayısıyla birinde lojik 1 değeri 100 mA iken, diğerinde 500 mA olabilir. Genel anlamda sayısal işaret iki sınıfı ayrılır:

- İkili işaret (*Binary signal*)
- Kodlanmış sayısal işaret (*Encoded digital signal*)

İkili işaret, herhangi bir anda 1 ya da 0 değerini alabilen bir bitlik işarettir. Kodlanmış sayısal işaret ise, herhangi bir anda 2'den çok ayrık durumdan birini alabilen işaretlerdir. Örneğin 7 sayısını ikili tabanda göstermek için kodlanmış ikili işaret kullanılır. Çünkü 1 bit ile tamsayılar kümesindeki 7'yi temsil etmek mümkün değildir; en azından 3 bit gereklidir. Çünkü 3 bit ile 2^3 bağıntısında 8 değişik durum gösterilebilir; 0, 1, 2, 3...7 gibi.



Şekil-2.2. Analog işaretten ikili işaret elde edilmesi.

İkili işaret ile kodlanmış sayısal işaret arasındaki fark aşağıdaki gibi de açıklanabilir: Şekil-2.2.a da tepe değeri 6 V olan bir analog işaret görülmektedir; bu işaretin 5 Volt'un ve üstü, lojik 1, altı ise lojik 0 ile temsil edilirse, Şekil-2.2.b deki gibi bir ikili işaret elde edilebilir. Artık lojik 1 değeri 5 Volt ve üstüdür; kaç Volt olduğu o kadar önemli değildir.

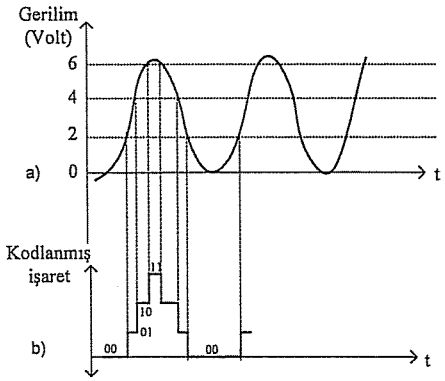
Bu şekilde ikili işaret, en temel sayısal işarettir ve 1 bit ile temsil edilebilir. Şekil-2.2.b deki ikili işaret aşağıdaki gibi gösterilebilir:

0 1 0 1 0 1

Ancak, Şekil-2.2.a daki analog işaretin ise, periyodik bir işaret olduğu varsayılırsa, aşağıdaki gibi gösterilir:

$Z = \sin(\omega t)$

Şekil-2.2.a da verilen analog işaret Şekil-2.3 de görüldüğü gibi merdiven basamağı şeklinde de gösterilebilir. Örneğin, her basamağın 2 Volt'a karşı düştüğü varsayılırsa 4 değişik basamak vardır ve işaret herhangi bir anda bu basamaklardan birinde olabilir; basamaklar arasında olamaz. Dolayısıyla bu basamaklar analog işaretin kodlandığı yerlerdir ve kuanta düzeyleri olarak isimlendirilirler.



Şekil-2.3. Analog işaretten kodlanmış sayısal işaret elde edilmesi.

Kodlanmış sayısal işaretteki basamakların gösterilmesi için birden çok bit kullanılır. Kaç bit kullanılacağını, doğrudan, basamak sayısı belirler.

Örneğin, Şekil-2.3 de olduğu gibi 4 değişik basamak varsa, 2 bit gereklidir.

0 → 00 (0 Volt)
1 → 01 (2 Volt)
2 → 10 (4 Volt)
3 → 11 (6 Volt)

Çünkü, 2 bit ile 4 değişik durum temsil edilebilir; 2 bit, 2 tabanında 2 haneli sayı anlamına gelir.

2.2. Analog/Sayısal Dönüşüm

Analog bir işaretin ikili işaret dönüştürülmesi oldukça kolaydır; bir kırpıcı devresiyle yapılabilir. Çünkü analog giriş işareti belirli bir seviyeyi geçince 1, altında kalınca 0 üretilerek gerçekleştirilebilir. Ancak, analog işaretten kodlanmış sayısal işaret karşılığının elde edilmesi veya tersine sayısal işaretten analog işaret üretilmesi için özel tasarlanmış birimler kullanılır. Bu birimler, analog'tan sayısal dönüştürmede ADC (*Analog-to-Digital Converter*), sayısaldan analog'a dönüştürmede DAC (*Digital-to-Analog Converter*) olarak adlandırılır:

- ADC : analog ⇨sayısal
- DAC : sayısal ⇨analog

Bu tür dönüştürme işi yapan birimlerde anahtar sözcükler kodların kaç bit olacağı ve dönüştürme hızıdır. Kodların bit sayısı doğrudan birimin kaç bitlik olduğunu gösterir. örneğin kod 2 bitlik olacaksa 2 bitlik ADC veya 2 bitlik DAC denilir. Uygulamada, 2 bitlik ADC veya DAC küçük kalmaktadır; bunun yerine 8, 10 veya 16 bitlik ADC veya DAC kullanılır. Dönüştürme hızı ise, bir dönüştürme işlemini en kötü durumda ne kadar sürede yapacağını belirtir.

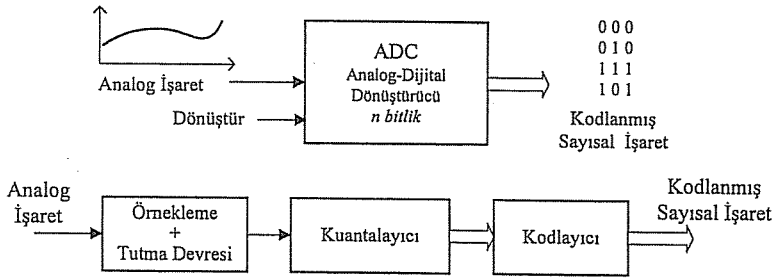
Analog sayısal dönüşümde kodların bit sayısı duyarlık olarak adlandırılır. Örneğin ses bilgileri 8-bit yerine 10-bit, 12-bit veya 16-bitlik kodlara dönüştürülürse daha kaliteli/duyarlı bir dönüşüm yapılmış olunur. Çünkü, aynı analog veriyi kodlamak için daha fazla kod kullanılmış olunur; 8-bitlikte 2^8=256, 10-bitlikte 2^{10}=1024, 16-bitlikte 2^{16}=65536 ayrık durum vardır. Ayrık durum sayısı fazla olacak şekilde kodlamakla daha duyarlı çalışılmış olunur; ancak, uygulamada çok fazla olmasının da anlamı yoktur. Olması gerektiği kadar yeterlidir.

Dönüştürme hızı (*conversion speed*), analog-sayısal veya sayısal-analog dönüştürmenin hızını ve hangi parametrelere bağlı olduğunu gösterir. Dönüştürme işlemi için birçok yöntem kullanılmaktadır ve dönüştürme hızları farklıdır. En hızlı dönüştürme şipşak (*flash*) dönüştürme yöntemidir. Ancak bu yöntemde dönüştürme yapan birimlerin eleman maliyeti yüksek olur; ardışıl yaklaşımla (*successive-approximation*) dönüştürme gibi dönüştürme hızını belirli bir periyod süresinde yapan birimler daha ekonomik olabilir.

2.2.1. ADC'nin Mimarisi

Bir ADC'nin gösterilimi ve temel birimleri Şekil-2.4 de gösterildiği gibidir. Kodlanmış sayısal işaret elde edilmesi, analog işaret üzerine kuantalama ve kodlama işlemlerinin uygulanmasıyla kotarılır. Analog/Sayısal dönüştürme işlemi, analog işaretin taşıdığı bilginin kuantalanmış ve kodlanmış şeklini sayısal bir kod sözcüğü ile ifade etmek anlamına gelmektedir.

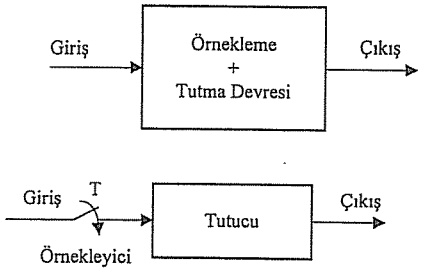
Analog işaret önce örneklenir ve tutulur; ardından, kuantalanır ve bir kodlayıcıyla karşılık düşen kod elde edilir; kod birden çok bit'ten oluşur.



Şekil-2.4. Analog-Dijital dönüştürme işlemi.

Örnekleme ve Tutma İşlemleri

Şekil-2.4 de verilen ADC yapısındaki ilk eleman örnekleme ve tutma elemanıdır. Bu elemanın görevi, belirli aralıklarla analog işaretten örnekler alıp bu değerleri belirli bir süre tutmaktır. Elemanın yapısı Şekil-2.5 te gösterildiği gibidir. Burada, T örnekleme zamanını belirtir; örnekleme süresi, örnekleme periyodundan çok küçük olduğu için sıfır kabul edilir.



Şekil-2.5. Örnekleme-Tutma devresi.

Örnekleme elemanının çalışma ilkesi *Shannon* teoremi olarak da bilinen örnekleme teoremine dayanmaktadır. Örnekleme teoremi, band sınırlı herhangi bir x(t) işareti, belirli aralıklarla örneklendikten sonra, alınan örnek değerlerden herhangi bir bozulma olmaksızın x(t) işaretini tekrar elde edebilmek için örnekleme frekansının seçilmesini bir kurala bağlamaktadır. Buna göre, x(t) işaretine ait en yüksek frekanslı bileşenin frekansı ω_c radyan/saniye ise seçilebilecek en düşük örnekleme frekansı, en yüksek frekans olan ω_c'in iki katı olmalıdır. Bu durumda, ω_s örnekleme frekansı olmak üzere,

$$\omega_s \geq 2\omega_c$$

örnekleme teoreminin en genel ifadesi olarak belirlenir.

Kuantalama

Kuantalama, sürekli bir büyüklüğü, belirli sayıda eşit aralıklı basamaklara ayırma işlemi olarak tanımlanabilir. Bu şekilde basamaklara ayrılan büyüklüğe ise kuantalanmış büyüklük adı verilir. Her bir kuanta düzeyini diğerinden ayırt etmek mümkündür. Ayrıca, kuantalama kombinasyonundaki eleman sayısı arttıkça duyarlılık artar. Duyarlılık, çıkış kodunda değişiklik oluşturabilecek en küçük giriş değeridir. Kuantalama işleminin duyarlılığı kullanılan Analog Dijital Çevirici (ADC)'nin yapısına göre değişir.

n bit ile kodlama yapılacağı düşünülürse, 2^n tane kuanta düzeyi ve 2^n-*1* tane kuantalama aralığı elde edilir. Maksimum ile minimum gerilim arasındaki fark V_{maks} olarak tanımlanırsa, *a* ile gösterilen her bir kuantalama aralığının yüksekliği (duyarlılık),

$$a = \frac{V_{maks}}{2^n}$$

olur (Duyarlık hesabı için bazı uygulamalarda, $a = V_{maks}/(2^n - 1)$ ifadesi de kullanılmaktadır).

Örnek-2.1.

-3 Volt ile +5 Volt arasında değişen bir analog işaretin 1 Volt duyarlılıkla sayısal olarak ifade etmek için kaç bit kullanmak gerekir?

Burada V_{maks}=5-(-3)=8 Volt'tur. İstenen ise $V_{maks}/2n \leq 1$ *Volt* olmasıdır. $8 \leq 2^n$ olmasından yararlanarak, bu analog gerilimin ancak 3 bit kullanılarak 1 Volt duyarlılıkla ifade edilebileceği bulunur.

Örnek-2.2.

Bir analog işaret 0 ile 15 Volt arasında kesintisiz değerler alabilmektedir; bu işaretin 100 mV duyarlılıkla kodlanmış sayısal işarete dönüştürülmesi için kaç bitlik bir ADC gerekir.

Burada, V_{maks}=(15-0)=15 V bulunur. Duyarlığın 100 mV olması istenmişti; bu 0,1 V'a karşılık gelir. Buradan *duyarlılık*=$V_{maks}/2^n$ bağıntısında parametreler yerine koyulursan kaç bit gerektiği aşağıdaki gibi bulunur:

$V_{maks}/2^n=0,1$

$2^n=V_{maks}/0,1=15/0,1=150$ çıkar.

Bir başka deyişle $2^n=150$ bulunmuştur. $n=\log_2 150=7,23$ bulunur. Buradan görüldüğü gibi 7,23 bit gereklidir; dolayısıyla 7,23'ün bir üst tamsayısı olan 8 bit gereklidir.

Kodlama

Kodlama kuantalama düzeylerine ikili sayı sisteminde birer kod verme işlemidir. Kuantalanmış işaretin kodlanması aşağıdaki gibi açıklanabilir:

S : Sonlu bir simge kümesi (0,1),

B : Sonlu veya sonsuz bir öğe kümesi ($K_0, K_1 \ldots K_7$),

Σ : Sıralanmış simge dizileri kümesi (Σ={000, 001, . . . , 111}) olsun.

Her $b_i \in B$ elemanına, Σ kümesinde bir ve yalnız bir kod sözcüğü karşı düşürülmesi işlemine *kodlama* adı verilir. Bir diğer geçerli tanım ise; her genlik ve karaktere 2 tabanında farklı bir sayı karşı düşürmeye kodlama denir. Bu sayıya kod sözcüğü, hane sayısına da kod uzunluğu adı verilir. Kodlanacak kuanta sayısı n ise kod uzunluğu en az $s=\lceil \log_2 n \rceil$'dır.

Örnek-2.3.

Bir analog işaret 0 ile 16 Volt arasıda değişmektedir. Bu işaretin 3 bit ile temsil edebilecek şekilde kodlanmış sayısal işarete dönüştürülmesi istenmektedir. Buna göre kodlanmış sayısal işaret dizisini elde ediniz.

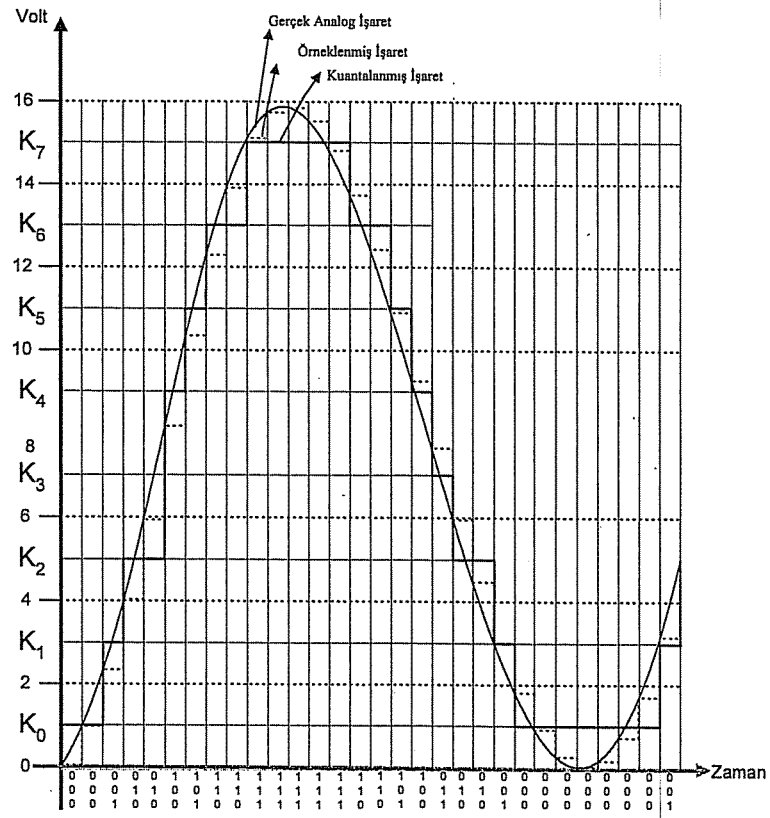
Kodlamanın $n=3$ bit ile yapılacağı düşünülürse, $2^n = 8$ kuanta düzeyi ve $2^n-1 = 7$ kuantalama aralığı olduğu görülür. Her bir kuantalama aralığı yüksekliği $V_{maks}/2^n = 16/8$ bağıntısından 2 Volt olarak bulunur. Kuanta düzeyleri ise,

[0,2)→1V,	[2,4)→3V,
[4,6)→5V,	[6,8)→7V,
[8,10)→9V,	[10,12)→11V,
[12,14)→13V,	[14,16]→15V

seviyelerine karşı düşürülmüştür. Bu kuanta düzeyleri K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7 şeklinde isimlendirilir. Bundan sonra yapılacak işlem her bir kuanta düzeyine kod anlamında bir ikili sayı karşı düşürmektir. Bu işlem ise,

K0 → 000,	K4 → 100,
K1 → 001,	K5 → 101,
K2 → 010,	K6 → 110,
K3 → 011,	K7 → 111,

olacak biçimde gerçeklenir; Şekil-2.6 da bütün anlatılanların grafiksel gösterilimi verilmeye çalışılmıştır.



Şekil-2.6. Analog işaretin kuantalanması ve kodlanması.

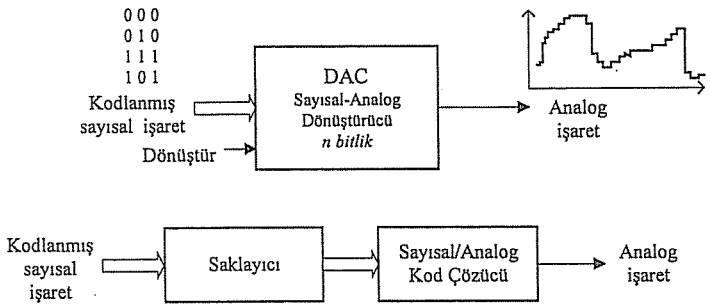
Sonuç olarak analog gerilim, bit katarı şeklinde ifade edilmiştir. Örnek dikkatli bir şekilde incelendiğinde kuanta düzeylerinin 1 V'tan başladığı ve 15 V'ta sonlandığı

görülür. Eğer kuanta düzeylerinin 0 V'dan başlayıp, 16 V'da sona ermesi istenirse kuanta aralığının yüksekliğini bulmak için kullanılan $a = V_{maks} / 2^n$ ifadesinin yerine $a = V_{maks} / (2^n - 1)$ şeklindeki ifadeyi kullanmak, yani kuanta aralığını daha yüksek seçmek gerekir.

2.2.2. DAC'ın Mimarisi

DAC sayısal veriden analog işaret üreten bir elemandır; girişlerine n bitlik sayısal işaret uygulanır ve çıkışlarında buna karşılık düşen analog işaret elde edilir. Girişlerindeki sayısal işaretin değişmesi belirli bir gecikmeyle çıkışa analog olarak yansır. DAC'ın iç mimarisi, sayısal verinin tutulması ve ilgili koda ait analog çıkışın üretilmesini yapacak şekilde elemanlar içerecek yapıdadır.

Tipik bir DAC mimarisi Şekil-2.7 de gösterildiği gibidir; şekilden görüleceği üzere sayısal giriş önce bir saklayıcıda tutulmakta ve orada tutulan sayının analog karşılığı üretilmektedir[1].



Şekil-2.7. Sayısal-Analog dönüştürme işlemi.

Örnek-2.4.

4-bitlik bir DAC devresinin çıkış gerilim aralığı 0 V ile 6 V arasında değişmektedir. 0000 sayısal girişi 0 V'a, 1111 sayısal girişi 6 V'a karşılık düşecek biçimde DAC'a ait dönüştürme tablosu oluşturunuz.

Bu sorunun çözümünde ilk önce analog gerilim değişim aralığını belirlemek gerekir. Bu da ADC'lerde kullanılan duyarlık formülüyle mümkün olur. Duyarlık hesabı için iki yaklaşım şekli vardır. Bu durumda analog çıkış değerini bulabilmek için ya $a = V_{maks} / 2^n = 6/16 = 0,375$ V ya da $a = V_{maks} / (2^n - 1) = 6/15 = 0,4$ V artımla dönüştürme tablosunu oluşturma yoluna gidilir. Heriki yaklaşım için de artımlar adım adım Tablo-2.1 de gösterilmiştir; $a = V_{maks} / (2^n - 1)$ şeklindeki yaklaşım, soruda verilen 0000 sayısal girişi 0 V'a, 1111 sayısal girişi 6 V'a karşılık düşmesi koşulunu sağlamaktadır. Birinci yaklaşımda 1111 sayısal işaretinin 5,750 V'a karşılık düştüğü görülmektedir.

[1] Sayısal veriyi analog'a dönüştürmek için birçok yöntem kullanılır. Çoğunda, ağırlık orantılama direnç devresiyle işlemsel kuvvetlendirici olarak adlandırılan bir elaman kullanılır.

Tablo-2.1. Analog dönüştürme tablosu.

Sayısal Girişler	Analog Çıkış $\frac{V_{maks}}{2^n}$	Analog Çıkış $\frac{V_{maks}}{2^n - 1}$
DCBA	**Vout**	**Vout**
0 0 0 0	0,000	0,0
0 0 0 1	0,375	0,4
0 0 1 0	0,750	0,8
0 0 1 1	1,125	1,2
0 1 0 0	1,500	1,6
0 1 0 1	1,875	2,0
0 1 1 0	2,250	2,4
0 1 1 1	2,625	2,8
1 0 0 0	3,000	3,2
1 0 0 1	3,375	3,6
1 0 1 0	3,750	4,0
1 0 1 1	4,125	4,4
1 1 0 0	4,500	4,8
1 1 0 1	5,000	5,2
1 1 1 0	5,375	5,6
1 1 1 1	5,750	6,0

2.3. Özet

Fiziksel bir olayın elektriksel olarak gösterilmesi işaret olarak adlandırılır. İşaretler genel olarak, biri analog diğeri sayısal olmak üzere iki sınıfa ayrılır. Doğada fiziksel olaylar genel olarak analog işaret olarak ortaya çıkmasına karşın, bilgisayar gibi sayısal sistemler bu analog işaretleri doğrudan kullanamazlar. Dolayısıyla, analog işaret önce sayısal hale dönüştürülür; üzerinde işlem yapılır ve isteniyorsa sonuç değerler tekrar analog hale dönüştürülerek fiziksel sisteme uygulanır. Analog'tan

sayısala dönüştüren birimler ADC, sayısal'dan analog'a dönüştüren birimler ise DAC olarak adlandırılır.

Analog işaret, biri ikili işaret diğeri ise kodlanmış ikili işaret, yani sayısal veri olmak üzere iki şekilde sayısal hale dönüştürülebilir. İkili işaret yalnızca iki genlik değerine sahip bir işaretken sayısal veri analog işaretin herhangi bir andaki mutlak değerini gösteren ve genel olarak birden çok bitten oluşan bir koddur.

Fiziksel dünya ile doğrudan bağlantısı olan sayısal sistemlerde birer adet ADC/DAC çifti bulunmalıdır. Bu tür birimlerde iki önemli anahtar sözcük vardır; biri duyarlılık diğeri dönüştürme hızıdır. Duyarlılık, doğrudan ilgili birimin kaç bit olduğuna bağlıdır ve bit sayısı arttıkça duyarlılık artar. Örneğin, 10-bitlik bir ADC, 8-bitlik bir ADC'den daha duyarlıdır.

2.4. Sorular

1. a) Analog İşaret, Sayısal İşaret, Analog Sistem ve Sayısal Sistem kavramlarını açıklayınız. Her birine bir örnek veriniz.

   b) +15 Volt ile -5 Volt arasında değişen bir gerilimi 0,1 Volt duyarlıkla ölçmek için kaç bitlik ADC kullanmak gerekir?
2. +6 Volt ile -4 Volt arasında değişen bir analog gerilimi en az 0,05 Volt duyarlılıkla sayısal olarak ifade etmek için en az kaç bit kullanmak gerekir? Kuantalama ve kodlama işini yapınız.
3. 60°C ile -10°C arasındaki sıcaklığı 1°C hassasiyetle gösterebilmek için en az kaç bitlik kodlanmış işarete ihtiyaç vardır? Bu soruyu, -10°C karşılığı sıfırıncı kuanta düzeyi, 60°C ise sonuncu kuanta düzeyine karşı gelecek biçimde çözünüz.
4. 5,12 metrelik mesafeyi 1 cm'lik duyarlılıkla gösterebilmek için en az kaç bit kullanmak gerekir ve 5,12 metreyi gösteren kombinasyonu belirleyiniz.
5. Örnek-2.2 de Şekil-2.2 ile verilen 0 ile 16 Volt arasında değişen analog gerilimi, kodlama uzunluğu 3 bit olacak şekilde ve sıfırıncı kuanta düzeyi 0 V'a, yedinci kuanta düzeyi 16 V'a gelecek biçimde kuantalayınız. Kodlama işlemi sonucunda analog gerilime karşı düşen bit katarını elde ediniz. Sonucu Örnek-2.2 de verilen ile karşılaştırınız ve farklılıkları belirtiniz.
6. Analog'dan sayısala dönüştürmek için birçok mimari vardır; örneğin bir tanesi ardışıl yaklaşım yöntemi olarak adlandırılır. Bunun dışında hangi yöntemler olduğunu araştırınız ve aralarındaki farkı dönüştürme hızı açısından karşılaştırınız.
7. Sayısaldan analog'a dönüştürmek için birçok mimari vardır. Bunların adlarını ve aralarındaki farkları araştırınız. Hangisi daha az maliyetle yapılabilmektedir.

3.

Sayı Sistemleri

Sayılar değişik tabanlara dayanılarak farklı şekillerde gösterilebilir. Örneğin, günlük yaşamda kullandığımız sayı sistemi 10 tabanına dayanır; yaptığımız toplama, çıkarma, çarpma gibi işlemler 10 tabanına göre yapılmaktadır ve buna oldukça alışığız. Ancak 10 tabanı dışında da sayı sistemleri vardır ve birçok uygulamada bu sayı sistemleri işin doğasına daha uygundur. Örneğin sayısal uygulamalarda ve doğal olarak bilgisayar uygulamalarında ikili (2 tabanı), sekizli (8 tabanı) ve onaltılık (16 tabanı) sayı sistemleri kullanılması yaygındır. Üstelik ikili sayı sistemi, bilgisayar sistemleri ve lojik devre tasarımının temelini, başlangıç noktasını teşkil eder. Bu bölümde 2'li, 8'li ve 16'lık sayı sistemlerinde sayıların gösterilimleri ve aritmetik işlemler aşağıdaki başlıklar altında ele alınmıştır:

3.1. Çeşitli Sayı Sistemleri
İkili Sayı Sistemi
İkili Sayılar Üzerinde Aritmetik İşlemler
Tümleyen Aritmetiği
Sekizli Sayı Sistemi
Onaltılı Sayı Sistemi
BCD Sayı
Sayıların Bilgisayarda Gösterimi
Bir Sayının Genel İfadesi
3.2. Sayı Sistemleri Arasında Dönüşüm
n Tabanından 10 Tabanına
10 Tabanından n Tabanına
m Tabanından n Tabanına
2 Tabanından 8/16 Tabanına
3.3. İşaretli Sayıların Gösterilimi
İşaretli Sayılarda Tümleme Aritmetiği
Taban'a Göre Tümleme
(Taban-1)'e Göre Tümleme
Çıkarma İşlemi ve Tümleyen Aritmetiği
Çıkarma İşlemi ve (Taban-1)'e Tüm. Ar.
3.4. Özet
3.5. Sorular

İkili sayı sisteminde 2 tane rakam vardır; biri 0 ve diğeri 1'dir. Dolayısıyla bir ikili sayı 0 ve 1'lerin dizisinden oluşur: 100111, 1100110, 10001 gibi. İkili sayı sisteminde hane yerine bit sözcüğü kullanılır. Örneğin biraz önce verilen 3 tane sayıdan ilki 6, ikincisi 7 ve üçüncüsü 5 haneli birer ikili sayıdır. Dolayısıyla bu sayılara, sırasıyla, 6, 7 ve 5 bitlik sayılar denir.

bit kavramı doğrudan 10 tabanındaki haneye karşılık düşer. Örneğin 1999 dört haneli 10 tabanında bir sayıdır. Dört haneli 10 tabanında bir sayı ile en büyük 9999 sayısı temsil edilebilir.

3.1. Çeşitli Sayı Sistemleri

Çeşitli sayı sistemleri vardır. Rakamların yanyana koyulmasıyla oluşturulan bir sayının gerçek değeri, o sayı için temel alınan tabana bağlıdır. Yani bir sayının ifade edilmesindeki en önemli nokta o sayının tabanıdır. Örneğin, 37 sayısının hangi tabanda yazılmış olduğu büyük önem taşımaktadır. Bu sayı 10 tabanında yazılmışsa farklı, 8 tabanında yazılmışsa farklı değere sahiptir. Genel olarak bir sayı, örneğin 37 sayısı, 37_{10}, 37_8 şeklinde yazılarak gösterilir; sayının sağ altına tabanı yazılır. Ancak, tabanı yazılmayan sayılar 10 tabanında olduğu kabul edilir.

Tablo-3.1 de 0 ile 15 arasındaki sayıların 10, 2, 8 ve 16 tabanlarındaki karşılıkları verilmiştir; görüleceği üzere 15_{10} sayısı 2'li tabanda 1111_2, 8'li tabanda 17_8, 16'lı tabanda F_{16} olarak gösterilmektedir. Sayısal sistem tasarımlarında veya bilgisayar uygulamaları geliştirilirken 10 tabanı yerine 2'li, 8'li veya 16'lı tabanlarda sayı kullanılması işin doğasına daha uygun olmaktadır. Dolayısıyla, sayıların, bu sayı sistemlerindeki gösterimleri ve aralarındaki dönüşümüm nasıl yapılacağının bilinmesi oldukça yararlı olur.

Tablo-3.1. Çeşitli tabanlarda verilmiş sayılar.

10 Tabanı	2 Tabanı	8 Tabanı	16 Tabanı
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

- **İkili Sayı Sistemi** (*Binary*)

İkili sayı sisteminde rakamlar 0 ve 1 şeklindedir; bunlar ardarda konularak ikili sayı oluşturulur. Bir ikili sayının en soldaki biti en anlamlı (en yüksek anlamlı), en sağdaki biti de en anlamsız (en düşük anlamlı) bit olarak adlandırılır. Gerçekten de en soldaki bit en anlamlıdır. Çünkü bu bit üzerindeki bir değişiklik sayının değerini büyük oranda değiştirir; ancak, en sağdaki bit üzerindeki bir değişiklik sayının değerini yalnız 1 azaltır veya arttırır.

en anlamlı → 1 0 0 0 1 1 ← en anlamsız

İkili-Ondalık Dönüşüm : Bir ikili sayı, $(b_{n-1}\ b_{n-2}\ b_{n-3} \ldots b_1\ b_0)_2$, $b_i \in \{0, 1\}$ ise, bu sayının ondalık karşılığı, $\sum_{i=0}^{n-1} b_i 2^i$ bağıntısıyla hesaplanır.

$$1000101_2 \Rightarrow 1\cdot2^6+0\cdot2^5+0\cdot2^4+0\cdot2^3+1\cdot2^2+0\cdot2^1+1\cdot2^0 \Rightarrow 64+8+1 \Rightarrow 73_{10}$$

Ondalık-İkili Dönüşüm : Bir ondalık sayı, $(d_{n-1}\ d_{n-2}\ d_{n-3} \ldots d_1\ d_0)_{10}$, $d_i \in \{0, 1 \ldots 9\}$ ise ve bu sayının ikili karşılığı $\sum_{i=0}^{n-1} b_i 2^i$ ise, $(d_{n-1}\ d_{n-2}\ d_{n-3} \ldots d_1\ d_0)_{10} = (b_{n-1}\ b_{n-2}\ b_{n-3} \ldots b_1\ b_0)_2$'dir. Bu yöntem ondalık-ikili dönüşümün genel yoludur; ancak, pratikte çok kestirme bir yol vardır. Örneğin, 10 tabanında verilen sayı, önce 2'ye tam bölme yapılarak bölünür; bölüm ve kalan bir yere yazılır. Ardından bölüm 2'ye bölünür ve yeni bölüm ve kalan bir yere yazılır. Bu işleme, bölüm 0 olana kadar devam edilirse, kalanlar 10 tabanında verilen sayının 2'li karşılığını gösterir.

Örnek-3.1.

21_{10} sayısının 2'li tabandaki karşılığı bulunmak istensin. Aşağıda görüleceği üzere önce sayı, ardından bölüm ard arda 2'ye bölünerek kalanlar bulunmuştur. Kalanlar aşağıdan yukarıya doğru değerlendirilmesiyle 2'li karşılık bulunmuş olunur.

Sayı		Bölen		Bölüm	Kalan
<u>21</u>	:	2	=	10	1
10	:	2	=	5	0
5	:	2	=	2	1
2	:	2	=	1	0
1	:	2	=	0	1

Sonuç olarak 21_{10} sayısının 2'li karşılığı 10101_2 olarak bulunur.

İkili Sayılar Üzerinde Aritmetik İşlemler

İkili sayılar üzerinde toplama, çıkarma gibi aritmetik işlemler, genel olarak 10 tabanında olduğu gibi gerçekleştirilir. Ancak çıkarma işleminde tutulacak bir ikinci yol daha vardır ve bu yol toplama işlemi yapılarak çıkarma yapılmasını sağlar.

- Toplama işlemi; günlük yaşamda 10 tabanı için kullanılan yöntemle gerçekleştirilir
- Çıkarma işlemi
  1. 10 tabanı için kullanılan yöntemle
  2. Tümleyen aritmetiğiyle

gerçekleştirilir. Burada öncelikle tümleyen aritmetiği açıklanarak çıkarma işleminin gerçekleştirilebilmesi için gerekli temel adımlar tanıtılmıştır.

Tümleyen Aritmetiği

Tümleyen aritmetiğinde, ikili sayı önce tümlenir. Biri 1'e tümleme ve diğeri 2'ye tümleme olarak adlandırılan iki tümleme şekli vardır.

- 1'e tümleme: 1 olan bitler 0, 0 olan bitler 1 yapılarak bulunur.
- 2'ye tümleme: 1'e tümlemeyle elde edilen sonuca 1 eklenerek bulunur.

Örnek-3.2.

21_{10} ve 5_{10} sayısını önce 2'li tabana dönüştürünüz ve 21_{10}'den 5_{10}'i ikili düzeyde çıkartınız.

21_{10} sayısı Örnek-3.1 de dönüştürülmüştü ve 10101_2 olarak bulunmuştu. 5_{10} sayısının ikili karşılığı ise 101_2'dir. Bunlara göre çıkartma işlemi için önce 101_2 (5_{10}) sayısının 2'ye tümleyeni bulunmalı ve 10101_2 (21_{10}) ile toplanmalıdır. Burada her ne kadar toplama işlemi yapılıyor olsa da aslında çıkartma yapılmıştır.

sayı	⇨	00101_2	
bir tümleyeni	⇨	11010_2	
ikiye tümleyeni	⇨	11011_2	
$21_{10}-5_{10}=16_{10}$	⇨	1 0 1 0 1	
		1 1 0 1 1	
	+	----------	
		1 0 0 0 0	⇨ $1\cdot2^4+0\cdot2^3+0\cdot2^2+0\cdot2^1+0\cdot2^0$ ⇨ 16

- **Sekizli Sayı Sistemi** (*Octal*)

İkili sayı düzenindeki sayının sağdan sola doğru 3'erli gruplanması ve gruba bir rakam düşürülmesiyle elde edilir. Örneğin 21_{10} sayısının sekizli tabandaki karşılığı şöyle bulunabilir:

$$(21_{10}) \Rightarrow \underset{2}{\underline{10}}\,\underset{5}{\underline{101}}_2 \Rightarrow 10\ 101 \Rightarrow 25_8$$

- **Onaltılık Sayı Sistemi** (*Hexadecimal*)

İkili sayı düzenindeki sayının sağdan sola doğru 4'erli gruplanması ve her gruba [0, 1 ... 9, A, B, C, D, E, F] kümesinden bir rakam düşürülmesiyle elde edilir. Örneğin 21_{10} sayısının onaltılık tabandaki karşılığı şöyle bulunabilir:

$$10101_2(21_{10}) \Rightarrow 1\ 0101 \Rightarrow 15_{16}$$

Örnek-3.3.

0101001111111010_2 sayısının 16'lık tabandaki gösterilimini bulunuz.

$$\underset{5}{\underline{0101}}\ \underset{3}{\underline{0011}}\ \underset{F}{\underline{1111}}\ \underset{A}{\underline{1010}} \Rightarrow 53FA_{16}$$

- **BCD Sayı**

BCD sayı ikili tabanda kodlanmış ondalık sayı demektir. BCD sayılar on tabanındaki rakamlara 4 bitlik ikili kod atanmasıyla elde edilir. Dolaysıyla BCD sayının bir hanesi 4 bit yer işgal eder; bu bit ile temsil edilebilen 0 ile 16 arasındaki sayı kombinezonlarından 0-9 arası kullanılırken 10-15 (onaltılık tabanda A-F) arası kullanılmaz. Tablo-3.2 de ondalık sayılara karşılık gelen ikili kodlanmış ondalık sayılar verilmiştir.

Tablo-3.2. BCD sayılarda rakamlar.

Rakam	İkili kodlanmış rakam
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Eğer 9'dan büyük ondalık sayı ikili kodlanmış ondalık gösterilim ile ifade edilmek istenirse her hanenin farklı değerlendirilerek BCD koduna çevrilmiş olması gerekir. Aksi taktirde yapılan işlem bir ondalık/ikili taban dönüşümü şekline gelir ki bu yaklaşım BCD sayılarda yanlış sonuçlar doğurur. Bu işlemin bir örnekle ifade etmek gerekirse; 21_{10} sayısının BCD olarak aşağıdaki gibi gösterilmesi gerekir:

$21_{10} \Rightarrow (0010\ 0001)_{BCD}$

- **Sayıların Bilgisayarda Gösterilimi**

Sayılar, bilgisayarda, belirli sayıda bit kullanılarak gösterilir. Örneğin 8, 16, 32, 64 bit kullanılabilir. Sayının temsil edilmesi için kullanılan bit sayısı arttıkça gösterilebilecek sayının maksimum değeri de artar.

Sayıların bilgisayar ortamındaki gösterilimi yapısı tamsayılar ve kesirli sayılar için farklıdır. Yani, farklı format kullanılır. Üstelik format yapısı, sayının işaretli olup olmaması, kayan noktalı veya sabit noktalı olmasına göre de sayının gösterilim formatı değişir.

- Tamsayı
  - *İşaretli*
  - *İşaretsiz*
- Kesirli sayı
  - *Kayan noktalı*
  - *Sabit noktalı*

Bilgisayar ortamında ikili tabanda kodlama kullanılır. Bilgisayar ortamında, en yalın sayı işaretsiz tamsayıdır. Çünkü, sayının doğrudan 2'li tabandaki karşılığı kullanılır. Örneğin, 39 sayısı işaretsiz tamsayı olarak aşağıdaki gibi bulunur:

$39_{10} = 100111_2$

Ancak sayı işaretli ise yani eksi değerler de alabiliyorsa gösterilim farklı olur. Çünkü işaretli sayılarda hem sayının mutlak değeri hem de işareti gösterilmektedir. Bunun için iki temel yöntem kullanılır. Birincisi, sayının mutlak değerini gösteren bitlerin en soluna (en anlamlı bitin soluna) bir adet işaret biti eklenir; bu bit 1 ise eksi, 0 ise artı diye değerlendirilir. Buna göre yukarıda verilen 39_{10} sayısı -39_{10} olarak aşağıdaki gibi gösterilebilir:

$39_{10} = \underline{0}100111_2$
$-39_{10} = \underline{1}100111_2$

İşaretli sayıların gösteriliminde ikinci yöntem, 2'ye tümleyen aritmetiği kullanılmasıdır. Burada sayının artı değerinin 2'ye tümleyeninin bulunması sayının eksi değerinin elde edilmesini sağlar. Örneğin, 39_{10} sayısının karşılığı aşağıdaki gibi bulunur.

$$39_{10} = 0100111_2 \xrightarrow{\text{1'e tümleme}} 1011000_2 \xrightarrow{\text{2'ye tümleme}} 1011001_2$$

Burada görüldüğü gibi mutlak değeri aynı olmasına rağmen eksi karşılıkları birinde 1100111_2, diğerinde ise 1011001_2 bulunmuştur. İkili tabanda gösterilimleri farklı olsa da bu sayılar 10 tabanında -39'a karşılık düşer, yalnızca format yapıları farklıdır.

Kesirli sayılarda sayının toplam bitlerinden bir kısmı tam, bir kısmı da kesirli tarafı göstermek için kullanılır. İki tarafı ayırmak için bir nokta vardır; noktanın sol tarafı tam, sağ tarafı kesirli kısmı gösterir. Aşağıda sabit noktalı gösterilim için format görülmektedir.

$$a_{q-1}a_{q-2}\cdots a_0 \ . \ a_{-1}\cdots a_{-p}$$

Bu sayının tam kısmı q, kesirli kısmı ise p basamaklıdır. Eğer sayı 10 tabanında ise a_0 birler basamağı, a_1 onlar basamağı, a_2 ise yüzler basamağıdır ve bu şekilde devam eder. Benzer şekilde a_{-1} onda birler basamağı, a_{-2} yüzde birler basamağı, a_{-3} binde birler basamağı adını alır ve diğer basamaklar da benzer şekilde isimlendirilirler.

Sabit noktalı kesirli sayılardan farklı olarak, özellikle bilimsel hesaplamalarda kayan noktalı kesirli sayılar kullanılmaktadır. Kayan noktalı kesirli bir sayı iki bölümden oluşur; işareti de içeren kesirli kısım ve yine işareti içeren üs kısmı. Örneğin, on tabanında verilmiş olan +39,345 sayısının kayan noktalı gösterilimi;

Kesir	Üs
+.39345	+02

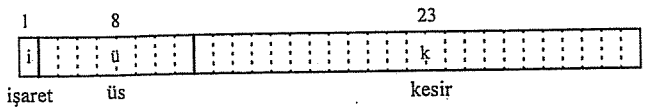
şeklindedir. Bu sayının bilimsel gösterilimi ise $+.39345 \times 10^{+02}$ (*kesir x taban*^{üs}) biçiminde ifade edilebilir. Eğer aynı işlem ikili tabandaki +1101,01 sayısı için 8-bit kesirli kısım ve 6-bit üs kullanılarak gerçekleştirilmek istenirse;

Kesir	Üs
01101010	000100

elde edilir. Buradaki üs kısmının on tabanında 4 olması gerekir. Bu üs değeri ikili tabanda ifade edilirse 000100 kullanılması yeterli olur. Kesir kısmında en soldaki hane sayının artı veya eksi olmasını ifade etmektedir. Eğer bu değer 0 ise sayı artı, 1 ise eksidir. Kesirli kısım için 8-bit ayrıldığı için sayı değeri işaret bitinden hemen sonra gelir. Hane sayısı 7'den az ise eksikler 0 ile doldurulur; hane sayısı 7'den fazla ise sayı kırpılır.

İkili tabandaki kayan noktalı sayıların gösterilimi için standart IEEE[1] tarafından önerilmiştir ve yaygın olarak kullanılmaktadır. Bu standarda göre kayan noktalı sayılar 32-bit ile aşağıdaki gibi ifade edilir:

[1] IEEE: Instutite of Electrical and Electronics Engineering



Bu gösterilimde, işaret kısmı, kesir'in işaret bilgisini taşır; 0 ise sayı pozitif, 1 ise sayı eksidir. Sayının değeri ise $(-1)^i\, 2^{ü-127} \times (1.k)$ formülüyle hesaplanır.

Bir Sayının Genel İfadesi

İster tamsayı, ister kesirli olsun bir sayı genel olarak aşağıdaki verilen polinomla ifade edilir. Burada, n tabanı, q tam kısmın hane sayısını, p ise kesirli kısmın hane sayısını ve a'larda katsayıları gösterir:

$$N_n = a_{q-1}n^{q-1} + \cdots + a_0 n^0 + a_{-1}n^{-1} + \cdots + a_{-p}n^{-p} = \sum_{i=-p}^{q-1} a_i n^i$$

Bu polinom n>1 ve $0 \le a_i \le n-1$ için tanımlıdır. Örneğin, 10 tabanında bir sayı ifade edilirken n=10 ve $0 \le a_i \le 9$; 2 tabanında bir sayı ifade edilirken $n=2$ ve $0 \le a_i \le 1$ olur. Aşağıda 10 ve 2 tabanında sayılar görülmektedir.

37_{10}	$45{,}125_{10}$
100101_{12}	$101101{,}001_2$

Bu sayılar aşağıda verilen şekilde bir yapıya sahiptirler:

$$N_n = a_{q-1}a_{q-2}\cdots a_0 \,.\, a_{-1}a_{-2}\cdots a_{-p}$$

3.2. Sayı Sistemleri Arasında Dönüşüm Kuralları

Bir sayının değişik tabanlarda karşılıklarının bulunması için dönüşüm işlemi yapılır. Uygulama böyle dönüşüm işlemlerine zaman zaman ihtiyaç duyulur. Yani herhangi bir tabandaki bir sayının diğer tabanlardaki karşılığına gerek duyulur.

3.2.1. n Tabanından 10 Tabanına Dönüşüm

Bu dönüşüm işlemi için iki yöntemden bir kullanılabilir. Birinci yöntemde, n tabanında verilmiş olan bir sayıyı 10 tabanına dönüştürmek için aşağıda verilen polinomun hesabı yapılır.

$$(N)_n = a_{q-1}n^{q-1} + a_{q-2}n^{q-2} + \cdots + a_0 n^0 + a_{-1}n^{-1} + \cdots + a_{-p}n^{-p}$$

İkinci yöntemde ise, önce sayının tam kısmı sonra da kesirli kısmı ayrı ayrı dönüştürülür. Sadece tamsayılara uygun yöntemde,

$$(N)_n = n(a_{q-1}n^{q-2} + a_{q-2}n^{q-3} + \cdots) + a_0$$
$$(N)_n = n[n(a_{q-1}n^{q-3} + a_{q-2}n^{q-4} + \cdots) + a_1] + a_0$$
$$(N)_n = n[n[n(a_{q-1}n^{q-4} + a_{q-2}n^{q-5} + \cdots) + a_2] + a_1] + a_0$$

ve sonuç olarak,

$$(N)_n = n[n[\cdots n(a_{q-1}n + a_{q-2})\cdots + a_2] + a_1] + a_0$$

polinomun sayısal değerini hesaplamak gerekir. Sadece kesirli sayılar için uygun yöntemde ise, tam sayılar için verilen ifadeye benzer şekilde

$$(N)_n = n^{-1}[n^{-1}[n^{-1}[\cdots n^{-1}(a_{-p}n^{-1} + a_{-(p-1)})\cdots + a_{-3}] + a_{-2}] + a_{-1}]$$

polinomun sayısal değerini hesaplamak gerekir.

Örnek-3.4.

$3CD8_{16}$ sayısının 10 tabanına dönüştürülmesi için,

$$3CD8_{16} = 3\times16^3 + 12\times16^2 + 13\times16^1 + 8\times16^0$$

polinomunun sayısal değeri bulunmalıdır. Bu durumda,

$$\begin{aligned} 3CD8_{16} &= 3\times16^3 + 12\times16^2 + 13\times16^1 + 8\times16^0 \\ &= 12288 + 3072 + 208 + 8 \\ &= 15576_{10} \end{aligned}$$

elde edilir.

Örnek-3.5.

$547{,}6_8$ sayısının 10 tabanına dönüştürülmesi için,

$$547{,}6_8 = 5\times8^2 + 4\times8^1 + 7\times8^0 + 6\times8^{-1}$$

polinomunun sayısal değeri bulunmalıdır. Bu durumda,

$$547,6_8 = 5\times8^2 + 4\times8^1 + 7\times8^0 + 6\times8^{-1}$$
$$= 320 + 32 + 7 + 0,75$$
$$= 359,75_{10}$$

elde edilir. Görüldüğü gibi bu tip problemlerde polinom değerini doğru olarak bulabilmek için katsayıların doğru basamak değerleriyle çarpılması gerekmektedir. Bunun için pratik bir yöntem virgülden sonra ve virgülden önce numaralandırma yapmaktır. Bu durumda;

210 – 1
547,6

5'in 8^2 ve takiben 4'ün 8^1, 7'nin 8^0 ve 6'nın 8^{-1} ile çarpılacağı anlaşılır.

Örnek-3.6.

$1001,0111_2$ sayısının 10 tabanına dönüştürülmesi için,

$$1001,0111 = 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2} + 1\times2^{-3} + 1\times2^{-4}$$

polinomunun sayısal değeri bulunmalıdır. Bu durumda,

$$1001,0111_2 = 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2} + 1\times2^{-3} + 1\times2^{-4}$$
$$= 8 + 0 + 0 + 1 + 0 + 0,25 + 0,125 + 0,0625$$
$$= 9,4375_{10}$$

elde edilir.

Örnek-3.7.

641_8 sayısının 10 tabanına dönüştürülmesi için,

$$641_8 = 8(6\times8+4)+1$$

polinomunun sayısal değeri bulunmalıdır. Bu durumda,

$$641_8 = 8(6\times8+4)+1$$
$$= 417_{10}$$

elde edilir.

Örnek-3.8.

$0,6725_8$ sayısının 10 tabanına dönüştürülmesi için,

$$0,6725_8 = 8^{-1}(8^{-1}(8^{-1}(5\times8^{-1}+2)+7)+6)$$

polinomunun sayısal değeri bulunmalıdır. Bu durumda,

$$0,6725_8 = 8^{-1}(8^{-1}(8^{-1}(5\times8^{-1}+2)+7)+6)$$
$$= 0,8645_{10}$$

elde edilir.

3.2.2. *10 Tabanından n Tabanına Dönüşüm*

10 tabanında verilen bir sayıyı n tabanına, tam ve kesirli kısımları ayrı ayrı işlemden geçirilerek dönüştürülür. $(N)_{10}$ sayısının tam kısmı n tabanında,

$$(N)_{10T} = a_{q-1}n^{q-1} + a_{q-2}n^{q-2} + \cdots + a_1n^1 + a_0n^0 = \sum_{i=0}^{q-1} a_i n^i$$

polinomu ile ifade edilir. a_i'leri bulmak için bu polinom önce n'e bölünür.

$$\frac{(N)_{10T}}{n} = Q_0 + \frac{a_0}{n} \quad \Rightarrow \quad (N)_{10T} = n\times Q_0 + a_0$$
$$(N)_{10T} = n\times(a_{q-1}n^{q-2} + a_{q-2}n^{q-3} + \cdots + a_1n^0) + a_0$$

Böylece eğer $(N)_{10T}$, n tabanına bölünürse sonuçta Q_0 gibi bir tamsayı ve a_0 gibi bir kalan elde edilmektedir. Buradaki a_0 kalanı, $(N)_n = a_{q-1}a_{q-2}\cdots\cdots a_0$ şeklinde olan sayının en sağındaki terimdir. a_1'i bulmak için Q_0, n'e bölünür.

$$\frac{Q_0}{n} = Q_1 + \frac{a_1}{n} \quad \Rightarrow \quad Q_0 = n\times Q_1 + a_1$$

Burada a_1 kalan sayıdır ve bu bölme işlemine Q_{q-1} sıfır olana kadar devam edilir.

Örnek-3.9.

21_{10} sayısının 2 tabanına dönüştürülmesi için,

		Bölen		Bölüm	Kalan
21	:	2	=	10	1
10	:	2	=	5	0
5	:	2	=	2	1
2	:	2	=	1	0
1	:	2	=	0	1

şeklinde tekrarlamalı bölme işlemi yapılır. Bu işlem bölüm sıfır olana kadar sürer. Kalan sütunundaki sayılar ters sırada yazılarak yeni tabana dönüşüm gerçeklenmiş olur. Sonuç olarak,

$$21_{10} = 10101_2$$

elde edilir.

Örnek-3.10.

16527_{10} sayısının 16 tabanına dönüştürülmesi için,

		Bölen		Bölüm	Kalan
16527	:	16	=	1032	15 (F)
1032	:	16	=	64	8
64	:	16	=	4	0
4	:	16	=	0	4

şeklinde tekrarlamalı bölme işlemi yapılır. Bu işlem bölüm sıfır olana kadar sürer. Kalan sütunundaki sayılar 16 tabanında ifade edilir ve ters sırada yazılarak yeni tabana dönüşüm gerçeklenmiş olur. Sonuç olarak,

$$16527_{10} = 408F_{16}$$

elde edilir.

$(N)_{10}$ sayısının kesirli kısmı n tabanında,

$$(N)_{10K} = a_{-1}n^{-1} + a_{-2}n^{-2} + \cdots + a_{-p}n^{-p}$$

polinomu ile ifade edilir. a_{-1}'i bulmak için bu polinom önce n ile çarpılır.

$$n \times (N)_{10K} = a_{-1} + a_{-2}n^{-1} + \cdots + a_{-p}n^{-p+1}$$
$$n \times (N)_{10K} = a_{-1} + P_1$$

Böylece a_{-1} gibi bir tamsayı ve P_1 gibi bir kesirli sayı elde edilmektedir. Bu durumda,

$n \times (N)_{10K} < 1$ ise $a_{-1}=0$'dır.

$n \times (N)_{10K} \geq 1$ ise $a_{-1} \neq 0$'dır ve tamsayılı kısım a_{-1}'e verilir.

a_{-2}'yi bulmak için $n \times (N)_{10K}$ çarpımının kesirli kısmı, P_1 sayısı, b_1 ile çarpılır.

$$\begin{aligned} n \times P_1 &= n \times (a_{-2}n^{-1} + a_{-3}n^{-2} + \cdots + a_{-p}n^{-p+1}) \\ &= a_{-2} + a_{-3}n^{-1} + \cdots + a_{-p}n^{-p+2} \\ &= a_{-2} + P_2 \end{aligned}$$

Böylece a_{-2} gibi bir tamsayı ve P_2 gibi bir kesirli sayı elde edilmektedir. Bu durumda,

$n \times P_1 < 1$ ise $a_{-} = 0$'dır.

$n \times P_1 \geq 1$ ise $a_{-} \neq 0$'dır ve tamsayılı kısım a_{-} 'ye verilir.

Genelde sonu olmayan bu işleme aynı adımlar kullanılarak devam edilir.

Kesirli sayılar için *10* tabanından n tabanına geçiş işlemi, çarpma işlemlerinde kesirli kısmın hiçbir zaman sıfır olmaması halinde sonsuza kadar gidebilir. Bu durumda, işlemi bir yerde sonlandırmak gerekir. m tabanında p hane ile gösterilen kesirli bir sayının, n tabanında kaç hane ile ifade edilebileceği,

$$m^{-p} = n^{-r}$$
$$-p \ln m = -r \ln n$$
$$r = p\frac{\ln m}{\ln n}$$

ilişkisi ile belirlenebilir. Eğer bu ifadede m=10 ise,

$$r = p\frac{\ln 10}{\ln n}$$

olacaktır.

Örnek-3.11.

$0,37_{10}$ sayısının 2 tabanına dönüştürülmesi için, n=2 ve p=2 olduğundan,

$$r = p\frac{\ln 10}{\ln n} = 2\frac{\ln 10}{\ln 2} = 6,64$$

haneye ihtiyaç vardır. Bunun anlamı 6 haneden büyük ifade edilebilmesidir. Bu durumda 2 tabanındaki sayının kesirli kısmını 7 haneye kadar hesaplamak yeterli olacaktır. On tabanındaki kesirli sayıyı iki tabanına dönüştürmek için,

$$\begin{array}{lllll}
0,37 \times 2 = 0,74 < 1 & \Rightarrow & a_{-1} = 0 \\
0,74 \times 2 = 1,48 \geq 1 & \Rightarrow & a_{-2} = 1 \\
0,48 \times 2 = 0,96 < 1 & \Rightarrow & a_{-3} = 0 \\
0,96 \times 2 = 1,92 \geq 1 & \Rightarrow & a_{-4} = 1 \\
0,92 \times 2 = 1,84 \geq 1 & \Rightarrow & a_{-5} = 1 \\
0,84 \times 2 = 1,68 \geq 1 & \Rightarrow & a_{-6} = 1 \\
0,68 \times 2 = 1,36 \geq 1 & \Rightarrow & a_{-7} = 1
\end{array}$$

işlemleri gerçeklenir ve a_{-7} bulunana kadar tekrarlanır. Sonuç olarak, $(0,37)_{10} = (0,0101111)_2$ elde edilir.

Örnek-3.12.

$0,25_{10}$ sayısının 2 tabanına dönüştürülmesi için, n=2 ve p=2 olduğundan,

$$r = p\frac{\ln 10}{\ln n} = 2\frac{\ln 10}{\ln 2} = 6,64$$

haneye ihtiyaç vardır. Bu durumda 2 tabanındaki sayının kesirli kısmını 7 haneye kadar hesaplamak gerekecektir. On tabanındaki kesirli sayıyı iki tabanına dönüştürmek için,

$$\begin{array}{lll}
0,25 \times 2 = 0,50 < 1 & \Rightarrow & a_{-1} = 0 \\
0,50 \times 2 = 1,00 < 1 & \Rightarrow & a_{-2} = 1 \\
\text{Kalan: } 0,00 & &
\end{array}$$

işlemleri gerçeklenir. Ancak kalan 0 olduğu için işlem burada otomatik olarak sonlanır. Dolayısıyla sayıyı 2 tabanında ifade etmek için 7 haneli kesirli sayıya ihtiyaç yoktur. Sonuç olarak,

$$0,25_{10} = 0,01_2$$

elde edilir.

Örnek-3.13.

$253,263_{10}$ sayısının 8 tabanına dönüştürülmesi için, önce tam kısım için,

		Bölen		Bölüm	Kalan
253	:	8	=	31	5
31	:	8	=	3	7
3	:	8	=	3	3

şeklinde tekrarlamalı bölme işlemi yapılır. Bu işlem bölüm sıfır olana kadar sürer. Kalan sütunundaki sayılar ters sırada yazılarak yeni tabana dönüşüm gerçeklenmiş olur. Sonuç olarak,

$$253_{10} = 375_8$$

elde edilir. Kesirli kısım için, n= 8 ve p=3 olduğundan,

$$r = p\frac{\ln 10}{\ln n} = 3\frac{\ln 10}{\ln 8} = 3,32$$

haneye ihtiyaç vardır. Bu durumda 8 tabanındaki sayının kesirli kısmını 4 haneye kadar hesaplamak gerekecektir. On tabanındaki kesirli sayıyı sekiz tabanına dönüştürmek için,

$$\begin{array}{lll}
0,263 \times 8 = 2,104 \geq 1 & \Rightarrow & a_{-1} = 2 \\
0,104 \times 8 = 0,832 < 1 & \Rightarrow & a_{-2} = 0 \\
0,832 \times 8 = 6,656 \geq 1 & \Rightarrow & a_{-3} = 6 \\
0,656 \times 8 = 5,248 \geq 1 & \Rightarrow & a_{-4} = 5
\end{array}$$

işlemleri gerçeklenir ve a_{-4} bulunana kadar tekrarlanır. Sonuç olarak,

$$0,263_{10} = 0,2065_8$$

bulunur. Sayının tam ve kesirli kısımları birleştirilirse,

$$253,263_{10} = 375,2065_8$$

elde edilir.

3.2.3. m Tabanından n Tabanına Dönüşüm

m tabanından *n* tabanına geçiş aynen Ayrıt 3.2.2 de anlatılan 10 tabanından *n* tabanına geçiş işlemleri gibi gerçeklenir. Ancak burada dikkat edilmesi gereken önemli nokta işlemlerin *m* tabanında yapılması gerekliliğidir. Gerek tam kısmın gerekse kesirli kısmın dönüştürülmesinde bölme ve çarpma işlemleri *m* tabanına göre gerçeklenmelidir.

Örnek-3.14.

$0{,}7304_8$ sayısının 2 tabanına dönüştürülmesi için, m=8, n= 2 ve p=4 olduğundan,

$$r = p\frac{\ln m}{\ln n} = 4\frac{\ln 8}{\ln 2} = 12$$

haneye ihtiyaç vardır. Sekiz tabanındaki kesirli sayıyı iki tabanına dönüştürmek için bundan sonra klasik yöntem takip edilir. Yalnız burada dikkat edilmesi gereken önemli nokta çarpma işlemlerinin 8 tabanında gerçekleştirilmesinin gerekliliğidir.

$$\begin{array}{lcccl}
0{,}7304 & \times\ 2 & = & 1{,}6610 \geq 1 & \Rightarrow\ a_{-1} = 1 \\
0{,}6610 & \times\ 2 & = & 1{,}5420 \geq 1 & \Rightarrow\ a_{-2} = 1 \\
0{,}5420 & \times\ 2 & = & 1{,}3040 \geq 1 & \Rightarrow\ a_{-3} = 1 \\
0{,}3040 & \times\ 2 & = & 0{,}6100 < 1 & \Rightarrow\ a_{-4} = 0 \\
0{,}6100 & \times\ 2 & = & 1{,}4200 \geq 1 & \Rightarrow\ a_{-5} = 1 \\
0{,}4200 & \times\ 2 & = & 1{,}0400 \geq 1 & \Rightarrow\ a_{-6} = 1 \\
0{,}0400 & \times\ 2 & = & 0{,}1000 < 1 & \Rightarrow\ a_{-7} = 0 \\
0{,}1000 & \times\ 2 & = & 0{,}2000 < 1 & \Rightarrow\ a_{-8} = 0 \\
0{,}2000 & \times\ 2 & = & 0{,}4000 < 1 & \Rightarrow\ a_{-9} = 0 \\
0{,}4000 & \times\ 2 & = & 1{,}0000 \geq 1 & \Rightarrow\ a_{-10} = 0 \\
\text{Kalan:} & 0{,}000 & & &
\end{array}$$

İşlemi kalan 0 olduğu için burada sonlandırılır. Çarpma işlemlerinin 8 tabanına göre yapıldığını burada belirtmek gerekmektedir. Sonuç olarak,

$$0{,}7304_8 = 0{,}111011000100$$

elde edilir.

Örnek-3.15.

$1234{,}567_8$ sayısının 4 tabanına dönüştürülmesi için, önce tam kısım için,

		Bölen		Bölüm	Kalan
1234	:	4	=	247	0
247	:	4	=	51	3
51	:	4	=	12	1
12	:	4	=	2	2
2	:	4	=	0	2

şeklinde tekrarlamalı bölme işlemi yapılır. Bu işlem bölüm sıfır olana kadar sürer. Kalan sütunundaki sayılar ters sırada yazılarak yeni tabana dönüşüm gerçeklenmiş olur. Sonuç olarak,

$$1234_8 = 22130_4$$

elde edilir. Kesirli kısım için, m= 8, n= 4 ve p=3 olduğundan,

$$r = p\frac{\ln m}{\ln n} = 3\frac{\ln 8}{\ln 4} = 4{,}5 \cong 5$$

haneye ihtiyaç vardır. Sekiz tabanındaki kesirli sayıyı dört tabanına dönüştürmek için,

$$\begin{array}{lcccl}
0{,}567 & \times\ 4 & = & 2{,}734 \geq 1 & \Rightarrow\ a_{-1} = 2 \\
0{,}734 & \times\ 4 & = & 3{,}560 \geq 1 & \Rightarrow\ a_{-2} = 3 \\
0{,}560 & \times\ 4 & = & 2{,}700 \geq 1 & \Rightarrow\ a_{-3} = 2 \\
0{,}700 & \times\ 4 & = & 3{,}400 \geq 1 & \Rightarrow\ a_{-4} = 3 \\
0{,}400 & \times\ 4 & = & 2{,}000 \geq 1 & \Rightarrow\ a_{-5} = 2 \\
\text{Kalan :} & 0{,}000 & & &
\end{array}$$

işlemi kalan 0 olduğu için burada sonlandırılır. Çarpma işlemlerinin 8 tabanına göre yapıldığını burada belirtmek gerekmektedir. Sonuç olarak,

$$0{,}567_8 = 0{,}23232_4$$

bulunur. Sayının tam ve kesirli kısımları birleştirilirse,

$$1234{,}567_8 = 22130{,}23232_4$$

elde edilir.

3.2.3.1. 2 Tabanından 8/16 Tabanına Dönüşüm

2 tabanında ifade edilen bir sayı, üçer üçer gruplanarak ve gruplanan terimlerin 8 tabanındaki karşılığı yazılarak 8 tabanına dönüştürülebilir. Aynı şekilde bu sayı dörderli gruplanarak ve gruplanan terimlerin 16 tabanındaki karşılığı yazılarak 16 tabanına dönüştürülebilir.

Benzer biçimde, 8 tabanında ifade edilen bir sayının her terimi, üç bitle ifade edilerek 2 tabanına, 16 tabanında ifade edilen bir sayının her terimi, dört bitle ifade edilerek 2 tabanına dönüştürülür.

Örnek-3.16.

$(0,7304)_8$ sayısının 2 tabanına dönüştürülmesi için, 8 tabanındaki bir sayının üç bitlik 2 tabanındaki bir sayı ile ifade edilebileceği gerçeği göz önünden bulundurularak,

Sekiz Tabanı		İki Tabanı
7	=	111
3	=	011
0	=	000
4	=	100
$0,7304_8$	=	$0,111011000100_2$

şeklinde yazılabilir.

Örnek-3.17.

$1234,567_8$ sayısının 4 tabanına dönüştürülmesi için, 8 tabanındaki bir sayının üç bitlik 2 tabanındaki bir sayı ile ifade edilebileceği gerçeği göz önünden bulundurularak,

Sekiz Tabanı : 1 2 3 4 , 5 6 7
İki Tabanı : 001 010 011 100 , 101 110 111

elde edilir. Daha sonra bu sayıyı 4 tabanında ifade etmek için 2 tabanındaki sayı hem tam kısım hem de kesirli kısım virgülden başlanarak ikişerli gruplanır. Bu durumda,

İki Tabanı : 001 010 011 100 , 101 110 111
Dört Tabanı : 0 2 2 1 3 0 , 2 3 2 3 2

elde edilir. Sonuç olarak,

$1234,567_8 = 22130,23232_4$

şeklinde yazılabilir.

3.3. İşaretli Sayıların Gösterilimi

İşaretli sayılarda sayının artı veya eksi olduğu da belirtilir. Bu amaçla, sayının mutlak değerine ek olarak işareti de belirtilmelidir; temelde iki yöntemden biri kullanılır. Birinde sayının en solundaki bit (en anlamlı bit) işaret biti olarak kullanılır. Bu bit 1 ise eksi, 0 ise artı olarak değerlendirilir. Diğer yöntemde ise, sayının işareti için 2'ye tümleyen aritmetiği kullanılır. 2'ye tümleme, sayının 1'e tümlemesine 1 eklenerek bulunur; 1'e tümleme ise, sayının tüm bitlerinin ters çevrilmesiyle (1 olan 0, 0 olanlar 1 yapılır) elde edilir.

Buna göre, örneğin -5_{10} sayısının gösterilim şöyle olabilir:

1. Yönteme göre;

$-5_{10} \Rightarrow \underline{1}0101_2$

2. Yönteme göre;

$-5_{10} \Rightarrow 0101 \Rightarrow$ 1'e tümlenmişi $\Rightarrow 1010 \Rightarrow$ 2'ye tümleme $\Rightarrow 1011$

den, -5_{10} sayısının işaretli ikili karşılığı,

$-5_{10} \Rightarrow 1011_2$

olarak bulunur.

3.3.1. İşaretli Sayılarda Tümleme Aritmetiği

Tümleme aritmetiği işaretli sayılar üzerinde aritmetik işlem yapılırken kullanılır. İkili sayı sisteminde 2'ye ve 1'e tümleme işlemi, çıkartma işleminin toplama işlemiyle kotarılmasını sağlar. *Taban*'a göre tümleme ve *taban-1*'e göre tümleme...

3.3.1.1. Taban'a Göre Tümleme

Genel olarak, n tabanında verilmiş, q-1 rakamlı tam kısımdan oluşan $(N)_n$ sayısının tabana göre tümleyeni,

$$(T)_n = n^q - (N)_n$$

şeklinde tanımlanır. n^q sayısı q adet 0 ve daha üst kademede 1'den oluşur.

Örnek-3.18.

23_{10} sayısının 10'a tümleyeni,

$$(T)_n = n^q - (N)_n = 10^2 - 23 = 100 - 23 = 77_{10}$$

olarak bulunur. Tümleyen işlemine genelde çıkartma işlemlerinde ihtiyaç duyulur. Örneğin, $47_{10} - 23_{10}$ işlemi, tümleyen yardımıyla toplama işlemine dönüştürülür. $47_{10} + 77_{10} = \underline{1}24_{10}$ bulunur. Burada elde oluşması sonucun pozitif çıktığını gösterir. Bu yüzden, sayı değerlendirilirken başında yer alan 1 sayısı dikkate alınmaz.

Örnek-3.19.

18684_{10} sayısının 10'a tümleyeni,

$$(T)_n = n^q - (N)_n = 10^5 - 18684 = 100000 - 18684 = 81316_{10}$$

olarak bulunur.

Örnek-3.20.

$0{,}6642_{10}$ sayısının 10'a tümleyeni,

$$(T)_n = n^q - (N)_n = 10^0 - 0{,}6642 = 1 - 0{,}6642 = 0{,}3358_{10}$$

olarak bulunur. Buradan da görüldüğü gibi tam kısım olmadığı için q=0 alınmıştır.

Örnek-3.21.

111101_2 sayısının 2'ye tümleyeni,

$$(T)_n = n^q - (N)_n = 2^6 - 111101 = 1000000 - 111101 = 000011_2$$

olarak bulunur. Bu noktada, ikili tabanındaki bir sayının 2'ye tümleyeni bulmak için bir kolay yöntemden bahsedilebilir. Sayıda 1'ler yerine 0, 0'lar yerine 1 yazılır ve sonuca 1 eklenir. Bu durumda 111101_2 sayısı için $000010_2 + 1 = 000011_2$ elde edilir. Sonuç yukarıdakiyle aynıdır.

Örnek-3.22.

$0{,}101_2$ sayısının 2'ye tümleyeni,

$$(T)_n = n^q - (N)_n = 2^0 - 0{,}101 = 1 - 0{,}101 = 0{,}011_2$$

olarak bulunur. Basit yöntemle aynı işlem gerçeklenirse, tam kısım 0 olduğu için dikkate alınmayarak, 101 => 010 + 1 = 011 ve sonuç olarak $0{,}011_2$ elde edilir.

3.3.1.2. (Taban-1)'e Göre Tümleme

n tabanında verilmiş, q-1 rakamlı tam ve p rakamlı kesirli kısımdan oluşan $(N)_n$ sayısının (*taban-1*)'e göre tümleyeni,

$$(T)_b = n^q - n^{-p} - (N)_n$$

şeklinde tanımlanır. Eğer sayıda kesirli kısım yok ise p=0 alınır.

Örnek-3.23.

18684_{10} sayısının 9'a tümleyeni,

$$(T)_n = n^q - n^{-p} - (N)_n = 10^5 - 10^0 - 18684 = 99999 - 18684 = 81315_{10}$$

olarak bulunur.

Örnek-3.24.

$0{,}6642_{10}$ sayısının 9'a tümleyeni,

$$(T)_n = n^q - n^{-p} - (N)_n = 10^0 - 10^{-4} - 0{,}6642 = 1 - 0{,}6642 = 0{,}3357_{10}$$

olarak bulunur. Buradan da görüldüğü gibi tam kısım olmadığı için q=0, kesirli kısım göz önünde bulundurularak p=4 alınmıştır.

Örnek-3.25.

111101_2 sayısının 1'e tümleyeni,

$$(T)_n = n^p - n^{-q} - (N)_n = 2^6 - 2^0 - 111101 = 1000000 - 111110 = 000010_2$$

olarak bulunur.

Bu noktada, iki tabanındaki bir sayının 1'e tümleyeni bulmak için bir kolay yöntemden bahsedilebilir. Sayıda 1'ler yerine 0, 0'lar yerine 1 yazılır. Ancak burada 2'ye tümleyeni bulma işlemindeki gibi sonuca 1 eklenmez, işlem burada tamamlanır. Bu durumda 111101_2 sayısı için 000010_2 elde edilir. Sonuç yukarıdakiyle aynıdır.

Örnek-3.26.

$0{,}101_2$ sayısının 1'e tümleyeni,

$$(T)_n = n^q - n^{-p} - (N)_n = 2^0 - 2^{-4} - 0{,}101 = 1 - 0{,}110 = 0{,}010_2$$

olarak bulunur. Basit yöntemle aynı işlem gerçeklenirse, tam kısım 0 olduğu için dikkate alınmayarak, 101 => 010 ve sonuç olarak $0{,}010_2$ elde edilir.

3.3.2. Çıkarma İşleminin Tümleyen Aritmetiğiyle Yapılması

Örnek-3.15 te de belirtildiği gibi çıkarma işleminin gerçekleştirilmesinde *taban*'a veya (*taban-1*)'e tümleme mantığından yararlanılır. Tabana göre tümlemeyle çıkarma işleminde, çıkarılacak sayı bir eksi sayı olarak değerlendirilir, bu sayının tabana göre tümleyeni alınır ve pozitif sayı ile toplanır. Eğer sonuçta bir elde oluşursa, kalan sayının pozitif bir sayı olduğu anlaşılır. Elde dikkate alınmaz ve sonuç yazılır. Eğer sonuçta elde oluşmazsa, kalan sayının eksi olduğu anlaşılır. Bu durumda kalan sayının tabana göre tümleyeni alınır ve sayının önüne bir eksi işareti konularak sonuç yazılır.

Örnek-3.27.

$$\begin{array}{r} 66358_{10} \\ -\ 2164_{10} \\ \hline \end{array}$$

çıkarma işleminin tümleyen aritmetiği ile gerçekleştirilmesi örnek olarak verilmiştir.

Bu amaçla ilk olarak 2164 sayısının tabana göre tümleyenini bulmak gerekir. Bu işlem yapılırken dikkat edilmesi gereken bir nokta bulunmaktadır. 2164 sayısının tam kısmı 4 haneden oluşmasına rağmen çıkartma işlemi yapılacak sayı 5 haneli bir tam kısma sahiptir. Bu durumda;

$$(T)_n = n^q - (N)_n = 10^5 - 2164 = 100000 - 2164 = 97836_{10}$$

olarak bulunur. Bu aşamadan sonra işlem bir toplama işlemine dönüştürülmüştür. Dolayısıyla ilk sayı ile tabana tümleyeni alınmış sayı toplanır. Sonuçta,

$$\begin{array}{r} 66358 \\ +\ 97836 \\ \hline 1\ 64194 \end{array}$$

elde edilir. Elde oluşmasından dolayı bu işlemin sonucunun pozitif olduğu anlaşılır. Elde göz önünde bulundurulmaksızın sonucun $+64194_{10}$ olduğu söylenir.

Örnek-3.28.

$$\begin{array}{r} 2164_{10} \\ -\ 66358_{10} \\ \hline \end{array}$$

şeklindeki çıkarma işlemi gerçekleştirilmeye çalışılsın. Bu amaçla ilk olarak 66358 sayısının tabana göre tümleyenini bulmak gerekir. Bu durumda;

$$(T)_n = n^q - (N)_n = 10^5 - 66358 = 100000 - 66358 = 33642_{10}$$

olarak bulunur. Bu aşamadan sonra işlem bir toplama işlemine dönüştürülmüştür. Dolayısıyla ilk sayı ile tabana tümleyeni alınmış sayı toplanır. Sonuçta,

$$\begin{array}{r} 2164 \\ +\ 33642 \\ \hline 35806 \end{array}$$

elde edilir. Elde oluşmadığından bu işlemin sonucunun eksi olduğu anlaşılır. Bu durumda elde edilen sayının tabana göre tekrar tümleyeni alınır,

$$(T)_n = n^q - (N)_n = 10^5 - 35806 = 100000 - 35806 = 64194_{10}$$

Bulunan sonuç kesinlikle eksi bir sayıdır. Bu durumda sonucun -64194_{10} olduğu söylenir.

Örnek-3.29.

$$\begin{array}{r} 1110011_2 \\ -\ 1101010_2 \\ \hline \end{array}$$

şeklindeki çıkarma işlemi tabana göre tümleme alarak gerçekleştirilmeye çalışılsın. Bu amaçla ilk olarak ikinci sayının tabana göre tümleyeni alınır. Bu durumda, basit yöntemle,

$$1101010_2 \xrightarrow{\text{1'e tümleme}} 0010101_2 \xrightarrow{\text{sonuca 1 eklenir}} 0010110_2$$

olarak bulunur. Bu aşamadan sonra işlem bir toplama işlemine dönüştürülmüştür. Dolayısıyla ilk sayı ile tabana tümleyeni alınmış sayı toplanır. Sonuçta,

$$\begin{array}{r} 1110011_2 \\ +\ 0010110_2 \\ \hline 1\ \ 0001001_2 \end{array}$$

elde edilir. Elde oluşmasından dolayı bu işlemin sonucunun pozitif olduğu anlaşılır. Elde göz önünde bulundurulmaksızın sonucun $+0001001_2$ olduğu söylenir.

Örnek-3.30.

$$\begin{array}{r} 1101010_2 \\ -\ 1110011_2 \\ \hline \end{array}$$

şeklindeki çıkarma işlemi tabana göre tümleme alarak gerçekleştirilmiştir. Bu amaçla ilk olarak ikinci sayının tabana göre tümleyeni alınır. Bu durumda, basit yöntemle,

$$1110011_2 \xrightarrow{\text{1'e tümleme}} 0001100_2 \xrightarrow{\text{sonuca 1 eklenir}} 0001101_2$$

olarak bulunur. Bu aşamadan sonra işlem bir toplama işlemine dönüştürülmüştür. Dolayısıyla ilk sayı ile tabana göre tümleyeni alınmış sayı toplanır. Sonuçta,

$$\begin{array}{r} 1101010_2 \\ +\ 0001101_2 \\ \hline 1110111_2 \end{array}$$

elde edilir. Sonuçta elde oluşmadığı için bu işlemin sonucunun negatif olduğu anlaşılır. Bu durumda, en son elde edilen sayının tabana göre tümleyeni alınır ve başına eksi işareti konarak sonuç elde edilir. Kısaca;

$$1110111_2 \xrightarrow{\text{1'e tümleme}} 0001000_2 \xrightarrow{\text{sonuca 1 eklenir}} 0001001_2$$

elde edilir ve -001001_2 şeklinde işlem sonucu bulunur.

3.3.3. Çıkarmanın (Taban-1)'e Tümleyen Aritmetiğiyle Yapılması

(*Taban-1*)'e göre tümlemeyle çıkarma işleminde, çıkarılacak sayının (*taban-1*)'e göre tümleyeni alınır ve pozitif sayı ile toplanır. Eğer bir elde oluşursa, oluşan elde sonuçta elde edilen sayıya eklenir. İşlemin sonucu pozitiftir. Bu durumda işlem tamamlanmış sayılır. Eğer sonuçta elde oluşmazsa, kalan sayının negatif olduğu anlaşılır. Bu durumda kalan sayının (taban-1)'e göre tümleyeni alınır ve sayının önüne bir eksi işareti konularak sonuç yazılır.

Örnek-3.31.

$$\begin{array}{r} 1101_2 \\ -\ 0111_2 \\ \hline \end{array}$$

çıkarma işlemi (taban-1)'e göre tümlenerek gerçekleştirilsin. Bu amaçla ilk olarak ikinci sayının (taban-1)'e göre tümleyeni alınır. Bu durumda, basit yöntemle,

$$0111_2 \xrightarrow{\text{1'e tümleme}} 1000_2$$

olarak bulunur. Bu aşamadan sonra işlem bir toplama işlemine dönüştürülmüştür. Dolayısıyla ilk sayı ile tabana tümleyeni alınmış sayı toplanır. Sonuçta,

$$\begin{array}{r} 1101_2 \\ -\ 1000_2 \\ \hline 1\ \ 0101_2 \\ +\ \ \ \ \ \ 1 \\ \hline 0110_2 \end{array}$$

elde edilir. Elde oluşmasından dolayı bu işlemin sonucunun pozitif olduğu anlaşılır. Oluşan elde sayıya eklenir ve sonucun $+00100_2$ olduğu söylenir.

Örnek-3.32.

$$\begin{array}{r} 0111_2 \\ -\ 1101_2 \\ \hline \end{array}$$

çıkarma işlemi (taban-1)'e göre tümlenerek gerçekleştirilsin. Bu amaçla ilk olarak ikinci sayının 1'e tümleyeni alınır. Bu durumda, basit yöntemle,

$$1101_2 \xrightarrow{\text{1'e tümleme}} 0010_2$$

olarak bulunur. Bu aşamadan sonra işlem bir toplama işlemine dönüştürülmüştür. Dolayısıyla ilk sayı ile tabana tümleyeni alınmış sayı toplanır. Sonuçta,

$$\begin{array}{r} 0111_2 \\ -\ 0010_2 \\ \hline 1001_2 \end{array}$$

elde edilir. Burada elde oluşmadığı için sayının negatif olduğu anlaşılır. Sonucu elde etmek için sayının 1'e tümleyeni alınır ve başına eksi işareti ilave edilir. Bu durumda,

$$1001_2 \xrightarrow{\text{1'e tümleme}} 0110_2 \xrightarrow{\text{Sayı}} -0110_2$$

elde edilir.

3.4. Özet

Sayı sistemlerini, özellikle ikili ve onaltılı tabandaki sayıları iyi bilmek ve bu sayı sistemlerini çok iyi kullanabilmek lojik devre tasarımcısının öğrenmesi gereken temel ve en önemli konulardan biridir. Lojik devre tasarımcısı, bakış açısını sadece lojik tümdevrelerle sınırlandırmayıp mikroişlemci tabanlı sistemlere yöneltme ihtiyacını er ya da geç duyacaktır. Bu durumda özellikle program geliştirme aşamasında hem ikili taban hem de onaltılı taban üzerinde işlemlere ihtiyaç duyacak, eğer iyi bir sayı sistemleri bilgisine sahipse bu zorlukları kolaylıkla aşacaktır. Benzer biçimde, lojik devre tasarımı sırasında sayı sistemlerine duyulan ihtiyaç sayı sistemlerini kullanabilme yeteneğine sahip veya bir başka deyişle sayı sistemleri konusunu özümsemiş kullanıcı için herhangi probleme sebep olmadan ortadan kalkacaktır. Bu aşamalardan değerlendirildiğinde, sayı sistemlerini bilmek; kullanılan tabanları, tabanlar arası dönüşüm kurallarını, işaretli sayıların gösterilimini, toplama ve tümleyen aritmetiği ile çıkarma işlemlerini bilmek demektir. Bu bilgilere sahip tasarımcı ise, gerek lojik devre tasarımında gerekse mikroişlemcili sistem tasarımında en önemli adımı atmış olacaktır.

3.5. Sorular

1. 65_{10} sayısının 2, 8 ve 16 tabanındaki karşılıklarını bulunuz.
2. $0,5714_8$ sayısını 2 tabanına dönüştürünüz.
3. $4,654_8$ sayısını 2 ve 4 tabanına dönüştürünüz.
4. $4,654_{10}$ sayısını 8 tabanına dönüştürünüz.
5. 8 tabanında verilmiş olan (3746,72) sayısını 8 tabanında işlem yaparak ve bu işlemleri göstererek 6 tabanındaki sayıya çeviriniz.
6. 16 tabanında verilmiş olan $(A3,FB)_{16}$ sayısının 8 tabanındaki karşılığını bulunuz.
7. 8 tabanında verilmiş olan (7364,64) sayısını 8 tabanında işlem yaparak ve bu işlemleri göstererek 6 tabanındaki sayıya çeviriniz.
8. 32K×8 şeklinde tanımlanmış bir bellek birimi bulunmaktadır. Bu bellek biriminin en yüksek adresi 32K=(32×1024)-1 şeklinde hesaplanır. Buna göre bellek biriminin en yüksek adresini 16 tabanında ifade ediniz.
9. 16K×8 şeklinde tanımlanmış bir bellek birimi bulunmaktadır. Bu bellek biriminin en yüksek adresi 16K=(16×1024)-1 şeklinde hesaplanır. Buna göre bellek biriminin en yüksek adresini 2 tabanında ifade ediniz. Bulduğunuz 2 tabanındaki adresi doğru sayıda gruplayarak 16 tabanına çeviriniz.
10. 10101011_2 sayısının 1 ve 2'ye tümleyenlerini bulunuz.
11. $0,11011_2$ sayısının 1 ve 2'ye tümleyenlerini bulunuz.

12. $$\begin{array}{r} 3269_{10} \\ -\ 65536_{10} \\ \hline \end{array}$$

şeklindeki çıkarma işlemi tümleyen aritmetiğiyle gerçekleştiriniz.

13. $$\begin{array}{r} 01001_2 \\ -\ 10100_2 \\ \hline \end{array}$$

çıkarma işlemi (taban-1)'e tümleyen aritmetiğiyle gerçekleştiriniz.

14. $(BF)_{16}$ sayısını 2-tabanında ifade ediniz.

15. $$\begin{array}{r} 25_{10} \\ -\ 84_{10} \\ \hline \end{array}$$

şeklinde verilen sayıları önce 8 tabanında ifade ediniz, daha sonra çıkarma işlemini tabana tümleme aritmetiği ile gerçekleştiriniz.

16. $$\begin{array}{r} 10110_2 \\ -\ 01111_2 \\ \hline \end{array}$$

şeklinde verilen çıkarma işlemini 2'ye tümleme alarak gerçekleştiriniz.

17. 9 tabanında verilmiş olan $(846,8)_9$ sayısını 9-tabanında işlem yaparak ve bu işlemleri göstererek 4-tabanına çeviriniz {ln 9 / ln 4 ≅ 1,6 alınabilir}.

18. $$\begin{array}{r} 10110_2 \\ -\ 01111_2 \\ \hline \end{array}$$

şeklinde verilen çıkarma işlemini önce 1'e sonra da 2'ye tümleme alarak gerçekleştiriniz.

4.

Kodlama

Kodlama, bilginin veya verinin sayısal olarak gösterilmesi için kullanılan yöntemdir. Lojik devre tasarımı temelde ikili sayı düzenine dayanır; dolayısıyla sayısal olarak gösterilmesi gereken bilgi/veri önce kodlanarak sayısal olarak sembolize edilmelidir. Örneğin alfabede bulunan harflerin sayısal olarak gösterilmesi için, önce alfabedeki her harfe bir kod verilmelidir; örneğin A harfine 65 kodu verilmişse, bu harfin sayısal bir sistemde saklanması için 65'in ikili karşılığı saklanır; veya iki sistem arasında A harfi aktarılacaksa 65 sayısının ikili karşılığı aktarılır. Benzer şekilde alfabedeki tüm harflere birer kod verilirse, bir metnin saklanması, aslında 65, 66, 67 gibi sayıların saklanması demektir. Değişik kodlama çeşitleri vardır; bazıların verinin saklanması için kullanılırken, bazıları da veri haberleşmesinde kullanılır. Bu bölümde, sayısal sistemlerde kullanılan kodlama yöntemleri ve kod çeşitleri aşağıdaki başlıklar altında ele alınmıştır:

4.1. Kodlama Tanımı
4.2. Sayısal Kodlama
BCD Gösterim
BCD Sayılarla Aritmetik İşlemler
BCD Toplama
BCD Çıkarma
Üç Fazlalık Kodu
Üç Fazlalık Kodunda Aritmetik İşl.
Aiken Kodu
Aiken Kodunda Aritmetik İşlemler
Bitişik Kodlar ve *Gray* Kodu
4.3. Hata Sezme ve Onarma
Eşlik Biti Ekleme
Boyuna Fazlalık Sınaması - LRC
Çevrimli Fazlalık Sınaması - CRC
Hamming Kodlaması
4.4. Alfanumerik Kodlar
ASCII
Unikod
Türkçe Karakter Tablosu
4.5. Özet
4.6. Sorular

Bir sayısal sistem ile insanın diyalog kurabilmesi için kodlamadan yararlanılır. Sayısal sistemler 0 ve 1'den oluşan ikili sayılarla işlem yapabilmekte, ancak insan on tabanındaki sayıları mantıklı bulup, bu sayılarla çalışabilmektedir. Bu iki sistemin bir arada çalışabilmesi için sürekli bir kod dönüşümünün yapılması gerekmektedir. Buna bir örnek olarak insanın matematiksel işlemler için kullandığı hesap makinası

verilebilir. Hesap makinasının işlemleri gerçekleştiren kısmı Merkezi İşlem Birimidir (ya da yalnızca işlemci de denir). Ancak bu sistem ikili sayı sistemiyle çalışabilmektedir. Oysaki insan onlu sayı sistemiyle işlemleri düşünmektedir. Bu durumda tuş takımından girilecek olan değerlerin işlemciye uygun hale getirilmesi gerekmektedir. İşte burada iki birimi uyumlaştıran bir Kodlayıcı kullanılır. Bu kodlayıcı, verilen onlu sayıyı ikili bir sayıya çevirir. İşlemci, sonucu bulduktan sonra bu sonucu insana bildirmek için 7-parçalı bir göstergeden yararlanabilir. Ancak bu işlem için de bir kod dönüşümüne ihtiyaç duyulur. Merkezi işlem biriminden alınan sonuç, 7-parçalı göstergenin istenen parçasının yakılması suretiyle insanın anlayabileceği bir biçime dönüştürülür. Bu örnekten de görülebileceği gibi, insan makina diyalogunda, insan dilini makina diline çeviren bir kodlayıcı ve makina dilini insanın anlayabileceği dile dönüştüren kod çözücü kullanmak zorunluluğu bulunmaktadır.

Bilgisayar ortamında da, yazılara ait karakterler ikili sayı olarak saklanırlar; bu amaçla alfabedeki karakterlere birer sayısal kod verilir. Örneğin A için 65_{10}, B için 66_{10} kodları seçilmişse BABA sözcüğünün saklanması için bilgisayar belleğine sırasıyla 66, 65, 66 ve 65 sayılarının ikili tabandaki karşılıkları saklanır; yani, 1000010, 100001, 1000010 ve 1000001 ikili sayıları saklanır. Benzer şekilde alfabede bulunan harf, rakam, nokta virgül ve özel işaretler gibi karakterlere sayısal kod atanır. Farklı birçok alfabe/karakter kümesi vardır; aynı karaktere farklı kodlar verilmiş olunabilir. A'ya 65, B'ye 66 kodu ASCII tabloda geçerlidir (bkz. Ayrıt 4.4).

4.1. Kodlama Tanımı

Kodlama, sonlu elemanlı bir kümenin her bir elemanına birer tane kod verilmesi olarak ifade edilebilir. Daha sonra, bilgi bu kodlara dayanılarak gösterilir. Örneğin Türkçe alfabede 29 tane harf vardır; yani, alfabe sonlu elemanlı bir kümedir. Alfabedeki her bir harfe 0 ile 28 arasında birer sayı karşılığı atanarak kodlama tablosu elde edilebilir. Bu aşamadan sonra, Türkçe bir metin bu kodlama tablosuna dayanılarak saklanabilir.

Diğer bir örnek Mors alfabesi verilebilir; Mors alfabesinde biri nokta, diğerleri çizgi ve boşluk olmak üzere üç karakter vardır. Mors alfabesine dayanılarak ifade edilen birşey bu üç öğenin kombinasyonu ile ifade edilir. Dolayısıyla kod tablosu üç elemanlıdır. Kodlamanın genel ifadesi aşağıdaki gibi verilebilir. Burada,

S: Sonlu bir simge kümesi (0,1)

Σ: Sıralanmış simge dizileri kümesi ($\Sigma=\{000, 001, \ldots, 111\}$)

B: Sonlu veya sonsuz bir öğe kümesi ($K_0, K_2, \ldots, K_7$) olsun.

ile tanımlanmış olsun. Buna göre, her $b_i \in B$ elemanına, Σ kümesinde bir kod sözcüğü karşı düşürülmesi işlemine Kodlama adı verilir. Burada 000,001,...,111 gibi

simgeler b_i'nin birer kod sözcüğüdür; b_i'ler için sıralı simge dizisi kümesinde yalnızca bir kod sözcüğü vardır. Farklı iki öğenin kod sözcükleri her zaman birbirlerinden farklı olur.

Örnek-4.1.

Dört öğeli bir B kümesinin 0 ve 1 işaretleriyle kodlanması aşağıdaki gibi gerçekleştirilebilir:

0 1 00 10

Eğer kod uzunluğu hepsi için iki olması istenirse,

00 01 10 11

elde edilir. Kod sözcüğü uzunluğu n ve simge kümesindeki simgelerin sayısı m ise, sabit uzunlukta bir kodda m^n tane kod sözcüğü bulunabilir. Örneğin, simge kümesindeki simge sayısı $m=2$ ve kod sözcüğündeki simge sayısı $n=2$ ise, oluşturulacak olan kodda $m^n = 2^2 = 4$ tane kod sözcüğü vardır denir.

Bazı durumlarda oluşturulan kodların hepsi birden kullanılmayabilir. örneğin kullanılan kod sözcüğü uzunluğu k ise,

$$m^{n-1} \leq k \leq m^n$$

olabilir. Eğer kodlamada $k=m^n$ tane kod sözcüğü de kullanılıyorsa, bu şekilde oluşturulan kodlara <u>Artıksız Kodlar</u> adı verilir. Eğer $k<m^n$ olarak seçilmişse <u>Artıklı Kodlar</u> adı verilir.

Örnek-4.2.

Bir alfabede 8 tane harf olduğu varsayılsın. Buna göre bu alfabenin kod tablosunu S={0, 1} kümesine dayanarak oluşturunuz.

Simge kümesinde 0 ve 1 var; dolayısıyla bir simge kümesindeki bir öğe ile iki değişik durum ifade edilebilir. Ancak alfabede 8 değişik harf var; yani, sekiz değişik durumun ifade edilmesi gerekir. Bunun için bir harf birden çok simge kullanılarak temsil edilmelidir. Dolayısıyla, $n^2=8 \Rightarrow n=3$ hesabıyla kod uzunluğu olan $n=3$ olarak bulunur. 0 ve 1 simgeleri kullanılarak 3 uzunluktaki kodlar aşağıdaki gibidir:

$$\Sigma = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

Her harfe Σ kümesinden bir kod verilerek, alfabedeki harflere karşılık düşen kodlar da şöyle olabilir:

Harf	Kod	Harf	Kod
H_1	000	H_5	100
H_2	001	H_6	101
H_3	010	H_7	110
H_4	011	H_7	111

Örnek-4.3.

Bir isim kümesinin üyeleri {Oğuzhan, Volkan, Batuhan, Alper, Ege, Barış, Pınar ve Tayfun} olsun. Bu bilgileri sayısal olarak saklamak için ya her harfe verilen kodlar kullanılır ya da buradaki her isme bağımsız bir kod verilir. Bağımsız kod verilmesi durumunda her isme verilecek kodları belirleyiniz ve birinci yöntemle arasındaki farkı bellek gereksinimi açısından açıklayınız.

Her isme bağımsız bir kod verilmesi isteniyorsa şöyle bir yaklaşım yapılabilir: Sekiz tane isim vardır; dolayısıyla n=8 olur. her isme ikili tabanda bir kod (sayı) verilecek olduğu için $n=2^k$'dan $k=\log_2 n$ bağıntısından k=3 bulunur. Yani her isim için 3 bitlik bir sayı kullanılmalıdır. Bu durumda her isme verilecek kodlar aşağıdaki gibi olur:

000	Oğuzhan	100	Ege
001	Volkan	101	Barış
010	Batuhan	110	Pınar
011	Alper	111	Tayfun

Böyle bir kod atama işlemi yapıldıktan sonra, artık, *Oğuzhan* bilgisi sayısal olarak 000, *Volkan* bilgisi 001 ile temsil edilir.

İsimlerin saklanması için birinci yöntem kullanılmış olsaydı, her karaktere bir kod verilecekti; Türkçe'de 29 tane harf olduğu düşünülürse herbir harf ancak 5 bit ile kodlanabilecektir. Dolayısıyla herbir isimdeki karakter sayısı çarpı 5 miktarında bit gerekecekti. Bu da, aynı bilgileri saklamak için daha çok bit, dolayısıyla bellek gerektirecektir.

4.2. Sayısal Kodlama

Sayısal kodlama, sayıların bellekte tutulması şeklini belirler; sayılar ya doğrudan ikili karşılıkları kullanılarak ya da sayının her hanesindeki rakama bir kod atanarak temsil edilirler. Sayıların doğrudan ikili tabandaki karşılıkları kullanıldığında, ilgili sayı doğrudan ikili tabana dönüştürülür ve bellekte öyle temsil edilir. Örneğin, 6 sayısı 0110, 32 sayısı 100000, 63 sayısı 111111 olan karşılıklarıyla temsil edilirler. Diğer bir saklama şekliyse, sayının hanelerine ait rakamlara birer kod karşı düşürülmesiyle yapılır. Örneğin BCD saklama şeklinde 0 ile 9 arasındaki herbir rakama 4 bitlik bir kod atanır ve sayı saklanırken sayının hanelerindeki rakamlara ait kodlar

kullanılır. Örneğin 6 sayısı 0110, 32 sayısı 0011 0010, 63 sayısı 0110 0011 olan karşılıklarıyla temsil edilirler. BCD dışında üç fazlalık, *Aiken* ve *Gray* kodlama şekli kullanılır. İlerleyen kısımlarda bu kodların genel yapısı ve temel işlemler ele alınmıştır. Sayıların bilgisayar ortamında saklanması için birçok format kullanılır; bu formatların durumu sayının tamsayı, kesirli sayı, işaretsiz, işaretli olmasına göre değişir. Bu konuda ayrıntılı bilgi Bölüm 3'de verilmiştir; ayrıntılı bilgi için bakınız Bölüm 3.

4.2.1. İkili Kodlanmış Ondalık (BCD) Gösterim

10 tabanındaki bir sayıyı göstermek için 0, 1, ... 9 simgeleri kullanılır; bu simgelerden bir veya daha fazlası yanyana yazılarak 10 tabanında sayı yazılabilir: 1999, 1962, 16, 37 gibi. Böyle bir 10 tabanında sayı, sayısal ortamda değişik şekillerde saklanabilir; nasıl saklanacağını uygulama belirler. Örneğin, bilgisayar uygulamalarında, 10 tabanında bir tamsayı doğrudan ikili karşılığı bulunarak gösterilir. Ancak, lojik devre tasarımında ise, 10 tabanında sayıların gösterilimi için kısaca BCD diye adlandırılan kodlama yöntemi kullanılır. BCD'de ondalık her rakam uzunluğu 4 bit olan simgelerle temsil edilirler. Yani, n=4'tür. Kod sözcüğündeki simge sayısı ise sadece 0 ve 1 'ler kullanıldığı için m=2'dir. Kullanılabilecek kod sözcüğü sayısı ise,

$$m^{n-1} \le k \le m^n \quad \rightarrow \quad 2^3 \le k \le 2^4$$
$$8 \le k \le 16$$

olarak belirlenebilir. Bunun anlamı, birbirinden farklı 8 ile 16 arasında kod sözcüğü kullanılabileceğidir. BCD'de bu 16 sözcükten sadece 10 tanesi kullanılır. Bu yüzden BCD kodu artık bir kodlamadır. Tablo-4.1 de BCD'de kullanılan kodlar verilmiştir. Görüleceği üzere 0-9 arasındaki her rakama karşılığı olan bir ikili sayı düşürülmüştür.

Tablo-4.1. İkili kodlanmış ondalık (BCD) gösterilim

SAYI	BCD KOD SÖZCÜĞÜ
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Burada belirtilmesi gereken bir önemli nokta da, 10 tabanında verilmiş olan bir sayının 2 tabanındaki ifadesi ile BCD kodundaki ifadesinin tamamen birbirinden farklı olduğudur. Çünkü BCD ifade, onlu sayının ikili kodlanmasıyla elde edilir ve bir taban dönüşümü kesinlikle değildir. Örneğin $(24)_{10}$ sayısının iki tabanındaki karşılığı dönüşüm kuralları aracılığıyla $(11000)_2$ olarak elde edilirken BCD kodlanmayla $(0010\ \ 0100)_{BCD}$ yazılır. Dikkat edilirse, BCD kodlama, on tabanındaki sayının ikili kodlanmasıyla elde dilmektedir. Bu durumda BCD kodlamada sadece 0-9 arasındaki sayıların değerlendirildiği diğer ifadelerin kullanılmadığı göze çarpar. Bu da BCD kodunun gerçekten bir artıklı kod olduğunu göstermektedir.

BCD Sayılar Üzerinde Aritmetik İşlemler

BCD sayılar üzerinde aritmetik işlem yapmak günlük yaşamda kullanılan aritmetik işlemlere benzer. örneğin 2 sayı toplamına en sağdaki haneden başlanır soldaki hanelere doğru ilerlenir. Bu işlemler sırasında, bir hanenin sonucu 9'dan büyük olabilir. Bu durumda, o hane için, sonuçtan 9 çıkarılır ve haneye kalan yazılır; bir soldaki haneye elde geçer.

- **BCD Toplama**

1. Hanelerin toplanması sırasında sonuç 9'a eşit veya küçükse sonuç zaten BCD'dir.

Örnek-4.4.

$(22)_{10}$ sayısı ile $(14)_{10}$ sayısının BCD kodunda ifade edilmesi ve toplama işlemi aşağıdaki gibi gerçeklenir:

$$\begin{array}{r} (22)_{10} \\ +\ (14)_{10} \\ \hline (36)_{10} \end{array} \qquad \begin{array}{r} (0010\ \ 0010) \\ +\ (0001\ \ 0100) \\ \hline (0011\ \ 0110) \end{array}$$

Elde edilen sonuç da BCD kodundandır ve toplama işlemi sırasında herhangi bir elde oluşmamıştır.

2. Toplama sonucu [10-16) arasında ise, sonuç BCD kodunda değildir. Sonucu BCD olarak ifade etmek için, ikili kodlanmış 6 ekleyip oluşan eldeyi bir üst kademeye aktarmak gerekir.

Örnek-4.5.

$(24)_{10}$ sayısı ile $(38)_{10}$ sayısının BCD ifade edilmesi ve toplama işleminin gerçeklenmesi aşağıdaki gibi yapılır:

$$\begin{array}{r} (24)_{10} \\ +\ (39)_{10} \\ \hline (63)_{10} \end{array} \qquad \begin{array}{r} (0010\ \ 0100) \\ +\ (0011\ \ 1001) \\ \hline (0101\ \ 1101) \\ +\ \ \ \ \ \ \ \ \ \ 0110 \\ \hline +\ \ \ \ \ \ \ \ 1\ \ 0011 \\ \hline (0110\ \ 0011) \end{array}$$

BCD kodunda değil

3. Toplama sonucu [16-20) arasında ise, 5 bitli bir sonuç oluşur. Bu durumda sonuca yine ikili kodlanmış 6 ekleyip, oluşan beşinci bit komşu haneye yazılır.

Örnek-4.6.

$(7)_{10}$ sayısı ile $(9)_{10}$ sayısının BCD kodunda ifade edilmesi ve toplama işleminin gerçeklenmesi aşağıdaki gibi gerçeklenir:

$$\begin{array}{r} (7)_{10} \\ +\ (9)_{10} \\ \hline (16)_{10} \end{array} \qquad \begin{array}{r} (0111) \\ +\ \ \ \ \ \ (1001) \\ \hline (0001\ \ 0000) \\ +\ \ \ \ \ \ \ \ \ \ 0110 \\ \hline (0001\ \ 0110) \end{array}$$

- **BCD Çıkarma**

BCD'de çıkarma işlemini gerçekleştirmek için öncelikle çıkarılacak sayının 10'a tümleyenini almak gerekir. Bu aşamadan sonra 10 tabanındaki sayı BCD olarak ifade edilir ve toplama işlemi gerçeklenir.

Örnek-4.7.

$$\begin{array}{r} (69)_{10} \\ -\ (32)_{10} \\ \hline \end{array}$$

işleminin gerçeklenmesi şu şekilde yapılır:

Öncelikle 32 sayısının 10'a tümleyeni: 10^2-32 = 68 şeklinde bulunur. bu aşamadan sonra işlem toplama işlemine indirgenmiş olur.

$$\begin{array}{r} (69)_{10} \\ +\ (68)_{10} \\ \hline (1\ 37)_{10} \end{array} \qquad \begin{array}{rl} & (0110\ \ 1001) \\ + & (0110\ \ 1000) \\ \hline & (1100\ \ 1\,0001) \\ + & (0110\ \ \ \ 0110) \\ \hline & 10010\ \ 10111 \\ & +\ \ \ \ 1 \\ \hline & 1\ (0011\ \ 0111) \end{array}$$

$\downarrow$

$(37)_{10}$

İkisi de BCD kodunda değil

Oluşan bu elde sayının pozitif olduğunu belirttiği için dikkate alınmaz ve işlem sonucu on tabanındaki gösterilim ile aynı ve doğru olarak bulunmuş olur.

4.2.2. Üç Fazlalık Kodu (Excess-3)

Tablo-4.2 de gösterildiği gibi ikili sistemin üç fazlası alınarak oluşturulan kodlama işlemidir. Örneğin 0 sayısı, ikili sayı sistemindeki 3 sayısına eşittir.

Tablo-4.2. Üç fazlalık kodu.

SAYI	ÜÇ FAZLALIK KODU
0	0 0 1 1
1	0 1 0 0
2	0 1 0 1
3	0 1 1 0
4	0 1 1 1
5	1 0 0 0
6	1 0 0 1
7	1 0 1 0
8	1 0 1 1
9	1 1 0 0

Üç Fazlalık Kodunda Aritmetik İşlemler

BCD kodunda olduğu gibi, üç fazlalık kodunda iki sayının toplamından elde edilen sayının yine üç fazlalık kodunda olması gerekmektedir. Ancak bazı durumlarda üç fazlalık kodunda bir sayıyı ifade etmek için ilave işlemlere ihtiyaç duyulur. Toplama işlemi şu aşamalardan geçer;

1. Toplamada mevcut hanelerin dışında üst haneye geçen bir sayı bulunmazsa, ikili tabanda 3 çıkarmak gerekir.

Örnek-4.8.

$(2)_{10}$ sayısı ile $(3)_{10}$ sayısının üç fazlalık kodunda ifade edilmesi ve toplama işlemi aşağıdaki gibi gerçeklenir:

$$\begin{array}{r} (2)_{10} \\ +\ (3)_{10} \\ \hline (5)_{10} \end{array} \qquad \begin{array}{rl} & (0101) \\ + & (0110) \\ \hline & (1011) \\ - & (0011) \\ \hline & (1000) \end{array}$$

Elde edilen sonuç da üç fazlalık kodundadır ve on tabanında 5'e karşılık gelir.

2. Toplamada üst haneye geçen bir sayı oluşursa, ikili tabanda 3 eklenir. Bu ekleme işlemi, üst haneye geçen sayı için de yapılır.

Örnek-4.9.

$(5)_{10}$ sayısı ile $(8)_{10}$ sayısının üç fazlalık kodunda ifade edilmesi ve toplama işlemi aşağıdaki gibi gerçeklenir:

$$\begin{array}{r} (5)_{10} \\ +\ (8)_{10} \\ \hline (13)_{10} \end{array} \qquad \begin{array}{rl} & (1000) \\ + & (1011) \\ \hline & (1\ \ 0011) \\ + & (0011) \\ \hline & (0001\ \ 0110) \\ + & (0011) \\ \hline & (0100\ \ 0110) \end{array}$$

Elde edilen sonuç da üç fazlalık kodundadır ve on tabanında 13'e karşılık gelir.

4.2.3. Aiken Kodu

Aiken kodu Tablo-4.3 de verildiği gibi 0 ile 9 arasındaki sayıların ilk beş ve son beş rakamlarının ikili karşılıklarından oluşur; simetrik bir koddur.

Tablo-4.3. *Aiken* kodu.

SAYI	AIKEN KODU
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	1 0 1 1
6	1 1 0 0
7	1 1 0 1
8	1 1 1 0
9	1 1 1 1

Bu kodlama türünün özelliği, [0-4] arasındaki ilk beş sayının bilinen ikili kodlamaya eşdeğer olduğu, (5-9) arasındaki ikinci beş sayının ise, ilk beş sayının "1"'e tümleyeni olduğu söylenebilir. Örneğin 3_{10} sayısı bu kodlamada 0 0 1 1 şeklinde ifade edilir. Bu sayının 1'e tümleyeni 1 1 0 0'dır ve bu sayı aynı zamanda *Aiken* kodunda 6_{10} sayısına karşılık gelmektedir.

Aiken Kodunda Aritmetik İşlemler

Aiken kodunda toplama işlemi şu aşamalardan geçer;

1. *Toplama işleminin sonucu yine Aiken kodunda olduğundan düzeltmeye gerek olmayan durumlar vardır.*

Örnek-4.10.

$(3)_{10}$ sayısı ile $(6)_{10}$ sayısının *Aiken* kodunda ifade edilmesi ve toplama işlemi aşağıdaki gibi gerçeklenir:

$$\begin{array}{rl} & (3)_{10} \\ + & (6)_{10} \\ \hline & (9)_{10} \end{array} \qquad \begin{array}{rl} & (0011) \\ + & (1100) \\ \hline & (1111) \end{array}$$

Elde edilen sonuç *Aiken* kodundadır ve on tabanında 9 sayısına karşılık gelir.

2. *Toplama işlemi sonucunda doğru netice ile karşılaşılmaması halinde üst kademeye geçen bir elde yoksa sonuca ikili tabanda 6 eklenir.*

Örnek-4.11.

$(2)_{10}$ sayısı ile $(3)_{10}$ sayısının *Aiken* kodunda ifade edilmesi ve toplama işlemi aşağıdaki gibi gerçeklenir:

$$\begin{array}{rl} & (2)_{10} \\ + & (3)_{10} \\ \hline & (5)_{10} \end{array} \qquad \begin{array}{rl} & (0010) \\ + & (0011) \\ \hline & (0101) \\ + & (0110) \\ \hline & (1011) \end{array}$$

Elde edilen sonuç da üç fazlalık kodundadır ve on tabanında 5'e karşılık gelir.

3. *Toplam işlemi sonucunda üst haneye geçen bir elde oluşursa, sonuçtan ikili tabanda 6 çıkarılır.*

Örnek-4.12.

$(8)_{10}$ sayısı ile $(9)_{10}$ sayısının *Aiken* kodunda ifade edilmesi ve toplama işlemi aşağıdaki gibi gerçeklenir:

$$\begin{array}{rl} & (8)_{10} \\ + & (9)_{10} \\ \hline & (17)_{10} \end{array} \qquad \begin{array}{rl} & (1011) \\ + & (1100) \\ \hline & (1\ 0111) \\ - & (0110) \\ \hline & (0001\ 1101) \end{array}$$

Elde edilen sonuç da üç fazlalık kodundadır ve on tabanında 17 sayısına karşılık düşer.

4.2.4. Bitişik Kodlar ve Gray Kodu

Birbirini izleyen sayılara karşılık alınan ikili kod sözcükleri arasındaki *Hamming* uzaklığı 1 ise, bu tür kodlara *"bitişik kodlar"* adı verilir. Eğer kod sözcüklerinin ilki ile sonuncusu arasındaki uzaklık *Hamming*'e göre yine 1 ise bu tür kodlamalara çevrimli bitişik kod denilir.

Hamming Uzaklığı

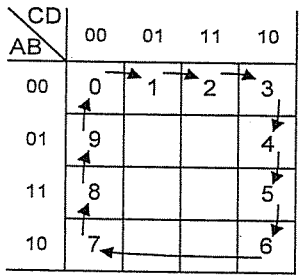
Hamming uzaklığı, Lojik Devre ifadelerin indirgenmesinde kullanılacak *Karnaugh* Diyagramlarının temelini oluşturan ve bu bakış açısıyla büyük önem taşıyan bir kavram olarak karşımıza çıkmaktadır. Matematiksel olarak tanımlamak gerekirse; n bileşenli bir $\underline{x}$ vektörü $\underline{x}=(x_1,\cdots,x_n)$ şeklinde tanımlansın. $\underline{x}^0$, B_2^n'de bir nokta veya bir kombinasyon olsun. Bu durumda

$$x_i \in B_2 \quad \Rightarrow \quad \underline{x} \in B_2^n$$

yazılabilir. Bu tanımlar çerçevesinde B_2^n'in iki noktası $\underline{x}^0$ ile $\underline{x}^1$ arasındaki uzaklık, noktaların farklı bileşenlerinin sayısı olarak tanımlanır, $d(\underline{x}^0,\underline{x}^1)$ ile gösterilir ve iki kod kelimesi arasındaki farklı bileşenlerin sayısı olarak yorumlanır. Örneğin, $d(01101,01111)=1$ dir.

Örnek-4.13.

Bu örnekte, dört bitten oluşan çevrimli bir BCD kodlama oluşturulacaktır. Bu amaçla, Şekil-4.1 de verilen, 2 girişli, 4 satırlı ve 4 sütunlu bir diyagram çizilir. Bu diyagrama *Karnaugh* Diyagramı adı verilir. Bu tablonun bir özelliği kenar hanelerin birbiriyle komşu olduğunun kabul edilmesidir. Dolayısıyla bu kenarların *Hamming* uzaklıkları bir olur. Oluşturulan kodlama Tablo-4.4 te gösterilmiştir. Şekil-4.1 den görülebileceği gibi oluşan kod, birbirine *Hamming* Uzaklığı 1 olan kod sözcüklerinden oluşmuştur. Bu yüzden oluşturulmuş olan kod çevrimli bir kod olmasına rağmen 16 kod sözcüğünden sadece 10 tanesi kullanıldığı için aynı zamanda artıklı bir kod olarak nitelendirilir.



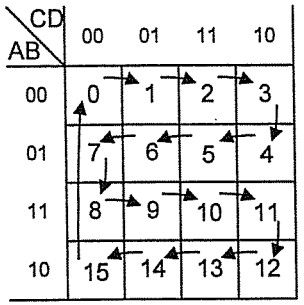
Şekil-4.1. *Karnaugh* diyagramı.

Tablo-4.4. Oluşturulan kodlama.

SAYI	KOD
0	0000
1	0001
2	0011
3	0010
4	0110
5	1110
6	1010
7	1000
8	1100
9	0100

- ***Gray* Kodu**

Gray kodu, 2^n öğeli bir küme için 2 tabanında artıksız ve çevrimli bir kodlama oluşturulmak suretiyle elde edilir. Bu amaçla, Şekil-4.2 de verilen, 2 girişli, 4 satırlı ve 4 sütunlu *Karnaugh* diyagramı oluşturulur. Bu diyagramda kenar hanelerinin birbiriyle *Hamming* uzaklıkları bir olur. Oluşturulan kodlar Tablo-4.5 te gösterilmiştir.



Şekil-4.2. *Gray* koduna ilişkin *Karnaugh* diyagramı

Tablo-4.5. *Gray* kodu rakamları.

SAYI	İKİLİ SİSTEM	GRAY KODU
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

4.3. Hata Sezme ve Onarma

Hata sezme ve onarma, sayısal ortamda saklanan veya iki nokta arasında aktarılan verinin bozulması durumunda hatanın sezilmesi ve onarılması konusunu kapsar. Sayısal verilerin saklanmasında ve aktarılmasında en çok karşılaşılan hata biçimi, herhangi bir yerdeki 1'in yerini 0'ın veya 0'ın yerini 1 almasıdır; veya bir grup bit'in değer değiştirmesi şeklinde kendisini gösterir. Böyle durumlarda, hatanın sezilmesi ve belirli düzeye kadar onarılması için, kod sözcüğü hata sezme ve onarma bilgilerini de içerecek şekilde oluşturulabilir.

Hata sezilmesi için kullanılabilecek birçok yöntem vardır; en basiti belirli sayıda bit grubuna 1 adet eşlik bit'i eklenmesidir. Eşlik bit'i ekleme, bir nebze hata oluştuğunu algılatabilir. Hatanın hissedilme olasılığının arttırılması için LRC, CRC olarak adlandırılan yöntemler kullanılır. Uygulama da, genel olarak, bilginin saklanmasında eşlik biti ekleme, bilgisayara ağları üzerinden veri haberleşmesinde CRC yöntemleri kullanılır.

Hata sezmenin yanında hatanın düzeltilmesi de gereklidir. Hata düzeltme, hatalı kısmın tekrar istenmesiyle veya kodlamanın hata düzeltme özellikleri içerecek biçimde yapılmasıyla kotarılır. Hata düzeltmeli kodlamada, en yaygın kullanılan yöntem *Hamming* kodlamasıdır.

İlerleyen ayrıtlarda sırasıyla eşlik biti ekleme ve LRC, CRC ve *Hamming* kodlama ele alınmıştır.

4.3.1. Eşlik biti Ekleme

Eşlik bit'i ekleme bit dizisi (veri) içerisindeki tek sayıda hatayı sezmek için kullanılır. Eşlik bitinin değeri katar içindeki 1'lerin sayısına göre belirlenir; 1'lerin sayısının çift veya tek olmasına göre eşlik bitinin değeri 1 ya da 0 yapılır. Çift (*even*) ve tek (*odd*) olarak adlandırılan iki ayrı uygulaması vardır: Çift eşlikte eşlik bitiyle beraber birlerin sayısı çift, tek eşlikte ise tek olur. Bu uygulama daha çok boyu 7, 8 bit gibi kısa olan verilerin saklanmasında veya aktarılmasında kullanılır. Ancak, hata sayısı çift olursa hata sezilemez; tek sayıda hata olursa sezilir. [ÇÖLKESEN ve ÖRENCİK-1999]

Örneğin, BCD kodunda tek eşlik için kodlar ve eşlik bitleri Tablo-4.6 da, çift eşlik için kodlar ve onlara ait eşlik bitleri Tablo-4.7 de gösterilmiştir.

Tablo-4.6. Tek eşlik biti ve BCD kodu.

SAYI	BCD KODU	Eşlik Biti (tek)
0	0000	1
1	0001	0
2	0010	0
3	0011	1
4	0100	0
5	0101	1
6	0110	1
7	0111	0
8	1000	0
9	1001	1

Tablo-4.7. Çift eşlik biti ve BCD kodu.

SAYI	BCD KODU	Eşlik Biti (çift)
0	0000	0
1	0001	1
2	0010	1
3	0011	0
4	0100	1
5	0101	0
6	0110	0
7	0111	1
8	1000	1
9	1001	0

Eşlik bitinin kullanılması halinde sadece hatanın varlığı sezilir; ancak hangi bitte olduğunu belirlemek mümkün değildir. Eğer hatanın hangi bitte olduğu öğrenilmek isteniyorsa, denetim biti olarak adlandırılan ikinci bir eşlik biti kullanılabilir. Bu yöntem Boyuna Fazlalık Sınaması (LRC) olarak adlandırılır.

4.3.2. Boyuna Fazlalık Sınaması - LRC

Boyuna Fazlalık sınaması, kısaca LRC (*Longitudinal Redundancy Check*) birden çok bit dizisi içeren veri üzerindeki hatayı sezmek için kullanılır. Bu yöntemde bit dizisi alt alta koyulur ve hem yatay hem de dikey olarak eşlik bitleri bulunur. Yatay olan eşlik biti, dikey olan sınama (kontrol) biti olarak adlandırılır. Bu yöntemde bir bitin

bozulması hem bir bit dizisine ait eşlik bitinin, dikey olarak elde edilen sınama bitinin değerini etkileyeceği için hatanın yerini de bulmak (hata düzeltme) mümkün olur. Bu yöntemde en kötü durum 2 ayrı bit dizisinde aynı bit pozisyonundaki ikişer bitinin birden bozulması durumudur; dört bitin bozulduğu bu durumda hata sezilemez.

Örnek-4.14.

7294 sayısı BCD kodunda kodlanacak ve bir noktadan diğerine, "1"lerin sayısını tek yapan eşlik biti ve "1"lerin sayısını çift yapan kontrol biti kullanılarak aktarılacaktır. Bu durumda,

Sayı	BCD Karşılığı	Eşlik biti
7	0 1 1 1	0
2	0 0 1 0	0
9	1 0 0 1	1
4	0 1 0 0	0
Sınama biti	1 0 0 0	

elde edilir. Eğer BCD sözcüğün aktarımı sırasında herhangi bir bit hatası oluşursa bu hatanın hangi sayıda ve hangi bit'te olduğunu belirlemek mümkündür. Örneğin, 2 sayısının kodu karşı tarafa 0010 yerine 0000 şeklinde aktarılırsa eşlik ve sınama bit'leri aracılığıyla hatanın yerini belirlemek olasıdır. Öncelikle eşlik bit'ini tek yapan yerler belirlenir ve hemen 2 sayısına karşılık gelen 0000 sözcüğünde hata olduğu anlaşılır. Sınama bit'lerine bakılarak yukarıdan aşağıya 1000 hanesinde sınama bit'inin çift olmadığı görülür. Aşağıda hatanın nasıl algılanacağı gösterilmiştir.

Sayı	BCD Kodu Karşılığı	Eşlik Biti
7	0 1 1 1	0
2	0 0 0 0	0
9	1 0 0 1	1
4	0 1 0 0	0
Sınama biti	1 0 0 0	

Hatalı aktarılan sözcükteki hatalı bit

4.3.3. Çevrimli Fazlalık Sınaması – CRC

Bu yöntemde, bir grup bit dizisi arkasına CRC (*Cyclic Redundancy Check*) olarak adlandırılan sınama bitleri eklenir. CRC sınama bitleri şöyle hesaplanır:

1. Veriye ait bit dizisi P(x) adı verilen bir polinom ile gösterilir. Polinomun katsayıları gönderilen veri katarının ilgili pozisyonundaki bitlerinin değeridir. Aktarılacak bilginin uzunluğu *n* bit ise, buna göre polinom aşağıdaki gibi olur:

$$P(x) = b_{n-1} \cdot x^{n-1} + b_{n-2} \cdot x^{n-2} + \ldots + b_1 \cdot x^1 + b_0 \cdot x^0$$

İlk bitin değeri, polinomun x^{n-1} teriminin katsayısı; 2. bitin değeri, x^{n-2} teriminin katsayısı, ..., bilginin son bitinin değeri de polinom x^0 teriminin katsayısı olur. Örneğin aşağıdaki veri katarına karşı düşen *P(x)* polinomu hemen yanında gösterildiği gibi olur.

1 0 1 0 0 1 0 1 1 1 bit dizisine *karşı düşen polinom*

$$P(x) = 1 \cdot x^9 + 0 \cdot x^8 + 1 \cdot x^7 + \ldots + 1 \cdot x^0$$
$$= x^9 + x^7 + x^4 + x^2 + x + 1$$

2. Bu *P(x)* polinomu x^p ile çarpılır. Bu işlem sonucu elde edilen yeni polinoma karşı düşen bit katarı, aslında, önceki bit katarı ile onun sonuna iliştirilmiş *p* tane 0 bitinden oluşur:

1 0 1 0 0 1 0 1 1 1 0 0 ..... 0 0

p tane sıfır

3. $x^p \cdot P(x)$ polinomu, *p.* dereceden *G(x)* adı verilen bir üreteç (*generating*) polinomuna bölünür. Üreteç polinomları belirli hata sezme özellikleri bulunan bir takım standart polinomlardır; aşağıda 3 tane örnek üreteç polinomu verilmiştir:

$$x^{16} + x^{15} + x^2 + 1;\ x^{16} + x^{12} + x^5 + 1;\ x^{12} + x^{11} + x^3 + x^2 + x + 1$$

4. $x^p \cdot P(x)\ /\ G(x)$ bölümünü kodlayan hesaplasın. Bu bölme işleminde bölümü *Q(x)*, kalanı *R(x)* ile gösterelim.

$$x^p \cdot P(x) = Q(x) \cdot G(x) + R(x)$$

İşlemlerde modülo 2 aritmetiği kullanalım. Bu durumda toplama ve çıkarma aynı işlemi tanımlar:

$0 + 0 = 0 ; 0 - 0 = 0$
$0 + 1 = 1 ; 0 - 1 = 1$
$1 + 0 = 1 ; 1 - 0 = 1$
$1 + 1 = 0 ; 1 - 1 = 0$ Yani,

$x^p \cdot P(x) - R(x) = Q(x) \cdot G(x)$
$x^p \cdot P(x) + R(x) = Q(x) \cdot G(x)$

olmalıdır.

Kodlayan, kod çözücüye $P(x)$ polinomuna karşı düşen bit dizisi yerine $x^p \cdot P(x) + R(x)$ polinomunu göndersin. Bu bit dizisi asıl gönderilecek bilgi bit dizisi ile bunun sonuna eklenmiş p uzunluğunda ek bir diziden ibarettir. Gönderilen toplam diziye ilişkin polinom, $G(x)$ polinomunun bir tam katı idi ve G(x) alıcı tarafından önceden bilinmekteydi. Herhangi bir bitin bozulmadığını ve $Q(x) \cdot G(x)$'in kod çözücüye aynen ulaştığını varsayalım: Kod çözücü, kendisine gelen bit dizisine karşı düşen polinomu $G(x)$'e böler. Bu bölmenin sonunda kalan olmamalıdır; kod çözen *kalan = 0* olması durumunda hatasız iletim olduğunu belirler; gelen bit dizisinin en sonundaki p tanesini atar; geriye kalan n bit bilgiyi belirtir. [ÇÖLKESEN ve ÖRENCİK-1999]

Örnek-4.14.

1010010111 bit dizisine eklenecek CRC sınama bitlerini hesaplayınız.

Buna karşı düşen polinom, $P(x) = x^9 + x^7 + x^4 + x^2 + x + 1$ olur. Üreteç fonksiyonu, $G(x) = x^4 + x^2 + x + 1$ seçilmiş olsun. Bu durumda polinom ile üreteç fonksiyonunun çarpımı şöyle olur:

$$x^4 P(x) = x^{13} + x^{11} + x^8 + x^6 + x^5 + x^4$$

Buradan CRC sınama bitleri aşağıdaki gibi hesaplanır:

$$\begin{array}{l|l}
x^{13} + x^{11} + x^8 + x^6 + x^5 + x^4 & x^4 + x^2 + x + 1 \\ \hline
x^{13} + x^{11} + x^{10} + x^9 & x^9 + x^6 + x^5 + x^2 \text{ bölüm} \\
\hline
x^{10} + x^9 + x^8 + x^6 + x^5 + x^4 & \\
x^{10} + x^8 + x^7 + x^6 & \\
\hline
x^9 + x^7 + x^5 + x^4 & \\
x^9 + x^7 + x^6 + x^5 & \\
\hline
x^6 + x^4 & \\
x^6 + x^4 + x^3 + x^2 & \\
\hline
x^3 + x^2 \text{ kalan} &
\end{array}$$

$x^p P(x) = x^{13} + x^{11} + x^8 + x^6 + x^5 + x^4$ - *bölünen*
$G(x) = x^4 + x^2 + x + 1$ - *bölen*
$Q(x) = x^9 + x^6 + x^5 + x^2$ - *bölüm*
$R(x) = x^3 + x^2$ - *kalan*

Bu işlemler sonucu sınama bitleri dahil toplam bit dizisi,

$$Q(x)\,G(x) = x^p P(x) + R(x) = x^4 (x^9 + x^7 + x^4 + x^2 + x + 1) + x^3 + x^2$$
$$= x^{13} + x^{11} + x^8 + x^6 + x^5 + x^4 + x^3 + x^2$$

polinomuna göre aşağıdaki gibi olmalıdır.

1 0 1 0 0 1 0 1 1 1 <u>1 1 0 0</u>
bilgi bitleri *CRC bitleri*

Kod çözücü, bu diziye karşı düşen polinomu $G(x) = x^4 + x^2 + x + 1$ 'e böler, doğal olarak kalanı 0 bulur; en son gelen 4 biti atar ve veriye ait bitlerini böylece doğru olarak elde etmiş olur. CRC hesabı, lojik olarak öteleme ve TVEYA işlemleriyle yapılabilir. CRC yönteminde p. dereceden bir üreteç polinomu kullanıldığında bit dizisi içinde ilk bozulan bit ile son bozulan bit arasındaki uzaklığın p'den kısa olması durumunda bu hata alıcıda kesin olarak sezilir; çünkü bu durumda $E(x)$'in, $G(x)$'e tam olarak bölünmesi mümkün değildir.

4.3.4. Hamming Kodlaması

Hata düzeltmek için kullanılan en yaygın yöntem *Hamming* kodlamasıdır. Bu kodlamada üretilen kod bir ölçüde bozulmuş bile olsa da yeniden doğru olarak elde edilebilir. Herbir kod sözcüğünün en çok j bitinin bozulması durumunda hatayı düzeltmenin mümkün olduğu koda j bit hata bağışıklığı olan kod denir. Örneğin kodlanan karakter kümesi {A, B }, kod kümesi {000, 111 } seçilmişse, 1 bitlik hata bağışıklığı olan veya 2 bitlik hataların sezilebildiği bir kod elde edilir. Bu kodun her sözcüğünün 1 veri biti ve 2 sınama biti vardır. Örneğin, *Hamming* (7, 4) kodunda, 4 adet bilgi biti ve 3 adet sınama biti kullanılarak 7 bitlik kod sözcükleri elde edilir. Kodlanan karakter kümesi 16 elemanlıdır. Kod sözcüğü: $I_1 I_2 I_3 I_4 C_1 C_2 C_3$

$C_1 = I_1 I_2 I_4$ 'e ilişkin çift eşlik biti
$C_2 = I_1 I_3 I_4$ 'e ilişkin çift eşlik biti
$C_3 = I_2 I_3 I_4$ 'e ilişkin çift eşlik biti

Bu kod, bir bitlik hata düzeltebilir veya 2 bitlik hata sezebilir. Bir başka açıdan ele alınırsa, en çok 1 bitin bozulacağını varsayalım. Bilgi bitlerini $I_1, I_2, \ldots, I_n$ ile gösterelim. Hatayı düzeltmek için bilgiye eklenen bitleri (sınama bitleri) de $C_1, C_2, \ldots, C_m$ ile gösterelim.

$C_i = f(I_1, I_2, ..., I_n)$ şeklinde hesaplanacaktır. f olarak bir TVEYA ⊕ fonksiyonu kullanılır:

$C_1 = I_1 \oplus I_2 \oplus I_4$

$C_2 = I_1 \oplus I_3 \oplus I_4$

$C_3 = I_2 \oplus I_3 \oplus I_4$

Kodlayıcı ve kod çözücü C_i 'lerin i ε {1, ..., m}, I_j 'ler j ε {1, ..., n} cinsinden nasıl hesaplanacağını bilmelidir.

Yukarıdaki örnekte 4 bilgi biti ve 3 denetim biti kullanılır. Gönderilen bit dizisi,

$I_1 I_2 I_3 I_4 C_1 C_2 C_3$

En çok 1 bit bozulmuş olarak geldiğini varsayalım. I_1 bozulduğunda hem C_1 hem C_2 'nin alıcıda hesaplanan değeri ile kendisine ulaşan değerleri farklı olacaktır. I_2'de aynı durum C_1 ve C_3 için, I_3'de C_2 ve C_3 için , I_4'de C_1 , C_2 ve C_3 için olur. C_1 bozulduğunda ise yalnız C_1' in hesaplanan ve alınan değerleri farklı olur. Aynı şey C_2 veya C_3 bozulduğunda da söylenebilir; 1 bit bozulduğunda alıcı kesin olarak hangi bitin bozulduğunu anlayabilir ve bunu düzeltebilir.

Örnek-4.15.

Kod çözücüye

I_1	I_2	I_3	I_4	C_1	C_2	C_3
1	0	1	0	1	0	1

gönderilmiş olsun, ancak 1 0 0 0 1 0 1 gelsin. Hatanın nasıl düzeltileceğini açıklayınız.

Kod çözücü C_1, C_2 ve C_3'ü, I_1 I_2 I_3 I_4 cinsinden hesaplar: $C_1 = 1$, $C_2 = 1$, $C_3 = 0$. Hem C_2, hem de C_3'ün hesaplanan ve gelen değerleri birbirinden farklı; o halde I_3 bozuk. Alıcı I_3'ün değerini değiştirerek doğru bilgiye ulaşır. [ÇÖLKESEN ve ÖRENCİK-1999]

4.4. Alfanumerik Kodlar

Alfanumerik kodlar, sayısal sistemlerde rakamlar dışında harf, simge ve özel karakterlerin kodlanması için kullanılır. Örneğin yazı, denklem alfanumerik kodlar aracılığıyla sayısallaştırılabilir; bu yöntemde bir karakter tablosu vardır ve herbir karaktere birer ayrık kod verilir. Böylece, o karakter kümesindeki (alfabeden) karakterlerden oluşan veri, karakterlere ait sayısal kodların saklanmasıyla oluşturulur. Bilgisayar sistemlerin yaygın olarak kullanılan alfanumerik kodlama ASCII olarak adlandırılan kod tablosuna göre yapılır; bir de *unikod* olarak adlandırılan kodlanma tablosu vardır.

ASCII *(American Standard Code for Information Interchange)*

ASCII kodlamada, temelde, herbir karakter için 7 bit kullanılır; dolayısıyla 128 farklı karakter vardır. Genişletilmiş ASCII (Extended ASCII) olarak adlandırılan ve toplam 256 karakter içeren uyarlamasında ise her karakter 8 bit'ten oluşur. Tablo-4.8 de ASCII tablodaki karakterler ve onların 10 ve 16'lık tabandaki sayısal kodları görülmektedir.

Tablo-4.8. ASCII tablodaki karakterler ve sayısal kodlar (7 bitlik).

On	Onaltı	kr.	On	Onaltı	kr.	On	Onaltı	kr	On	Onaltı	kr
0	*0*	NULL	32	*20*	boşluk	64	*40*	@	96	*60*	`
1	*1*	SOH	33	*21*	!	65	*41*	A	97	*61*	a
2	*2*	STX	34	*22*	"	66	*42*	B	98	*62*	b
3	*3*	ETX	35	*23*	#	67	*43*	C	99	*63*	c
4	*4*	EOT	36	*24*	$	68	*44*	D	100	*64*	d
5	*5*	ENQ	37	*25*	%	69	*45*	E	101	*65*	e
6	*6*	ACK	38	*26*	&	70	*46*	F	102	*66*	f
7	*7*	BELL	39	*27*	'	71	*47*	G	103	*67*	g
8	*8*	BS	40	*28*	(	72	*48*	H	104	*68*	h
9	*9*	HT	41	*29*	)	73	*49*	I	105	*69*	i
10	*A*	LF	42	*2A*	*	74	*4A*	J	106	*6A*	j
11	*B*	VT	43	*2B*	+	75	*4B*	K	107	*6B*	k
12	*C*	FF	44	*2C*	,	76	*4C*	L	108	*6C*	l
13	*D*	CR	45	*2D*	-	77	*4D*	M	109	*6D*	m
14	*E*	SO	46	*2E*	.	78	*4E*	N	110	*6E*	n
15	*F*	SI	47	*2F*	/	79	*4F*	O	111	*6F*	o
16	*10*	DLE	48	*30*	0	80	*50*	P	112	*70*	p
17	*11*	DC1	49	*31*	1	81	*51*	Q	113	*71*	q
18	*12*	DC2	50	*32*	2	82	*52*	R	114	*72*	r
19	*13*	DC3	51	*33*	3	83	*53*	S	115	*73*	s
20	*14*	DC4	52	*34*	4	84	*54*	T	116	*74*	t
21	*15*	NAK	53	*35*	5	85	*55*	U	117	*75*	u
22	*16*	SYN	54	*36*	6	86	*56*	V	118	*76*	v
23	*17*	ETB	55	*37*	7	87	*57*	W	119	*77*	w
24	*18*	CAN	56	*38*	8	88	*58*	X	120	*78*	x
25	*19*	EM	57	*39*	9	89	*59*	Y	121	*79*	y
26	*1A*	SUB	58	*3A*	:	90	*5A*	Z	122	*7A*	z
27	*1B*	ESC	59	*3B*	;	91	*5B*	[	123	*7B*	{
28	*1C*	FS	60	*3C*	<	92	*5C*	\	124	*7C*	|
29	*1D*	GS	61	*3D*	=	93	*5D*	]	125	*7D*	}
30	*1E*	RS	62	*3E*	>	94	*5E*	^	126	*7E*	~
31	*1F*	US	63	*3F*	?	95	*5F*	_	127	*7F*	del

ASCII tablodaki ilk 32 karakter kontrol karakterleri olarak adlandırılır; daha çok belirli bir işleve sahiptirler. Örneğin 7 nolu karakter *bip* sesi üreten bir kontrol karakteridir. Kontrol karakterleri uygulama alanına göre farklı işlevlere sahip olabilir. Örneğin bilgisayar ortamında yapılan dosyalama işlemlerinde farklı bilgisayar haberleşmesinde farklı olabilir.

Tablo-4.8 den görüleceği üzere büyük harflerle küçük harfler farklı kodlara sahiptirler. Örneğin büyük A, 65; küçük a, 97 koduna sahiptirler.

Örnek-4.16.

ASCII tabloya göre "Baba bana top al" cümlesi hangi sayısal kodların ard arda gelmesiyle oluşur?

Yapılması gereken herbir karaktere ait sayısal kodların bulunması ve ard arda yazılmasıdır. Buna göre "Baba bana top al" cümlesi"

66 97 98 97 98 97 110 97 116 111 112 97 108

sayısal kodlardan oluşur.

Unikod (*Unicode*)

Unikod'da herbir karakter için 16 bit kullanılır; dolayısıyla 2^{16}'dan 65536 tane farklı kod olabilir. ASCII tabloda 128 tane birbirinden farklı kod vardı; genişletilmişte ise 256 tane. Unikod da bu rakam çok çok artmıştır. Unikod tablo içerisine birçok dile ait özel karakterler, çok çok özel simgeler koyulabilir. ASCII'de kod sayısı sınırlı olduğu için ancak gerekli karakterler ve simgeler koyulabilmiştir. Örneğin, Tablo-4.8 den görüleceği üzere normal ASCII'de Türkçe karakterler yoktur. Ancak Türkçe karakterlerin de oluştuğu ASCII tablo yapılabilir.

Türkçe Karakter Tablosu

Özel olarak Türkçe karakter tablosu oluşturulabilir. Örneğin Tablo-4.8 de görülen ve İngilizce için olan ASCII tabloda Türkçe'de kullanılmayan veya az kullanılan karakterler çıkarılıp yerine Ç, Ö, Ş gibi Türkçe'de olan harfler koyulabilir. Tablo-4.9 da olabilecek örnek Türkçe ASCII tablo verilmiştir. Bu tablo herhangi bir yer kullanılmayıp tamamen alfanumerik kodlamanın gösterilmesi için üretilmiştir. Tablodan görüleceği gibi Türkçe karakterler, TL simgesi ve papatya tabloya birer simge olarak eklenmiştir. Örneğin, "10 Milyon TL" cümlesinin Tablo-4.9 a göre kodlanması durumunda aşağıdaki sayısal kodlar bulunur:

45 44 76 104 107 120 110 109 35

Tablo-4.9. Örnek Türkçe ASCII tablo.

On	Onaltı	kr.	On	Onaltı	kr.	On	Onaltı	kr	On	Onaltı	kr.
0	*0*	NULL	32	*20*	boşluk	64	*40*	Ç	96	*60*	ç
1	*1*	SOH	33	*21*	!	65	*41*	D	97	*61*	d
2	*2*	STX	34	*22*	"	66	*42*	E	98	*62*	e
3	*3*	ETX	35	*23*	TL	67	*43*	F	99	*63*	f
4	*4*	EOT	36	*24*	%	68	*44*	G	100	*64*	g
5	*5*	ENQ	37	*25*	'	69	*45*	Ğ	101	*65*	ğ
6	*6*	ACK	38	*26*	(	70	*46*	H	102	*66*	h
7	*7*	BELL	39	*27*	)	71	*47*	I	103	*67*	ı
8	*8*	BS	40	*28*	+	72	*48*	İ	104	*68*	i
9	*9*	HT	41	*29*	,	73	*49*	J	105	*69*	j
10	*A*	LF	42	*2A*	-	74	*4A*	K	106	*6A*	k
11	*B*	VT	43	*2B*	.	75	*4B*	L	107	*6B*	l
12	*C*	FF	44	*2C*	0	76	*4C*	M	108	*6C*	m
13	*D*	CR	45	*2D*	1	77	*4D*	N	109	*6D*	n
14	*E*	SO	46	*2E*	2	78	*4E*	O	110	*6E*	o
15	*F*	SI	47	*2F*	3	79	*4F*	Ö	111	*6F*	ö
16	*10*	DLE	48	*30*	4	80	*50*	P	112	*70*	p
17	*11*	DC1	49	*31*	5	81	*51*	R	113	*71*	r
18	*12*	DC2	50	*32*	6	82	*52*	S	114	*72*	s
19	*13*	DC3	51	*33*	7	83	*53*	Ş	115	*73*	ş
20	*14*	DC4	52	*34*	8	84	*54*	T	116	*74*	t
21	*15*	NAK	53	*35*	9	85	*55*	U	117	*75*	u
22	*16*	SYN	54	*36*	:	86	*56*	Ü	118	*76*	ü
23	*17*	ETB	55	*37*	;	87	*57*	V	119	*77*	v
24	*18*	CAN	56	*38*	<	88	*58*	Y	120	*78*	y
25	*19*	EM	57	*39*	=	89	*59*	Z	121	*79*	z
26	*1A*	SUB	58	*3A*	>	90	*5A*	[	122	*7A*	papatya
27	*1B*	ESC	59	*3B*	?	91	*5B*	\	123	*7B*	{
28	*1C*	FS	60	*3C*	@	92	*5C*	]	124	*7C*	|
29	*1D*	GS	61	*3D*	A	93	*5D*	a	125	*7D*	}
30	*1E*	RS	62	*3E*	B	94	*5E*	b	126	*7E*	~
31	*1F*	US	63	*3F*	C	95	*5F*	c	127	*7F*	del

4.5. Özet

Kodlama, lojik devre tasarımının ve sayısal sistemlerinin önemli bir yanıdır. Herşeyden önce, üzerinde yapılacak bilgi veya veri belirli bir düzen altında sayısal

olarak kodlanmalıdır. Bir bilgi, farklı kodlama teknikleri kullanılarak değişik şekillerde kodlanabilir. BCD kodu lojik devre tasarımında önemli bir konuma sahiptir. Özellikle sayıcılarda ve insan-makina diyalogunun yoğun olarak yaşandığı bütün elektronik sistemlerde (hesap makinası gibi) BCD kodu kullanılmaktadır. Bunun yanı sıra, haberleşme/ veri iletimi (Üç fazlalık, *Aiken* kodları gibi) ve lojik devrelerin indirgenmesinde kullanılan yöntemlerde yararlanılan özel kodlar (Bitişik kodlar gibi) da yer almaktadır. Alfanumerik kodlar, yazı ve denklem gibi verilerin sayısallaştırılmasında kullanılır. En çok bilenen alfanumerik kod tablosu ASCII'dir; bir de Unikod olarak adlandırılan kodlama tablosu vardır. Karakterler, ASCII'de 7 bit, genişletilmiş ASCII'de 8 bit ve Unikod'da 16 bit uzunluğundadır.

4.6. Sorular

1. Örnek 4.2 de verilen soruda herbir isme sayısal kod verilerek birer kez saklandığında toplam 24 bit gerekir. Çünkü 8 isim vardır ve kod uzunluğu 3 bit'tir. Eğer herbir karaktere kod verilerek saklanmış olsaydı, Türkçe'de 29 harf olduğu varsayılarak herbir isim birer kez saklanırsa toplam kaç bitlik bellek gerekir?
2. Toplam 16 sözcükten oluşan bir dil olduğunu varsayarak bu dilde bulunan herbir sözcüğe bir kod atayınız ve örnek cümleler vererek kodlamanın nasıl yapıldığını gösteriniz. Dilde bulunan sözcükleri de kendinize göre belirleyiniz.
3. Sayısal Kodlama ile alfanumerik kodlama arasındaki farkı açıklayınız. 123 sayısı heriki kodlama yöntemiyle de kodlanabilir mi? Kodlanırsa aralarındaki fark nedir?
4. 1024 sayısın hem BCD hem de yalın ikili olarak nasıl kodlandığını gösteriniz. Hangisi daha çok bit gerektirir? Neden?
5. Üç fazlalık ve *Aiken* kodları ne tür uygulamalar için daha uygundur? Araştırınız.
6. Başlangıç sayısı 011 olan 3 bitlik *Gray* kodu geliştiriniz.
7. Hata sezme ile hata onarma arasındaki farkı açıklayınız. Eşlik, LRC, CRC ve *Hamming*; hangileri hata sezme ve hangisi hata onarma yöntemleridir.
8. LRC, boyuna fazlalık sınmasında hatalı bit'in nerede olduğu anlaşılabilir mi? Nasıl?
9. "Baba bana top al" cümlesi İngilizce ASCII tabloya göre, 66 97 98 97 98 97 110 97 116 111 112 97 108 olarak bulunur (bakınız Tablo-4.8.). Aynı cümle Türkçe ASCII tabloya göre nasıl kodlanır. Neden iki kodlama sonucu elde edilen kodlar farklı çıkmıştır?
10. Türkçe ASCII tablosuna göre (bkz. Tablo-4.9.) "papatya" ve "PAPATYA" sözcükleri nasıl kodlanır? Hangisi daha az kodla temsil edilmiştir? Neden?
11. BCD olarak kodlanmış olan 2 sayının çarpımını en az iki hanelik sayı üzerinde gösteriniz.
12. BCD kodlama sayısal uygulamalarda nerelerde kullanılır; BCD kodlamanın kullanıldığı bir uygulamayla karşılaştınız mı? Açıklayınız.

5.

Lojik Devre Temelleri

Lojik devreler ikili işaretler veya ikili kodlanmış veriler üzerinde çalışan ve temeli *Boole Cebrine* dayanan düzeneklerdir. Temel tasarım elemanı bildiğimiz modern mantıktaki *ve*, *veya* gibi işlemlere atanmış ve kapı olarak adlandırılan birimlerdir. Bu birimler ve türetilen birimler kullanılarak giriş ve çıkış fonksiyonları belirli ve ikili sayı sistemine dayanan hemen hemen her devrenin tasarımı yapılabilir. Lojik devrelerde biri lojik 0, diğeri lojik 1 olarak adlandırılan iki durum vardır. Bilgisayar dahil tüm sayısal sistemler bu iki lojik değerin farklı şekilde kombinasyonları yapılarak tasarlanırlar. Bu bölümde lojik devre temellerine ait temel lojik işlemler, Venn diyagramları, Boole Cebri aksiyom ve teoremleri, Boole Cebri'nin kanonik biçimleri ve lojik kapı elemanlarıyla devre gerçekleştirme, aşağıdaki başlıklar altında ele alınmıştır:

5.1. Lojik İşlemlerin Temeli
VE İşlemi (AND)
VEYA İşlemi (OR)
TÜMLEME İşlemi (NOT)

5.2. Türetilen İşlemler
TÜMLEYEN-VE (TVE) İşlemi (NAND)
TÜMLEYEN-VEYA (TVEYA) İşlemi (NOR)
YA DA İşlemi (XOR)
EŞDEĞERLİK (TYA DA) İşlemi (XNOR)

5.3. Temel İşlemlerin Kümeler Cebriyle Gösterilişi
Küme Kavramı
Kümeler Cebrinde Kesişim
Kümeler Cebrinde Birleşim
Kümeler Cebrinde Tümleme
Kümeler Cebrindeki Diğer Kavramlar

5.4. Boole Cebrinin Aksiyom ve Teoremleri
Boole Cebri Aksiyomları
Boole Cebri Teoremleri

5.5. Boole Cebri Fonksiyonları ve Standart Biçimleri
Minimum ve Maksimum Terimler
Minimum Terimler Kanonik Biçimi
Maksimum Terimler Kanonik Biçimi

5.6. Kanonik Biçimler Arasındaki Dönüşüm

5.7. Doğruluk Tablosundan Kanonik Biçimlerin Bulunması

5.8. Kanonik Biçimlerin Kapılarla Gerçeklenmesi

5.9. Özet

5.10. Sorular

5.1. Lojik İşlemlerin Temeli

Lojik işlemlerin temelini, çoğu kuralı geleneksel cebire benzeyen ama kendine has aksiyom ve teoremleri olan Boole Cebri oluşturur; 19 yüzyılda George Boole tarafından ikili sistemlerdeki bir takım kuralları matematiksel olarak ifade edilmiş ve buradan Boole Cebri'nin temelleri ortaya çıkmıştır. Boole Cebri, *De Morgan* ve *Huntington*'un yeni kurallar eklemesi ve *Shannon*'un Boole Cebri'ni kontaklı anahtar devrelerine uygulamasıyla günümüzdeki halini almıştır.

Boole Cebri işlemlerinin görsel olarak ifade edilmesi Venn diyagramları aracılığıyla yapılır. Boole Cebri içinde üç tane temel lojik işlemi tanımlanmıştır. Bu işlemler VE, VEYA ve TÜMLEME işlemleridir. Herşeyden önce temel mantık işlemleri tanımları, doğruluk tabloları, anahtar devreleri, lojik devre gösterilişleri, zamanlama diyagramları üzerinde durulması yararlı olacaktır.

5.1.1. VE İşlemi (AND)

VE işlemi, Boole Çarpımı, Kesişim, lojik VE ya da AND işlemi olarak da isimlendirilir. Birbirine VE işlemi ile bağlı iki önermeden oluşan bir birleşik önermenin doğru olması, her iki önermenin de doğru olmasına bağlıdır. Burada adı geçen önermeler, *a, b, c, d,* ... gibi değişkenlerle ifade edilirler ve sadece "0" veya "1" değerinden birini alabilirler. *a* ve *b* ile ifade edilen önermelerin doğruluğu "1", yanlışlığı ise "0" ile gösterilir. VE işlemi " · " ile ifade edilir ya da herhangi bir özel işaret kullanılmaz. Birleşik önerme ise;

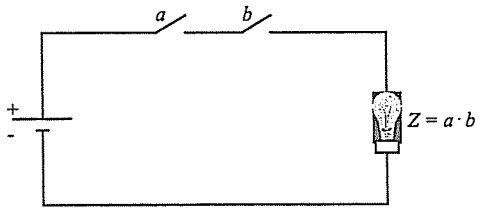
$Z = a \cdot b$ ya da $Z = ab$

ile gösterilir. VE işlemine ilişkin doğruluk tablosu Tablo-5.1 de verilmiştir.

Tablo-5.1. VE işleminin doğruluk tablosu.

a	b	$Z=a \cdot b$
0	0	0
0	1	0
1	0	0
1	1	1

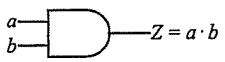
VE işleminin anahtar devreleriyle gösterilmesinde birbirine seri bağlanmış iki anahtar ele alınır. Anahtarların açık olması durumu "0", kapalı olması durumu "1", lambanın yanması durumu "1", lambanın sönük olması durumu "0" ile gösterilir. Bu yapıya ilişkin doğruluk diyagramı zaten Tablo-5.1 de verilmişti, anahtar devreleriyle gösterilim ise Şekil-5.1 de görülmektedir.



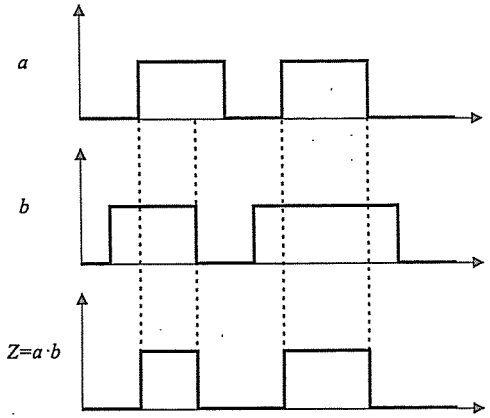
Şekil-5.1. VE işleminin anahtar devrelerindeki karşılığı.

Seri anahtar devresinde *a* ve *b* anahtarlarının her ikisinin de kapalı olması durumunda çıkış ifadesinin değeri doğru olmakta, devre kapanmakta ve lambanın yanması sağlanmaktadır.

VE işleminin ANSI/IEEE 1973 standardına göre lojik devre gösterilişi Şekil-5.2 de verilmiştir. Bu aşamadan sonra VE işlemi yerine VE kapısı tanımlaması kullanılacaktır. VE işlemi için yapılan bütün açıklamalar, VE kapısı için de geçerlidir. Rasgele seçilmiş *a* ve *b* girişlerine göre VE kapısının zamanlama diyagramı ise Şekil-5.3 de verilmiştir.



Şekil-5.2. ANSI/IEEE 1973 Standardına göre VE kapısı lojik gösterilimi.



Şekil-5.3. VE kapısının zamanlama diyagramı.

5.1.2. VEYA İşlemi (OR)

VEYA işlemi, Boole Toplamı, Birleşim, lojik VEYA ya da OR işlemi olarak da isimlendirilir. Birbirine VEYA işlemi ile bağlı iki önermeden oluşan bir birleşik önermenin doğru olması, birleşik önermeyi meydana getiren önermelerden en az birinin doğru olmasına bağlıdır. VEYA işlemi "+" ile ifade edilir. Birleşik önerme ise

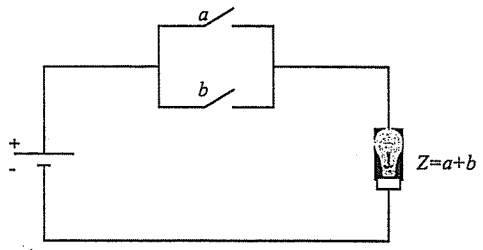
$$Z = a+b$$

ile gösterilir. VEYA işlemine ilişkin doğruluk tablosu Tablo-5.2 de verilmiştir.

Tablo-5.2. VEYA işleminin doğruluk tablosu.

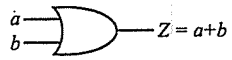
a	b	$Z=a+b$
0	0	0
0	1	1
1	0	1
1	1	1

VEYA işleminin anahtar devreleriyle gösterilmesinde birbirine paralel bağlanmış iki anahtar ele alınır. Anahtarların açık olması durumu "0", kapalı olması durumu "1", lambanın yanması durumu "1", lambanın sönük olması durumu "0" ile gösterilir. Bu yapıya ilişkin doğruluk diyagramı Tablo-5.2 de verilmişti, anahtar devreleriyle gösterilim ise Şekil-5.4 de görülmektedir.

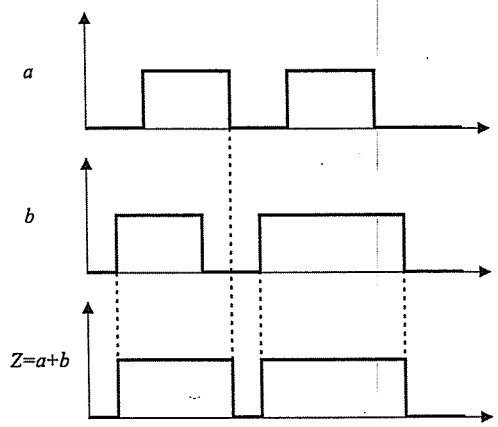


Şekil-5.4. VEYA işleminin anahtar devrelerindeki karşılığı.

Paralel anahtar devresinde a ve b anahtarlarından sadece birinin kapalı olması durumunda çıkış ifadesinin değeri doğru olmakta, devre kapanmakta ve lambanın yanması sağlanmaktadır. VEYA işleminin ANSI/IEEE 1973 standardına göre lojik devre gösterilişi olan VEYA kapısı Şekil-5.5 de, rasgele seçilmiş a ve b girişlerine göre zamanlama diyagramı ise Şekil-5.6 de verilmiştir.



Şekil-5.5. ANSI/IEEE 1973 Standardına göre VEYA kapısı lojik gösterilimi.



Şekil-5.6. VEYA kapısının zamanlama diyagramı.

Bu noktaya kadar anlatılan VE ile VEYA işlemleri birden çok değişkene uygulanan işlemlerdir. Tek bir değişkene uygulanabilen Lojik İşlemleri de bulunmaktadır. Bir sonraki alt bölümde tek bir değişkene uygulanabilen TÜMLEME işlemi tanıtılacaktır.

5.1.3. TÜMLEME İşlemi (NOT)

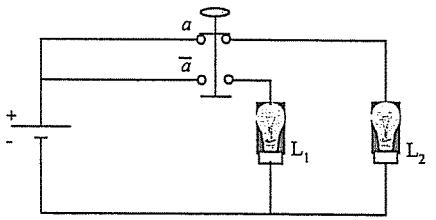
TÜMLEME işlemi, Boole komplementi, lojik tersleme olarak ya da NOT işlemi olarak da isimlendirilmektedir. TÜMLEME işlemi uygulanan önerme, başlangıçta doğru ise yanlış, yanlış ise doğru olacaktır. İkili gösterilimde, önermenin aldığı değer "1" ise tümleme işlemi uygulanmış önerme "0" değerini, önermenin aldığı değer "0" ise tümleme işlemi uygulanmış önerme "1" değerini alır. a önermesine uygulanan TÜMLEME işlemi a' ya da $\overline{a}$ ile ifade edilir. TÜMLEME işlemine ilişkin doğruluk tablosu Tablo-5.3 de verilmiştir.

Tablo-5.3. TÜMLEME işleminin doğruluk tablosu.

a	$Z=\overline{a}$
0	1
1	0

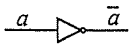
TÜMLEME işleminin anahtar devreleriyle gösterilmesinde birbirine mekanik olarak bağlı, başlangıç durumunda biri açık diğeri kapalı bulunan iki anahtar kullanılır.

Başlangıçta a önermesi doğru olduğu için L_2 yanar, $\overline{a}$ yanlış olduğundan L_1 ise sönüktür. Düğmeye basılınca tersine durum söz konusu olur. Her iki durumda da a ile $\overline{a}$ birbirinin tümleyeni olarak davranır. Bu yapıya ilişkin doğruluk diyagramı Tablo-5.3 de verilmişti, anahtar devreleri eşdeğeri ise Şekil-5.7 de görülebilir.

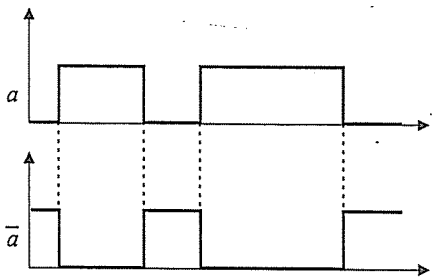


Şekil-5.7. TÜMLEME işleminin anahtar devrelerindeki karşılığı.

TÜMLEME işleminin ANSI/IEEE 1973 standardına göre lojik devre gösterilişi olan TÜMLEME kapısı Şekil-5.8 de, zamanlama diyagramı ise Şekil-5.9 da verilmiştir.



Şekil-5.8. ANSI/IEEE 1973 Standardına göre TÜMLEME kapısı lojik gösterilimi.



Şekil-5.9. TÜMLEME kapısının zamanlama diyagramı.

VE, VEYA ile TÜMLEME temel lojik kapılarıdır. Bu kapı elemanlarının yanı sıra, yine bu elemanlardan türetilmiş işlemler, dolayısıyla kapı elemanları bulunur. TÜMLEYEN-VE, TÜMLEYEN-VEYA, YA DA ve EŞDEĞERLİK olarak isimlendirilen bu kapı elemanları, lojik devre tasarımında temel lojik kapı elemanlarıyla birlikte yaygın şekilde kullanılırlar. Tasarımcıların temel kriterlerinden biri olan maliyet faktörünü en aza indirmek için tek tip kapı elemanlarıyla yapılan tasarımlarda özellikle türetilen işlemlerden yararlanılır. Türetilen işlemler Ayrıt 5.2 de ayrıntılı olarak açıklanmıştır; tıpkı temel işlemlerde olduğu gibi, doğruluk tabloları, lojik gösterilimleri ve zamanlama diyagramları verilmiştir.

5.2. Türetilen İşlemler

Lojik devre tasarım dünyasında, temel lojik işlemlerden yararlanarak yeni işlemler türetilmektedir. Bu bir anlamda lojik devre tasarımcısının ufkunu genişletmek ve esneklik sağlamaktır. Şu ana kadar sadece üç kapı elemanı kullanarak tasarım yapmak zorunda olan tasarımcıya türetilen kapı elemanları ile birlikte toplam yedi kapılık bir üretim alanı sunulmaktadır. İlerleyen bölümlerde lojik devre dünyasının bu kadarla sınırlı kalmayacağı, hatta özel tasarımlara bile imkan sunulacağı ayrıntılı olarak ifade edilmiştir. Ancak öncelikle burada, temel lojik işlemlerinden türetilen TÜMLEYEN-VE (TVE), TÜMLEYEN-VEYA (TVEYA), YA DA ile EŞDEĞERLİK lojik işlemleri ve dolayısıyla lojik kapıları tanıtılmıştır.

5.2.1. TÜMLEYEN-VE (TVE) İşlemi (NAND)

TÜMLEYEN-VE (NAND) işlemi, VE işlemine TÜMLEME işleminin uygulanması ile elde edilir. VE işleminin tersine, birbirine TÜMLEYEN-VE işlemi ile bağlı iki önermeden oluşan bir birleşik önermenin yanlış olması, her iki önermenin de doğru olmasına bağlıdır. Aksi taktirde birleşik önerme sonucu doğrudur. Burada adı geçen önermeler, *a, b, c, d, ...* gibi değişkenlerle ifade edilirler ve sadece "0" veya "1" değerinden birini alabilirler. *a* ile *b* şeklinde ifade edilen önermelerin doğruluğu "1", yanlışlığı ise "0" ile gösterilir. TÜMLEYEN-VE işlemi TVE olarak da isimlendirilir ve TVE birleşik önermesi,

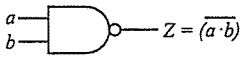
$$Z=(\overline{a\cdot b}) \quad \text{ya da} \quad Z=(\overline{ab})$$

ile gösterilir. TVE işlemine ilişkin doğruluk tablosu Tablo-5.4 de verilmiştir.

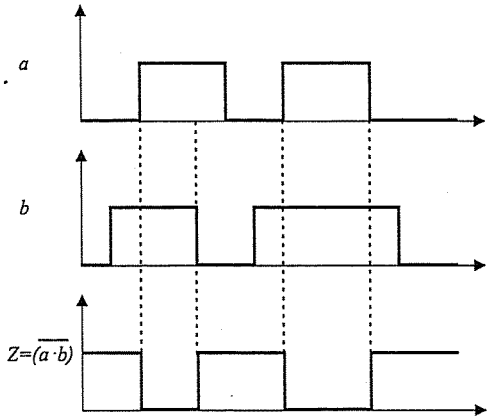
Tablo-5.4. TVE işleminin doğruluk tablosu.

a	b	$Z=(\overline{a\cdot b})$
0	0	1
0	1	1
1	0	1
1	1	0

TVE işleminin ANSI/IEEE 1973 standardına göre lojik devre gösterilişi olan TVE kapısı Şekil-5.10 da, rasgele seçilmiş *a* ve *b* girişlerine göre zamanlama diyagramı ise Şekil-5.11 de verilmiştir.



Şekil-5.10. ANSI/IEEE 1973 Standardına göre TVE kapısı lojik gösterilimi.



Şekil-5.11. TVE kapısının zamanlama diyagramı.

5.2.2. TÜMLEYEN-VEYA (TVEYA) İşlemi (NOR)

TÜMLEYEN-VEYA (NOR) işlemi, VEYA işlemine TÜMLEME işleminin uygulanması ile elde edilir. Birbirine TVEYA işlemi ile bağlı iki önermeden oluşan bir birleşik önermenin doğru olması, birleşik önermeyi meydana getiren önermelerden her ikisinin de yanlış olmasına bağlıdır. TVEYA işlemine ilişkin birleşik önerme,

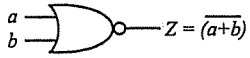
$$Z = \overline{(a+b)}$$

ile gösterilir. TVEYA işlemine ilişkin doğruluk tablosu Tablo-5.5 te verilmiştir:

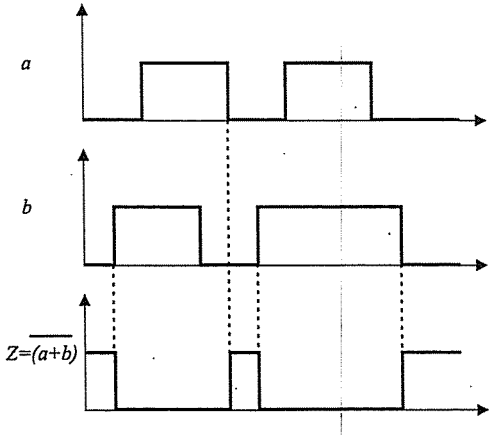
Tablo-5.5. TVEYA işleminin doğruluk tablosu.

a	b	$Z = \overline{(a+b)}$
0	0	1
0	1	0
1	0	0
1	1	0

TVEYA işleminin ANSI/IEEE 1973 standardına göre lojik devre gösterilimi olan TVEYA kapısı Şekil-5.12 de, rasgele seçilmiş a ve b girişlerine göre zamanlama diyagramı ise Şekil-5.13 de verilmiştir.



Şekil-5.12. ANSI/IEEE 1973 Standardına göre TVEYA kapısı lojik gösterilimi.



Şekil-5.13. TVEYA kapısının zamanlama diyagramı.

5.2.3. YA DA İşlemi (XOR)

YA DA (XOR) işlemi ile birbirine bağlı iki önermeden oluşan bir birleşik önermenin doğru olması, birleşik önermeyi meydana getiren önermelerden birinin doğru diğerinin ise yanlış olmasına bağlıdır. YA DA işlemi ⊕ simgesiyle ifade edilir ve bu işleme ilişkin birleşik önerme,

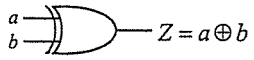
$$Z = a \oplus b = (\bar{a} \cdot b + a \cdot \bar{b}) \quad Z = a \oplus b$$

ile gösterilir. YA DA işlemine ilişkin doğruluk tablosu Tablo-5.6 da verilmiştir.

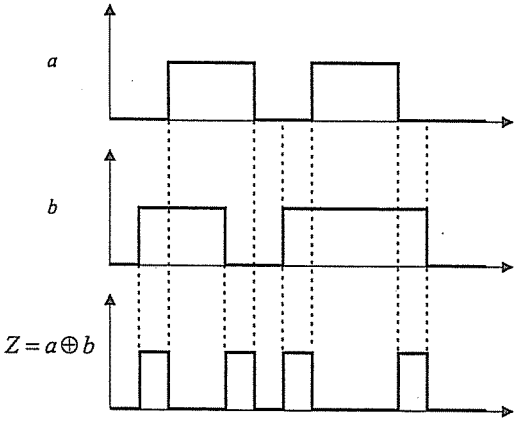
Tablo-5.6. YA DA işleminin doğruluk tablosu.

a	b	$Z = a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

YA DA işleminin ANSI/IEEE 1973 standardına göre lojik devre gösterilimi olan YA DA kapısı Şekil-5.14 de, rasgele seçilmiş a ve b girişlerine göre zamanlama diyagramı ise Şekil-5.15 da verilmiştir.



Şekil-5.14. ANSI/IEEE 1973 Standardına göre YA DA kapısı lojik gösterilimi.



Şekil-5.15. YA DA kapısının zamanlama diyagramı.

5.2.4. EŞDEĞERLİK (TYA DA) İşlemi (XNOR)

EŞDEĞERLİK (XNOR) işlemi ile birbirine bağlı iki önermeden oluşan bir birleşik önermenin doğru olması, birleşik önermeyi meydana getiren önermelerden her ikisinin de doğru veya her ikisinin de yanlış olmasına bağlıdır. EŞDEĞERLİK işlemine ilişkin birleşik önerme,

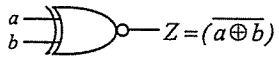
$$Z = \overline{(a \oplus b)} = (\bar{a} \cdot \bar{b} + a \cdot b)$$

ile gösterilir. Bu işlem YA DA'nın TÜMLEYEN'idir ve TYA DA şeklinde ifade edilebilir. EŞDEĞERLİK işlemine ilişkin doğruluk tablosu Tablo-5.7 de verilmiştir.

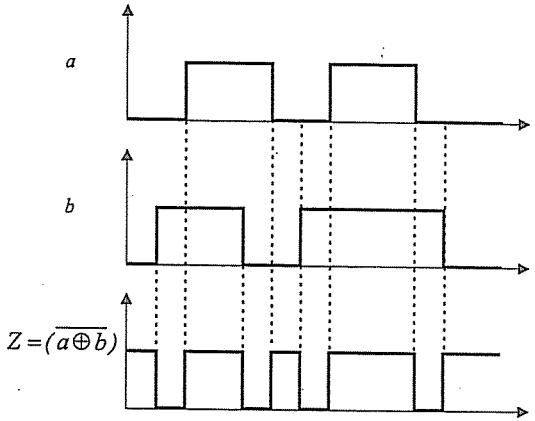
Tablo-5.7. EŞDEĞERLİK işleminin doğruluk tablosu.

a	b	$Z = \overline{(a \oplus b)}$
0	0	1
0	1	0
1	0	0
1	1	1

EŞDEĞERLİK işleminin ANSI/IEEE 1973 standardına göre lojik devre gösterilimi olan EŞDEĞERLİK kapısı Şekil-5.16 da, rasgele seçilmiş a ve b girişlerine göre zamanlama diyagramı ise Şekil-5.17 de verilmiştir.



Şekil-5.16. TYA DA kapısı gösterilimi.



Şekil-5.17. TYA DA kapısının zamanlama diyagramı.

Tanıtılan Lojik kapı elemanları toplu olarak Tablo-5.8 de verilmiştir. (Bu Tablo Kombinezonsal Devrelerin anlatıldığı Bölüm 7'de de ayrıca verilmiştir.)

5.3. Temel Lojik İşlemlerin Kümeler Cebri ile Gösterilişi

Lojik işlemlerle kümeler cebri ifadeleri çok benzerlik göstermektedir. Kümeler cebrinin grafik olarak gösterilimi olan Venn Diyagramları, lojik işlemlerin sağladığı ifadeleri görsel olarak ifade etmeye arayan bir araç olarak kullanılabilir. Dolayısıyla, kümeler cebri ve Venn diyagramlarından lojik işlemlerin doğruluğunu göstermek için yararlanılabilir. Burada, öncelikle kümeler cebrinin temel kavramları açıklanmış daha sonra Venn diyagramlarına ilişkin gösterilimler verilmiştir. Bu bilgiler ileride tanımlanmış olan Boole Cebri için iyi bir temel oluşturmaktadır.

5.3.1. Küme Kavramı

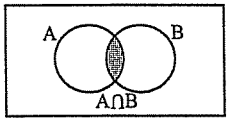
Küme, en az bir ortak özelliği bulunan elemanlar topluluğuna denir. A kümesinin bir elemanı olan x'in gösterilimi $x \in A$ şeklindedir ve x, A kümesinin elemanıdır şeklinde yorumlanır. Kümeler üzerine tanımlanan işlemler aynı zamanda Boole Cebri'nin işlemlerini de gerçeklerler. Bu anlamda kümeler üzerinde Kesişim, Birleşim ve Tümleme işlemlerinden bahsedilir. Bu işlemler Boole Cebri'ndeki VE, VEYA ve TÜMLEME işlemlerine birebir karşılık düşmektedir.

Tablo-5.8. Lojik kapı elemanları.

Lojik İşlem	Lojik İfadesi	ANSI/IEEE-1973 Std.	ANSI/IEEE-1984 Std.
VE (AND)	$Z = x_1 \cdot x_2 \cdot \ldots \cdot x_n$		
VEYA (OR)	$Z = x_1 + x_2 + \cdots + x_n$		
TVE (NAND)	$Z = \overline{(x_1 \cdot x_2 \cdot \ldots \cdot x_n)}$		
TVEYA (NOR)	$Z = \overline{(x_1 + x_2 + \cdots + x_n)}$		
YA DA (XOR)	$Z = x_1 \oplus x_2 \oplus \cdots \oplus x_n$		
EŞ DEĞERLİK (TYA DA) (XNOR)	$Z = \overline{(x_1 \oplus x_2 \oplus \cdots \oplus x_n)}$		

5.3.2. Kümeler Cebrinde Kesişim

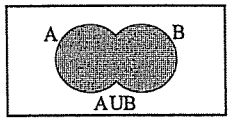
A ve B gibi iki küme gözönünde bulundurulduğunda, A ve B kümelerinin ortak elemanları A ve B kümelerinin kesişim kümesini oluşturur. Kesişim kümesi içinde yer alan elemanlar hem A hem de B kümesinin birer elemanı olmak zorundadırlar. Kesişme işlemi $\cap$ ile ifade edilir ve A ile B kümesinin kesişimi $A \cap B$ şeklinde gösterilir. Bu işlemin *Venn* diyagramlarındaki karşılığı Şekil-5.18 de verilmiştir.



Şekil-5.18. Kümelerin kesişmesi işleminin *Venn* diyagramı gösterilimi.

5.3.3. Kümeler Cebrinde Birleşim

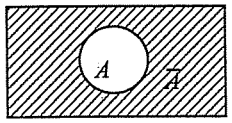
A ve B gibi iki küme gözönünde bulundurulduğunda, A ve B kümelerinin elemanlarından oluşan yeni küme, A ve B kümelerinin birleşim kümesini oluşturur. Birleşim kümesi içinde yer alan elemanlar A ve B kümesinin bütün elemanlarını içermelidir. Birleşme işlemi $\cup$ ile ifade edilir ve A ile B kümesinin birleşimi $A \cup B$ şeklinde gösterilir. Bu işlemin *Venn* diyagramlarındaki karşılığı Şekil-5.19 da verilmiştir.



Şekil-5.19. Kümelerin birleşim işleminin *Venn* diyagramı gösterilimi.

5.3.4. Kümeler Cebrinde Tümleme

Bir A kümesi gözönünde bulundurulduğunda, A kümesinin elemanı olmayan elemanlardan oluşan yeni küme, A kümesinin tümleyen kümesini oluşturur. Tümleyen küme içinde yer alan elemanlar, A kümesine ait olmayan elemanları içermelidir. A kümesinin Tümleyeni $\overline{A}$ ile gösterilir. Bu işlemin *Venn* diyagramlarındaki karşılığı Şekil-5.20 de verilmiştir.



Şekil-5.20. Bir kümenin tümlenmesi işleminin *Venn* diyagramı.

Lojik işlemlerin kümeler cebrindeki karşılıkları böylece tanımlanmış oldu. Kümeler cebrinde bazı temel kavramlar bir sonraki ayrıtta açıklanmıştır.

5.3.5. Kümeler Cebrindeki Diğer Kavramlar

Kümeler cebrinde, A kümesinin elemanlarıyla bu kümenin tümleyeninin oluşturduğu elemanların birleşiminden oluşan kümeye *Evrensel Küme* adı verilir ve "1" ile gösterilir. Bu durumda,

$$A+\overline{A}=1$$

ifadesi yazılabilir. Bir kümeye ait olan elemanlarla, o küme haricindeki elemanların birleşimi evrensel kümeyi tanımlar.

A kümesiyle, bu kümenin tümleyeninin kesişimi *Boş Küme* olarak adlandırılır ve "0" ile gösterilir. Bu durumda aşağıdaki ifade yazılabilir:

$$A\cdot\overline{A}=0$$

Evrensel küme bütün elemanları içinde barındırdığı için, bu kümenin herhangi bir kümeyle birleşimi yine evrensel kümedir. Evrensel kümenin bir A kümesiyle kesişiminde, bu A kümesinin elemanları ortak elemanları oluşturduğu için işlem sonucu yine A kümesidir. Bu işlemler aşağıdaki gibi gösterilebilir:

$$1+A=1$$
$$1\cdot A=A$$

Boş kümenin hiçbir elemanı bulunmamaktadır. Bu yüzden, herhangi bir A kümesiyle boş kümenin birleşimi boş küme, herhangi bir A kümesiyle boş kümenin birleşimi, yine bu A kümesidir. Bu işlemler sırasıyla,

$$0+A=A$$
$$0\cdot A=0$$

şeklinde ifade edilebilir.

5.4. Boole Cebrinin Aksiyom ve Teoremleri

Her disiplinde olduğu gibi lojik devre tasarımı da belirli kurallar çerçevesinde yapılır. Lojik devrelerde bu kurallar kümesinin dayanağı Boole Cebri'dir. Bütün cebirsel yapılarda olduğu gibi Boole Cebri'nde de, doğru olarak kabul edilen ve doğruluğu ispatlanabilen önermeler olmak üzere iki temel kurallar dizisi vardır. Doğru olarak kabul edilen önermelere *aksiyom*, doğruluğu ispat edilebilen önermelere ise *teorem* adı verilir.

5.4.1. Boole Cebri Aksiyomları

"0" ve "1" ikilisinden oluşan bir B kümesine " + " ve " · " işlemleri uygulanmış olsun. Bu durumda Boole Cebri'ne ilişkin aksiyomlar:

1. Her bir değişken, "0" ya da "1" değerinden sadece birini alabilir. Değişken eğer "0" değerini almıyorsa değeri "1", "1" değerini almıyorsa değeri "0" 'dır. Bir diğer değişle,

   $$a\neq 0\Rightarrow a=1$$
   $$a\neq 1\Rightarrow a=0$$

   yazılabilir. Bundan sonraki aksiyomlar Tablo-5.1, Tablo-5.2 ve Tablo-5.3 de verilen doğrulük tablolarından yararlanılarak oluşturulmuştur.

2. a) $1+1=1$'dir. Birbirine VEYA işlemi ile bağlı iki önermenin her biri doğru ise birleşik önerme de doğrudur.

   b) $0\cdot 0=0$'dir. Birbirine VE işlemi ile bağlı iki önermenin ikisi de yanlış ise birleşik önerme de yanlıştır.

3. a) $0+0=0$'dir. Birbirine VEYA işlemi ile bağlı iki önermenin ikisi de yanlış ise birleşik önerme de yanlıştır.

   b) $1\cdot 1=1$'dir. Birbirine VE işlemi ile bağlı iki önermenin her biri doğru ise birleşik önerme de doğrudur.

4. a) $1+0=1$'dir. Birbirine VEYA işlemi ile bağlı iki önermeden biri doğru diğeri yanlış ise birleşik önerme doğrudur.

   b) $0\cdot 1=0$'dir. Birbirine VE işlemi ile bağlı iki önermeden biri doğru diğeri yanlış ise birleşik önerme yanlıştır.

5.4.2. Boole Cebri Teoremleri

1. a) $a+b=b+a$ — *Değişme Özelliği*
   b) $a\cdot b=b\cdot a$
2. a) $a+b+c=(a+b)+c=a+(b+c)$ — *Birleşme Özelliği*
   b) $a\cdot b\cdot c=(a\cdot b)\cdot c=a\cdot(b\cdot c)$
3. a) $a+b\cdot c=(a+b)\cdot(a+c)$ — *Dağılma Özelliği*
   b) $a\cdot(b+c)=a\cdot b+a\cdot c$

4. a) $a+a=a$ *Değişkende Fazlalık Özelliği*
   b) $a\cdot a=a$

5. a) $a+a\cdot b=a$ *Yutma Özelliği*
   b) $a\cdot(a+b)=a$

6. a) $\overline{(a)}=\overline{a}$ *İşlemde Fazlalık Özelliği*
   b) $\overline{(\overline{a})}=a$

7. a) $\overline{(a+b+c+\ldots)}=\overline{a}\cdot\overline{b}\cdot\overline{c}\cdot\ldots$ ***De Morgan Kuralı***
   b) $\overline{(a\cdot b\cdot c\cdot\ldots)}=\overline{a}+\overline{b}+\overline{c}+\ldots$

8. a) $a+\overline{a}=1$ *Sabit Özelliği*
   b) $a\cdot\overline{a}=0$

9. a) $0+a=a$ *Etkisizlik Özelliği*
   b) $1\cdot a=a$

10. a) $1+a=1$ *Yutan Sabit Özelliği*
    b) $0\cdot a=0$

11. a) $(a+\overline{b})\cdot b=a\cdot b$
    b) $a\cdot\overline{b}+b=a+b$

12. a) $(a+b)\cdot(\overline{a}+c)\cdot(b+c)=(a+b)\cdot(\overline{a}+c)$
    b) $a\cdot b+\overline{a}\cdot c+b\cdot c=a\cdot b+\overline{a}\cdot c$

13. a) $(a+b)\cdot(\overline{a}+c)=a\cdot c+\overline{a}\cdot b$
    b) $a\cdot b+\overline{a}\cdot c=(a+c)\cdot(\overline{a}+b)$

14. a) $f(a,b,c,d,\ldots)=[a+f(0,b,c,d,\ldots)]\cdot[\overline{a}+f(1,b,c,d,\ldots)]$
    b) $f(a,b,c,d,\ldots)=a\cdot f(1,b,c,d,\ldots)+\overline{a}\cdot f(0,b,c,d,\ldots)$ ***Shannon Teoremi***

Örnek-5.1.

Boole Cebrinde tanımlanmış olan teoremlerin, Boole cebrine ilişkin aksiyom ve özellikler kullanılarak ispatlanması mümkündür.

Bu örnekte, dağılma özelliği 3a)'nın doğruluk tablosu ile gerçeklenen ispatı verilmiştir. Bu amaçla a,b,c giriş değerlerine karşılık, $a+b\cdot c=(a+b)\cdot(a+c)$ ifadesinin sol tarafına ilişkin ifadeler yazılır ve her iki tarafın da birbirine eşit olduğu ifade edilir. Bu durumda Tablo-5.9 da oluşturulan doğruluk tablosundan yararlanılır.

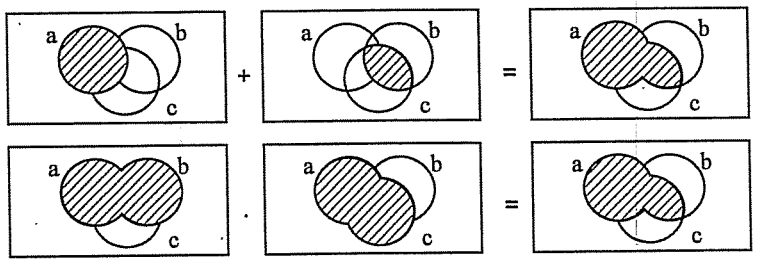
Tablo-5.9. a + b · c = (a + b) · (a + c) işlemine ilişkin doğruluk tablosu.

a	b	c	$b\cdot c$	$(a+b)$	$(a+c)$	$a+b\cdot c$	$(a+b)\cdot(a+c)$
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Bu tablonun sağ tarafında verilen iki ifadedeki lojik değerler karşılaştırılır. Bu durumda, $a+b\cdot c$ ifadesi ile $(a+b)\cdot(a+c)$ ifadesinin birbirine eşdeğer olduğu görülür.

Örnek-5.2.

Bu örnekte dağılma özelliği 3a)'nın *Venn* diyagramlarıyla ispatı yapılacaktır. Bu amaçla a,b,c kümelerinden türetilen $a+b\cdot c$ ve $(a+b)\cdot(a+c)$ ifadelerinin *Venn* Diyagramlarındaki karşılıkları verilecektir. Bu yapılar Şekil-5.21 de gösterilmiştir.



Şekil-5.21. $a+b\cdot c$ ve $(a+b)\cdot(a+c)$ eşdeğerliğinin *Venn* diyagramlarıyla ifadesi.

Örnek-5.3.

Bu örnekte Yutma özelliği 5a)'nın $a+a\cdot b=a$ şeklinde verilen ifadesinin doğruluğu Boole Cebri Aksiyom ve özellikleri kullanılarak ispatlanmıştır. Bu durumda,

$$a+a\cdot b=a\cdot(1+b)=a\cdot 1=a$$

yazılabilir. Eğer lojik devrelerle bu işlem gerçeklenirse, $a+a\cdot b$ ifadesini gerçekleştirmek için bir adet VEYA kapısına, bir adet de VE kapısına ihtiyaç olur. Oysa

sadeleştirme neticesinde geriye sadece a değişkeni kaldığından herhangi bir lojik kapı elemanına ihtiyaç olmaksızın ifade gerçeklenir. Buradan da görülebileceği üzere lojik ifadelerin indirgenmesi maliyet faktörü ve devrenin karmaşıklığı konusunda son derece etkili olmaktadır. Lojik ifadelerin indirgenmesi konusu Bölüm 6'da, devrelerin karmaşıklık hesabı konusu ise Bölüm 8'de ayrıntılı bir şekilde ele alınmıştır.

Örnek-5.4.

Bu örnekte 7) ile verilen *De Morgan* Teoremlerinin doğruluğu Boole Cebri Aksiyom ve özellikleri kullanılarak ispatlanmıştır. Önce 7a) ile verilen $\overline{(a+b+c+\ldots)}=\bar{a}\cdot\bar{b}\cdot\bar{c}\cdot\ldots$ eşitliği ele alınır. Bu ifade öncelikle sadece iki değişken için ispatlanır $[\overline{(a+b)}=\bar{a}\cdot\bar{b}]$ ve sonra genelleştirilir. 8) teoremi ile verilen $a+\bar{a}=1$ ve $a\cdot\bar{a}=0$ göz önünde bulundurulur ve $b\triangleq\bar{a}$ olarak kabul edilirse, 8) teoremindeki ifadeler $a+b=1$ ve $a\cdot b=0$ şekline gelir. Bu durumda, $(a+b)+\overline{(a+b)}=1$ ve $(a+b)\cdot\overline{(a+b)}=0$ olduğunu göstermek gerekir; birinci ifadenin sağlandığını göstermek için;

$$\begin{aligned}(a+b)+\overline{(a+b)}&=(a+b)+\bar{a}\cdot\bar{b}=a+(b+\bar{a}\cdot\bar{b})=a+(b+\bar{a})\cdot(b+\bar{b})\\&=a+b+\bar{a}=a+\bar{a}+b=1+b\\&=1\end{aligned}$$

yazılır. İkinci ifadenin doğruluğu için:

$$\begin{aligned}(a+b)\cdot\overline{(a+b)}&=(a+b)\cdot\bar{a}\cdot\bar{b}=a\cdot\bar{a}\cdot\bar{b}+b\cdot\bar{a}\cdot\bar{b}\\&=0\cdot\bar{b}+0\cdot\bar{a}\\&=0\end{aligned}$$

yazılır. Sonuçta, $(a+b)+\overline{(a+b)}=1$ ve $(a+b)\cdot\overline{(a+b)}=0$ dolayısıyla $(a+b)+\bar{a}\cdot\bar{b}=1$ ve $(a+b).\bar{a}\cdot\bar{b}=0$ sağlandığından $\overline{(a+b)}=\bar{a}\cdot\bar{b}$ dir. Bu ifade çok değişken olması durumunda,

$$\overline{(a+b+c+\ldots)}=\bar{a}\cdot\bar{b}\cdot\bar{c}\cdot\ldots$$

olduğu söylenebilir.

İkinci olarak 7b) ile verilen $\overline{(a\cdot b\cdot c\cdot\ldots)}=\bar{a}+\bar{b}+\bar{c}+\ldots$ eşitliğini ele alınır. Bu ifade önce, sadece iki değişken için ispatlanır $[\overline{(a\cdot b)}=\bar{a}+\bar{b}]$ ve sonra genelleştirilir. 8) teoremi ile verilen $a+\bar{a}=1$ ve $a\cdot\bar{a}=0$ göz önünde bulundurulur ve $b\triangleq\bar{a}$ olarak

kabul edilirse, 8) teoremindeki ifadeler $a+b=1$ ve $a\cdot b=0$ şekline gelir. Bu durumda, $(a\cdot b)+\overline{(a\cdot b)}=1$ ve $(a\cdot b)\cdot\overline{(a\cdot b)}=0$ olduğunu göstermek gerekir; birinci ifadenin sağlandığını göstermek için;

$$\begin{aligned}(a\cdot b)+\overline{(a\cdot b)}&=(a\cdot b)+(\bar{a}+\bar{b})=(a\cdot b+\bar{a})+\bar{b}=(a+\bar{a})\cdot(b+\bar{a})+\bar{b}\\&=b+\bar{a}+\bar{b}=1+\bar{a}\\&=1\end{aligned}$$

Şimdi ise ikinci ifadenin doğruluğunu irdelemek gerekir;

$$\begin{aligned}(a\cdot b)\cdot\overline{(a\cdot b)}&=(a\cdot b)\cdot(\bar{a}+\bar{b})=a\cdot b\cdot\bar{a}+a\cdot b\cdot\bar{b}\\&=0\cdot a+a\cdot 0\\&=0\end{aligned}$$

Sonuçta, $(a\cdot b)+\overline{(a\cdot b)}=1$ ve $(a\cdot b)\cdot\overline{(a\cdot b)}=0$ dolayısıyla $(a\cdot b)+(\bar{a}+\bar{b})=1$ ve $(a\cdot b)\cdot(\bar{a}+\bar{b})=0$ sağlandığından $\overline{(a\cdot b)}=\bar{a}+\bar{b}$ dir. Bu ifade çok değişken olması durumunda aşağıdaki gibi yazılır:

$$\overline{(a\cdot b\cdot c\cdot\ldots)}=\bar{a}+\bar{b}+\bar{c}+\ldots$$

Örnek-5.5.

Bu örnekte 12a) ile verilen $(a+b)\cdot(\bar{a}+c)\cdot(b+c)=(a+b)\cdot(\bar{a}+c)$ teoreminin doğruluğu Boole Cebri Aksiyom ve özellikleri kullanılarak ispatlanmıştır. Sol tarafa ilişkin ifadedeki ilk iki parantez çarpılıp düzenlenir, c ortak parantezine alındıktan sonra (b+c) ile çarpılır ve terimler birleştirilerek yazılırsa,

$$\begin{aligned}(a+b)\cdot(\bar{a}+c)\cdot(b+c)&=(a\cdot\bar{a}+a\cdot c+\bar{a}\cdot b+b\cdot c)\cdot(b+c)\\&=(a\cdot c+\bar{a}\cdot b+b\cdot c)\cdot(b+c)\\&=[c\cdot(a+b)+\bar{a}\cdot b]\cdot(b+c)\\&=[c\cdot(a+b)\cdot(b+c)+\bar{a}\cdot b\cdot(b+c)]\\&=[c\cdot(b+c)\cdot(a+b)+\bar{a}\cdot b\cdot(b+c)]\\&=[c\cdot(a+b)+\bar{a}\cdot b+\bar{a}\cdot b\cdot c]\\&=[c\cdot(a+b)+\bar{a}\cdot b\cdot(1+c)]\\&=[c\cdot(a+b)+\bar{a}\cdot b]\\&=a\cdot c+b\cdot c+\bar{a}\cdot b\\&=(a+b)\cdot(\bar{a}+c)\end{aligned}$$

sonucuna ulaşılır.

Örnek-5.6.

Bu örnekte 12b) ile verilen $a\cdot b+\overline{a}\cdot c+b\cdot c=a\cdot b+\overline{a}\cdot c$ teoreminin doğruluğu Boole Cebri Aksiyom ve özellikleri kullanılarak ispatlanır. Sol tarafa ilişkin ifadedeki $b\cdot c$ terimi $(a+\overline{a})$ ile çarpılırsa ifadenin değeri değişmez ancak, bu işlem kolaylığı getirir. Bu durumda,

$$\begin{aligned} a\cdot b+\overline{a}\cdot c+b\cdot c &= a\cdot b+\overline{a}\cdot c+b\cdot c\cdot(a+\overline{a}) \\ &= a\cdot b+\overline{a}\cdot c+b\cdot c\cdot a+b\cdot c\cdot\overline{a} \end{aligned}$$

elde edilir. Sağ taraftaki ilk terim ile üçüncü terim $a\cdot b$, ikinci terim ile dördüncü terim $\overline{a}\cdot c$ ortak parantezine alınırsa,

$$\begin{aligned} a\cdot b+\overline{a}\cdot c+b\cdot c &= a\cdot b+\overline{a}\cdot c+b\cdot c\cdot a+b\cdot c\cdot\overline{a} \\ &= a\cdot b\cdot(1+c)+\overline{a}\cdot c\cdot(1+b) \\ &= a\cdot b+\overline{a}\cdot c \end{aligned}$$

elde edilir ve teoremin doğruluğu ispatlanmış olunur.

Örnek-5.7:

Bu örnekte 14) ile verilen Shannon Teoremlerinin doğruluğu Boole Cebri Aksiyom ve özellikleri kullanılarak ispatlanmıştır. Shannon Teoremleri, Minimum Terimler ve Maksimum Terimler Kanonik Biçimlerine temel oluşturduklarından oldukça önemli teoremlerdir. Bu yüzden, ispatları da büyük önem taşır. Bu teorem, $a=1$, $\overline{a}=0$ ve $a=0$, $\overline{a}=1$ verildiğinde eşitliğin sağ ve sol taraflarından aynı fonksiyonun elde edildiğini göstermek suretiyle ispatlanır. Öncelikle, 14a) ile verilen

$$f(a,b,c,d,\ldots)=[a+f(0,b,c,d,\ldots)]\cdot[\overline{a}+f(1,b,c,d,\ldots)]$$

şeklindeki ilk teoremi göz önünde bulundurulsun.

$a=1$, $\overline{a}=0$ için,

$$\begin{aligned} f(1,b,c,d,\ldots) &= [1+f(0,b,c,d,\ldots)]\cdot[0+f(1,b,c,d,\cdots)] \\ &= f(1,b,c,d,\cdots) \end{aligned}$$

ve $a=0$, $\overline{a}=1$ için,

$$\begin{aligned} f(0,b,c,d,\ldots) &= [0+f(0,b,c,d,\ldots)]\cdot[1+f(1,b,c,d,\cdots)] \\ &= f(0,b,c,d,\cdots) \end{aligned}$$

elde edilir. Her iki durumda da fonksiyonun sağ ve sol tarafları birbirine eşit çıktığından ifade doğrudur.

14b) ile verilen $f(a,b,c,d,\ldots)=a\cdot f(1,b,c,d,\ldots)+\overline{a}\cdot f(0,b,c,d,\ldots)$ şeklindeki ikinci teorem göz önünde bulundurulursa, $a=1$, $\overline{a}=0$ için,

$$\begin{aligned} f(1,b,c,d,\ldots) &= 1\cdot f(1,b,c,d,\ldots)+0\cdot f(1,b,c,d,\cdots)] \\ &= f(1,b,c,d,\cdots) \end{aligned}$$

ve $a=0$, $\overline{a}=1$ için,

$$\begin{aligned} f(0,b,c,d,\ldots) &= 0\cdot f(1,b,c,d,\ldots)+1\cdot f(0,b,c,d,\cdots) \\ &= f(0,b,c,d,\cdots) \end{aligned}$$

elde edilir. Her iki durumda da fonksiyonun sağ ve sol tarafları birbirine eşit çıktığı için ifade doğrudur.

5.5. Boole Cebri Fonksiyonları ve Standart Biçimler

$x_1,x_2,\cdots,x_n$ değişkenlerine ve sabitlere, Boole Cebrinin VE, VEYA ile TÜMLEME işlemleri uygulanarak elde edilen *n-değişkenli* bir f fonksiyonuna *Boole Fonksiyonu* denilir.

Boole Cebri işlemlerinin iki temel biçimi bulunmaktadır. Bu biçimler, Minimum Terimler Kanonik Biçimi ve Maksimum Terimler Kanonik Biçimi olarak isimlendirilirler. Bu biçimler doğruluk tablosundan doğrudan doğruya elde edilebilen ifadelerdir. Bu biçimler, lojik ifadelerin en sade şekilleri veya başka bir değişle indirgenmiş ifadeleri değildirler. Bu iki biçime ilişkin tanımları vermeden önce, *minimum terim* (*minterm*) ve *maksimum terim* (*maksterm*) kavramlarını açıklamak yararlı olur.

Minimum ve Maksimum Terimler

Oluşturulan doğruluk tablolarından da görülebileceği üzere, x ve y şeklindeki iki değişkenden VE işlemi ile birbirine bağlı $\overline{x}\overline{y}$, $\overline{x}y$, $x\overline{y}$ ve xy şeklinde dört değişik kombinasyon tanımlanabilir. Bu kombinasyonların her birine *minimum terim* (*minterm*) adı verilir. *Venn* diyagramıyla ifade edilirse bu ifadelerin her birinin farklı bir alanı tanımladığı görülür. n değişken 2^n adet minimum terim ile ifade edilebilir ve minimum terimler m_i ile gösterilir. Benzer şekilde, x ve y şeklindeki iki değişkenden VEYA işlemi ile birbirine bağlı $x+y$, $x+\overline{y}$, $\overline{x}+y$ ve $\overline{x}+\overline{y}$ şeklinde dört değişik kombinasyon tanımlanabilir. Bu kombinasyonların her birine *maksimum terim* (*maksterm*) adı verilir. *Venn* diyagramıyla ifade edilirse bu ifadelerin her

birinin farklı bir alanı tanımladığı görülür. n değişken 2^n adet maksimum terim ile ifade edilebilir ve minimum terimler M_i ile gösterilir. Tablo-5.10 da iki değişken için *minimum terimler* ve *maksimum terimler* gösterilmiştir.

Tablo-5.10. İki değişken için minimum ve maksimum terimler.

x	y	*Minterm*	*m'ye indis*	*Maksterm*	*M'ye İndis*
0	0	$\bar{x}\cdot\bar{y}$	m_0	$x+y$	M_0
0	1	$\bar{x}\cdot y$	m_1	$x+\bar{y}$	M_1
1	0	$x\cdot\bar{y}$	m_2	$\bar{x}+y$	M_2
1	1	$x\cdot y$	m_3	$\bar{x}+\bar{y}$	M_3

Bir lojik işlem, minimum terimlerin toplamı veya maksimum terimlerin çarpımıyla ifade edilebilir. Minimum terimlerin toplamından oluşmuş ifadeye *Minimum Terimler Kanonik Biçimi*, maksimum terimlerin çarpımından oluşan ifadeye ise *Maksimum Terimler Kanonik Biçimi* adı verilir. Ayrıt 5.5.2 de bu biçimler ayrıntılı olarak açıklanacaktır.

5.5.1. Minimum Terimler Kanonik Biçimi

Minimum terimlerin toplamından oluşmuş ya da bir diğer değişle mintermlerin toplamından oluşan ifadeye *Minimum Terimler Kanonik Biçimi* adı verilir. Minimum Terimler çarpımlardan oluştuğu için Minimum Terimler Kanonik Biçime, *Çarpımlar Toplamı Kanonik Biçimi* adı da verilir. Minimum terimler m_i şeklinde gösterilebildiklerine göre, bir Boole fonksiyonuna ilişkin Minimum Terimler Kanonik Biçimi $\sum m_i$ ile gösterilir. Bir diğer gösterilim ise m_i 'deki terim numarasıdır. Bu durum da Boole fonksiyonu $\sum i$ şeklinde ifade edilir. Kanonik kelimesi, lojik fonksiyonu oluşturan terimlerin ya kendilerinin ya da tümleyenlerinin çarpım terimlerin içinde mutlaka bulunması anlamına gelmektedir.

Örnek-5.8.

Bu örnekte, VEYA işlemine ilişkin minimum terimler kanonik biçimi elde edilecektir. Bu amaçla Tablo-5.11 de işleme ilişkin doğruluk tablosu verilmiş ve minimum terimler gösterilmiştir.

Tablo-5.11. VEYA işlemine ilişkin doğruluk tablosu ve minimum terimler.

a	b	$F=a+b$	*Minterm*	*m'ye İndis*
0	0	0	$\bar{a}\cdot\bar{b}$	m_0
0	1	1	$\bar{a}\cdot b$	m_1
1	0	1	$a\cdot\bar{b}$	m_2
1	1	1	$a\cdot b$	m_3

Minimum terimler kanonik biçimine ilişkin ifade yazılırken, VEYA işleminin lojik ifadesinde "1" olan minimum terimler değerlendirilir ve bu terimlerin toplamı minimum terimler kanonik biçimini oluşturur. Buna göre VEYA işleminin minimum terimler kanonik biçimi için;

$$F=a+b=\bar{a}\cdot b+a\cdot\bar{b}+a\cdot b=m_1+m_2+m_3=\sum(1,2,3)$$

yazılır.

Aslında bir fonksiyonun, minimum terimler kanonik biçimini oluşturmak, o fonksiyona ilişkin 14-b)'de verilen *Shannon* teoremini yazmak ile eşdeğerdir. Bu durumda, n değişkenli ifade için,

$$\begin{aligned}f(x_1,x_2,\cdots,x_n)&=\bar{x}_1\cdot f(0,x_2,\cdots,x_n)+x_1\cdot f(1,x_2,\cdots,x_n)\\&=\bar{x}_1\cdot[\bar{x}_2\cdot f(0,0,x_3,\cdots,x_n)+x_2\cdot f(0,1,x_3,\cdots,x_n)]\\&\quad+x_1\cdot[\bar{x}_2\cdot f(1,0,x_3,\cdots,x_n)+x_2\cdot f(1,1,x_3,\cdots,x_n)]\end{aligned}$$

$$\begin{aligned}f(x_1,x_2,\cdots,x_n)&=\bar{x}_1\cdot\bar{x}_2\cdot f(0,0,x_3,\cdots,x_n)+\bar{x}_1\cdot x_2\cdot f(0,1,x_3,\cdots,x_n)\\&\quad+x_1\cdot\bar{x}_2\cdot f(1,0,x_3,\cdots,x_n)+x_1\cdot x_2\cdot f(1,1,x_3,\cdots,x_n)]\\&\quad\vdots\\&=\bar{x}_1\cdot\bar{x}_2\cdot\cdots\cdot\bar{x}_n\cdot f(0,0,0,\cdots,0)+\bar{x}_1\cdot\bar{x}_2\cdot\cdots x_n\cdot f(0,0,0,\cdots,1)\\&\quad+\cdots+x_1\cdot x_2\cdot\cdots x_n\cdot f(1,1,1,\cdots,1)\end{aligned}$$

yazılır. Bu ifadede yer alan değişkenler, lojik ifadeler olarak değerlendirilirken, f fonksiyon değeri lojik 0 veya lojik 1 değerini alarak, hangi terimlerin Minimum Terimler Kanonik Biçimine dahil olacağını belirler. Örneğin, a, b ve c gibi üç değişken için,

$$\begin{aligned}f(a,b,c)&=\bar{a}\cdot\bar{b}\cdot\bar{c}\cdot f(0,0,0)+\bar{a}\cdot\bar{b}\cdot c\cdot f(0,0,1)+\bar{a}\cdot b\cdot\bar{c}\cdot f(0,1,0)+\bar{a}\cdot b\cdot c\cdot f(0,1,1)\\&\quad+a\cdot\bar{b}\cdot\bar{c}\cdot f(1,0,0)+a\cdot\bar{b}\cdot c\cdot f(1,0,1)+a\cdot b\cdot\bar{c}\cdot f(1,1,0)+a\cdot b\cdot c\cdot f(1,1,1)\end{aligned}$$

ifadesi bulunur.

Örnek-5.9.

Bu örnekte, $f(a,b,c)=a\cdot b+\bar{a}\cdot c$ fonksiyonunun minimum terimler kanonik biçimi elde edilmiştir. Bu amaçla,

$$f(a,b,c)=\bar{a}\cdot\bar{b}\cdot\bar{c}\cdot f(0,0,0)+\bar{a}\cdot\bar{b}\cdot c\cdot f(0,0,1)+\bar{a}\cdot b\cdot\bar{c}\cdot f(0,1,0)+\bar{a}\cdot b\cdot c\cdot f(0,1,1)$$
$$+a\cdot\bar{b}\cdot\bar{c}\cdot f(1,0,0)+a\cdot\bar{b}\cdot c\cdot f(1,0,1)+a\cdot b\cdot\bar{c}\cdot f(1,1,0)+a\cdot b\cdot c\cdot f(1,1,1)$$

fonksiyonu göz önüne alınır. Bu fonksiyondaki her bir f değeri aşağıdaki gibi belirlenir.

$$\begin{aligned}
f(0,0,0) &= 0\cdot 0+1\cdot 0 = 0+0 = 0\\
f(0,0,1) &= 0\cdot 0+1\cdot 1 = 0+1 = 1\\
f(0,1,0) &= 0\cdot 1+1\cdot 0 = 0+0 = 0\\
f(0,1,1) &= 0\cdot 1+1\cdot 1 = 0+1 = 1\\
f(1,0,0) &= 1\cdot 0+0\cdot 0 = 0+0 = 0\\
f(1,0,1) &= 1\cdot 0+0\cdot 1 = 0+0 = 0\\
f(1,1,0) &= 1\cdot 1+0\cdot 0 = 1+0 = 1\\
f(1,1,1) &= 1\cdot 1+0\cdot 1 = 1+0 = 1
\end{aligned}$$

f fonksiyonlarından "1" değerini alan terimlerin toplamı, minimum terimler kanonik biçimini oluşturur. Bu durumda,

$$f(a,b,c)=\bar{a}\cdot\bar{b}\cdot c+\bar{a}\cdot b\cdot c+a\cdot b\cdot\bar{c}+a\cdot b\cdot c=m_1+m_3+m_6+m_7=\sum(1,3,6,7)$$

elde edilir.

5.5.2. Maksimum Terimler Kanonik Biçimi

Maksimum terimlerin çarpımından oluşmuş ya da bir diğer değişle makstermlerin çarpımından oluşan ifadeye *Maksimum Terimler Kanonik Biçimi* adı verilir. Maksimum Terimler toplamlardan oluştuğu için Maksimum Terimler Kanonik Biçime, *Toplamlar Çarpımı Kanonik Biçimi* adı da verilir. Maksimum terimler M_i şeklinde gösterilebildiklerine göre, bir Boole fonksiyonuna ilişkin Maksimum Terimler Kanonik Biçimi $\prod M_i$ ile gösterilir. Bir diğer gösterilim ise M_i ifadesindeki terim numarasıdır. Bu durum da Boole fonksiyonu $\prod i$ şeklinde ifade edilir.

Örnek-5.10.

Bu örnekte, VE işlemine ilişkin maksimum terimler kanonik biçimi elde edilecektir. Bu amaçla Tablo-5.12 de işleme ilişkin doğruluk tablosu verilmiş ve maksimum terimler gösterilmiştir.

Tablo-5.12. VE işlemine ilişkin doğruluk tablosu ve maksimum terimler.

a	b	$F=a\cdot b$	*Maksterm*	*M'ye İndis*
0	0	0	$a+b$	M_0
0	1	0	$a+\bar{b}$	M_1
1	0	0	$\bar{a}+b$	M_2
1	1	1	$\bar{a}+\bar{b}$	M_3

Maksimum terimler kanonik biçimine ilişkin ifade yazılırken, VE işleminin lojik ifadesinde 0 olan minimum terimler değerlendirilir ve bu terimlerin çarpımı maksimum terimler kanonik biçimini oluşturur. Buna göre VE işleminin maksimum terimler kanonik biçimi için

$$F=a\cdot b=(a+b)\cdot(a+\bar{b})\cdot(\bar{a}+b)=M_0\cdot M_1\cdot M_2=\prod(0,1,2)$$

yazılır.

Aslında bir fonksiyonun, maksimum terimler kanonik biçimini oluşturmak, o fonksiyona ilişkin 14-a)'da verilen Shannon teoremini yazmak ile eşdeğerdir. Bu durumda, n değişkenli ifade için,

$$\begin{aligned}
f(x_1,x_2,\cdots,x_n) &= [x_1+f(0,x_2,\cdots,x_n)]\cdot[\bar{\bar{x}}_1+f(1,x_2,\cdots,x_n)]\\
&= [x_1+[\bar{\bar{x}}_2+f(1,1,\cdots,x_n)]\cdot[x_2+f(0,0,\cdots,x_n)]]\\
&\quad [\bar{x}_1+[\bar{x}_2+f(1,1,\cdots,x_n)]\cdot[x_2+f(0,0,\cdots,x_n)]]\\
&\quad \vdots\\
&= [x_1+x_2+\cdots+x_n+f(0,0,\cdots,0)]\cdot\\
&\quad [\bar{x}_1+\bar{x}_2+\cdots+x_n+f(1,1,\cdots,0)]\cdots\\
&\quad [(\bar{x}_1+\bar{x}_2+\cdots+\bar{x}_n)+f(1,1,\cdots,1)]
\end{aligned}$$

yazılır. Bu ifadede yer alan değişkenler, lojik ifadeler olarak değerlendirilirken, f fonksiyon değeri lojik 0 veya lojik 1 değerini alarak, hangi terimlerin maksimum terimler kanonik biçimine dahil olacağını belirler. Örneğin, a, b ve c gibi üç değişken için,

$$\begin{aligned}
f(a,b,c) &= [a+b+c+f(0,0,0)]\cdot[a+b+\bar{c}+f(0,0,1)]\cdot[a+\bar{b}+c+f(0,1,0)]\\
&\cdot[a+\bar{b}+\bar{c}+f(0,1,1)].[\bar{a}+b+c+f(1,0,0)]\cdot[\bar{a}+b+\bar{c}+f(1,0,1)]\\
&\cdot[\bar{a}+\bar{b}+c+f(1,1,0)]\cdot[\bar{a}+\bar{b}+\bar{c}+f(1,1,1)
\end{aligned}$$

ifadesi bulunur.

Örnek-5.11:

Bu örnekte, $f(a,b,c)=a\cdot b+\bar{a}\cdot c$ fonksiyonunun maksimum terimler kanonik biçiminin elde edilme aşamaları gösterilmiştir. Bu amaçla,

$$\begin{aligned} f(a,b,c) = {} & [a+b+c+f(0,0,0)]\cdot[a+b+\bar{c}+f(0,0,1)]\cdot[a+\bar{b}+c+f(0,1,0)] \\ & \cdot[a+\bar{b}+\bar{c}+f(0,1,1)]\cdot[\bar{a}+b+c+f(1,0,0)]\cdot[\bar{a}+b+\bar{c}+f(1,0,1)] \\ & \cdot[\bar{a}+\bar{b}+c+f(1,1,0)]\cdot\left[\bar{a}+\bar{b}+\bar{c}+f(1,1,1)\right] \end{aligned}$$

fonksiyonu göz önüne alınır. Bu fonksiyondaki her bir f değeri aşağıdaki gibi belirlenir.

$$\begin{aligned} f(0,0,0) &= 0\cdot0+1\cdot0 = 0+0 = 0 \\ f(0,0,1) &= 0\cdot0+1\cdot1 = 0+1 = 1 \\ f(0,1,0) &= 0\cdot1+1\cdot0 = 0+0 = 0 \\ f(0,1,1) &= 0\cdot1+1\cdot1 = 0+1 = 1 \\ f(1,0,0) &= 1\cdot0+0\cdot0 = 0+0 = 0 \\ f(1,0,1) &= 1\cdot0+0\cdot1 = 0+0 = 0 \\ f(1,1,0) &= 1\cdot1+0\cdot0 = 1+0 = 1 \\ f(1,1,1) &= 1\cdot1+0\cdot1 = 1+0 = 1 \end{aligned}$$

f fonksiyonlarından 0 değerini alan terimlerin çarpımı, Maksimum Terimler Kanonik Biçimini oluşturur. Bu durumda,

$$\begin{aligned} f(a,b,c) &= [a+b+c]\cdot[a+\bar{b}+c]\cdot[\bar{a}+b+c]\cdot[\bar{a}+b+\bar{c}] \\ &= M_0\cdot M_2\cdot M_4\cdot M_5 \\ &= \prod(0,2,4,5) \end{aligned}$$

elde edilir. Bir lojik fonksiyonun sadece bir tane minimum terimler kanonik biçimi ve bir tane de maksimum terimler kanonik biçimi mevcuttur.

5.6. Kanonik Biçimler Arasındaki Dönüşüm

Minimum terimler kanonik biçimi verilen bir ifadenin maksimum terimler kanonik biçimini bulmak ve bunun tersini gerçekleştirmek mümkündür. Tablo-5.10 da verildiği üzere minimum terimlerin tümleyenini almakla maksimum terimleri elde etmek mümkün olabilmektedir. Bu işlem yapılırken Boole Cebri teoremleri 7)a ve b) de verilen *De Morgan* Kuralından yararlanılır.

Örnek-5.12.

Bu örnekte, $f(a,b,c)=\sum(1,3,5,7)$ şeklinde minimum terimleriyle verilmiş fonksiyonunun maksimum terimlerinin belirlenmesi anlatılmıştır. Bu amaçla öncelikle minimum terimlerin tümleyenleri olan $\bar{f}(a,b,c)=\sum(0,2,4,6)$ fonksiyonu belirlenir. Buradaki terimler

$$\bar{f}(a,b,c)=\sum(0,2,4,6)= m_0+m_2+m_4+m_6$$

şeklinde minimum terimlerin toplamıdır. *De Morgan* kuralı yardımıyla bu ifade farklı bir şekilde yazılabilir. Bu durumda,

$$f=\overline{(m_0+m_2+m_4+m_6)}=\bar{m}_0\cdot\bar{m}_2\cdot\bar{m}_4\cdot\bar{m}_6$$

elde edilir. Minimum terimlerin tümleyeni maksimum terimleri tanımladığından bu ifade yerine

$$f=\overline{(m_0+m_2+m_4+m_6)}=\bar{m}_0\cdot\bar{m}_2\cdot\bar{m}_4\cdot\bar{m}_6=M_0\cdot M_2\cdot M_4\cdot M_6$$

yazılır. Bu durumda Minimum Terim Kanonik Şekli $\sum(1,3,5,7)$ olan bir $f(a,b,c)$ fonksiyonunun, Maksimum Terimler Kanonik Şekli $\prod(0,2,4,6)$ olarak yazılabilir.

Örnek-5.12 den de görülebileceği üzere, minimum terimler ile maksimum terimler arasında, m'ye indis ile gösterilişinde,

$$\bar{m}_i=M_i$$

ilişkisi geçerli olur.

5.7. Doğruluk Tablosundan Kanonik Biçimleri Bulunması

Verilen bir lojik fonksiyonda değişkenlerin aldıkları değerler yerine konularak, bu fonksiyonun değerleri belirlenir ve buradan Minimum ya da Maksimum Terimler Kanonik Biçimi oluşturulabilir. Bir fonksiyonun aldığı değerler bir araya getirilerek doğruluk tablosu oluşturulabilir. Doğruluk tablosunda "1" değerini alanlar lojik toplanarak (VEYA) minimum terimler kanonik biçimi, "0" değerini alanlar lojik çarpılarak (VE) maksimum terimler kanonik biçimi oluşturulur.

Örnek-5.13.

Burada Örnek-5.9 ve Örnek-5.11 de verilen $f(a,b,c)=a\cdot b+\bar{a}\cdot c$ lojik fonksiyon ele alınmış ve doğruluk tablosu oluşturulduktan sonra, minimum ve maksimum terimler kanonik biçimi elde edilmesi aşamaları irdelenmiştir. Bu amaçla, her bir değişkenin aldığı değer göz önünde bulundurularak,

$$
\begin{aligned}
f(0,0,0) &= 0\cdot 0+1\cdot 0 = 0+0 = 0\\
f(0,0,1) &= 0\cdot 0+1\cdot 1 = 0+1 = 1\\
f(0,1,0) &= 0\cdot 1+1\cdot 0 = 0+0 = 0\\
f(0,1,1) &= 0\cdot 1+1\cdot 1 = 0+1 = 1\\
f(1,0,0) &= 1\cdot 0+0\cdot 0 = 0+0 = 0\\
f(1,0,1) &= 1\cdot 0+0\cdot 1 = 0+0 = 0\\
f(1,1,0) &= 1\cdot 1+0\cdot 0 = 1+0 = 1\\
f(1,1,1) &= 1\cdot 1+0\cdot 1 = 1+0 = 1
\end{aligned}
$$

elde edilir. f fonksiyonunun aldığı değerler, Tablo-5.13 de gösterildiği gibi, doğruluk tablosunda doğrudan yerine konulur.

Tablo-5.13. VE işlemine ilişkin doğruluk tablosu ve maksimum terimler.

a	b	c	$f(a,b,c)=a\cdot b+\bar{a}\cdot c$	*Minterm*	*Maksterm*
0	0	0	0	-	$a+b+c$
0	0	1	1	$\bar{a}\cdot\bar{b}\cdot c$	-
0	1	0	0	-	$a+\bar{b}+c$
0	1	1	1	$\bar{a}\cdot b\cdot c$	-
1	0	0	0	-	$\bar{a}+b+c$
1	0	1	0	-	$\bar{a}+b+\bar{c}$
1	1	0	1	$a\cdot b\cdot\bar{c}$	-
1	1	1	1	$a\cdot b\cdot c$	-

Bu tabloda fonksiyonun "0" değerini aldığı noktalarda Minimum Terimleri ve fonksiyonun "1" değerini aldığı noktalarda Maksimum Terimleri yazmaya gerek yoktur. Kanonik biçimleri elde etmek için, Minimum Terimler toplanır veya Maksimum terimler çarpılır. Bu durumda, $f(a,b,c)=a\cdot b+\bar{a}\cdot c$ fonksiyonuna ilişkin Minimum Terimler Kanonik Biçimi, doğruluk tablosundaki fonksiyonun "1" değerini aldığı noktalardaki mintermlerin toplamıyla,

$$f(a,b,c)=\bar{a}\cdot\bar{b}\cdot c+\bar{a}\cdot b\cdot c+a\cdot b\cdot\bar{c}+a\cdot b\cdot c$$

şeklinde elde edilir. Benzer şekilde, aynı fonksiyona ilişkin Maksimum Terimler Kanonik Biçimi, yine fonksiyonun doğruluk tablosundaki 0 değerini aldığı noktalardaki makstermlerin çarpımıyla,

$$f(a,b,c)=[a+b+c]\cdot[a+\bar{b}+c]\cdot[\bar{a}+b+c]\cdot[\bar{a}+b+\bar{c}]$$

şeklinde elde edilir. Bulunan bu değerler, Örnek-5.9. ve Örnek-5.11 de elde edilen sonuçlarla aynıdır.

Bazı durumlarda doğruluk tablosu doğrudan verilmiş olabilir. Bu durumda gerek Minimum Terimler Kanonik Biçimin, gerekse Maksimum Terimler Kanonik Biçimini elde etmek mümkündür.

Örnek-5.14.

Bu örnekte, Tablo-5.14 te doğruluk tablosu verilen lojik fonksiyona ilişkin, Minimum ve Maksimum Terimler Kanonik Biçimi elde edilmiştir.

Tablo-5.14. VE işlemine ilişkin doğruluk tablosu ve maksimum terimler.

a	b	c	$f(a,b,c)$	*Minterm*	*Maksterm*
0	0	0	1	$\bar{a}\cdot\bar{b}\cdot\bar{c}$	-
0	0	1	1	$\bar{a}\cdot\bar{b}\cdot c$	-
0	1	0	0	-	$a+\bar{b}+c$
0	1	1	0	-	$a+\bar{b}+\bar{c}$
1	0	0	1	$a\ \bar{b}\ \bar{c}$	-
1	0	1	1	$a\ \bar{b}\ c$	-
1	1	0	0	-	$\bar{a}+\bar{b}+c$
1	1	1	0	-	$\bar{a}+\bar{b}+\bar{c}$

Bu tabloda fonksiyonun "1" olduğu noktalarda minimum terimler, "0" olduğu noktalarda maksimum terimler ifade edilir. Fonksiyonun "0" değerini aldığı noktalarda minimum terimleri ve 1 değerini aldığı noktalarda maksimum terimleri yazmaya gerek yoktur. Kanonik biçimleri elde etmek için, minimum terimler toplanır veya maksimum terimler çarpılır. Bu durumda, $f(a,b,c)$ fonksiyonuna ilişkin minimum terimler kanonik biçimi, doğruluk tablosundaki fonksiyonun 1 değerini aldığı noktalardaki mintermlerin toplamıyla,

$$f(a,b,c)=\bar{a}\cdot\bar{b}\cdot\bar{c}+\bar{a}\cdot\bar{b}\cdot c+a\cdot\bar{b}\cdot\bar{c}+a\cdot\bar{b}\cdot c$$

şeklinde elde edilir. Benzer şekilde, aynı fonksiyona ilişkin maksimum terimler kanonik biçimi, yine fonksiyonun doğruluk tablosundaki 0 değerini aldığı noktalardaki makstermlerin çarpımıyla,

$$f(a,b,c)=[a+\bar{b}+c]\cdot[a+\bar{b}+\bar{c}]\cdot[\bar{a}+\bar{b}+c]\cdot[\bar{a}+\bar{b}+\bar{c}]$$

şeklinde elde edilir.

5.8. Kanonik Biçimlerin Lojik Kapılarla Gerçeklenmesi

Bir lojik fonksiyonun kanonik biçimi elde edildikten sonra, lojik kapı elemanları kullanılarak fiziksel olarak gerçeklenebilir. Tabi bu noktada kullanılacak eleman ve lojik tasarım aşamaları çok büyük önem taşımaktadır. Kanonik biçim ifadeleriyle yapılan bazı tasarımlar maliyeti çok arttırdığı için tercih edilmezler. Bu yöntem yerine indirgenmiş fonksiyonlarla gerçeklenen tasarım yöntemleri kullanılır. Lojik fonksiyonları indirgeyerek ve maliyet faktörü göz önünde bulundurularak yapılan tasarımlara ilerleyen bölümlerde yer verilecektir. Bununla birlikte, lojik kapıları daha yakından tanımak ve bu kapıları kullanarak basit devreler kurmak amaciyla, bu bölümde kanonik biçimleri kullanarak gerçeklenen tasarımlara yer verilmiştir.

Kanonik biçim ifadelerinde iki kademeli bir lojik devre yapısından söz etmek doğru olur; ya önce çarpımlar gerçeklenir sonra toplamları alınır ya da önce toplamlar gerçeklenir sonra çarpımları alınır. Nitekim bu devre, gerek lojik fonksiyonun ifadesi en sade biçimde yazılmadığı, gerekse lojik devre fiziksel olarak gerçeklenirken tek tip eleman kullanılmadığı için ekonomik olarak tasarlanmış bir devre olarak yorumlanmamalıdır. Mühendislikte yapılan tasarım kadar tasarımın ekonomik ya da bir başka deyişle en düşük maliyetli olması temel ilkelerden birisidir. Bu durumda doğrudan kanonik biçimleri elde edilmiş lojik fonksiyonları doğrudan kullanmak yerine ya bu fonksiyonlar tek tip eleman kullanılacak biçimde düzenlenir ya da lojik ifade en az terime indirgenerek tasarım yapılır.

Bu bölümde öncelikle kanonik biçimde verilmiş bir lojik ifadenin lojik kapı elemanları kullanılarak fiziksel gerçeklenmesi açıklanmış, daha sonrada mevcut kapı elemanları ile bilinen kapı elemanlarının tasarlanması için gerekli bilgiler verilmiştir. Lojik Devrelerde tek tip kapı elemanları kullanılarak tasarım yapılması Bölüm 6'da, karmaşıklık hesabı ve maliyet faktörü kavramları Bölüm 8'de ayrıntılı olarak incelenmiştir.

Örnek-5.15.

Bu örnekte, minimum terimler kanonik biçimi,

$$f(a,b,c)=\bar{a}\cdot\bar{b}\cdot c+\bar{a}\cdot b\cdot c+a\cdot b\cdot\bar{c}+a\cdot\bar{b}\cdot c$$

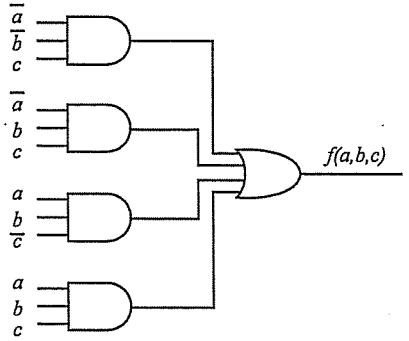
ve maksimum terimler kanonik biçimi,

$$f(a,b,c)=[a+b+c]\cdot[a+\bar{b}+c]\cdot[\bar{a}+b+c]\cdot[\bar{a}+b+\bar{c}]$$

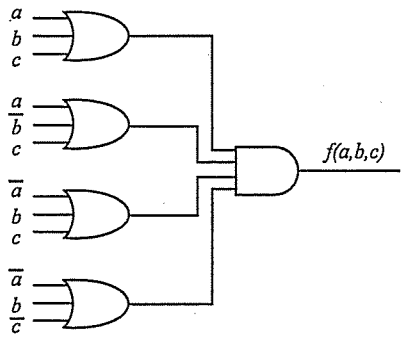
şeklinde verilmiş olan lojik fonksiyonun kapı elemanlarıyla fiziksel gerçeklenmesi verilmektedir.

Her iki yapı da iki düzeyli bir lojik devre ile gerçeklenir. Minimum terimler kanonik biçimi ifadesinde önce çarpım terimleri VE kapılarıyla gerçeklenir, sonra VEYA kapıları ile toplamlar alınır. Dolayısıyla Şekil-5.22 de lojik devre şeması verilen çarpımlar toplamı ifadesi elde edilir.

Maksimum Terimler Kanonik Biçimi ifadesinde önce toplam terimleri VEYA kapılarıyla gerçeklenir, sonra VE kapıları ile çarpımlar alınır. Dolayısıyla Şekil-5.23.'de lojik devre şeması verilen toplamlar çarpımı ifadesi elde edilmiş olunur.



Şekil-5.22. Minimum terimler kanonik biçimiyle oluşturulmuş bir lojik devre.



Şekil-5.23. Maksimum terimler kanonik biçimiyle oluşturulmuş bir lojik devre.

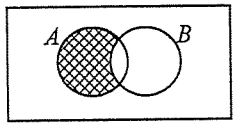
Kanonik biçimlerin gerçeklenmesinde kullanılan iki kademeli devre yapısı Şekil-5.22 ve Şekil-5.23 de daha ayrıntılı olarak gözükmektedir. Tek tür kapı elemanları kullanılarak indirgenmiş ifadeler üzerinde yapılan tasarımlar mühendislik açısından değerlendirilirse daha iyi gerçeklenmiş tasarımlar olarak yorumlanırlar. Durum böyle olunca da öncelikle lojik fonksiyonların indirgenmesi sonra da tek tür kapı elemanlarıyla tasarım yapılması konularının incelenmesi gerekir; Bölüm 6'da da bu konu üzerinde durulmuştur.

5.9. Özet

Lojik kapılar, lojik devre tasarımının temel taşlarıdır ve bilgisayar mühendisliğinin başlangıç noktasını oluşturmaktadır. Boole cebri ise lojik elemanlar için temel kavramların ve özelliklerin belirlendiği cebir tabanlı bir kurallar kümesidir. Lojik devrelerle yapılan bütün tasarımlar Boole cebri kurallarına uygun olmak zorundadırlar. En temel tasarım elemanları VE, VEYA, TÜMLEME gibi kapı elemanlarıdır. Bunların dışında TVE, TVEYA, YA DA, TYA DA gibi türetilen kapı elemanları da vardır. Bu kapıların kümeler cebri ve Boole Cebri aksiyom ve teoremleri kullanılarak yalın lojik ifadelerin gerçekleştirilmesi yapılabilir. VE kapısına dayanan lojik ifadeler minimum terimler, VEYA kapısına dayanan lojik ifadeler maksimum terimler kanonik biçimi olarak adlandırılır. Tümleşik lojik devrelerde bile, kapılarla gerçekleştirilen yalın lojik ifadeler bulunur.

5.10. Sorular

1. $a\cdot(b+c)=a\cdot b+a\cdot c$ şeklinde verilen eşitliğin sağlanıp sağlanmadığını doğruluk tablosu oluşturarak ve *Venn* diyagramlarını kullanarak gösteriniz.
2. $a\cdot(a+b)=a$ şeklinde verilen eşitliğin sağlanıp sağlanmadığını doğruluk tablosu oluşturarak ve *Venn* diyagramlarını kullanarak gösteriniz.
3. a) Aşağıda *Venn* diyagramı ile verilen taralı alana ilişkin ifadeyi yazınız.



   b) Bu ifadeye ilişkin minimum terimler kanonik biçimini elde ediniz.
   c) Kanonik biçime ilişkin ifadeyi *Venn* diyagramları ile çizerek iki ifadenin eşdeğer olup olmadığını tartışınız.
4. $(a+b)\cdot(\overline{a}+c)=a\cdot c+\overline{a}\cdot b$ şeklinde verilen eşitliğin sağlanıp sağlanmadığını Boole Cebrine ilişkin aksiyom ve teoremleri kullanarak gösteriniz.

5. $(a+\overline{b})\cdot b=a\cdot b$ olduğunu Boole Cebrine ilişkin aksiyom ve teoremleri kullanarak gösteriniz.
6. $(a\cdot\overline{b}+b)=a+b$ şeklinde bir lojik eşitlik verilmiştir. Bu ifadenin sağlanıp sağlanmadığını;
   a) Doğruluk tablosu ile
   b) *Venn* diyagramları ile
   c) Boole cebrine ait aksiyom ve teoremleri kullanarak gösteriniz.
7. $\overline{(x_1+x_2+...+x_n)}=\overline{x_1}.\overline{x_2}.....\overline{x_n}$ Boole Cebri teoremini ispatlayınız. (Yol Gösterme: Boole Cebrinin diğer özelliklerini kullanarak önce iki değişken için ispatlayıp, sonra genelleştiriniz.)
8. $f(a,b,c,d)=a\cdot b+\overline{a}\cdot\overline{c}$ şeklinde verilen fonksiyonun minimum terimler kanonik biçimini elde ediniz.
9. $f(a,b,c)=\overline{a}\cdot b+a\cdot c$ fonksiyonu verilmiştir.
   a) Bu fonksiyona ilişkin doğruluk tablosu oluşturunuz.
   b) Fonksiyona ilişkin minimum ve maksimum terimler kanonik biçimlerini elde ediniz.
   c) Lojik kapı elemanlarını kullanarak her iki kanonik şekli ilişkin ifadeleri gerçekleyecek devre şemalarını çiziniz.
10. $f(a,b,c)=\overline{c}\cdot d\cdot(a+b)+c\cdot d\cdot(a+b)$ fonksiyonu verilmiştir.
   a) Bu fonksiyona ilişkin doğruluk tablosu oluşturunuz.
   b) Fonksiyona ilişkin minimum terimler kanonik biçimini elde ediniz.
   c) Fonksiyona ilişkin maksimum terimler kanonik biçimini elde ediniz.
   d) Lojik kapı elemanlarını kullanarak her iki kanonik şekli ilişkin ifadeleri gerçekleyecek devre şemalarını çiziniz.
11. $f(a,b,c)=\sum(0,1,3,5,6)$ fonksiyonu verilmiştir.
   a) Bu fonksiyona ilişkin doğruluk tablosu oluşturunuz.
   b) Fonksiyona ilişkin minimum terimler kanonik biçimini değişkenler cinsinden ifade ediniz.
   c) Fonksiyona ilişkin maksimum terimler kanonik biçimini değişkenler cinsinden ifade ediniz.
   d) Lojik kapı elemanlarını kullanarak her iki kanonik şekli ilişkin ifadeleri gerçekleyecek devre şemalarını çiziniz.

6.

Lojik Fonksiyonların İndirgenmesi

Lojik fonksiyonların indirgenmesi, genel olarak, "*lojik ifadenin farklı giriş değerlerine göre çıkış değerinin değişikliğe uğratılmadan daha az sayıda terimle ifade edilmesidir*" denilebilir. Böylece daha az maliyetli tasarımlar yapılabilir ve yalın ifadelerle uğraşma imkanı doğar. Mühendislikte temel faktörlerden biri tasarım maliyetidir. Daha doğrusu, tasarım, karmaşıklık ve maliyet faktörü göz önünde tutularak yapılır. Bu durumda, tasarımcı devre tasarımında elde ettiği lojik fonksiyonu aynı tür kapı elemanları kullanarak gerçekleştirme ve bunu yaparken de lojik ifadenin yalın halini kullanma yoluna gidebilir. Kısacası, doğrudan lojik ifadenin kullanılması maliyet ve devre karmaşıklığı açısından pek uygun bir çözüm olmayabilir. Bu durumda lojik fonksiyonun indirgenmesi akla gelir; indirgemeyle hem kullanılan lojik kapı sayısı hem de değişken sayısı azaltılmaya çalışılır. Bu bölümde, lojik fonksiyonların indirgenmesi konusu aşağıdaki başlıklar altında ele alınmıştır:

6.1. Görüşe Dayanarak İndirgeme
6.2. *Karnaugh* Diyagramı Yöntemi
6.3. *Quine-McCluskey*/Tablo Yöntemi
6.4. Eksik Boole İşlevleri
6.5. Türetilmiş Kapılarla Temel Lojik Kapıları
6.6. İndirgenmiş İfadelerin Aynı Tür Kapılarla Gerçeklenmesi
6.7. Özet
6.8. Sorular

Lojik fonksiyonlar çeşitli şekillerde indirgenebilir. Örneğin, görüşe dayanılarak indirgeme yapılabilir; tabi ki, bu yöntemde tasarımcının iyi bir görüş düzeyi ve deneyimi olmalıdır. Diğer yöntemler ise diyagram tasarımını dayanan *Karnaugh* diyagramı yöntemi ve tablo yöntemi olarak bilinen *Quine-McCluskey* yöntemidir. İlerleyen kısımlarda görüşe ve *Karnaugh* diyagramı yöntemine dayanan indirgeme yöntemleri ayrıntılı olarak ele alınmış olup *Quine-McCluskey* yöntemi kavramsal olarak incelenmiştir.

Lojik fonksiyonların değişik birçok cebirsel ifadeleri vardır. İndirgemeden amaç bu ifadelerden en uygun olanını seçmektir. En uygun olanı maliyet/karmaşıklık unsuruna dayanılarak yapılmalıdır. O halde, indirgeme, cebirsel ifadeler açısından, maliyet / karmaşıklık ölçütünü en küçük düzeyde tutan ifadeyi seçme işlemidir; bu ölçüt, terim sayısı, transistör sayısı, ifadedeki değişken sayısı, değişkenlerin tekrarlanma sayısı, bir terimdeki değişken sayısı, aynı tür elemandan mümkün olduğunca çok kullanılması gibi parametrelerden bir veya birkaçına bağlı olabilir. [HARMANCI- 1987]

6.1. Görüşe Dayanarak İndirgeme

Boole Cebrine ilişkin aksiyom ve teoremlerden yararlanarak bir lojik fonksiyonu daha az terimle ifade etmek mümkündür. Eğer indirgeme işlemi bu şekilde gerçeklenirse bu işlem tasarımcının görüşüne bağlı olduğu için, görüşe dayanarak indirgeme olarak isimlendirilir. Sonuçta elde edilen ifadelere Boole Cebrinde geçerli bütün özellikler kullanılsa da lojik fonksiyon daha fazla basitleştirilemez. Dolayısıyla bu yöntemde kullanıcının görüşü ve konuya hakimiyeti ön plandadır.

Örnek-6.1.

Bu örnekte, $f(a,b,c,d)=\bar{a}\cdot b\cdot\bar{c}+a\cdot d+b\cdot\bar{c}\cdot\bar{d}$ şeklinde verilen ifadenin minimal elemanla ifade edilip edilmediği sorgulanmıştır. Eğer verilen ifade minimal bir ifade değilse bu ifade daha da basitleştirilmeye çalışılır ve bu işlem tamamen tasarımcının görüşü çerçevesinde gerçekleştirilir. Verilen ifadede üç terim yer almaktadır. Bu terimlerden ortak paranteze alınabilen değişkenler değerlendirilir. Bu durumda birinci ve üçüncü terimlerde $b\cdot\bar{c}$ değişkenlerinin ortak olduğu görülmektedir. Bu değişkenler ortak paranteze alınırsa,

$$\begin{aligned} f(a,b,c,d) &= \bar{a}\cdot b\cdot\bar{c}+a\cdot d+b\cdot\bar{c}\cdot\bar{d} \\ &= b\cdot\bar{c}\cdot(\bar{a}+\bar{d})+a\cdot d \end{aligned}$$

elde edilir. Buradaki $(\bar{a}+\bar{d})$ terimi $(\overline{a\cdot d})$ ifadesine eşdeğerdir. Bu durumda,

$$f(a,b,c,d)=b\cdot\bar{c}\cdot(\overline{a\cdot d})+a\cdot d$$

yazılabilir. Buradaki $(a\cdot d)$ terimini diğer iki terim üzerine dağıtma imkanı mümkündür. Bu özellik Boole Cebri teoremlerinden 3)a koşulundan görülebilir. Bu durumda,

$$f(a,b,c,d)=(b\cdot\bar{c}+a\cdot d)\cdot(\overline{a\cdot d}+a\cdot d)$$

elde edilir. $(\overline{a\cdot d})$ terimi $(a\cdot d)$'nin tümleyenidir. Bu yüzden $(\overline{a\cdot d}+a\cdot d)=1$ değerini alır ve ifade,

$$f(a,b,c,d)=(b\cdot\bar{c}+a\cdot d)$$

terimine indirgenir. Bu ifadeye Boole Cebrinin diğer bütün özellikleri uygulansa bile daha basit bir ifade elde edilemez. Sonuçta bulunan bu ifade minimal bir ifadedir ve en başta verilen ifadenin indirgenmiş şeklidir. En baştaki ifade ise daha basit terimlere indirgendiği için minimal değildir.

Örnek-6.2.

Bu örnekte, $f(x,y,z)=\sum(0,4)$ şeklinde minimum terimler kanonik şekliyle verilen ifadenin minimal bir ifade olup olmadığı irdelenmiştir. Eğer ifade minimal değilse bu ifade indirgenmeye çalışılır. Öncelikle minimum terimler kanonik biçimini oluşturan terimler belirlenir. Bu durumda,

$$f(x,y,z)=\sum(0,4)=\bar{x}\cdot\bar{y}\cdot\bar{z}+x\cdot\bar{y}\cdot\bar{z}$$

elde edilir. Bu terimler $(\bar{y}\cdot\bar{z})$ ortak parantezine alınabilir. Bu durumda,

$$f(x,y,z)=\bar{y}\cdot\bar{z}\cdot(\bar{x}+x)$$

yazılabilir. Buradaki $(\bar{x}+x)=1$ olduğundan ifade,

$$f(x,y,z)=\bar{y}\cdot\bar{z}$$

terimine indirgenir. Bu ifadeye Boole Cebrinin diğer bütün özellikleri uygulansa bile daha basit bir ifade elde edilemez. Sonuçta bulunan bu ifade minimal bir ifadedir ve en başta verilen minimum terimler kanonik şeklindeki ifadenin indirgenmiş şeklidir. Minimum terimler kanonik şeklindeki ifade daha basit terimlere indirgendiği için minimal değildir.

Örnek-6.3.

Bu örnekte, $f(x,y,z)=\prod(1,5)$ şeklinde maksimum terimler kanonik şekliyle verilen ifadenin minimal bir ifade olup olmadığı irdelenmiştir. Eğer ifade minimal değilse bu ifade indirgenmeye çalışılır. Öncelikle maksimum terimler kanonik biçimini oluşturan terimler belirlenir. Bu durumda,

$$f(x,y,z)=\prod(1,5)=[x+y+\bar{z}]\cdot[\bar{x}+y+\bar{z}]$$

elde edilir. Parantez içindeki terimler dağıtılırsa,

$$f(x,y,z)=x\cdot\bar{x}+x\cdot y+x\cdot\bar{z}+y+y\cdot\bar{z}+\bar{z}\cdot\bar{x}+\bar{z}\cdot x+\bar{z}$$

elde edilir. Bu ifadede $x \cdot \bar{x} = 0$'dır. Geriye kalan terimler y ve $\bar{z}$ ortak parantezine alınırsa,

$$f(x,y,z) = y \cdot (x + \bar{x} + 1 + \bar{z} + \bar{z}) + \bar{z} \cdot (x + \bar{x} + 1)$$

bulunur. Her iki parantezin içi de 1 olduğundan sonuçta,

$$f(x,y,z) = y + \bar{z}$$

indirgenmiş terimi elde edilir. Bu ifadeye Boole Cebrinin diğer bütün özellikleri uygulansa bile daha basit bir ifade elde edilemez. Sonuçta bulunan bu ifade minimal bir ifadedir ve en başta verilen maksimum terimler kanonik şeklindeki ifadenin indirgenmiş şeklidir. Maksimum terimler kanonik şeklindeki ifade daha basit terimlere indirgenebildiği için minimal değildir.

6.2. *Karnaugh* Diyagramı Yöntemi

1952 yılında *Veitch* tarafından önerilen ve son halini 1953 yılında *Karnaugh* tarafından geliştirilen bu yöntemde lojik işlemlerin ifade edildiği doğruluk tablosu çevrimsel bir biçimde gösterilmiştir. Bu yöntem sayesinde lojik fonksiyonları indirgemek kolay ve sistematik bir şekle getirilmiştir.

Yöntem açıklanırken, öncelikle doğruluk tablosundan *Karnaugh* diyagramlarının oluşturulması açıklanmış, daha sonra komşuluk kavramı anlatıldıktan sonra *Karnaugh* diyagramlarının indirgenmesi işleminin adımları ayrıntılı olarak irdelenmiştir.

Öncelikle iki değişkenli bir lojik fonksiyon için doğruluk tablosundan *Karnaugh* diyagramına geçiş işlemi anlatılmıştır. Bu amaçla, Tablo-6.1 de verilen doğruluk tablosundan yararlanılmıştır.

Tablo-6.1. Doğruluk tablosu gösterilimi.

x	y	$f(x,y)$
0	0	m_0
0	1	m_1
1	0	m_2
1	1	m_3

Bu doğruluk tablosu için *Karnaugh* diyagramı Şekil-6.1 de oluşturulmuştur.

$x \backslash y$	0	1
0	m_0	m_1
1	m_2	m_3

Şekil-6.1. Tablo-6.1 de verilen doğruluk tablosuna ilişkin *Karnaugh* diyagramı.

Örnek-6.4.

Bu örnekte, $f(x,y) = x + y$ şeklindeki lojik fonksiyona ilişkin doğruluk tablosu oluşturulmuş ve bu tablodan yararlanarak *Karnaugh* diyagramı elde edilmiştir. $f(x,y) = x + y$ fonksiyonuna ilişkin doğruluk tablosu ve bu doğruluk tablosu için *Karnaugh* diyagramı aşağıda verilmiştir.

x	y	$f(x,y)$
0	0	0
0	1	1
1	0	1
1	1	1

$x \backslash y$	0	1
0	0	1
1	1	1

Üç değişkenli bir lojik fonksiyon için doğruluk tablosundan *Karnaugh* diyagramına geçiş işlemi de benzer şekilde gerçeklenir. Bu amaçla, Tablo-6.2 de verilen doğruluk tablosu kullanılır.

Tablo-6.2. Doğruluk tablosu gösterilimi.

x	y	z	$f(x,y,z)$
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7

Bu doğruluk tablosu için *Karnaugh* diyagramı Şekil-6.2 de verilmiştir. Şekilden de görülebileceği üzere *Karnaugh* diyagramı iki şekilde oluşturulabilir.

x \ yz	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

xy \ z	0	1
00	m_0	m_1
01	m_2	m_3
11	m_6	m_7
10	m_4	m_5

Şekil-6.2. Tablo-6.2 de verilen doğruluk tablosuna ilişkin *Karnaugh* diyagramı.

Örnek-6.5.

Bu örnekte, $f(x,y,z)=x+y+z$ şeklindeki lojik fonksiyona ilişkin doğruluk tablosu oluşturulmuş ve bu tablodan yararlanarak *Karnaugh* diyagramı elde edilme yoluna gidilmiştir. $f(x,y,z)=x+y+z$ fonksiyonuna ilişkin doğruluk tablosu,

x	y	z	f(x,y)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

şeklinde elde edilir ve bu doğruluk tablosu için *Karnaugh* diyagramı,

x \ yz	00	01	11	10
0	0	1	1	1
1	1	1	1	1

xy \ z	0	1
00	0	1
01	1	1
11	1	1
10	1	1

olarak bulunur.

Dört değişkenli bir lojik fonksiyon için doğruluk tablosundan *Karnaugh* diyagramına geçiş işlemi için de aynı yöntem kullanılır. Bu amaçla, Tablo-6.3 de verilen doğruluk tablosundan yararlanılmış ve *Karnaugh* diyagramı Şekil-6.3 deki gibi elde edilmiştir.

Tablo-6.3. Doğruluk tablosu gösterilimi.

x	y	z	w	F(x,y,z,w)
0	0	0	0	m_0
0	0	0	1	m_1
0	0	1	0	m_2
0	0	1	1	m_3
0	1	0	0	m_4
0	1	0	1	m_5
0	1	1	0	m_6
0	1	1	1	m_7
1	0	0	0	m_8
1	0	0	1	m_9
1	0	1	0	m_{10}
1	0	1	1	m_{11}
1	1	0	0	m_{12}
1	1	0	1	m_{13}
1	1	1	0	m_{14}
1	1	1	1	m_{15}

xy \ zw	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

Şekil-6.3. Tablo-6.3 de verilen doğruluk tablosuna ilişkin *Karnaugh* diyagramı.

Örnek-6.6.

Bu örnekte, $f(x,y,z,w)=x\cdot y+w\cdot z$ şeklindeki lojik fonksiyona ilişkin doğruluk tablosu oluşturularak *Karnaugh* diyagramı bulunmuştur. $f(x,y,z,w)=x\cdot y+w\cdot z$ fonksiyonuna ilişkin doğruluk tablosu ve bu doğruluk tablosu için *Karnaugh* diyagramı aşağıda verilmiştir.

x	y	z	w	$f(x,y,z,w)$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

xy \ zw	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	1	1	1	1
10	0	0	1	0

Karnaugh Yönteminde Komşuluk Kavramı

Bu noktaya kadar, doğruluk tablosu oluşturulmuş bir lojik fonksiyon için *Karnaugh* Diyagramının nasıl oluşturulacağı gösterilmiştir. Ancak bu noktadan sonra önemli olan lojik fonksiyonların indirgenmiş ifadelerini elde etmektir.

Karnaugh Diyagramını doğru oluşturmak, fonksiyonu indirgeyebilmenin ilk ve en önemli adımıdır. Zira ilk aşamada *Karnaugh* dizilişindeki farklılığı belirtmek yararlı olur. İkili düzende iki bitlik bir sayının dizisi 00-01-10-11 şeklinde sıralanırken, *Karnaugh* Diyagramında *Hamming* uzaklığının en fazla bir olmasına izin verilen bitişik kodların özelliklerinden iki bitlik bir sayı dizisi *Karnaugh* diyagramında 00-01-11-10 şeklinde ifade edilmek zorundadır.

Karnaugh Diyagramlarının indirgenmesine imkan sağlayan komşuluk kavramı şu şekilde açıklanabilir: n değişkenli bir fonksiyonda 2^k $(k=1,2,3,...)$ tane terim kısalıp $(n-k)$ çarpanlı bir tek terime dönüşürse bu 2^k tane terim k'ıncı mertebeden komşudur. *Karnaugh* diyagramı gösteriliminden önce lojik fonksiyon üzerinde bir örnek aşağıda verilmiştir.

Örnek-6.7.

n=4 değişkenli $2^k=4$ tane terim indirgenip 2. dereceden bir komşuluk oluşturabilir.

$$f(x,y,z,w)=x\cdot\bar{y}\cdot\bar{z}\cdot\bar{w}+x\cdot\bar{y}\cdot\bar{z}\cdot w+x\cdot y\cdot\bar{z}\cdot\bar{w}+x\cdot y\cdot\bar{z}\cdot w$$

şeklindeki lojik fonksiyonda $x\cdot\bar{z}$ ortak çarpan parantezine alınarak,

$$f(x,y,z,w)=x\cdot\bar{z}.(\bar{y}\cdot\bar{w}+\bar{y}\cdot w+y\cdot\bar{w}+y\cdot w)$$

şeklinde yazılabilir. Buradaki dört terimden oluşan ifadenin değeri 1'dir. Bu durumda 4 terim 2. dereceden komşuluk oluşturarak ortadan kaldırılmıştır.

Aşağıda verilen teoremlerde dört değişkenli bir fonksiyon için birinci, ikinci, üçüncü ve dördüncü dereceden komşuluklar tanımlanmıştır.

Teorem-6.1: *İki terimin birinci dereceden komşu olabilmesi için gerek ve yeter koşul bunlara karşı düşen fonksiyonların alan üzerinde komşu olmasıdır.*

Dört değişken için $2^k=2$ terimin (k=1) birinci dereceden komşu olması durumunda n-k=4-1=3 çarpanlı bir terim oluşur.

Örnek-6.8.

Şekil-6.4 de verilen *Karnaugh* diyagramını indirgeyiniz.

ab \ cd	00	01	11	10
00				
01		1	1	
11				
10				

Şekil-6.4. Karnaugh diyagramı.

Bu örneğin çözümünde öncelikle komşuluğu mertebesi ve özellikleri belirlenmelidir. Görüldüğü gibi lojik fonksiyon dört değişkenlidir ve iki terim birbirine komşudur. Bu durumda, n=4, k=1 olarak bulunur. Bu terimler birinci mertebeden komşuluk oluşturmakta olup, sonuçta üç çarpanlı bir terim elde edilecektir.

Buradaki terimler bir çerçeve içine alınarak gösterilirler ve ifadeleri lojik fonksiyonların aldıkları değerlere bağlı olarak verilir. Bu komşulukta a'nın tümleyeni, b ve d değerleri VE işlemine tabi tutularak ifade edilirken c değeri hem 0 hem de 1 değerini aldığı için düşer. Bu durumda indirgenmiş ifade,

$$f(a,b,c,d)=\overline{a}\cdot b\cdot d$$

olarak elde edilir. Daha önce de belirtildiği gibi üç çarpanlı bir terim oluşması beklenmektedir. Bir *Karnaugh* diyagramında birden fazla birinci derecede komşuluk bulunabilir. Örneğin,

ab \ cd	00	01	11	10
00	1			1
01		1	1	
11			1	
10				

şeklinde verilen *Karnaugh* Diyagramında tam üç tane birinci dereceden komşuluk bulunmaktadır. Bu durumda, her bir komşuluğun ifadesi belirlenir. Bu ifadeler VEYA, bir başka değişle "+" işlemiyle birleştirilerek indirgenmiş fonksiyon elde edilir. Sonuçta indirgenmiş ifade olarak aşağıdaki bağıntı elde edilir:

$$f(a,b,c,d)=\overline{a}\cdot b\cdot d+b\cdot c\cdot d+\overline{a}\cdot\overline{b}\cdot\overline{d}$$

Teorem-6.2: *Dört terimin ikinci dereceden komşu olabilmesi için gerek ve yeter koşul bunlara karşı düşen fonksiyonların alan üzerinde bir sıra şeklinde olması veya alan üstünde bir kare oluşturmasıdır.*

Dört değişken için $2^k=4$ terimin (k=2) ikinci dereceden komşu olması durumunda n-k=4-2=2 çarpanlı bir terim oluşur.

Örnek-6.9.

Şekil-6.5 de verilen *Karnaugh* Diyagramını indirgeyiniz.

ab \ cd	00	01	11	10
00				
01		1	1	
11		1	1	
10				

Şekil-6.5. *Karnaugh* diyagramı.

Bu örneğin çözümünde öncelikle komşuluğu mertebesi ve özellikleri belirlenmelidir. Görüldüğü gibi lojik fonksiyon dört değişkenlidir ve dört terim birbirine komşudur. Bu durumda, n=4, k=2 olarak bulunur. Bu terimler ikinci mertebeden komşuluk oluşturmakta olup, sonuçta iki çarpanlı bir terim elde edilecektir.

Buradaki terimler bir çerçeve içine alınarak gösterilirler ve ifadeleri lojik fonksiyonların aldıkları değerlere bağlı olarak verilir. Bu komşulukta b ve d değerleri VE işlemine tabi tutularak ifade edilirken a ve c değişkenleri hem 0 hem de 1 değerini aldığı için düşer. Bu durumda indirgenmiş ifade,

$$f(a,b,c,d)=b\cdot d$$

olarak elde edilir. Daha önce de belirtildiği gibi iki çarpanlı bir terim oluşması beklenmektedir. Bir *Karnaugh* diyagramında birden fazla ikinci derecede komşuluk bulunabilir. Örneğin,

ab \ cd	00	01	11	10
00	1			
01	1	1		
11	1	1		
10	1			

şeklinde verilen *Karnaugh* Diyagramında iki tane ikinci dereceden komşuluk bulunmaktadır. Bu durumda, her bir komşuluğun ifadesi belirlenir. Bu ifadeler VEYA, bir başka değişle "+" işlemiyle birleştirilerek indirgenmiş fonksiyon elde edilir. Sonuçta indirgenmiş ifade olarak,

$$f(a,b,c,d) = \overline{c} \cdot \overline{d} + b \cdot \overline{c}$$

elde edilir. Bir başka örnek ise,

ab \ cd	00	01	11	10
00	1			1
01				
11	1			1
10	1			1

şeklinde verilebilir ve bu *Karnaugh* Diyagramında da iki tane ikinci dereceden komşuluk bulunmaktadır. Bu durumda, her bir komşuluğun ifadesi belirlenir. Bu ifadeler "VEYA", bir başka değişle "+" işlemiyle birleştirilerek indirgenmiş fonksiyon elde edilir. Sonuçta indirgenmiş ifade olarak,

$$f(a,b,c,d) = \overline{b} \cdot \overline{d} + a \cdot \overline{d}$$

elde edilir.

Teorem-6.3. *Sekiz terimin üçüncü dereceden komşu olabilmesi için gerek ve yeter koşul bunlara karşı düşen fonksiyonların alan üzerinde komşu iki sıra oluşturmasıdır (İki değişkenli fonksiyonlarda mevcut değildir).*

Sekiz değişken için $2^k = 8$ terimin (k=3) ikinci dereceden komşu olması durumunda n-k=4-3=1 diğer bir deyişle tek bir terim oluşur.

Örnek-6.10.

Şekil-6.6 da verilen *Karnaugh* Diyagramını indirgeyiniz.

Bu örneğin çözümünde öncelikle komşuluğu mertebesi ve özellikleri belirlenmelidir. Görüldüğü gibi lojik fonksiyon dört değişkenlidir ve sekiz terim birbirine komşudur. Bu durumda, n=4, k=3 olarak bulunur. Bu terimler üçüncü mertebeden komşuluk oluşturmakta olup, sonuçta tek bir terim elde edilecektir.

ab \ cd	00	01	11	10
00		1	1	
01		1	1	
11		1	1	
10		1	1	

Şekil-6.6. ***Karnaugh*** **diyagramı.**

Buradaki terimler bir çerçeve içine alınarak gösterilirler ve ifadeleri lojik fonksiyonların aldıkları değerlere bağlı olarak verilir. Bu komşulukta sadece d değeri 1 değerini alır diğer bütün değişkenler 0 ve 1 değerlerinin ikisini birden aldıklarından dikkate alınmazlar. Bu durumda indirgenmiş ifade,

$$f(a,b,c,d) = d$$

olarak elde edilir. Daha önce de belirtildiği gibi iki çarpanlı bir terim oluşması beklenmektedir. Bir *Karnaugh* diyagramında birden fazla ikinci derecede komşuluk bulunabilir. Örneğin,

ab \ cd	00	01	11	10
00	1	1	1	1
01	1			1
11	1			1
10	1	1	1	1

şeklinde verilen *Karnaugh* Diyagramında iki tane ikinci dereceden komşuluk bulunmaktadır. Bu durumda, her bir komşuluğun ifadesi belirlenir. Bu ifadeler "VEYA", bir başka değişle "+" işlemiyle birleştirilerek indirgenmiş fonksiyon elde edilir. Sonuçta indirgenmiş ifade olarak,

$$f(a,b,c,d)=\overline{d}+\overline{b}$$

elde edilir.

Teorem-6.4. *Onaltı terimin dördüncü dereceden komşu olabilmesi için gerek ve yeter koşul bunlara karşı düşen fonksiyonların bütün alanı kaplıyor olması gerekmektedir (Üç değişkenli fonksiyonlarda mevcut değildir).*

Örnek 6.11.

$f(a,b,c,d)=\overline{a}\cdot\overline{b}\cdot\overline{c}\cdot\overline{d}+a\cdot\overline{b}\cdot\overline{c}\cdot\overline{d}+a\cdot b\cdot\overline{c}\cdot d+a\cdot b\cdot c\cdot d+\overline{a}\cdot\overline{b}\cdot c\cdot\overline{d}+a\cdot\overline{b}\cdot c\cdot\overline{d}$ şeklinde minimum terimler kanonik biçimiyle verilen fonksiyona ilişkin doğruluk tablosunun oluşturulması ve bu fonksiyonun *Karnaugh* diyagramı yardımıyla indirgenmesi örnek olarak seçilsin. İlk olarak alınan fonksiyona ilişkin doğruluk tablosu,

a	b	c	d	f(a,b,c,d)
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

şeklinde oluşturulur. Bu aşamadan sonra *Karnaugh* diyagramına geçilir ve oluşan komşuluklar göz önünde bulundurularak ifade indirgenir. Bu durumda *Karnaugh* diyagramı,

ab \ cd	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	1	1	0
10	1	0	0	1

şeklinde elde edilir ve ortada yer alan iki terim $a\cdot b\cdot d$ ifadesine, kenarlardaki dört terim ise $\overline{b}\cdot\overline{d}$ ifadesine indirgenir. Sonuç olarak,

$$f(a,b,c,d)=a\cdot b\cdot d+\overline{b}\cdot\overline{d}$$

indirgenmiş ifadesi elde edilir.

Örnek-6.12.

$f(a,b,c,d)=\bar{a}\cdot\bar{b}\cdot\bar{c}+\bar{b}\cdot c\cdot\bar{d}+\bar{a}\cdot b\cdot c\cdot\bar{d}+a\cdot\bar{b}\cdot\bar{c}$ şeklinde verilen fonksiyonun *Karnaugh* Diyagramı yardımıyla indirgeyiniz.

Bir lojik fonksiyon ifadesi verildikten sonra bu ifadeyi doğruluk tablosu oluşturmadan *Karnaugh* Diyagramına aktarmak mümkündür. Lojik değişkenlerin aldıkları değerlere bakarak *Karnaugh* diyagramı oluşturulur ve bundan sonra komşuluklar belirlenerek indirgeme işlemine geçilir.

Karnaugh Diyagramı oluşturulurken, minimum terimler kanonik biçiminde olduğu gibi bütün değişkenlerin ifadede bulunması istenir. Örneğin, $f(a,b,c,d)$ fonksiyonundun ilk terimi olan $\bar{a}\cdot\bar{b}\cdot\bar{c}$ aslında $\bar{a}\cdot\bar{b}\cdot\bar{c}\cdot(d+\bar{d})$ şeklinde ifade edilebilir. Bu durumda $(d+\bar{d})=1$ olduğundan fonksiyona bir etkisi yoktur. Ancak bu gösterilim *Karnaugh* Diyagramına geçiş açısından kolaylık sağlar. Diğer terimler de aynı şekilde değerlendirilirse,

ab \ cd	00	01	11	10
00	1	1	0	1
01	0	0	0	1
11	0	0	0	0
10	1	1	0	1

şeklinde *Karnaugh* Diyagramı elde edilir ve komşuluklar,

ab \ cd	00	01	11	10
00	1	1	0	1
01	0	0	0	1
11	0	0	0	0
10	1	1	0	1

şeklinde belirlenerek, indirgenmiş lojik fonksiyon,

$$f(a,b,c,d)=\bar{b}\cdot\bar{d}+\bar{b}\cdot\bar{c}+\bar{a}\cdot c\cdot\bar{d}$$

olarak elde edilir. Aynı indirgeme işlemi 0'ların bulunduğu terimler üzerinde komşuluklar belirlenerek de gerçeklenebilir. Ancak bu durumda toplamlar çarpımı şeklinde bir ifade bulunacaktır. 0'lar seçilerek oluşturulan *Karnaugh* diyagramında,

ab \ cd	00	01	11	10
00	1	1	0	1
01	0	0	0	1
11	0	0	0	0
10	1	1	0	1

şeklinde komşuluklar belirlenir ve indirgenmiş lojik fonksiyon bu durumda,

$$f(a,b,c,d)=[\bar{c}+\bar{d}]\cdot[\bar{a}+\bar{b}]\cdot[c+\bar{b}]$$

olarak bulunur.

Örnek-6.13.

$f(a,b,c,d)=\sum(0,3,4,5,6,7,13,14)$ şeklinde Minimum Terimlerin Toplamı şeklinde verilen fonksiyonun *Karnaugh* diyagramı yardımıyla indirgenmesi örneği incelenecektir.

Bir lojik fonksiyon ifadesi verildikten sonra bu ifadeyi doğruluk tablosu oluşturmadan *Karnaugh* diyagramına aktarmak mümkündür. Lojik değişkenlerin aldıkları değerlere bakarak *Karnaugh* diyagramı oluşturulur ve bundan sonra komşuluklar belirlenerek indirgeme işlemine geçilir. Bu durumda minimum terimlerden yararlanarak,

ab \ cd	00	01	11	10
00	1	0	1	0
01	1	1	1	1
11	0	1	0	1
10	0	0	0	0

şeklinde *Karnaugh* diyagramı elde edilir ve komşuluklar,

ab \ cd	00	01	11	10
00	1	0	1	0
01	1	1	1	1
11	0	1	0	1
10	0	0	0	0

şeklinde belirlenerek, indirgenmiş lojik fonksiyon aşağıdaki gibi elde edilir:

$$f(a,b,c,d)=\overline{a}\cdot\overline{c}\cdot\overline{d}+b\cdot\overline{c}\cdot d+\overline{a}\cdot c\cdot d+b\cdot c\cdot\overline{d}$$

Bu örneğin bir özelliği, mevcut olan en büyük komşuluğun değil de bütün terimleri kapsayan komşulukların seçilmiş olmasıdır. Gerçekten de dikkat edilirse yatay olarak yerleşen dört adet komşu, ikinci dereceden komşuluk oluşturmalarına rağmen dikkate alınmamış bunun yerine dört tane birinci dereceden komşuluk seçilmiş, bu komşuluklar bütün terimleri kapsadığı için ikinci dereceden komşuluk oluşturan terimler dikkate alınmamıştır.

Örnek-6.14.

$f(a,b,c)=\sum(0,1,4,5)$ şeklinde minimum terimlerin toplamı şeklinde verilen fonksiyonun üç değişkenli *Karnaugh* Diyagramı yardımıyla indirgenmesi örneği incelenecektir.

Öncelikle bu ifade üç değişkenli *Karnaugh* diyagramına aktarılır. Sonra da komşuluklar belirlenerek indirgeme işlemine geçilir. Minimum terimlerden yararlanarak üç değişkenli *Karnaugh* diyagramı aşağıda iki farklı biçimde ifade edilmiş ve komşuluklar her iki diyagramda da belirlenmiştir.

a \ bc	00	01	11	10
0	1	1	0	0
1	1	1	0	0

ab \ c	0	1
00	1	1
01	0	0
11	0	0
10	1	1

Bu aşamadan sonra, komşuluklardan yararlanarak fonksiyonun indirgenmiş ifadesini belirlenmelidir; ifade aşağıdaki gibi bulunur:

$$f(a,b,c)=\overline{b}$$

6.3. *Quine-McCluskey*/Tablo Yöntemi

Tablo yöntemiyle indirgeme yapılması "*algoritmik ifadeye daha yatkındır*" denilebilir; lojik ifadenin var olan terimleri, içerdikleri 1'lerin sayısına göre sınıflanıp bir tablo üzerine gruplanarak yerleştirilir. Daha sonra, gruplar arasındaki ilişkilere bakılarak, aralarında ilişki olup sadece bir değişkenle ilişkisi bozuk olan terimlerde ilişkiyi bozan değişkenler ayıklanarak terimler yalınlaştırılır; aralarında hiç ilişki olmayan terimler ise birbiriyle işleme sokulmaz. Bu işlem, tüm terimlerin aralarında ilişki kalmayana kadar tekrarlanır; indirgenmiş lojik ifade tablo üzerinde işleme sokulmamış olan terimlerden elde edilir.

Tablo yöntemi, diğer adıyla kendisini geliştiren bilim adamlarına atfen verilmiş ismiyle *Quine-McCluskey* yöntemi olarak da bilinir; ilk olarak *Quine* tarafından algoritmik hale getirilmiş olup daha sonra *McCluskey* tarafından geliştirilmiştir. Tablo yöntemi, lojik ifadedeki değişken sayısı arttığında diğer indirgeme yöntemlerine göre avantaja sahiptir; algoritmik bir ifadeye sahip olduğu için bilgisayar ortamında programı yapılarak çok değişkenli lojik ifadelerin bile indirgenmesi yapılabilir. Değişken sayısı arttığında görüşe dayanarak indirgeme imkansız hale gelir; *Karnaugh* diyagramıyla indirgemede de değişkenler arasındaki komşuluk ilişkisinin görülmesi zorlaşır. Lojik ifadedeki değişken sayısı 2, 3 ise görüşe; 3, 4 ise

Karnaugh veya tablo yöntemlerine; 4, 5 , 6 veya daha fazla ise tablo yöntemine indirgeme yapmak uygundur.

Quine-McCluskey yöntemiyle indirgeme için yapılması gerekenler Örnek-6.15 de adım adım açıklanmıştır. Bu örnekte 4 değişken vardır ve Örnek-6.17 de *Karnaugh* yöntemiyle indirgenen lojik ifadenin benzeridir. Örnek-6.16 da ise 3 değişkenli bir lojik ifadenin indirgenmesi yapılmıştır. Örneğe dikkatlice bakılırsa, ifadedeki tüm terimlerin sonucu 1 yapacak şekilde sadeleştiği görülür. Böyle bir örneğe uygulamada pek karşılaşılmaz; ancak, *Quine-McCluskey* yöntemini açıklamak için her durumu gösteren güzel bir örnektir.

Örnek-6.15.

$f(a, b, c, d) = \Sigma_1(3, 7, 11, 12, 13, 14, 15)$ şeklinde verilen lojik ifadeyi tablo/*Quine-McCluskey* yöntemiyle indirgeyerek yalın lojik ifadeyi bulunuz.

Quine-McCluskey yöntemiyle indirgeme yapmak için yapılması gerekenler adım adım şöyledir:

1. adım: İlk adım, terimleri (mintermlerin) sahip oldukları birlerin sayısına göre sıfır tane 1, bir tane 1, iki tane 1, üç tane 1 ve dört tane 1 içerenler şeklinde gruplamaktır. Bunun için lojik ifadeye ait mintermlerin ikili tabandaki karşılıkları ve bunların içerdikleri lojik 1'lerin sayısı tespit edilir. Verilen örnek için mintermler, onların ikili tabanda karşılıkları ve sahip oldukları birlerin sayısı şöyledir:

minterm	ikili tabanda	lojik 1'lerin sayısı
3	0 0 1 1	2 tane
7	0 1 1 1	3 tane
11	1 0 1 1	3 tane
12	1 1 0 0	2 tane
13	1 1 0 1	3 tane
14	1 1 1 0	3 tane
15	1 1 1 1	4 tane

Yukarıdaki tabloya göre mintermler Tablo-6.4 de 1. adım sütunundaki gibi gruplanır; görüldüğü gibi tablonun en üstünde en az sayıda lojik 1'e sahip olan 3 ve 12 mintermleri; daha sonra sırasıyla üç tane lojik 1'e sahip olan 7, 11, 13 ve 14 mintermleri ve dört tane lojik 1'e sahip olan 15 mintermi yer almıştır. Tablodaki yerleştirmeye göre iki adet lojik 1 içeren grup üç tane lojik 1 içeren gruba alttan komşudur; benzer şekilde, üç tane lojik 1 içeren grup iki tane içerene üstten, 4 tane içerene alttan komşudur; dört tane lojik 1 içeren grup da üç tane içerene üstten komşudur.

Tablo-6.4. *Quine McCluskey*/Tablo Yöntemi ile indirgeme.

1. Adım						2. Adım						3. Adım					
	a	*b*	*c*	*d*			*a*	*b*	*c*	*d*			*a*	*b*	*c*	*d*	
3	0	0	1	1	✓	3,7	0	-	1	1	✓	3,7,11,15	-	-	1	1	
12	1	1	0	0	✓	3,11	-	0	1	1	✓	3,11,7,15	-	-	1	1	
7	0	1	1	1	✓	12,13	1	1	0	-		12,13,14,15	1	1	-	-	
11	1	0	1	1	✓	12,14	1	1	-	0		12,14,13,15	1	1	-	-	
13	1	1	0	1	✓	7,15	-	1	1	1	✓						
14	1	1	1	0	✓	11,15	1	-	1	1	✓						
15	1	1	1	1	✓	13,15	1	1	-	1							
						14,15	1	1	1	-							

2. Adım: İkinci adımda, komşu gruplar arasındaki terimlerde ilişki aranır; şöyle ki, eğer birbirine komşu olan gruba ait iki minterm arasında, bitlerin sıralanmasında bir bitlik fark varsa, onlar arasında ilişki olduğu varsayılır. Bu tanıma, Tablo-6.4 İKİNCİ adım sütunundan görüleceği üzere 3 ile 7, 3 ile 11, 12 ile 13, 12 ile 14 arasında ilişki vardır; benzer şekilde üç tane lojik 1 ve dört lojik 1 içeren gruplar arasında da 7 ile 15, 11, 15, 13 ile 15 ve 14 ile 15 arasında ilişki vardır. İkinci adımda komşu gruplara ait terimler arasında ilişkiyi bozmaya çalışan değişken, yerine - karakteri koyularak yok edilir; benzer olan bitler olduğu gibi bırakılır. Bu koşullar tüm *mintermler* üzerine uygulandığında Tablo-6.4 de ikinci adım sütunundaki gibi bir sonuç elde edilir. (aynı numaralara ait terim-çiftleri oluşursa bir tanesi alınır!)

3. Adım: Üçüncü adımda, ikinci adımda elde edilen sütunda bulunan terim-çiftleri arasında yine ilişki aranır. Buradaki ilişkinin tanımı şöyledir: ❶ terim-çiftleri komşu gruplar içerisinde olmalıdır, ❷ ilişkisi olan terim-çiftlerinin yok edilmiş değişkeni aynı hizada olmalıdır, ❸ yalnızca bir tane bit'i farklı olmalıdır. Bu koşullar altında, örneğin, 3,7 terim-çiftiyle 11,15 terim-çifti arasında ilişki vardır. Çünkü her ikisinde de *b* değişkeni yok edilmiş olup *a* değişkeni farklılık göstermektedir; *b* ve *c* değişkenleriyse aynıdır. Ancak 3,7 terim-çiftiyle 14,15 terim-çifti arasında ilişki yoktur. Çünkü, yok edilen değişkeni (- karakteri) farklı hizada, birden çok farklı bit değerleri var...

Yukarıda açıklanan ilişki uyarınca Tablo-6.4 de 3. sütundaki gibi terim-dörtlüleri elde edilir. Burada komşu grup kalmadığı için tablo sonuna gelinilmiş demektir. (aynı numaralara ait terim-dörtlüleri oluşursa bir tanesi alınır!)

4. Adım: Dördüncü adımda, ilişkilendirme sürecinde ilişkiye sokulamayan terim, terim-çiftleri ve terim-dörtlüleri ele alınarak indirgenmiş lojik ifade bulunur. Tablo-6.4 de 1, 2 ve 3. adımlara ait sütunlara bakıldığında, sadece, 3.adım sütununda

terim-dörtlülerinde ilişki bulunamamıştır. Bu durumda indirgenmiş lojik ifade aşağıdaki gibi bulunur:

$$f(a, b, c, d) = a\,b + c\,d$$

Örnek-6.16.

$f(a, b, c) = \Sigma_1(0, 1, 2, 3, 4, 5, 6, 7)$ şeklinde verilen üç değişkenli lojik ifadeyi tablo/*Quine-McCluskey* yöntemini kullanarak indirgeyiniz. Sonucun ne olacağını görüşe dayanarak görebiliyor musunuz?

Evet, sonucun ne olacağı görülebilmelidir. Çünkü üç değişkenli bir lojik ifade de 8 değişik *minterm* vardır ve ifadede bu *mintermlerin* hepsi bulunmaktadır. Bir lojik ifadede *mintermlerin* hepsi varsa sonuç daima lojik 1 değerindedir. Dolayısıyla tablo/*Quine-McCluskey* yöntemiyle indirgemede sonuç lojik 1 olmalıdır. Örnek-6.15 de yapılan adımlara benzer adımlar burada da yapılmalıdır.

1. Adım: İlk adım *mintermlerin* sahip oldukları birlerin sayısına göre sıfır tane 1, bir tane 1, iki tane 1, üç tane 1 ve dört tane 1 içerenler şeklinde gruplamaktır. Verilen örnek için *mintermler*, onların ikili tabanda karşılıkları ve sahip oldukları birlerin sayısı şöyledir:

minterm	ikili tabanda	lojik 1'lerin sayısı	Grup No
0	0 0 0	0 tane	0
1	0 0 1	1 tane	1
2	0 1 0	1 tane	1
3	0 1 1	2 tane	2
4	1 0 0	1 tane	1
5	1 0 1	1 tane	1
6	1 1 0	2 tane	2
7	1 1 1	3 tane	3

Yukarıdaki tabloya göre *mintermler* Tablo-6.5 de 1.adım sütunundaki gibi gruplanır; görüldüğü gibi tablonun en üstünde sıfır tane lojik 1'e sahip olan 0 *mintermi*; daha sonra sırasıyla bir tane lojik 1'e sahip olan 1, 2 ve 4 *mintermleri*, iki tane lojik 1'e sahip olan 3, 5 ve 6 *mintermleri* ve üç tane lojik 1'e sahip olan 7 *mintermi* yer almıştır. Grup 0, grup 1'e ; grup 1, grup 2'ye; grup 2, grup 3'e komşudur.

Tablo-6.5. ***Quine McCluskey*/Tablo Yöntemi ile İndirgeme.**

1. Adım					2. Adım					3. Adım					4. Adım				
	a	*b*	*c*			*a*	*b*	*c*			*a*	*b*	*c*			*a*	*b*	*c*	
0	0	0	0	✓	0,1	0	0	-	✓	0,1,2,3	0	-	-	✓	0,1,2,3,4,5,6,7	-	-	-	
1	0	0	1	✓	0,2	0	-	0	✓	0,1,4,5	-	0	-	✓	0,1,4,5,2,3,6,7	-	-	-	
2	0	1	0	✓	0,4	-	0	0	✓	0,2,1,3	0	-	-	✓	0,2,4,6,1,3,5,7	-	-	-	
4	1	0	0	✓	1,3	0	-	1	✓	0,2,4,6	-	-	0	✓					
3	0	1	1	✓	1,5	-	0	1	✓	0,4,1,5	-	0	-	✓					
5	1	0	1	✓	2,3	0	1	-	✓	0,4,2,6	-	-	0	✓					
6	1	1	0	✓	2,6	-	1	0	✓	1,3,5,7	-	-	1	✓					
7	1	1	1	✓	4,5	1	0	-	✓	1,5,3,7	-	-	1	✓					
					4,6	1	-	0	✓	2,3,6,7	-	1	-	✓					
					3,7	-	1	1	✓	2,6,3,7	-	1	-	✓					
					5,7	1	-	1	✓	4,5,6,7	1	-	-	✓					
					6,7	1	1	-	✓	4,6,5,7	1	-	-	✓					

2. Adım: İkinci adımda, komşu gruplar arasındaki terimlerde ilişki aranır; şöyle ki, eğer birbirine komşu olan gruba ait iki *minterm* arasında, bitlerin sıralanmasında bir bitlik fark varsa, onlar arasında ilişki olduğu varsayılır. Bu tanıma, Tablo-6.5. 2. adım sütunundan görüleceği üzere grup 0 ile grup 1'de bulunan *mintermlerden* 0 ile 1, 0 ile 2 ve 0 ile 4 arasında ilişki vardır; benzer şekilde grup 1 ile grup 2'de bulunan *mintermlerden* 1 ile 3, 1 ile 5, 2 ile 3, 2 ile 6, 4 ile 5 ve 4 ile 6 arasında; grup 2 ile grup 3 te bulunan *mintermlerden* 3 ile 7, 5 ile 7 ve 6 ile 7 arasında ilişki vardır. İkinci adımda komşu gruplara ait terimler arasında ilişkiyi bozmaya çalışan değişken, yerine – karakteri koyularak yok edilir; benzer olan bitler olduğu gibi bırakılır. Bu koşullar tüm *mintermler* üzerine uygulandığında Tablo-6.5 de ikinci adım sütunundaki gibi bir sonuç elde edilir. (aynı numaralara ait terim-çiftleri oluşursa bir tanesi alınır!)

3. Adım: Üçüncü adımda, ikinci adımda elde edilen sütunda bulunan terim-çiftleri arasında yine ilişki aranır. Buradaki ilişkinin tanımı şöyledir: ❶ terim-çiftleri komşu gruplar içerisinde olmalıdır ❷ ilişkisi olan terim-çiftlerinin yok edilmiş değişkeni aynı hizada olmalıdır ❸ yalnızca bir tane bit'i farklı olmalıdır. Bu koşullar altında, örneğin, 0,1 terim-çiftiyle 2, 3 terim-çifti arasında ilişki vardır. Çünkü her ikisinde de *c* değişkeni yok edilmiş olup *b* değişkeni farklılık göstermektedir; *a* değişkeniyse aynıdır. Ancak 0,1 terim-çiftiyle 2,6 terim-çifti arasında ilişki yoktur. Çünkü, yok edilen değişkeni (- karakteri) farklı hizada, birden çok farklı bit değerleri var...

Yukarıda açıklanan ilişki uyarınca Tablo-6.5 üçüncü sütundaki gibi terim-dörtlüleri elde edilir. Aynı sayılardan oluşan terim-dörtlüleri birden fazlaysa bir tanesi alınır diğerleri önemsenmez. Burada da 0, 1, 2, 3 terim-dörtlüsüyle 0, 2, 1, 3 terim dörtlüsü aynıdır; yalnızca bir tanesi alınır. (Aynı numaralara ait terim-dörtlüleri oluşursa bir tanesi alınır!)

4. Adım: Dördüncü adımda da 3. adımdakine benzer ilişki aranarak terim-sekizlileri oluşturulur. Farklı grubun *mintermleri* olan ve yok edilmiş değişkenleri aynı hizada olan *mintermler* arasında sadece bir bitlik farka sahip terim-dörtlülerinden terim-sekizlileri elde edilir. Tablo-6.5 dördüncü sütundan görüleceği gibi terim-sekizlilerinin hepsinde yok edilmiş değişkenler kalmıştır. Bu durum, cebirsel ifade için herhangi bir değişkenin etkisi yoktur anlamına gelir.

5. Adım: Bir önceki adımda, terim, terim-çifti, terim-dörtlülerinin hepsi işleme sokulduğundan ve terim-sekizlilerinin hepsi yok edilmiş değişkenlere sahip olduğundan indirgenmiş lojik ifade,

$$f(a, b, c) = 1$$

şeklinde bulunur.

6.4. Eksik *Boole* İşlevleri

Şimdiye kadar ele alınan *Boole* fonksiyonlarının eksiksiz olduğu kabul edilmiştir. Diğer bir deyişle, "1" değerini almayan çıkış ifadeleri "0", "0" değerini almayan çıkış ifadeleri ise "1" değerini almak zorundadır. Bazı durumlarda bazı kombinasyonlara karşılık düşen çıkış fonksiyonları uygulamada hiçbir zaman gerçekleşmez. Bu durumda *Karnaugh* Diyagramında bu kombinasyonlara karşılık düşen çıkış fonksiyonları *keyfi* olarak nitelendirilirler ve ϕ ile gösterilirler. Bu çıkış değerlerinin bazıları "1" değerini, bazıları ise "0" değerini alabilir ve indirgeme bu şekilde gerçekleştirilir.

Örnek-6.17.

$f(a,b,c,d) = \sum(3,7,11,12,15) + \sum_{\phi}(0,10,13,14)$ şeklinde eksik terimlerle verilen fonksiyonun *Karnaugh* diyagramı yardımıyla indirgenmesine ilişkin örnek göz önünde bulundurulsun.

Bu ifade hem minimum terimleri hem de eksik terimleri içermektedir. Minimum terimler "1" değerini, eksik terimler ϕ değerini ve geri kalan ifadeler 0 değerini alır. Her ϕ değeri birbirinden bağımsızdır, bazıları 1 bazıları ise 0 olarak seçilebilir. Öncelikle fonksiyona ilişkin *Karnaugh* diyagramı oluşturulursa,

ab \ cd	00	01	11	10
00	ϕ	0	1	0
01	0	0	1	0
11	1	ϕ	1	ϕ
10	0	0	1	ϕ

elde edilir. Bu diyagramda yukarıdan aşağı sıralanan dört tane 1 ikinci mertebeden bir komşuluk oluşturur. Bunun haricinde 1100 hanesindeki 1 değeri mutlaka dikkate alınmalıdır. Bu ifade tek başına alınırsa dört terimli bir ifade oluşur. Bu durumda giriş sayısını azaltmak tasarım açısından önemli bir faktör olduğundan bu istenmez. Bu sırada bulunan iki tane keyfi değer 1, geriye kalan, köşelerdeki, keyfi değerler ise 0 değeri alacak şekilde seçilir. Kısaca göstermek gerekirse, önce sol tarafta verilen *Karnaugh* diyagramı oluşturulur ve daha sonra sağ taraftaki gibi uygun komşuluklar seçilir.

ab \ cd	00	01	11	10
00	ϕ=0	0	1	0
01	0	0	1	0
11	1	ϕ=1	1	ϕ=1
10	0	0	1	ϕ=0

ab \ cd	00	01	11	10
00	ϕ=0	0	1	0
01	0	0	1	0
11	1	ϕ=1	1	ϕ=1
10	0	0	1	ϕ=0

Sonuç olarak indirgenmiş lojik fonksiyon,

$$f(a,b,c,d) = c \cdot d + a \cdot b$$

olarak elde edilir.

6.5. Türetilmiş Kapılarla Temel Lojik Kapıların Yapılması

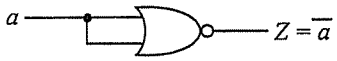
TVE ile TVEYA şeklinde aynı tür kapı elemanları kullanarak VE, VEYA ile TÜMLEME şeklindeki temel lojik kapıları gerçekleştirmek mümkündür. Bunun amacı, maliyeti en aza indirgemek için tercih edilen aynı tür kapılarla tasarım yapma yönteminin temel kapıların gerçeklenmesinde de kullanılabildiğini göstermektir. Maliyeti azaltmak için ilk olarak karmaşık fonksiyonlar indirgenir ve bu indirgenmiş ifadeler TVE, TVEYA şeklindeki aynı tür kapılar kullanılarak gerçeklenir. Bu sırada herhangi bir şekilde bir TÜMLEME kapısına ihtiyaç duyulursa bu eleman mevcut TVE ya da TVEYA kapılarıyla gerçeklenebilir. Bu ayrıtta TÜMLEME, VE ile VEYA temel kapı elemanlarının TVE ile TVEYA kapılarıyla gerçeklenmesi açıklanmıştır.

TÜMLEME Kapısı

Lojik fonksiyonu $f=\overline{a}$ şeklinde olan TÜMLEME işlemi TVE ya da TVEYA kapılarıyla gerçeklenebilir. Tasarımda iki girişli TVE kapısı kullanılabilir. Eğer her iki girişe de aynı değer verilirse TVE kapısı bir TÜMLEME kapısı gibi çalışır; $f=\overline{(a\cdot b)}=\overline{(a\cdot a)}=\overline{a}$. Bu durumda lojik fonksiyona ilişkin lojik şema;

şeklinde gerçeklenebilir.

Tasarımda ikinci alternatif olarak iki girişli TVEYA kapısı da kullanılabilir. TVEYA kapısının her iki girişine de aynı değer verilirse TVEYA kapısı bir TÜMLEME kapısı gibi çalışır: $f=\overline{(a+b)}=\overline{(a+a)}=\overline{a}$. Bu durumda lojik fonksiyon;

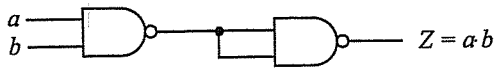


şeklinde gerçeklenebilir.

VE Kapısı

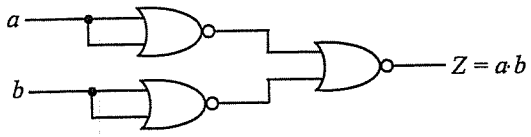
Lojik fonksiyonu $f=a\cdot b$ şeklinde olan VE işlemi TVE ya da TVEYA kapılarıyla gerçeklenebilir.

Tasarımda TVE kullanılırsa, $f=a\cdot b=\overline{\overline{a\cdot b}}=\overline{(\overline{a\cdot b})}$ şeklinde kolayca yazılabilir. Bu durumda TVE kapısına ilişkin uygun bir tasarım bir VE kapısı gibi çalışır. Bu durumda lojik fonksiyona ilişkin lojik şema,



şeklinde gerçeklenebilir.

Tasarım aşamasında TVEYA kullanılırsa, $f=a+b=\overline{\overline{a+b}}=\overline{(\overline{a}+\overline{b})}$ şeklinde yazılır. Bu durumda TVEYA kapısı bir VE kapısı gibi çalışır. Bu durumda lojik fonksiyon;

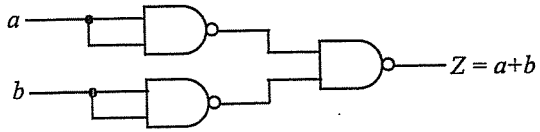


şeklinde gerçeklenebilir.

VEYA Kapısı

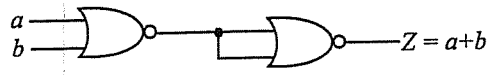
Lojik fonksiyonu $f=a+b$ şeklinde olan VEYA işlemi TVE ya da TVEYA kapılarıyla gerçeklenebilir.

Tasarımda TVE kullanılırsa, $f=a+b=\overline{\overline{a+b}}=\overline{(\overline{a}\cdot\overline{b})}$ şeklinde kolayca yazılabilir. Görüldüğü gibi, TVE kapılarının uygun bir kombinasyonu VEYA kapısı gibi çalışabilir. Bu durumda lojik fonksiyon;



şeklinde gerçeklenebilir.

Tasarım aşamasında TVEYA kullanılırsa, $f=a+b=\overline{\overline{a+b}}=\overline{(\overline{a+b})}$ şeklinde yazılır. Bu durumda TVEYA kapılarının uygun bir kombinasyonu bir VEYA kapısı gibi çalışır. Lojik fonksiyona ilişkin lojik şema ise,



şeklinde gerçeklenebilir.

6.6. İndirgenmiş İfadelerin Aynı Tür Kapılarla Gerçeklenmesi

Ayrıt 5.3.6 da, Kanonik biçimlerin lojik kapılarla gerçeklenmesi anlatılmıştı. Bu tasarımda eğer minimum terimler kanonik biçiminde bir ifade gerçekleniyorsa önce VE kapıları sonra bunların lojik toplamını alan bir VEYA kapısı, eğer maksimum terimler kanonik biçiminde bir ifade gerçekleniyorsa önce VEYA kapıları ve sonra bunların lojik çarpımını alan bir VE kapısı kullanılmıştı. Ancak bu tasarım yönteminde önemli, bir noktayı açıklamak gerekir; o da maliyet faktörünün göz önünde bulundurulmadığıdır. Halbuki kanonik biçimleri gerçekleştirmek yerine karmaşık lojik fonksiyonları indirgemek ve indirgenmiş fonksiyonları aynı tür kapı elemanları kullanarak tasarımın yapmak maliyet faktörünü en aza indirmek açısından gerekli ve son derece önemli bir tasarım adımıdır.

Bu ayrıtta çarpımlar toplamı veya toplamlar çarpımı şeklinde verilmiş veya indirgenerek elde edilmiş bir lojik ifadenin aynı tür kapı elemanlarıyla gerçeklenmesi açıklanmıştır. Bu amaçla öncelikle bu ifadeler üzerindeki birkaç dönüşüm işlemini vurgulamak yararlı olur. Çarpımlar toplamı şeklinde indirgenmiş bir ifade, sadece TVE ya da sadece TVEYA kapıları kullanılarak gerçeklenebilir.

Çarpımlar Toplamı – TVE Tasarımı

Çarpımlar toplamı şeklinde verilmiş bir ifadeyi sadece TVE kapılarıyla gerçeklemek için,

$$f = \sum \prod x_i$$

şeklinde verilmiş çarpımlar toplamı ifadesinin tümleyeni alınırsa,

$$\overline{f} = \overline{\sum \prod x_i} = \prod \overline{\prod x_i}$$

elde edilir. Bu ifadenin tekrar tümleyeni alınırsa f fonksiyonuna ulaşılır. Bu durumda,

$$\overline{(\overline{f})} = f = \overline{\prod \overline{\prod x_i}}$$

elde edilir. Bulunan bu ifade, tamamıyla çarpımların tümleyeni bir başka değişle TVE ifadelerinden oluşmuştur.

Örnek-6.18.

$f(a,b,c) = \overline{a}\cdot\overline{b}\cdot c + \overline{a}\cdot b\cdot c + a\cdot b\cdot\overline{c} + a\cdot b\cdot c$ şeklinde minimum terimler kanonik biçimiyle verilen lojik fonksiyonu indirgeyiniz ve sadece iki girişli TVE lojik kapı elemanları (74LS00) kullanılarak gerçekleştiriniz.

Problemin çözümü için önce fonksiyona ilişkin *Karnaugh* diyagramı,

a \ bc	00	01	11	10
0	0	1	1	0
1	0	0	1	1

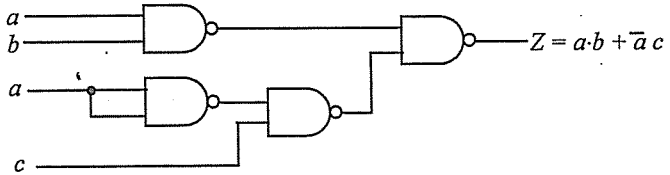
şeklinde oluşturulur ve indirgenir. Bu durumda fonksiyona ilişkin indirgenmiş ifade

$$f(a,b,c) = a\cdot b + \overline{a}\cdot c$$

şeklinde bulunur. Bu aşamadan sonra, bulunan ifade TVE kapılarıyla tasarım yapılacak biçimde düzenlenir ve

$$f(a,b,c) = a\cdot b + \overline{a}\cdot c$$
$$= \overline{\overline{(a\cdot b + \overline{a}\cdot c)}} = \overline{(\overline{a\cdot b})\cdot(\overline{\overline{a}\cdot c})}$$

elde edilir. İndirgenmiş fonksiyona ilişkin devre şeması Şekil-6.7 de verilmiştir.



Şekil-6.7. Sadece TVE lojik kapıları kullanılarak gerçeklenen devre.

Çarpımlar Toplamı – TVEYA Tasarımı

Çarpımlar toplamı şeklinde verilmiş bir ifadeyi sadece TVEYA kapılarıyla gerçeklemek için,

$$f = \sum \prod x_i$$

şeklinde verilmiş çarpımlar toplamı ifadesinde, $\prod x_i$ yerine $\overline{\sum \overline{x_i}}$ yazılabileceğini göstermek gerekir. $\prod x_i$ ifadesinin tümleyeni alınır ve bu tümleyen işlemi çarpma terimine ve x_i 'ye dağıtıldıktan sonra tekrar tümleyeni alınarak ilk ifade düzenlenirse,

$$\prod x_i = \overline{\sum \overline{x_i}} = \prod \overline{\overline{x_i}} = \prod x_i$$

elde edilir. Bu durumda $\prod x_i$ yerine $\overline{\sum \overline{x_i}}$ yazılabilir. Verilen f fonksiyonu ifadesinde bu yerine konursa,

$$f = \sum \prod x_i = \sum \overline{\sum \overline{x_i}}$$

bulunur. Ancak bu ifadenin ancak tümleyeni alınırsa sadece TVEYA kapılarını kullanmak mümkün olabilecektir. Sadece TVEYA kapılarıyla gerçeklenmiş fonksiyona ilişkin ifade,

$$\overline{f} = \overline{\sum \overline{\sum \overline{x_i}}}$$

olarak yazılır. Ancak bu durumda f yerine $\overline{f}$ gerçeklenir. f fonksiyonunu bulabilmek için çıkışın tümleyenini almak gerekir.

Örnek-6.19.

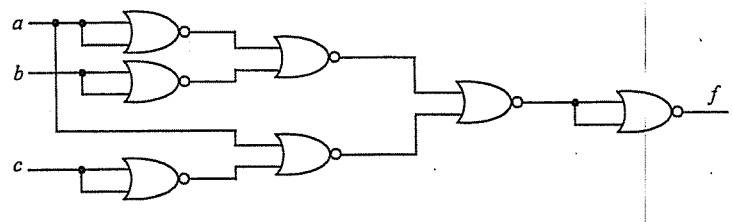
Bu örnekte, $f(a,b,c) = a \cdot b + \overline{a} \cdot c$ şeklinde verilen lojik fonksiyon sadece iki girişli TVEYA lojik kapı elemanları (74LS02) kullanılarak gerçekleştirilecektir. Bu durumda, yukarıdaki tanımlar gereği,

$$\begin{aligned} f(a,b,c) &= a \cdot b + \overline{a} \cdot c \\ &= \overline{(\overline{a} + \overline{b})} + \overline{\overline{(a + \overline{c})}} \\ &= \overline{(\overline{a} + \overline{b})} + \overline{(a + \overline{c})} \end{aligned}$$

yazılabilir. Ancak bu iki TVEYA kapısını birleştiren eleman olarak VEYA kapısı kullanıldığı için yeterli bir işlem değildir. Sonucunu bulmak için $\overline{f}$ gerçeklenir. f fonksiyonunu bulabilmek için çıkışın tümleyenini

$$f(a,b,c) = \overline{\overline{\overline{(\overline{a} + \overline{b})} + \overline{(a + \overline{c})}}}$$

şeklinde almak gerekir. Bu fonksiyona ilişkin devre şeması Şekil-6.8 de verilmiştir.



Şekil-6.8. Sadece TVEYA kapıları kullanılarak gerçeklenen devre.

Toplamlar Çarpımı – TVE Tasarımı:

Toplamlar çarpımı şeklinde verilmiş bir ifade sadece TVE veya sadece TVEYA kapılarıyla gerçeklenebilir. Toplamlar çarpımı şeklinde verilmiş bir ifadeyi sadece TVE kapılarıyla gerçeklemek için,

$$f = \prod \sum x_i$$

şeklinde verilmiş çarpımlar toplamı ifadesinde, $\sum x_i$ yerine $\overline{\prod \overline{x_i}}$ yazılabileceğini göstermek gerekir. $\sum x_i$ ifadesinin tümleyeni alınır ve bu tümleyen işlemi toplama terimine ve x_i'ye dağıtıldıktan sonra tekrar tümleyeni alınarak ilk ifade düzenlenirse,

$$\sum x_i = \overline{\prod \overline{x_i}} = \sum \overline{\overline{x_i}} = \sum x_i$$

elde edilir. Bu durumda $\sum x_i$ yerine $\overline{\prod \overline{x_i}}$ yazılabilir. Verilen f fonksiyonu ifadesinde bu yerine konursa,

$$f = \prod \sum x_i = \prod \overline{\prod \overline{x_i}}$$

bulunur. Ancak bu ifadenin tümleyeni alınırsa sadece TVE kapılarını kullanmak mümkün olabilecektir. Sadece TVE kapılarıyla gerçeklenmiş fonksiyona ilişkin ifade,

$$\overline{f} = \overline{\prod \overline{\prod \overline{x_i}}}$$

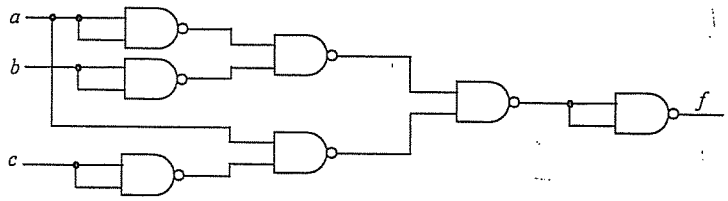
olarak yazılır. Ancak bu durumda f yerine $\overline{f}$ gerçeklenir. f fonksiyonunu bulabilmek için çıkışın tümleyenini almak gerekir.

Örnek-6.20.

Bu örnekte, $f(a,b,c)=[a+b]\cdot[\bar{a}+c]$ şeklinde verilen lojik fonksiyon sadece TVE lojik kapı elemanları (74LS00) kullanılarak gerçekleştirilmiştir. Bu durumda, öncelikle lojik fonksiyonun düzenlenmesi gerekir. VEYA kapıları TVE kapısı cinsinden ifade edilir ve arkasından iki kere tümleyeni alınırsa fonksiyon

$$\begin{aligned} f(a,b,c) &= [a+b]\cdot[\bar{a}+c] \\ &= (\overline{\overline{a}\cdot\overline{b}})\cdot(\overline{a\cdot\overline{c}}) \\ &= \overline{\overline{(\overline{a}\cdot\overline{b})\cdot(\overline{a\cdot\overline{c}})}} \end{aligned}$$

şeklinde elde edilir. Bu fonksiyona ilişkin devre şeması Şekil-6.9.'da verilmiştir.



Şekil-6.9. Sadece TVE kapıları kullanılarak gerçeklenen devre

Toplamlar Çarpımı – TVEYA Tasarımı

Toplamlar çarpımı şeklinde verilmiş bir ifadeyi sadece TVEYA kapılarıyla gerçeklemek için,

$$f=\prod\sum x_i$$

şeklinde verilmiş çarpımlar toplamı ifadesinin tümleyeni alınırsa,

$$\bar{f}=\overline{\prod\sum x_i}=\sum\overline{\sum x_i}$$

elde edilir. Bu ifadenin tekrar tümleyeni alınırsa f fonksiyonuna ulaşılır.

Bu durumda,

$$\overline{(\bar{f})}=f=\overline{\sum\overline{\sum x_i}}$$

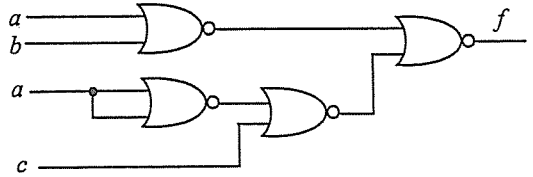
elde edilir. Bulunan bu ifade, tamamıyla çarpımların tümleyeni bir başka değişle TVEYA ifadelerinden oluşmuştur.

Örnek-6.21.

Bu örnekte, $f(a,b,c)=[a+b]\cdot[\bar{a}+c]$ şeklinde verilen lojik fonksiyon sadece TVEYA lojik kapı elemanları kullanılarak gerçekleştirilmiştir. Burada öncelikle lojik fonksiyonun düzenlenmesi gerekir. Fonksiyonun iki kere tümleyeni alınır ve düzenlenirse,

$$\begin{aligned} f(a,b,c) &= [a+b]\cdot[\bar{a}+c] \\ \overline{f(a,b,c)} &= \overline{[a+b]\cdot[\bar{a}+c]} \\ &= \overline{(a+b)}+\overline{(\bar{a}+c)} \\ \overline{\overline{f(a,b,c)}} = f(a,b,c) &= \overline{\overline{(a+b)}+\overline{(\bar{a}+c)}} \end{aligned}$$

elde edilir. Bu fonksiyona ilişkin devre şeması Şekil-6.10 da verilmiştir.



Şekil-6.10. Sadece TVEYA kapıları kullanılarak gerçeklenen devre

6.7. Özet

Bir Lojik devreye ait lojik fonksiyonun indirgenerek kullanılması maliyeti azaltır ve devredeki gecikme sürelerini en aza düşürür. Ayrıca bazı uygulamalarda hiç bir zaman gerçeklenmeyen çıkış değerleri bulunabilir ve bu değerler keyfi olarak yorumlanır. Bu uygulamalara ait lojik fonksiyonların indirgenmesi sırasında ortaya çıkacak keyfi değerler indirgenme işleminin daha kolay gerçeklenmesini sağlayabilir. Lojik devre tasarımcısı, özellikle kullanılacak devre elemanının seçimi sırasında maliyet faktörünü göz önünde bulundurmalıdır. Bu tasarımcının en önemli kriteridir. Problemin çözümü sırasında ortaya çıkabilecek bir keyfi değişken çözümün çok kolay olmasını sağlayabileceği için indirgeme işlemlerinde çok büyük önem taşır. Benzer şekilde, kullanılacak kapı tipi ve aynı tür kapı elemanının kullanılması da özellikle maliyetin belirlenmesi noktasında önemlidir ve iyi bir lojik devre tasarımcısı her iki noktayı da en iyi biçimde değerlendirmesini bilen kişidir.

6.8. Sorular

1. $f(a,b,c,d)=\bar{a}\cdot\bar{b}\cdot\bar{c}+\bar{a}\cdot b\cdot\bar{c}+\bar{a}\cdot b\cdot c+a\cdot b\cdot d$ şeklinde verilen lojik fonksiyonu *Karnaugh* diyagramıyla indirgeyiniz. İndirgenen fonksiyonu sadece iki girişli TVEYA (74LS02) kapıları kullanarak gerçekleyiniz.

2. Aşağıdaki lojik ifadeleri cebirsel olarak sadeleştiriniz.

   a) $f(u,x,v,y,z)=y\cdot z+u\cdot y+(y+z)\cdot(u+y)+(x+v+z)\cdot(x+u+v)\cdot(x+v+z)\cdot(x+u+v)$

   b) $f(a,b,c,d,e)=b\cdot c\cdot d+a\cdot b\cdot c\cdot e+a\cdot d+b\cdot c\cdot e+b\cdot c\cdot d$

   c) $f(a,b,c,d)=(b+c)\cdot(c+d)+a\cdot c+b\cdot c+(c+a)\cdot(c+b)$

3. $f(a,b,c,d)=\sum(5,7,9,10,11,13,14)+\sum_{\phi}(6,15)$ lojik fonksiyonunu Karnaugh diyagramıyla önce çarpımlar toplamı biçiminde, sonra da toplamlar çarpımı biçiminde indirgeyiniz. Daha basit olan indirgenmiş ifadeyi sadece TVEYA kapıları ile gerçekleyip devre şemasını çiziniz.

4. $f(x,y,w,z)=(w+x+y)\cdot(x+z)\cdot(w+x)$ şeklinde verilen fonksiyonun *Karnaugh* diyagramını bulunuz ve fonksiyonu indirgeyiniz.

5. a) Aşağıda verilen *Karnaugh* diyagramında değişkenlerin aldıkları değerler boş bırakılmıştır. *Karnaugh* diyagramının çarpımlar toplamı şeklindeki indirgenmiş ifadesi $f=\bar{a}\cdot\bar{c}$ olacak biçimde değişken değerlerini belirleyiniz.

ab \ cd	10	..	..	..
10	0	0	0	0
..	0	1	1	0
..	0	1	1	0
..	0	0	0	0

   b) Oluşturduğunuz özel tipteki diyagram ile $f=\bar{a}\cdot\bar{b}\cdot\bar{c}+\bar{a}\cdot b\cdot\bar{c}+\bar{a}\cdot b\cdot c+a\cdot b\cdot d$ fonksiyonunu indirgeyiniz.

   c) Aynı fonksiyonu klasik *Karnaugh* diyagramı ile indirgeyiniz.

6. Aşağıda verilen *Karnaugh* diyagramında değişkenlerin aldıkları değerler boş bırakılmıştır. *Karnaugh* diyagramının çarpımlar toplamı şeklindeki indirgenmiş ifadesi $f=\bar{a}\cdot\bar{c}$ olacak biçimde değişken değerlerini belirleyiniz.

ab \ cd	01	..	..	..
01	1	0	0	1
..	0	0	0	0
..	0	0	0	0
..	1	0	0	1

7. $f=w\cdot x\cdot y+y\cdot z+w\cdot y\cdot z+x\cdot y\cdot z$ ve

   $g=(w+x+y+z)\cdot(x+y+z)\cdot(w+y+z)$ fonksiyonları veriliyor.

   $F=f\cdot g$ fonksiyonunu çarpımlar toplamı ifadesini elde edecek şekilde indirgeyiniz. İndirgenen fonksiyonu sadece TVE kapıları kullanarak gerçekleştiriniz.

8. $f(a,b,c,d)=\bar{a}\cdot\bar{b}\cdot\bar{c}+\bar{a}\cdot b\cdot\bar{c}+\bar{a}\cdot b\cdot c+a\cdot b\cdot d$ şeklinde verilen lojik fonksiyonu *Karnaugh* diyagramı ile indirgeyiniz. İndirgenen fonksiyonu sadece iki girişli TVEYA kapıları kullanarak gerçekleştiriniz.

9. $f(a,b,c,d)=\sum(5,7,9,10,11,13,14)+\sum_{\phi}(6,15)$ lojik fonksiyonunu *Karnaugh* diyagramıyla önce çarpımlar toplamı biçiminde, sonra da toplamlar çarpımı biçiminde indirgeyiniz. Daha basit olan indirgenmiş ifadeyi sadece TVEYA kapıları ile gerçekleyip devre şemasını çiziniz.

10. $f(a,b,c,d)=\sum(1,3,5,7,8)+\sum_{\phi}(0,2,10)$ şeklinde verilen lojik fonksiyonu *Karnaugh* diyagramıyla,

    a) Çarpımlar toplamı şeklindeki ifadeyi,

    b) Toplamlar çarpımı şeklindeki ifadeyi bulacak biçimde indirgeyiniz.

    c) Çarpımlar toplamı şeklindeki ifadeyi, sadece iki girişli TVE (74LS00) kapıları kullanarak gerçekleyiniz.

11. $f(a,b,c,d)=\sum(0,9,10,12,13)+\sum_{\phi}(2,6,8,14)$ şeklinde bir lojik fonksiyon verilmiştir.

    a) Bu fonksiyonu, Çarpımlar toplamı şeklindeki ifadeyi bulacak biçimde *Karnaugh* diyagramı ile indirgeyiniz. İndirgenmiş ifadeyi sadece TVEYA kapıları kullanarak gerçekleyiniz.

    b) Toplamlar çarpımı şeklindeki ifadeyi bulacak biçimde indirgeyiniz. İndirgenmiş ifadeyi sadece TVE kapıları kullanarak gerçekleyiniz.

    c) Tasarım kriterleri açısından a) ve b) tasarımlarını karşılaştırınız. Hangi tasarımın neden daha uygun bir seçim olacağını açıklayınız.

12. Aşağıda verilen fonksiyonları indirgeyiniz.

a) $E_1 = y \cdot z + y \cdot z \cdot t + z \cdot t$

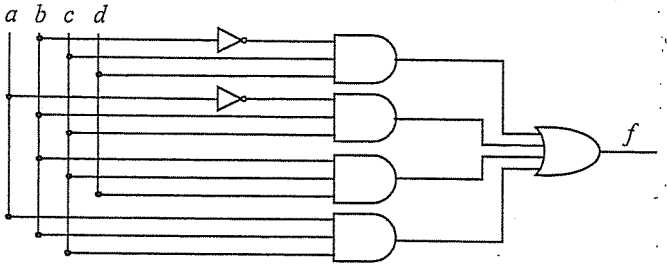
b) $E_2 = x \cdot \overline{y} \cdot z + x \cdot y + \overline{x} \cdot y \cdot z + x \cdot y \cdot z$

c) $E_3 = x \cdot y + \overline{x} \cdot y \cdot z + x \cdot y \cdot z$

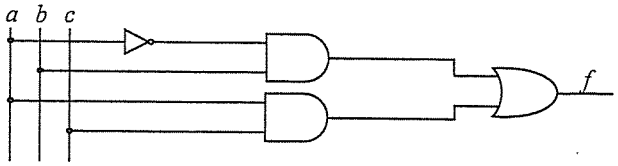
13. a) Aşağıda verilen devreye ilişkin f lojik ifadesini giriş değişkenleri cinsinden elde ediniz.

b) İfadeyi *Karnaugh* diyagramıyla indirgeyiniz.

c) İndirgenmiş ifadeyi sadece TVE kapıları ile gerçekleştiriniz.



14. Aşağıda bir lojik devre verilmiştir.



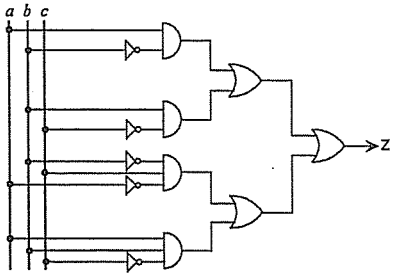
a) Devreye ilişkin çıkış fonksiyonu f'nin ifadesini a,b ve c değişkenleri cinsinden bulunuz.

b) Devreyi sadece iki girişli TVE kapılarıyla gerçekleştiriniz.

c) Devreyi sadece iki girişli TVEYA kapılarıyla gerçekleştiriniz.

d) Verilen devreyi, TVE ile TVEYA tasarımlarını, tasarım kriterleri açısından karşılaştırınız. Hangi tasarımın neden daha uygun bir seçim olacağını açıklayınız.

15. Aşağıda verilen devrenin çıkışını, girişler cinsinden ifade ediniz. Elde edilen fonksiyonu görüşe dayanarak indirgeyiniz. İndirgenen fonksiyonu sadece TVEYA kapıları kullanarak gerçekleyiniz.



16. a) Aşağıda doğruluk tablosu verilen lojik ifadeye ilişkin çarpımlar toplamı şeklinde indirgenmiş ifadeyi elde ediniz.

b) İfadeyi sadece iki girişli TVE kapılarıyla oluşturacak biçimde düzenleyiniz.

c) İfadeyi sadece iki girişli TVEYA kapılarıyla oluşturacak biçimde düzenleyiniz.

d) Eleman karmaşıklığı açısından hangi tür tasarımın daha uygun olduğunu bulunuz.

a	*b*	*c*	*Z*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

17. Aşağıdaki ifadenin tümleyenin alınız. Tümlenmiş ifadeyi en az sayıda değişken bulunduracak şekilde indirgeyiniz.

$$f = A \cdot B + C \cdot D + A \cdot C \cdot D + C \cdot D \cdot (\overline{A} \cdot B + A \cdot \overline{B}) + B \cdot D \cdot (\overline{A} \cdot \overline{C} + A \cdot C)$$

18. $f(x, y, z) = \sum 0(0, 1, 4, 9, 11, 13, 15) + \sum \phi(2, 3, 10, 12)$ şeklinde verilen fonksiyonun indirgenmiş şeklini elde ediniz.

7.

Kombinezonsal Devreler

Kombinezonsal devreler, çıkışı yalnızca o andaki giriş değerlerine bağlı olan ve bilinen kapıların biraraya getirilmesiyle oluşan lojik devrelerdir. Kombinezonsal devrelerde, devrenin geçmiş bilgileri veya durumları tutulmaz; çıkışlar doğrudan o andaki girişlerin fonksiyonudur. Dolayısıyla, yalın lojik ifadelerin gerçeklenmesi kombinezonsal devrelerle gerçekleştirilir. Bu bölümde kombinezonsal devrelerin temeli, devre elemanları ve çeşitli tasarım örnekleri aşağıdaki başlıklar altında ele alınmıştır:

7.1. Kombinezonsal Devreler
7.2. Kombinezonsal Devre Elemanları
7.3. Kombinezonsal Devre Tasarımı
7.4. Kombinezonsal Tümleşik Devreler
Aritmetik Toplama/Çıkarma Devreleri
Yarı Toplayıcı (*Half Adder*)
Tam Toplayıcı (*Full Adder*)
Yarı Çıkarıcı (*Half Subtractor*)
Tam Çıkarıcı (*Full Subtractor*)
Toplayıcı/Çıkarıcı (*Adder/Subtractor*)
Seçiciler (*Multiplexer*)
Dağıtıcılar (*Demultiplexer*)
Kod Çözücüler (*Decoder*)
Kodlayıcılar (*Encoder*)
ALU Tasarımı (*Arithmetic Logic Unit*)
7-Parçalı Gösterge (*7-Segment Display*)
7.5. Çıkış Bağlantı Şekilleri
3-Durumlu ve Açık Kollektörlü
7.6. Özet
7.7. Sorular

Bundan önceki bölümlerde, temel lojik işlemleri tanıtılmış ve bu işlemlerin görsel olarak ifade edilmesini sağlayan kümeler cebriyle *Venn* diyagramı gösterilimleri verilmiştir. Daha sonra lojik devrelerin temelini oluşturan Boole Cebri aksiyom ve teoremleri açıklanmış ve *Boole* Cebri'nin standart biçimleri olan minimum terimler kanonik biçimi ve maksimum terimler kanonik biçimi gösterilmiştir. Ayrıca, kanonik biçimler arasındaki dönüşüm kuralları ile kanonik biçimde verilen ifadelerin tek tip lojik kapı elemanlarıyla gerçeklenme şartları ele alınmıştır. Son olarak lojik fonksiyonların görüşe dayanarak ve *Karnaugh* diyagramları ile indirgenmesi ayrıntılı bir şekilde incelenmiş ve maliyet faktörünü azaltmak amacıyla indirgenmiş

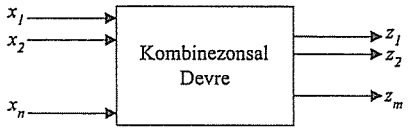
ifadelerin tek tür kapı elemanları ile gerçeklenmesi irdelenmiştir. Tüm bu bilgiler, bu ve bundan sonraki bölümler için gereklidir denilebilir.

Lojik devreler yapılarına göre, *kombinezonsal devreler* ve *ardışıl devreler* şeklinde iki sınıfa ayrılırlar. Kombinezonsal devrelerde çıkış işareti bulunurken sadece o andaki giriş değerleri dikkate alındığı için daha önceki çıkış değerlerini bilmeye ya da bir birimde saklamaya gerek yoktur. Bu yüzden, kombinezonsal devrelere aynı zamanda *belleksiz* veya *statik devreler* de denir. Buna karşılık ardışıl devrelerde önceki çıkış değerlerini bilmek gerektiği için bir bellek elemanına ihtiyaç vardır, bu yüzden ardışıl devrelere *bellekli* veya *dinamik devreler* olarak isimlendirilir.

7.1. Kombinezonsal Devreler

Bir lojik devrede, belirli bir andaki çıkışları, tamamen o andaki girişler belirliyor ise o devre kombinezonsal bir devredir. Sadece girişler çıkışı belirleyemiyor ve daha önceki giriş-çıkış işaretlerine ihtiyaç duyuluyorsa bu tür devrelere "*ardışıl devre*" adı verilmektedir. Bu bölümde kombinezonsal devreler anlatılacak ve kombinezonsal devre elemanları tanıtılacaktır. Bölüm 10, 11 ve 12'de ise ardışıl devreler incelenmiştir.

Bir kombinezonsal devrede çıkış işareti sadece o anki giriş işaretlerinden türetilmektedir. Kombinezonsal devreler Şekil-7.1 de verilen yapı ile ifade edilirler.



Şekil-7.1. Kombinezonsal devrelerin gösterilişi.

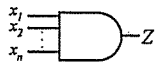
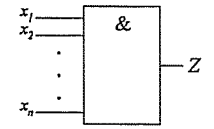
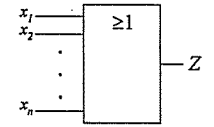
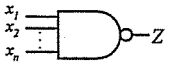
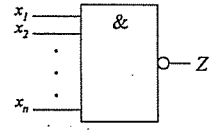
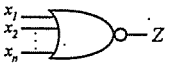
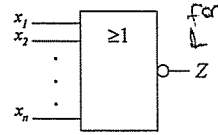
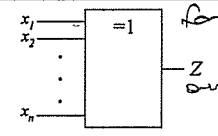
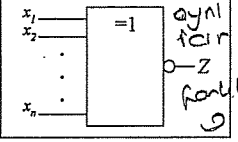
Kombinezonsal devre tasarımının gerçeklenebilmesi için ilk önce problemin sözle tanımlanması gerekmektedir. Bu aşamadan sonra, giriş ve çıkış değişkenleri belirlenir, isimlendirilir ve probleme ilişkin doğruluk tablosu oluşturulur. Daha sonra bilinen indirgeme yöntemlerinden biri kullanılarak her bir çıkış fonksiyonu indirgenir. Son olarak kullanılacak lojik devre elemanı belirlenir ve indirgenmiş çıkış fonksiyonu üzerinde değişiklikler yapılarak tasarım tamamlanır ve lojik devre şeması çizilir. Kombinezonsal devre tasarım yöntemi Ayrıt-7.3 de adım adım anlatılmıştır. Kombinezonsal devre tasarımında kullanılan kapı elemanları daha önce tanıtılmıştı. Bu elemanlara ilave olarak daha geniş çapta hazırlanmış elemanlar elektronik sektöründe kullanıldıkları için bu elemanların tanıtılması yararlı olur.

Kombinezonsal = hafızasız
Ardışıl = hafızalı

7.2. Kombinezonsal Devre Elemanları

Kombinezonsal devre tasarımında daha önceden tanıtılmış bulunan lojik kapı elemanları kullanılmaktadır. Bu elemanların ifadeleri ve lojik sembolleri hatırlatmak amacıyla Tablo-7.1 de gösterilmiştir (Bu tablo Bölüm 5 de daha önce verilmişti!).

Tablo-7.1. Kombinezonsal devreleri oluşturan lojik kapı elemanları.

Lojik İşlem	Lojik İfadesi	ANSI/IEEE-1973 Standardı	ANSI/IEEE-1984 Stadardı
VE (AND)	$Z = x_1 \cdot x_2 \cdot \ldots \cdot x_n$		
VEYA (OR)	$Z = x_1 + x_2 + \cdots + x_n$		
TVE (NAND)	Değillerin toplamı $Z = \overline{(x_1 \cdot x_2 \cdot \ldots \cdot x_n)}$		
TVEYA (NOR)	değillerin çarpımı $\cancel{(\bar{a}.b+a.\bar{b})}$ $Z = \overline{(x_1 + x_2 + \cdots + x_n)}$		 farklı durum ↳ 1 olanın
YA DA (XOR)	$Z = x_1 \oplus x_2 \oplus \cdots \oplus x_n$	$(\bar{a}.b+a.\bar{b})$	 farklı durum değ. Z aynı yapıs
EŞ DEĞERLİK TYA DA (XNOR)	$Z = \overline{(x_1 \oplus x_2 \oplus \cdots \oplus x_n)}$	$(a+\bar{b}).(\bar{a}+b)$ $(1+0)$	 aynı durum 1 farklı ise 0

7.3. Kombinezonsal Devre Tasarımı

Bu ayrıta kadar lojik devre elemanları tanıtıldı. Bu noktadan sonra yapılması gereken, bu elemanları kullanarak kombinezonsal devre tasarımı yapmaya ilişkin sistematik bir yöntemin tanıtılmasıdır. Kombinezonsal devre tasarımında kullanılabilecek yönteme ilişkin adımlar aşağıda sıralanmıştır:

1. Problem sözle tanıtılır.
2. Giriş ve çıkış değişkenlerinin sayısı belirlenir ve isimlendirilir.
3. Probleme ilişkin doğruluk tablosu oluşturulur.
4. Her bir çıkışa ait lojik fonksiyon, bilinen yöntemlerden biriyle (Görüşe dayanarak, *Karnaugh* Diyagramı *veya Quine McCluskey* yöntemi) indirgenir.
5. Kullanılacak lojik devre elemanları belirlenir; eğer tek tip kapılar kullanılacaksa indirgenmiş çıkış fonksiyonları düzenlenir, eğer başka tip bir eleman kullanılacaksa çalışma biçimine göre düzenlemeler yapılır.
6. Devreye ilişkin lojik devre şeması çizilir.

Bütün bu aşamalar adım adım gerçeklendikten sonra kombinezonsal devre tasarımı da tamamlanmış olacaktır. Kombinezonsal devre tasarımına ilişkin ayrıntılı örnekler aşağıda verilmiştir.

Örnek-7.1.

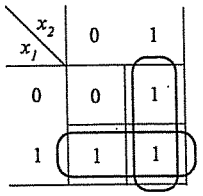
Bu örnekte, iki girişli, bir çıkışlı bir kombinezonsal devre tasarımı yapılmıştır. Girişlerin herikisi de sıfır ise devrenin çıkışında lojik 0, aksi durumda devrenin çıkışında lojik 1 olması istenmektedir (Problemin sözle tanımlanması kombinezonsal devre tasarımının başlangıç adımını teşkil eder ❶).

Devreye ilişkin girişleri x_1 ve x_2 ile devrenin çıkışını ise Z ile gösterilsin. Bu durumda iki girişli, bir çıkışlı bir devre elde edilmiş olur (Devrenin giriş/çıkış sayısı tespit edilir ve bu giriş/çıkışlara bir isim verilir ❷). Bundan sonra yapılması gereken, devreye ilişkin doğruluk tablosunun oluşturulmasıdır (❸). Tablo-7.1 de devreye ilişkin doğruluk diyagramı verilmiştir.

Tablo-7.1. Tasarlanacak devreye ilişkin doğruluk tablosu

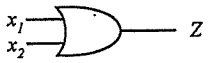
x_1	x_2	Z
0	0	0
0	1	1
1	0	1
1	1	1

Bu aşamadan sonra, çıkışa ilişkin ifade iki değişkenli *Karnaugh* diyagramı ile indirgenerek gerçekleştirilecek devredeki ifadeler en aza düşürülür. İndirgeme işlemi Şekil-7.1 de verilen *Karnaugh* diyagramı vasıtasıyla gerçeklenir (❹).



Şekil-7.1. Örnek-7.1.'e ilişkin Karnaugh diyagramı.

Şekil-7.1 de oluşturulan *Karnaugh* diyagramında yer alan iki terime ait indirgenmiş değerler yazılırsa devrenin çıkış fonksiyonu $Z = x_1 + x_2$ şeklinde elde edilir. Bu değerler tamamen indirgenmiş değerlerdir, daha da basitleştirmek mümkün değildir. Zaten bu yapının bir VEYA kapısı ile gerçekleştirilebileceği görülmektedir (❺). Son aşamada ise devre şeması çizilir. Devre sadece bir VEYA kapısı kullanılarak tasarlanmaktadır. Bu durumda Şekil-7.22 deki devre elde edilir.



Şekil-7.2. Tanımlanan problem için tasarlanan devre.

Örnek-7.2.

Bu örnekte, yine iki girişli, bir çıkışlı bir kombinezonsal devre tasarımı yapılmıştır. Girişler birbirine eşitse devrenin çıkışında lojik 1, girişler birbirinden farklıysa devrenin çıkışında lojik 0 olması istenmektedir (Problemin sözle tanımlanması kombinezonsal devre tasarımının başlangıç adımını teşkil eder ❶).

Devreye ilişkin girişleri x_1 ve x_2 ile devrenin çıkışını ise Z ile gösterilsin. Bu durumda iki girişli, bir çıkışlı bir devre elde edilmiş olur (Devrenin giriş/çıkış sayısı tespit edilir ve bu giriş/çıkışlara bir isim verilir ❷). Bundan sonra yapılması gereken, devreye ilişkin doğruluk tablosunun oluşturulmasıdır (❸). Tablo-7.2 de devreye ilişkin doğruluk tablosu verilmiştir.

Tablo-7.2. Tasarlanacak devreye ilişkin doğruluk tablosu.

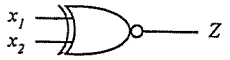
x_1	x_2	Z
0	0	1
0	1	0
1	0	0
1	1	1

Bu aşamadan sonra, çıkışa ilişkin ifade iki değişkenli *Karnaugh* diyagramı ile indirgenerek gerçekleştirilecek devredeki ifadeler en aza düşürülür. İndirgeme işlemi Şekil-7.3 de verilen *Karnaugh* diyagramı vasıtasıyla gerçeklenir (❹).

$x_1 \backslash x_2$	0	1
0	1	0
1	0	1

Şekil-7.3. Örnek-7.2 ye ilişkin *Karnaugh* diyagramı.

Şekil-7.3 de oluşturulan *Karnaugh* diyagramında yer alan iki terime ait indirgenmiş değerler yazılırsa devrenin çıkış fonksiyonu $Z = \overline{x_1} \cdot \overline{x_2} + x_1 \cdot x_2$ şeklinde elde edilir. Bu değerler tamamen indirgenmiş değerlerdir, daha da basitleştirmek mümkün değildir. Ancak kapı elemanları ile gerçekleme aşamasında mevcut kapı elemanları göz önünde bulundurulmalıdır. Bu durumda iki VE kapısı, bir VEYA kapısı ve iki de TÜMLEME kapısı kullanılarak devre gerçekleştirilmelidir veya tek tip kapı elemanları ile gerçekleme için fonksiyon üzerinde düzenleme yapılmalıdır. Ancak, durum böyle gözükse de lojik devre tasarımcısı bütün kapı elemanlarını gözönünde bulundurmalıdır. Bu durumda yukarıdaki fonksiyonu ifade eden lojik bir kapı bulunup bulunmadığı araştırmak tasarımcının *Eşdeğerlik* (TYA DA) kapısıyla problemi çözmesini sağlayacaktır (❺). Son aşamada ise devre şeması çizilir (❻). Devre sadece bir TYA DA kapısı kullanılarak tasarlanmaktadır. Bu durumda Şekil-7.4 deki devre elde edilir.



Şekil-7.4. Tanımlanan problem için tasarlanan devre.

Örnek-7.3.

Bu örnekte, A_1A_0 ve B_1B_0 şeklinde verilen iki bitlik iki sayının birbirine eşit, küçük veya büyük olduğunu bulan iki-bitlik Genlik Karşılaştırıcı devrenin tasarımı (❶) gerçeklenmiştir.

Devreye ilişkin girişler A_1A_0 ve B_1B_0 şeklinde zaten problemin içinde belirlenmişti. Çıkışları ise $t_1 \Rightarrow A>B$, $t_2 \Rightarrow A=B$ ve $t_3 \Rightarrow A<B$ olacak biçimde tanımlayabiliriz. Bu durumda dört girişli, üç çıkışlı bir devre oluşturulacağı görülebilir (❷). Bundan

sonra yapılması gereken, devreye ilişkin doğruluk tablosunun oluşturulmasıdır (❸). Tablo-7.3 de devreye ilişkin doğruluk tablosu verilmiştir.

Tablo-7.3. Genlik karşılaştırıcıya ilişkin doğruluk tablosu.

Girişler				Çıkışlar		
A_1	A_0	B_1	B_0	$t_1 \Rightarrow A>B$	$t_2 \Rightarrow A=B$	$t_3 \Rightarrow A<B$
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Bu aşamadan sonra yapılması gereken, her bir çıkışa ilişkin ifadenin indirgenmesi işlemidir. İndirgeme işlemi *Karnaugh* diyagramları vasıtasıyla gerçeklenir (❹). Bu durumda dört değişkenli *Karnaugh* diyagramları oluşturulur ve indirgenir. Aşağıda bu işlem sonuçları verilmiştir.

$B_1B_0 \backslash A_1A_0$	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

$t_1 \Rightarrow A>B$ olması durumunda, yani A sayısının B sayısından daha büyük olması durumunda lojik fonksiyonun indirgenmiş şekli,

$$t_1 = A_1 \cdot \overline{B_1} + A_0 \cdot \overline{B_1} \cdot \overline{B_0} + A_1 \cdot A_0 \cdot \overline{B_0}$$

olarak elde edilir.

B_1B_0 \ A_1A_0	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

$t_2 \Rightarrow$ A=B olması durumunda, yani A sayısının B sayısına eşit olması durumunda lojik fonksiyonun indirgenmiş şekli,

$$t_2 = \overline{A}_1 \cdot \overline{A}_0 \cdot \overline{B}_1 \cdot \overline{B}_0 + \overline{A}_1 \cdot A_0 \cdot \overline{B}_1 \cdot B_0 + A_1 \cdot A_0 \cdot B_1 \cdot B_0 + A_1 \cdot \overline{A}_0 \cdot B_1 \cdot \overline{B}_0$$

olarak elde edilir.

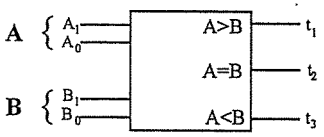
$t_3 \Rightarrow$ A<B olması durumunda, yani A sayısının B sayısından küçük olması durumunda lojik fonksiyonun indirgenmiş şekli,

$$t_3 = \overline{A}_1 \cdot B_1 + \overline{A}_1 \cdot \overline{A}_0 \cdot B_0 + \overline{A}_0 \cdot B_1 \cdot B_0$$

olarak elde edilir.

B_1B_0 \ A_1A_0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

Bu durumda lojik fonksiyonlara karşı düşen işlemler lojik kapılarla gerçeklenirler (❺). Bu adıma ilişkin çizim burada verilmemiştir, ancak genlik karşılaştırıcı devreye ilişkin şema sembolik gösterilimle (❻), dört giriş, üç çıkışlı kombinezonsal devre yapısıyla, Şekil-7.5 deki gibi tanımlanır.



Şekil-7.5. Tanımlanan problem için tasarlanan devre.

Bu devre tek bir tümdevre olarak gerçeklenmiştir. Kataloglardan bulunabileceği üzere 4 bitlik genlik karşılaştırıcı devre 74HC85 (HC: *High Speed CMOS*) koduyla 16 uçlu bir tümdevre olarak gerçeklenmiştir. Devrenin 8 girişi, 3 karşılaştırma çıkışı bulunmaktadır. Ayrıca 3 tane de genişletilmiş çıkışı yer almaktadır. Dolayısıyla lojik kapılarla gerçekleme yerine bu tümdevre elemanı da kullanılabilir. Buradan da görülebileceği gibi lojik devre tasarımcısına pratikte mevcut tümdevreleri çok iyi takip etmek / bilmek gibi çok önemli görevler düşmektedir.

7.4. Kombinezonsal Tümleşik Devreler

Lojik kapıların haricinde bazı özel elemanlar da mevcuttur. Toplama ve çıkarma devresi gibi bazı devreler de ihtiyaçtan ortaya çıkmalarına rağmen tek bir tümdevre ile gerçekleştirilmek suretiyle tasarımcıların ve üreticilerin kullanımına sunulmuşlardır. Bu tür tasarımlar, orta ölçekli tümdevreler (MSI- *Medium Scale Integration Components*) olarak isimlendirilmektedir. Lojik kapılar haricindeki kombinezonsal tümleşik devre elemanları şunlardır:

- Aritmetik Toplama / Çıkarma Devreleri:
  - Yarı Toplayıcı (*Half Adder*)
  - Tam Toplayıcı (*Full Adder*)
  - Yarı Çıkarıcı (*Half Subtractor*)
  - Tam Çıkarıcı (*Full Subtractor*)
  - Toplama ve Çıkarma Devresi (*Adder and Subtractor*)
- Seçiciler (*Multiplexer* ya da kısaca MUX)
- Dağıtıcılar (*Demultiplexer* ya da kısaca DEMUX)
- Kod Çözücüler/Kodlayıcılar (*Decoder/Encoder*)
- Aritmetik Lojik Birimler (ALU)
- 7-Parçalı Göstergeler (*7-Segment Display*)

7.4.1. Aritmetik Toplama / Çıkarma Devreleri

Matematiksel bir toplama veya çıkarma işleminde, iki sayının önce en düşük anlamlı hanelerine ilişkin toplama veya çıkarma işlemi gerçeklenir, sonra bir elde veya borcun oluşup oluşmamasına göre daha yüksek hanede toplama veya çıkarma işlemi tekrarlanır. Tabii bu işlemleri bir insan yapıyorsa alışkın olduğu on tabanı kullanacaktır. Eğer işlemi yapan bir makina ise doğal olarak iki tabanında bu işlemleri yerine getirir. Burada, iki tabanında aritmetik toplama ve çıkarma işlemi yapan devreler tanıtılmıştır. Tasarım yöntemi ve çalışma prensibi açısından bu devreler de birer kombinezonsal devredir.

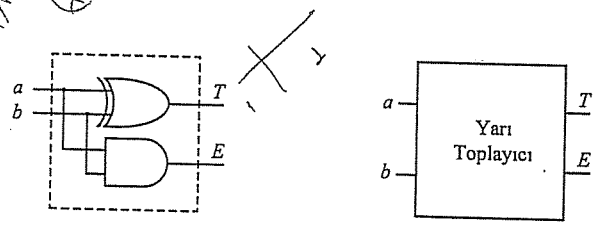
- Yarı Toplayıcı (*Half Adder*)
- Tam Çıkarıcı (*Full Subtractor*)
- Tam Toplayıcı (*Full Adder*)
- Toplama ve Çıkarma Devresi (*Adder and Subtractor*)
- Yarı Çıkarıcı (*Half Subtractor*)

Yarı Toplayıcı (*Half Adder*)

Yarı toplayıcı, elde girişi olmaksızın bir bitlik iki sayının toplamını bulan bir kombinezonsal lojik devredir ve doğruluk tablosu Tablo-7.3 de verilmiştir. Burada a ve b bir bitlik iki sayıyı göstermekte olup, toplam sonucu T, oluşabilecek elde ise E ile ifade edilmiştir. Toplam sonucu doğrudan doğruya *Karnaugh* indirgemesi ile bir YA DA (XOR) kapısı, Elde işlemi ise bir VE (AND) kapısı ile ifade edilebilir. Bu durumda devre şeması Şekil-7.6 da verildiği gibi elde edilir.

Tablo-7.4. Yarı toplayıcıya ilişkin doğruluk tablosu.

a	b	*Toplam* T	*Elde* E
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Şekil-7.6. Yarı toplayıcı devre ve simgesel gösterilimi.

Tam Toplayıcı (*Full Adder*)

Yarı toplayıcıdan farklı olarak, girişinde elde girişinin bulunduğu ve bir bitlik iki sayı ile birlikte toplandığı bir kombinezonsal devredir. Tam toplayıcıya ilişkin doğruluk tablosu Tablo-7.5 de verilmiştir.

Tablo-7.5. Tam toplayıcıya ilişkin doğruluk tablosu.

E_g	a	b	*Toplam* T	*Elde* E
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Burada a ve b gibi bir bitlik iki sayının ve E_g elde girişinin toplam sonucu T ile, oluşabilecek elde ise E ile gösterilmiştir. *Toplam* ve *elde* sonucu doğrudan doğruya *Karnaugh* indirgemesi ile elde edilir. Bu durumda,

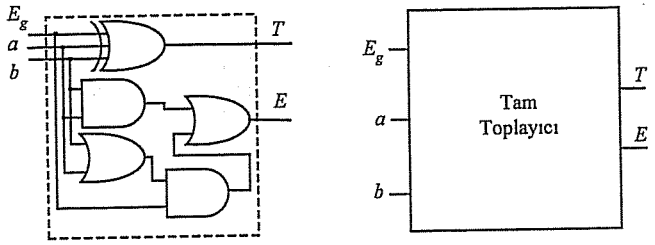
a \ bc	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$T = a \oplus b \oplus c$

a \ bc	00	01	11	10
0	0	0	1	0
1	0	1	1	1

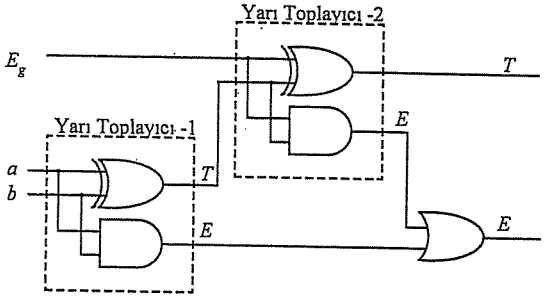
$E = a.b + E_g.(a+b)$

elde edilir. Tam toplayıcı devreye ilişkin şema Şekil-7.7 de verilmiştir.



Şekil-7.7. Tam toplayıcı devre ve simgesel gösterilimi.

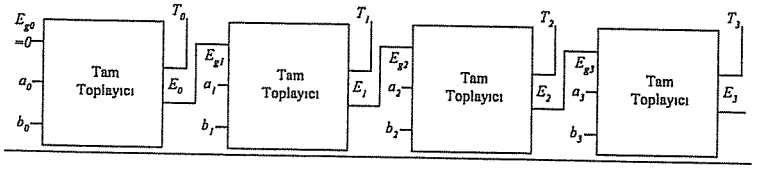
Daha kolay yöntem olarak, Tam toplayıcının iki adet yarı toplayıcıyla tasarımı önerilir. Bu durumda devre şeması Şekil-7.8 deki gibi oluşturulur.



Şekil-7.8. Yarı toplayıcılarla gerçeklenen tam toplayıcı devre.

Örnek-7.4.

Bu örnekte, tam toplayıcılar kullanarak 4 bitlik iki sayının toplamını bulan bir paralel toplayıcı devrenin tasarımı gerçeklenmiştir. Bu amaçla, dört tane tam toplayıcı seri bağlanarak ve her birinin toplam çıkışı toplam sayının çıkışı olarak düşünülerek,



şeklinde elde edilir. Buradaki E_3 çıkışı ise taşma kontrolü için kullanılır. İşte kombinezonsal bir devrenin tasarım işlemi bu biçimde gerçekleştirilmiş oldu. Yalnız bir noktayı açıklamak çok yararlı olacaktır. Bazı kombinezonsal devrelere olan ihtiyaç bu devrelerin özel olarak tek bir entegre devre ile gerçekleştirilmesine olanak sağlamaktadır. Nitekim 4-bitlik toplama işlemini yapacak bu devre de özel bir devredir ve 74LS283 devresi adıyla tek bir entegre devreyle gerçeklenmiştir ve *4-bit Full Adder with fast Carry* orijinal adıyla isimlendirilmiştir.

Yarı Çıkarıcı (*Half Subtractor*)

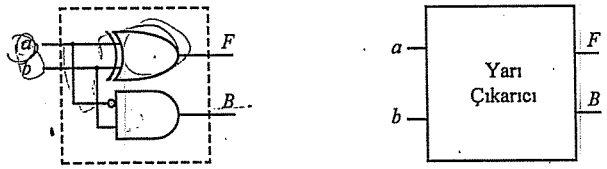
Mikroişlemcide iki sayıyı birbirinden çıkarma işlemi, ikinci sayının tabana tümleyenini alıp ilk sayı ile toplayarak gerçeklenebilir. Toplam sonucunda elde oluşursa

sonuç pozitiftir, elde oluşmaz ise sayı eksidir ve yine tümleyeni alınır ve önüne bir eksi işareti konulur. Lojik devrelerde çıkarma işleminin gerçeklenmesi için bu işlemi yerine getirecek bir Kombinezonsal Devre tasarlamak gerekir. Elde girişi olmadan çıkarma işlemini yapacak devre Yarı Çıkarıcı olarak isimlendirilir ve yarı çıkarıcıya ilişkin doğruluk tablosu Tablo-7.6 da verilmiştir.

Tablo-7.6. Yarı Çıkarıcıya ilişkin doğruluk tablosu.

a	*b*	*Fark* *F*	*Borç* *B*
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Burada *a* ve *b* gibi bir bitlik iki sayının çıkarma işleminin sonucu *F* ile, oluşabilecek borç ise *B* ile gösterilmiştir. *Fark* doğrudan doğruya *Karnaugh* indirgemesi ile bir YA DA (XOR) kapısı, *Borç* ise bir VE (AND) kapısı ile ifade edilebilir. Bu durumda devre şeması Şekil-7.9 da verildiği gibi elde edilir.



Şekil-7.9. Yarı Çıkarıcı Devre ve Sembolik Gösterilimi

Tam Çıkarıcı (*Full Subtractor*)

Çıkarma işlemi sırasında bir önceki haneden bir borç biti geldiği düşünülürse, bu durumda bir borç girişini yarı çıkarıcıya eklemek gerekir. Oluşan bu yeni devreye tam çıkarıcı adı verilir ve tam çıkarıcıya ilişkin doğruluk tablosu Tablo-7.7 de gösterilmiştir.

Tablo-7.7. Tam çıkarıcıya ilişkin doğruluk tablosu.

a	b	B_g	*Fark* F	*Borç* B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Bu doğruluk tablosunda, a ve b gibi bir bitlik iki sayıyı, B_g borç girişini, F oluşan farkı, B ise oluşabilecek bir borç çıkışını göstermektedir. *Fark* ve *borç* doğrudan doğruya *Karnaugh* indirgemesi ile elde edilir. Bu durumda,

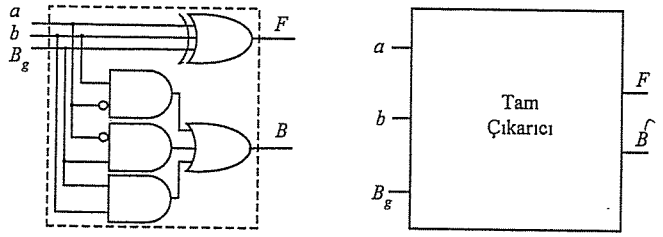
a \ bB_g	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$F = a \oplus b \oplus B_g$$

a \ bB_g	00	01	11	10
0	0	1	1	1
1	0	0	1	0

$$B = \bar{a}b + \bar{a}B_g + b \cdot B_g$$

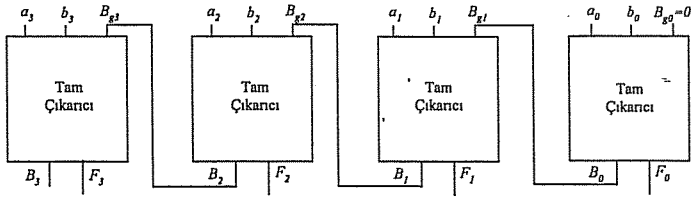
elde edilir. Tam çıkarıcı devreye ilişkin şema Şekil-7.10 da verilmiştir.



Şekil-7.10. Tam çıkarıcı devre ve Simgesel gösterilimi.

Örnek-7.6.

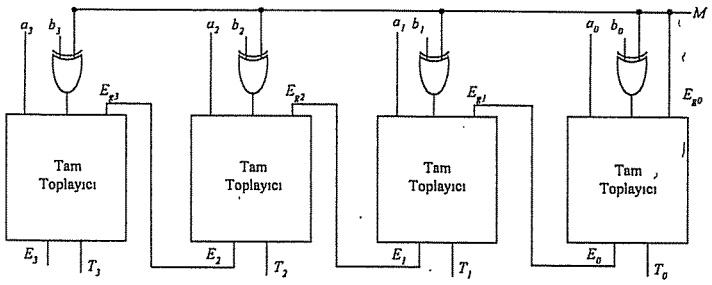
Bu örnekte, tam toplayıcılar kullanarak 4 bitlik iki sayının farkını bulan bir paralel çıkarıcı devrenin tasarımı gerçeklenmiştir. Bu amaçla, dört tane Tam Çıkarıcı seri bağlanarak ve her birinin Fark çıkışı Fark sayının çıkışı olarak düşünülerek,



şeklinde elde edilir. Buradaki B_3 çıkışı ise taşma kontrolü için kullanılır.

Toplama/Çıkarma Devresi (*Adder and Subtractor*)

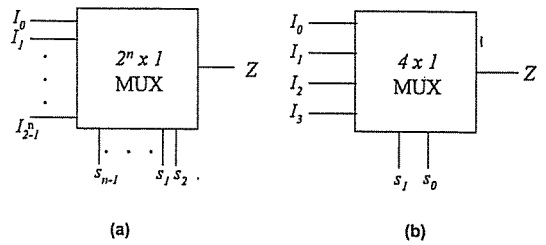
Toplama/Çıkarma devresi, bir kontrol girişi vasıtasıyla toplama ya da çıkarma işlemini seçen ve seçilen işlemi yerine getiren bir kombinezonsal devredir. Tasarım isteğe göre bir bitlik, iki bitlik yapılabilir, ancak en çok kullanılanı dört bitlik olanıdır. Toplama/Çıkarma devresinin tasarımında çıkarma işleminin gerçeklenmesi aşamasında ikiye tümleyen mantığı kullanılmaktadır. Eğer toplama işlemi yapılacaksa işlem doğrudan gerçeklenir, ancak eğer çıkarma işlemi yapılacaksa çıkarılacak sayı ikiye tümlenir ve ilk sayı ile toplanır. Her iki durumda da bir toplama işleminin gerçeklenmesi gerektiğinden toplama ve çıkarma işlemini yapacak devre tasarımında tam toplayıcılar kullanılır. Hangi işlemin gerçekleneceği ise tanımlanacak olan ve M ile gösterilen bir kontrol girişi ile belirlenir. Kontrol girişi, ikiye tümleme işlemini de gerçeklemektedir. Bir sayının lojik anlamda TÜMLEYEN'i alınırsa 1'e tümleyeni ve bu sayıya 1 eklenirse 2'ye tümleyeni bulunur. Sonuçta tam toplayıcının *Elde* girişine bir XOR kapısı ile M ve *Elde* girişleri uygulanır. Bu durumda M=0 ise toplama, M=1 ise çıkarma işlemi gerçeklenecektir. Toplama/Çıkarma devresine ilişkin lojik şema,



şeklinde oluşturulur. Bu devrede M=1 olduğunda ikinci sayı girişi $1 \oplus b_0 = 1 \cdot \overline{b_0} + 0 \cdot b_0 = \overline{b_0}$ şekline dönüşür. Bu durumda E_{g0} girişi 1 olduğundan sayının 2'ye tümleyeni alınmış olur. İşlem bundan sonra toplama işlemine dönüştürülmüştür.

7.4.2. Seçiciler (Multiplexer-MUX)

Sayısal veri iletiminde bir çok olası giriş yolundan birini seçip, bu yolu çıkışa veren özel kombinezonsal bir devredir. MUX elemanında n tane seçme girişi belirlenir ve bu durumda olası giriş yolu sayısı 2^n tanedir. Devrenin toplam giriş sayısı ise, $2^n + n$ tane olmasına karşılık çıkış sayısı sadece 1 tanedir. Seçicilere bazı uygulamalarda çoklayıcı, veri seçici ya da veri toplayıcı adı da verilmektedir. Bir seçici devreye ait gösterilim Şekil-7.11 de verilmiştir.



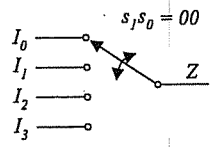
Şekil-7.11. a) Seçiciye ilişkin lojik devre şeması b) 4x1 MUX örneği.

Bu devrede $s_{n-1}, \cdots, s_1, s_0$ seçme girişlerini, $I_0, I_1, \cdots, I_{2^n-1}$ girişleri belirtmek üzere $m_i = s_{n-1} \cdot \ldots \cdot s_1 \cdot s_0$ şeklinde tanımlanarak seçici devreye ilişkin çıkışı ifadesi,

$$Z = m_0 \cdot I_0 + m_1 \cdot I_1 + \ldots + m_{2^n-1} \cdot I_{2^n-1}$$

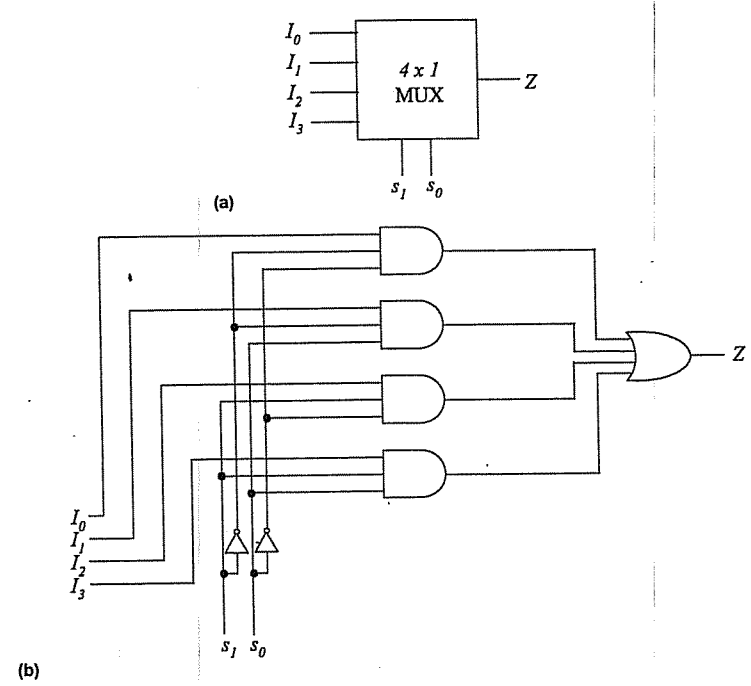
şeklinde ifade edilir. Fiziksel olarak ifade etmek gerekirse, seçici devre aslında seçme girişlerine göre birinci, ikinci ya da $2^n - 1$ inci terimi çıkışa veren devre olarak adlandırılabilir. Örneğin, 2 tane seçme (n=2) ucu olan devrenin 4 tane olası giriş yolu mevcuttur. Seçme girişlerinin aldıkları değerlere göre,

eğer $s_1 s_0 = 00 \Rightarrow$ birinci giriş çıkışa verilir,
eğer $s_1 s_0 = 01 \Rightarrow$ ikinci giriş çıkışa verilir,
eğer $s_1 s_0 = 10 \Rightarrow$ üçüncü giriş çıkışa verilir,
eğer $s_1 s_0 = 11 \Rightarrow$ dördüncü giriş çıkışa



verilir.

Bu örneğe ilişkin devrenin simgesel gösterilimi Şekil-7.12.a, kapılarla gerçekleştirilmiş iç yapısı Şekil-7.12.b de verilmiştir.



Şekil-7.12. a) n=2 olan seçiciye ilişkin simgesel gösterilim b) Lojik kapılarla seçici iç yapısı.

Örnek-7.7.

Doğruluk Tablosu Tablo-7.8 de verilen kombinezonsal devreyi, x_3 ve x_4'ü seçme girişleri olacak şekilde Standart 4×1 Seçici (MUX) birimi ile gerçekleştiriniz.

Tablo-7.8. Lojik fonksiyona ilişkin doğruluk tablosu.

x_1	x_2	x_3	x_4	$f(x_1,x_2,x_3,x_4)$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Problemin çözümünü için ilk olarak Minimum Terimler kanonik Biçimi ifadesi bulunur. Buradan seçme girişlerine ilişkin ifadeler ortak paranteze alınır. Ortak paranteze alınan terimler en sade biçime indirgenerek bilinen lojik kapılar cinsinden ifade edilir. Bundan sonra, MUX girişlerinde bu kapı elemanları kullanılarak Tablo-7.8 deki doğruluk tablosunu sağlayacak devreye ilişkin şema çizilir.

Bu amaçla, doğruluk tablosundan lojik fonksiyona ilişkin Minimum Terimler Kanonik Biçimi,

$$\begin{aligned} f(x_1,x_2,x_3,x_4) = {} & \overline{x}_1\cdot\overline{x}_2\cdot x_3\cdot\overline{x}_4+\overline{x}_1\cdot\overline{x}_2\cdot x_3\cdot x_4+\overline{x}_1\cdot x_2\cdot\overline{x}_3\cdot x_4 \\ & +\overline{x}_1\cdot x_2\cdot x_3\cdot x_4+x_1\cdot\overline{x}_2\cdot\overline{x}_3\cdot\overline{x}_4+x_1\cdot\overline{x}_2\cdot\overline{x}_3\cdot x_4 \\ & +x_1\cdot\overline{x}_2\cdot x_3\cdot\overline{x}_4+x_1\cdot\overline{x}_2\cdot x_3\cdot x_4 \end{aligned}$$

şeklinde elde edilir. Bundan sonra yapılması gereken, seçme girişleri cinsinden ifadeyi ortak çarpan parantezine almak gerekmektedir. Seçme girişlerinin aldıkları değerler: $\overline{x}_3\cdot\overline{x}_4$, $\overline{x}_3\cdot x_4$, $x_3\cdot\overline{x}_4$ ve $x_3.x_4$ şeklindedir. Ortak paranteze alınırsa,

$$\begin{aligned} f(x_1,x_2,x_3,x_4) = {} & \overline{x}_3\cdot\overline{x}_4\cdot(x_1\cdot\overline{x}_2)+\overline{x}_3\cdot x_4\cdot(\overline{x}_1\cdot x_2+x_1\cdot\overline{x}_2) \\ & +x_3\cdot\overline{x}_4\cdot(\overline{x}_1\cdot\overline{x}_2+x_1\cdot\overline{x}_2)+x_3\cdot x_4\cdot(\overline{x}_1\cdot\overline{x}_2+\overline{x}_1\cdot x_2+x_1\cdot\overline{x}_2) \end{aligned}$$

elde edilir. Ortak paranteze alınan terimlerin sadeleştirilmesi ile de

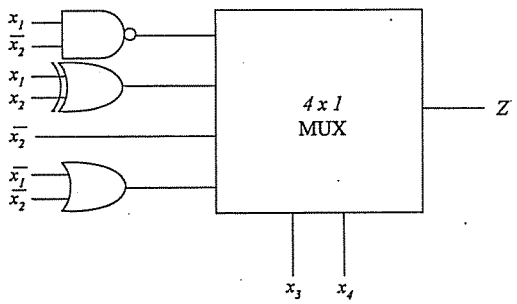
$$\begin{aligned} f(x_1,x_2,x_3,x_4) = {} & \overline{x}_3\cdot\overline{x}_4\cdot(x_1\cdot\overline{x}_2)+\overline{x}_3\cdot x_4\cdot(\overline{x}_1\cdot x_2+x_1\cdot\overline{x}_2)+x_3\cdot\overline{x}_4\cdot(\overline{x}_2\cdot(\overline{x}_1+x_1)) \\ & +x_3\cdot x_4\cdot((\overline{x}_2+x_2)\cdot\overline{x}_1+x_1\cdot\overline{x}_2) \end{aligned}$$

$$\begin{aligned} f(x_1,x_2,x_3,x_4) = {} & \overline{x}_3\cdot\overline{x}_4\cdot(x_1\cdot\overline{x}_2)+\overline{x}_3\cdot x_4\cdot(\overline{x}_1\cdot x_2+x_1\cdot\overline{x}_2)+x_3\cdot\overline{x}_4\cdot(\overline{x}_2) \\ & +x_3\cdot x_4\cdot(\overline{x}_1+\overline{x}_2) \end{aligned}$$

bulunur. Bu durumda MUX'a ilişkin girişler,

x_3	x_4	I
0	0	$x_1\cdot\overline{x}_2$
0	1	$x_1\oplus x_2$
1	0	$\overline{x}_2$
1	1	$\overline{x}_1+\overline{x}_2$

şeklinde belirlenmiştir ve bu girişlere karşılık düşen 4x1 MUX elemanı Şekil-7.13 de verilmiştir.



Şekil-7.13. Örnek-7.1 e ilişkin 4x1 MUX elemanı girişleri.

Örnek-7.8.

Örnek-7.7 de verilen problemi, (Doğruluk Tablosu için Tablo-7.8 e bakınız); x_1 ve x_2 'yi seçme girişleri alarak 4x1 MUX elemanı ile gerçekleştiriniz.

Bu amaçla, doğruluk tablosundan lojik fonksiyona ilişkin Minimum Terimler Kanonik Biçimi,

$$\begin{aligned} f(x_1,x_2,x_3,x_4) = {} & \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 \cdot \bar{x}_4 + \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 \cdot x_4 + \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 \cdot x_4 \\ & + \bar{x}_1 \cdot x_2 \cdot x_3 \cdot x_4 + x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot \bar{x}_4 + x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot x_4 \\ & + x_1 \cdot \bar{x}_2 \cdot x_3 \cdot \bar{x}_4 + x_1 \cdot \bar{x}_2 \cdot x_3 \cdot x_4 \end{aligned}$$

şeklinde elde edilir. Bundan sonra yapılması gereken, seçme girişleri cinsinden ifadeyi ortak çarpan parantezine almak gerekmektedir. Seçme girişlerinin aldıkları değerler: $\bar{x}_1 \cdot \bar{x}_2$, $\bar{x}_1 \cdot x_2$, $x_1 \cdot \bar{x}_2$ ve $x_1 \cdot x_2$ şeklindedir. Ortak paranteze alınırsa,

$$\begin{aligned} f(x_1,x_2,x_3,x_4) = {} & \bar{x}_1 \cdot \bar{x}_2 \cdot (x_3 \cdot \bar{x}_4 + x_3 \cdot x_4) + \bar{x}_1 \cdot x_2 \cdot (\bar{x}_3 \cdot x_4 + x_3 \cdot x_4) \\ & + x_1 \cdot \bar{x}_2 \cdot (\bar{x}_3 \cdot \bar{x}_4 + \bar{x}_3 \cdot x_4 + x_3 \cdot \bar{x}_4 + x_3 \cdot x_4) + x_1 \cdot x_2 \cdot (0) \end{aligned}$$

elde edilir. Ortak paranteze alınan terimlerin sadeleştirilmesi ile de

$$\begin{aligned} f(x_1,x_2,x_3,x_4) = {} & \bar{x}_1 \cdot \bar{x}_2 \cdot (x_3 \cdot (\bar{x}_4 + x_4)) + \bar{x}_1 \cdot x_2 \cdot (x_4 \cdot (\bar{x}_3 + x_3)) \\ & + x_1 \bar{x}_2 \cdot (\bar{x}_3 \cdot (\bar{x}_4 + x_4) + x_3 \cdot (\bar{x}_4 + x_4)) + x_1 \cdot x_2 \cdot (0) \end{aligned}$$

$$f(x_1,x_2,x_3,x_4) = \bar{x}_1 \bar{x}_2 \cdot (x_3) + \bar{x}_1 \cdot x_2 \cdot (x_4) + x_1 \cdot \bar{x}_2 \cdot (\bar{x}_3 + x_3) + x_1 \cdot x_2 \cdot (0)$$

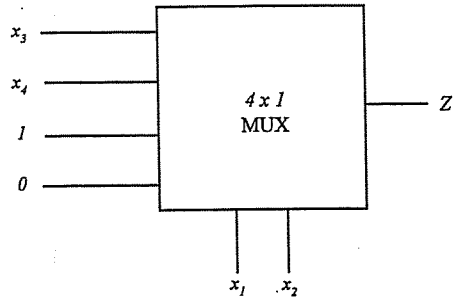
$$f(x_1,x_2,x_3,x_4) = \bar{x}_1 \cdot \bar{x}_2 \cdot (x_3) + \bar{x}_1 \cdot x_2 \cdot (x_4) + x_1 \cdot \bar{x}_2 \cdot (1) + x_1 \cdot x_2 \cdot (0)$$

bulunur. 4x1 MUX elemanında mevcut iki seçme girişi ve dört MUX girişine ilişkin uyarma tablosu aşağıda verilmiştir. ilişkin girişler Herbir seçme girişi çiftine ilişkin bulunan değerler,

x_1	x_2	I
0	0	x_3
0	1	x_4
1	0	1
1	1	0

şeklinde belirlenmiştir. Burada dikkat edilirse seçme girişlerinin belirlenmesinin çok önemli bir adım olduğu görülmektedir. Örnek-7.7 de x ve x_4 'ün seçme giriş

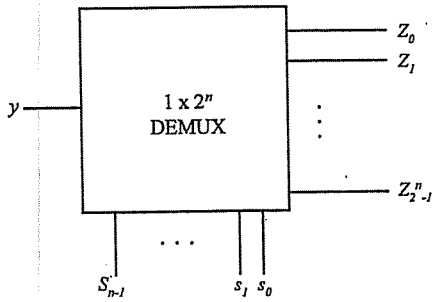
olarak belirlenmesi durumunda MUX girişlerinde 1'er adet TVE, YA DA ile VEYA kapısı kullanmak gerekmektedir. Oysa x_1 ve x_2 'nin seçme girişi olarak belirlenmesi durumunda hiç bir ilave kapıya gerek duyulmayacaktır. Maliyet faktörünün önemli olması nedeniyle Örnek-7.8 de gerçeklenen tasarım Örnek-7.7 de gerçeklenene göre çok daha başarılıdır. Belirlenen girişlere karşılık düşen 4x1 MUX elemanı Şekil-7.14 te verilmiştir.



Şekil-7.14. Örnek-7.2 ye ilişkin 4x1 MUX elemanı girişleri.

7.4.3. Dağıtıcılar (Demultiplexer - DEMUX)

Sayısal veri iletiminde bir noktadan gelen giriş işaretini bir çok olası çıkış yolundan birini seçip, bu yola veren özel Kombinezonsal bir devredir. DEMUX elemanında da n tane seçme girişi belirlenir. Giriş işareti bir tanedir. Olası çıkış yolu sayısı 2^n tanedir. Devrenin toplam giriş sayısı $n+1$ tane olmasına karşılık, çıkış sayısı 2^n tanedir. Veri dağıtıcı şeması Şekil-7.15 de verilmiştir.



Şekil-7.15. Veri dağıtıcıya ilişkin lojik devre şeması.

Bu devrede $s_{n-1},\cdots,s_1,s_0$ seçme girişlerini, $Z_0,Z_1,\cdots,Z_{2^n-1}$ çıkışları belirtmek üzere, $m_i = s_{n-1}\cdot\ldots\cdot s_1\cdot s_n$ şeklinde tanımlanarak Dağıtıcı devreye ilişkin çıkış ifadesi;

$$Z = m_i \cdot y \qquad i = 0,1,\cdots,2^n-1$$

şeklinde ifade edilir. Fiziksel olarak ifade etmek gerekirse, Veri Dağıtıcı devre aslında seçme girişlerine göre giriş işaretini, birinci, ikinci ya da 2^n-1 inci çıkışa veren devre olarak adlandırılabilir. Örneğin, 2 tane seçme girişi (n=2) için devrenin 4 tane olası çıkış yolu mevcuttur. Seçme girişlerinin aldıkları değerlere göre,

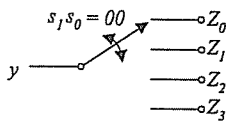
eğer $s_1s_0 = 00 \Rightarrow$ giriş birinci çıkışa verilir,

eğer $s_1s_0 = 01 \Rightarrow$ giriş ikinci çıkışa verilir,

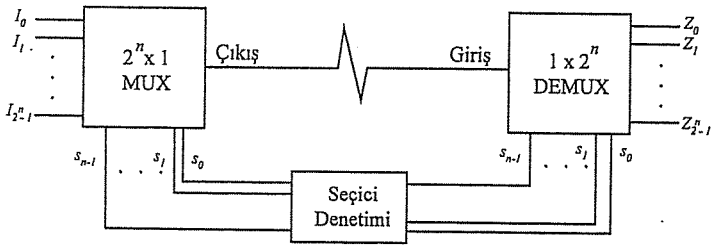
eğer $s_1s_0 = 10 \Rightarrow$ giriş üçüncü çıkışa verilir,

eğer $s_1s_0 = 11 \Rightarrow$ giriş dördüncü çıkışa verilir.

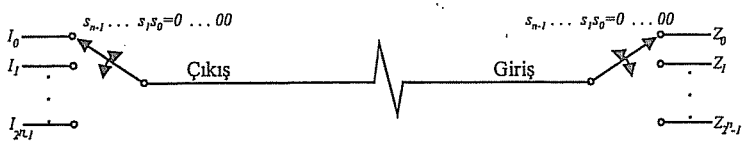
Bu anahtar yapısı seçme girişlerinin 00 değeri için aşağıda gösterilmiştir,



MUX ile DEMUX elemanlarının birlikte kullanılmasına, özellikle ilkel telefon santralleri iyi bir örnek oluşturmaktadır. Bu durumda, 2^n tane kullanıcı karşı tarafta bulunan 2^n tane kullanıcıya erişebilmektedir. Buna ilişkin lojik şema gösterilimi Şekil-7.16 da, simgesel anahtar devresi eşdeğeri ise Şekil-7.17 de verilmiştir.



Şekil-7.16. Haberleşme sistemlerinde kullanılan MUX ve DEMUX elemanları.

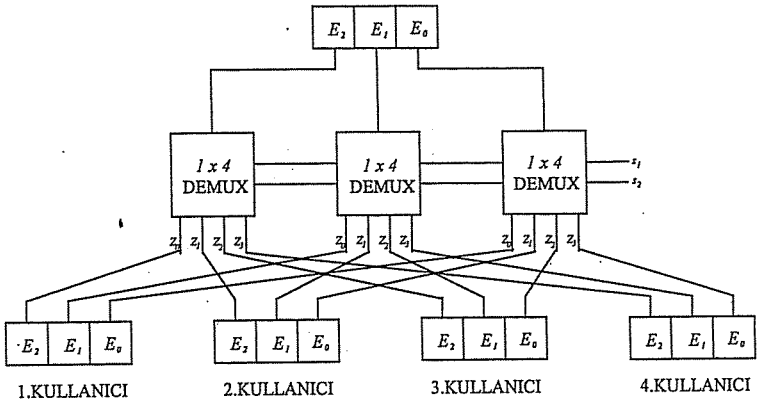


Şekil-7.17. Haberleşme sistemlerindeki eşdeğer yapı.

Örnek-7.9.

3 bitlik bir ifade dört kullanıcıya aktarılacaktır; bu işlem her kullanıcının seçilmesi ve bilginin sadece bu kullanıcılara ulaştırılması şeklinde olacaktır. Bu devrenin DEMUX elemanları kullanılarak gerçekleştiriniz.

Aktarılacak ifadenin 3-bitlik ve kullanıcı sayısının dört olması durumunda, her bir bit için bir tane DEMUX kullanılacak ve kullanıcı sayısı kadar çıkışa sahip olması istenecektir. Dolayısıyla 3 adet 1x4 DEMUX kullanmak gerekecektir. Bu durumda devreye ilişkin lojik şema Şekil-7.18 deki gibi çizilebilir.



Şekil-7.18. DEMUX elemanları ile gerçeklenen tasarım.

Bu devre s_1 ve s_2'nin aldığı değerlere göre 3-bitlik veri dört kullanıcıdan birine iletilir. Devrenin çalışması şu şekildedir:

$s_1s_2 = 00$ ise $E_2E_1E_0$ verisi 1. kullanıcıya,
$s_1s_2 = 01$ ise E_2E_1E verisi 2. kullanıcıya,
$s_1s_2 = 10$ ise E_2E_1E verisi 3. kullanıcıya,
$s_1s_2 = 11$ ise E_2E_1E verisi 4. kullanıcıya gönderilir.

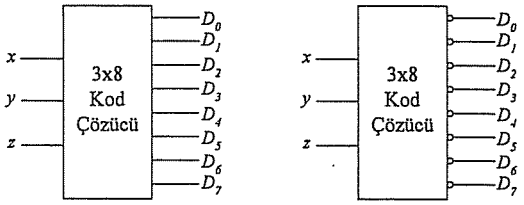
7.4.4. *Kod Çözücü (Decoder)*

Kod çözücü, n-bitlik bir sözcüğün kodunu çözüp olası en çok 2^n çıkış yolundan sadece birini aktif hale getiren bir kombinezonsal devredir. $n=3$ girişli, $2^n = 2^3 = 8$ çıkışlı bir kod çözücüye ilişkin doğruluk tablosu Tablo-7.9 da verilmiştir.

Tablo-7.9. Kod çözücüye ilişkin doğruluk tablosu.

Girişler			Çıkışlar							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Kod çözücü devre, x, y, z girişlerinin aldıkları değerlere göre $D_0D_1D_2D_3D_4D_5D_6D_7$ çıkış yollarından sadece birisinin aktif hale gelmesi ilkesine dayanarak çalışır. Üç giriş-sekiz çıkışlı kod çözücü devre, 3x8 (veya 3'e 8) kod çözücü olarak isimlendirilir. Bu devreye ilişkin sembolik gösterilim Şekil-7.19 da verilmiştir.



Şekil-7.19. 3x8 kod çözücüye ilişkin simgesel gösterilim
a) Normal çıkışlı b) Tümleyen çıkışlı.

Bazı kod çözücü devrelerde çıkışlar tümlenmiştir. Bunun anlamı; aktif olan çıkış lojik 0 değerini alırken diğer bütün çıkışların lojik 1 seviyesinde bulunmasıdır. Bu yapıya sahip bir kod çözücünün doğruluk tablosu Tablo-7.10 da, devreye ilişkin sembolik gösterilim Şekil-7.19 da verilmiştir.

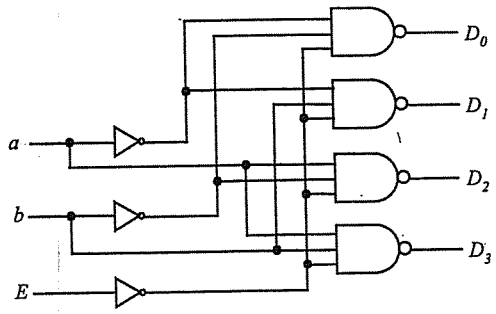
Bütün bunların yanı sıra, kod çözücülerin pratikte kullanımında dikkat edilmesi gereken durumlar söz konusudur. Kod çözücünün veri girişlerinin yanı sıra bir de

devrenin çalışıp çalışmamasını sağlayan etkin girişleri de bulunmaktadır. Özellikle 2'ye 4 kod çözücülerde bir tane tümleyen girişli, 3'e 8 kod çözücülerde ikisi tümleyen girişli, biri normal girişli olmak üzere üç tane etkin girişi yer almaktadır. Örneğin 74LS138-3'e 8 kod çözücünün $\overline{E}_1$, $\overline{E}_2$ ve E_3 şeklinde üç adet etkinleştirme ucu bulunmakta olup, devrenin çalışabilmesi için $\overline{E}$, $\overline{E}_2$'nin Lojik 0, E_3'ün lojik 1 değerlerine çekilmiş olması gerekmektedir. Bu durumda etkin girişleri aktif olmadan devrenin çalışması, yani giriş değerlerini değerlendirerek çıkış vermesi mümkün değildir. Kod çözücü bu durumda, eğer tümleyen çıkışlı ise bütün çıkışlarında lojik 1, normal çıkışlı ise bütün çıkışlarında lojik 0 üretecektir.

Tablo-7.10. Tümleyen çıkışlı kod çözücüye ilişkin doğruluk tablosu.

Girişler			Çıkışlar (Tümleyen)							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

Şekil-7.20 de 2x4 kod çözücünün kapılarla gerçeklenmiş iç yapısı görülmektedir. Dikkatlice bakıldığında ve kapıların olası giriş kombinasyonları verildiğinde aynı anda bir çıkışın aktif, diğerlerinin pasif olduğu görülür.



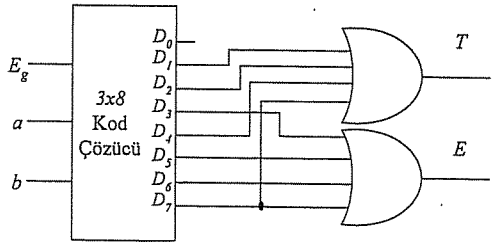
Şekil-7.20. 2x4 kod çözücünün kapılarla gerçekleştirilmiş iç yapısı.

Örnek-7.10.

Bu örnekte, Tam Toplayıcı devrenin kod çözücü kullanılarak gerçeklenmesi anlatılmıştır. Tam toplayıcıya ilişkin doğruluk tablosu Tablo-7.5 te verilmişti. Bu doğruluk tablosundan yararlanarak *Toplam* ve *Elde* çıkışlarına ilişkin fonksiyonlar

$$T(E_g, a, b) = \sum(1,2,4,7) \quad , \quad E(E_g, a, b) = \sum(3,5,6,7)$$

yazılabilir. Bu durumda Toplam ve Elde çıkışları birer VEYA kapısı ve bir tane 3x8 Kod Çözücü kullanılarak elde edilebilir. Tam toplayıcının kod çözücü kullanılarak tasarlanmış şekli Şekil-7.21 de verilmiştir.



Şekil-7.21. 3x8 kod çözücüyle tasarlanmış tam toplayıcı devre.

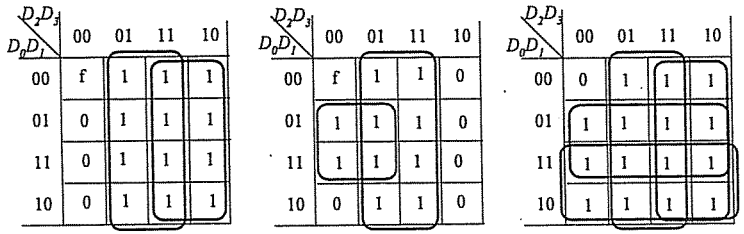
7.4.5. *Kodlayıcı (Encoder)*

Kod Çözücülerin tersine çalışan devrelerdir. Kodlayıcı, bir işareti başka bir lojik devre tarafından işlenebilecek şekle getiren devre olarak da tanımlanabilir. Özellikle sayı sistemleri arasında dönüşüm yapan onaltı, on ve sekiz tabanından ikili sayı sistemine kodlayıcı devreler bu sınıfa girmektedirler. Bunların haricinde öncelik kodlayıcı devre, yüksek öncelikli bitlerin değişiminin diğer bitlere göre önem kazandığı uygulamalarda tercih edilen kombinezonsal devrelerdir.

4-bitlik ($D_3D_2D_1D_0$) bir öncelik kodlayıcının çalışma şekli şöyledir: D_3 en yüksek anlamlı ve öncelik kodlayıcıda en yüksek öncelikli bittir. Sonra sırasıyla D_2D_1 ve D_0 gelir. Öncelik kodlayıcı *x,y* ve *hazır* olmak üzere üç çıkışa sahiptir. *x* ve *y* devrenin çıkışlarını gösterir. *hazır* girişi ise öncelik kodlayıcının girişindeki verinin hepsinin sıfır olması durumunda lojik 0 (hazır değil), aksi durumda lojik 1 (hazır) değerini alır. Herhangi bir şekilde D_3 girişi lojik 1 seviyeye çekilirse diğer girişlerin bir anlamı yoktur; bunlar keyfi değer alabilirler. Bu durumda devreye ilişkin doğruluk tablosu,

Girişler				Çıkışlar		
D_3	D_2	D_1	D_0	*x*	*y*	*hazır*
0	0	0	0	ϕ	ϕ	0
0	0	0	1	0	0	1
0	0	1	ϕ	0	1	1
0	1	ϕ	ϕ	1	0	1
1	ϕ	ϕ	ϕ	1	1	1

şeklinde oluşturulur. Her bir çıkışa ilişkin *Karnaugh* diyagramı;

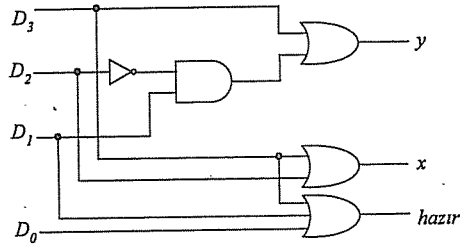


$x=D_2+D_3$ $\quad$ $y=D_3+D_1\overline{D}_2$ $\quad$ $hazır=D_0+D_1+D_2+D_3$

şeklinde oluşturulur ve uygun komşuluklar seçilerek indirgeme yapılırsa, çıkış fonksiyonları

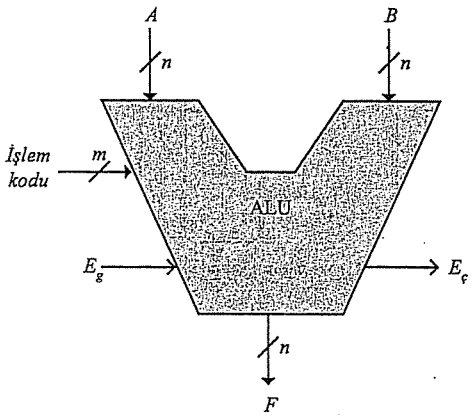
$$x = D_2 + D_3$$
$$y = D_3 + D_1 \cdot \overline{D}_2$$
$$hazir = D_0 + D_1 + D_2 + D_3$$

olarak bulunur. Bu durumda öncelik kodlayıcıya ilişkin devre şeması aşağıda verildiği gibi olur.



7.4.6. *ALU Tasarımı (Arithmetic Logic Unit)*

Kısaca ALU olarak anılan Aritmetik Lojik Birimler (*Arithmetic Logic Unit*), tek bir birim aracılığıyla toplama, çıkarma, tümleme, karşılaştırma ve öteleme gibi birçok aritmetik ve mantıksal işlemlerin yapıldığı/kotarıldığı tümleşik lojik elemanlardır. Bilgisayarların önemli bir parçasını oluşturan işlemciler içerisinde de tüm aritmetik ve lojik işlemler ALU üzerinden yapılır. Bir ALU'nun temelde, Şekil-7.22 de görüldüğü gibi veri/parametre girişleri, sonuç çıkışları ve parametreler üzerinde hangi işlemin yapılacağını gösteren işlem kodu (kısaca işkod) girişleri vardır. Dolayısıyla, girişlerine gelen parametreler üzerinde, o anda işlem kodu girişleriyle belirtilen işlem yapılır.



Şekil-7.22. ALU'nun genel gösterilimi.

Uygulamada çok farklı özelliklerde ALU'lar tasarlanmıştır; hem tümdevre olarak üretilmiş olup ayrı bir tümdevre olarak satın alınabilir, hem de ASIC yardımcı araçlarının kütüphanelerinde vardır. Dolayısıyla tasarımcı, aritmetik mantıksal işlemlerin çok olduğu lojik devre tasarımlarında hazır ALU birimler/tümdevreleri kullanabilir. Orta ölçekli tümdevre olarak, 74181, bir ALU tümdevresidir; 4 bitlik iki tane veri üzerinde 16'şar tane aritmetik ve mantıksal işlemden birini gerçekleştirir. Veriler üzerinde hangi işlemin yapılacağını belirleyen dört tane işlem kodu girişi ve bir tane mod (*M*) girişi vardır. Eğer *M*=1 ise işlem kodu girişindeki koda bağlı olarak 16 tane mantıksal işlemden birinin gerçekleştirir; *M*=0 ise, 16 tane aritmetik işlemden birini gerçekleştirir. Sanki 32 tane işlem varmış gibidir. 74181 hakkında ayrıntılı bilgi için [SGS-THOMSON] kataloguna bakınız. Tasarımcı 74181 ve buna benzer ALU tümdevrelerini tasarımlarında kullanabilir.

Örnek-7.11.

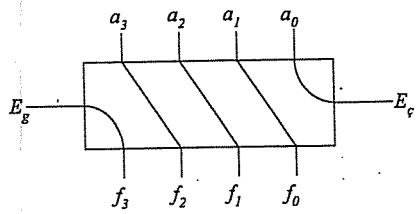
Bir ALU'nun iç mimarisini ve genel tasarımını görmek amacıyla iki tane 4 bitlik veri, bir tane elde girişleri olan ve bu veriler üzerinde 4 işlemden birini yapan bir ALU tasarlanmak istenmektedir; 4 bitlik sonuç çıkışı ve bir tane de elde çıkışı vardır. İşlem kodu listesi ve kodları Tablo-7.11 de verilen ALU'yu gerektiği kadar kapı, MUX ve buna benzer lojik elemanlar kullanarak tasarlayınız. ALU giriş ve çıkışları aşağıdaki isimlendirilmişlerdir:

- Veri girişleri : $A\ (a_3, a_3, a_1, a_0)$ ve $B\ (b_3, b_2, b_1, b_0)$
- Elde girişi ve çıkışı sırasıyla : E_g ve $E_ç$
- Sonuç çıkışı : $F\ (f_3, f_2, f_1, f_0)$
- İşlem kodu : İşkod (k_1, k_0)

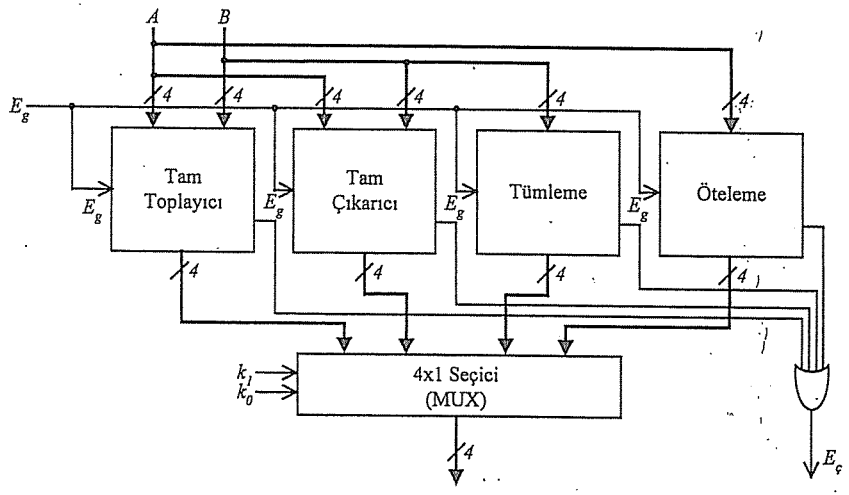
Tablo-7.11. Örnek ALU İşlem listesi ve kodları.

İşlem Kodu $k_1\ k_2$	İşlem	Açıklama
0 0	$F=A+B$	A ve B'nin içerikleri toplanır.
0 0	$F=A-B$	A'dan B çıkarılır.
0 0	$F=\overline{B}$	B'nin tüm bitleri tümlenir; 1'ler 0, 0'lar 1 yapılır.
0 0	$F=A_{1\ \text{bit sağa öteleme}}$	A'nın içeriği 1 bit sağa ötelenir; 0110 ise 0011 olur.

ALU 4 tane işlem yapmaktadır; toplama, çıkarma, tümleme ve öteleme, bu 4 işleme ait birimlerin nasıl yapılacağı biliniyor. Toplama ve çıkarma devreleri Ayrıt-7.2 de incelenmişti; tümleme işlemi de her bir bit için birer TÜMLEME kapısı kullanılır; öteleme işlemi de girişlerdeki verinin 1 bit sağa ötelenmesidir. Öteleme işlemi dışında diğer işlemleri yapan 4 bitlik altbirimler hazır demektir. Şimdi, önce 4 bitlik bir veriyi 1 bit sağa öteleyen bir altbirim tasarlansın ve sonra herbir işlemi yapan birimler ve MUX kullanılarak istenen ALU Şekil-7.23 deki gibi tasarlanabilir.



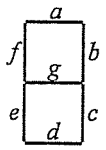
Aşağıda girişlerine gelen 4 bitlik veriyi çıkışına 1 bit sağa öteleyen lojik devre görülmektedir; elde girişi olan E_g, sol taraftan boşalan bite yerleşirken, en sağdaki bit olan en anlamsız bit elde çıkışına, $E_ç$'ye yansıtılmaktadır.



Şekil-7.23. Örnek-7.11 in çözümü olan ALU şeması.

7.4.7. 7-Parçalı Gösterge (7-Segment Display)

7-Parçalı Gösterge, BCD kodunda verilmiş olan bir ifadenin, sayı biçiminde gösterilebilmesi amacıyla kullanılan ışıklı bir kombinezonsal devredir. Bu işlemin gerçeklenebilmesi için yedi parçalı göstergenin her parçası *a*'dan *g*'ye kadar kodlanmıştır. Bu durumda, 7-parçalı göstergeye ilişkin sembolik gösterilim Şekil-7.22 de oluşturulabilecek kombinasyonlar ise Şekil-7.23 de görüldüğü gibidir.



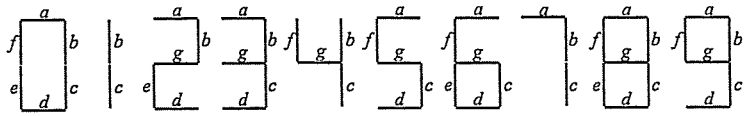
Şekil-7.22. 7-parçalı gösterge

Şekil-7.23. 7-parçalı gösterge ile oluşturulabilecek kombinasyonlar.

Örnek-7.12.

BCD kodunu 7-parçalı gösterge elemanını sürmek üzere yedi lojik işarete dönüştürecek bir kod çözücü devre tasarlamak amacıyla bu örnek verilmiştir. Gelen işaretler 0'dan 9'a kadar olan BCD sayılara karşı düşmektedir. Aydınlatılması istenen parçacığa lojik 1, aksi durumda lojik 0 uygulanır. Bu durumda giriş değişkenleri x_1, x_2, x_3, x_4 ve 7-parçalı göstergeye ilişkin parçalar, bir başka değişle çıkış değişkenleri a, b, c, d, e, f, g olmak üzere devreye ilişkin doğruluk tablosunu oluşturunuz ve tabloyu *Karnaugh* Diyagramı ile çarpımlar toplamı biçiminde indirgeyerek çıkış ifadelerini belirleyiniz.

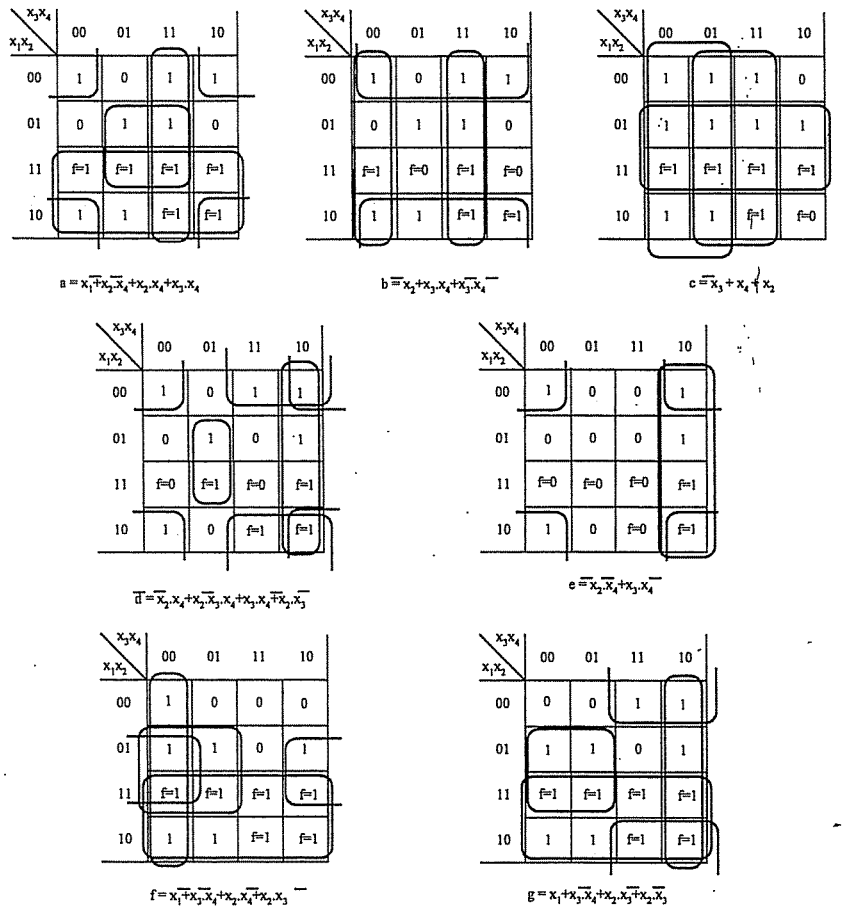
Problemin çözümünde ilk olarak, ilgili rakamları oluşturmak için aydınlatılması gereken parçaların belirlenmesi gerekir. Bu durumda, her sayı değerinde aydınlatılacak parçalara karşılık düşen değişkenler,



şeklinde belirlenir. Bu işlemden sonra giriş değişkenlerine karşılık düşen çıkış değişkenlerini gösteren doğruluk tablosu;

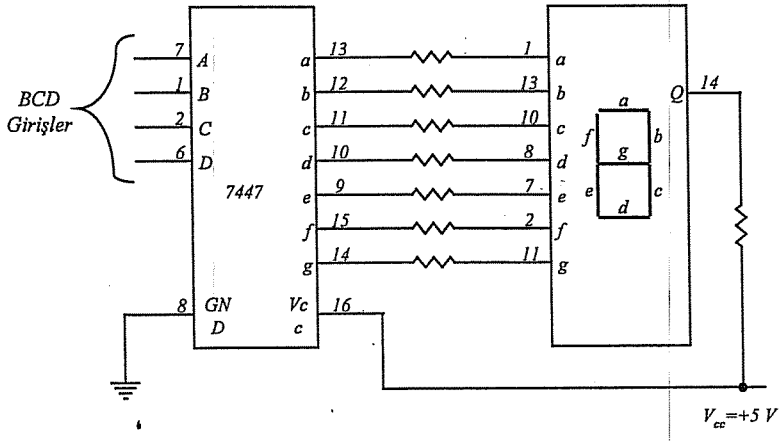
Girişler				Çıkışlar						
x_1	x_2	x_3	x_4	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	0	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1
1	0	1	0	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
1	0	1	1	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
1	1	0	0	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
1	1	0	1	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
1	1	1	0	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
1	1	1	1	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ

şeklinde oluşturulur ve her bir parçaya ilişkin çarpımlar toplamı şeklindeki lojik ifade *Karnaugh* diyagramları yardımıyla indirgenerek herbir segmente ilişkin aydınlatılma kombinasyonu elde edilir. Bu *Karnaugh* diyagramları ve indirgenmiş ifadeler Şekil-7.24 de verildiği gibi olur.



Şekil-7.24. 7-parçalı göstergede herbir segmentin parlatılması koşulları.

Uygulamada 7-parçalı göstergenin kullanımı bu kadar karmaşık değildir. Bir BCD/7-parçalı gösterge kod çözücü devre olan 74LS47 tümdevresi, verilmiş olan BCD koduna bakarak hangi segmentlerin aktif hangilerinin pasif olacağını belirler ve segmentleri birer direnç üzerinden sürer. Bu durumda BCD girişler bu kod çözücü tümdevreye uygulanır, sonuçta tümdevre çıkışında BCD koda karşılık gelen sayının gösterilimindeki segmentlerin bilgisi bulunmaktadır; örneğin 0011_{BCD} sayısı için tümdevre çıkışında 3 sayısının karşılığı olan *a, b, c, d* ve *g* segmentlerinin aydınlatılması beklenir. 74LS47 çıkışında bu segment uçlarına gerekli lojik bilgi gönderilir. Şekil-7.25 de 7447 ve 7-parçalı göstergenin bağlantıları gösterilmiştir.

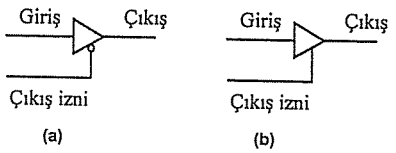


Şekil-7.25. Göstergenin 7447 BCD/7-parçalı gösterge kod çözücüsü ile sürülmesi.

7.5. Çıkış Bağlantı Şekilleri

Bir lojik devre, genel olarak, kapı, MUX, saklayıcı, bellek vs. gibi birden çok birimin birbiriyle bağlanmasıyla oluşturulur. Dolayısıyla birinin çıkışı diğerinin girişine bağlanır. Aksi söylenmediği sürece, lojik birimlerin çıkışları birbirine bağlanmaz; bağlandığı taktirde kısa devre oluşur. Bu tür çıkış özelliği totem-pole olarak adlandırılır. Ancak, ortak yol kullanımı gibi bazı özel hallerde devre çıkışlarını tek bir yol üzerinden bir başka yere aktarmak istenebilir. Bu durumda bir eleman bu ortak yolu kullanırken diğer elemanların sanki bu hatta bağlı değilmiş gibi çalışması beklenir. Bu çalışma biçimi iki farklı yapı ile gerçekleştirilebilir. Bu yolların birincisi 3-durumlu elemanlar (3-*state buffer*)'ın kullanılmasıdır. 3-durumlu elemanın çalışması tümleme işlemi yapmayan bir TÜMLEME elemanı gibidir. Bu elemanlara tampon (*buffer*) adı verilir. Ancak bu devrenin yapısına ilave olarak devreyi aktif ya da pasif hale getiren bir üçüncü giriş de mevcuttur. Bu girişe izin ucu adı verilir.

3-durumlu elemanın özelliği, eğer izin aktif ise girişteki değer çıkışa aktarılır. Ancak izin pasif yapılırsa bu durumda devrenin çıkışı yüksek empedans durumu alır ve sanki bağlı değilmiş gibi davranır. Yani, başka bir deyişle açık devre gibi davranır. 3-durumlu elemanın yapısı Şekil-7.26 da, giriş değerlerine göre çıkış değerlerinin değişimi ise Tablo-7.11 de verilmiştir. Şekil-7.26.a daki lojik 0 (tümleyen) çıkış izinli 3-durumlu eleman Tablo-7.11.a daki doğruluk tablosu ile ifade edildiği biçimde çalışır. Kısaca, izin ucu lojik 0 ise girişteki bilgi çıkışa aktarılır, çıkış izni lojik 1 ise 3-durumlu elemanın çıkışı yüksek empedans durumuna çekilerek açık devreymiş gibi; sanki çıkışı bir hatta bağlı değilmiş gibi çalışır.



Şekil-7.26. 3-durumlu elemanın genel yapısı

Şekil-7.26.b deki lojik 1 çıkış izinli 3-durumlu eleman Tablo-7.11.b deki doğruluk tablosu ile ifade edildiği biçimde çalışır. Kısaca, çıkış izni lojik 1 ise girişteki bilgi çıkışa aktarılır; çıkış izni lojik 0 ise 3-durumlu elemanın çıkışı yüksek empedans durumuna çekilerek açık devreymiş gibi; sanki bir hatta bağlı değilmiş gibi çalışır.

Tablo-7.11. 3-durumlu elemanın çalışma biçimi.

Çıkış izni	Giriş	Çıkış
0	0	0
0	1	1
1	0	Yüksek Empedans
1	1	Yüksek Empedans

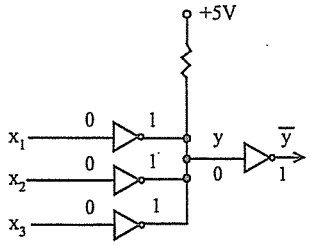
(a)

Çıkış izni	Giriş	Çıkış
0	0	Yüksek Empedans
0	1	Yüksek Empedans
1	0	0
1	1	1

(b)

Ortak yol kullanımında tercih edilecek ikinci tür elemanlar "*açık kollektörlü*" (*Open Collector*) çıkışa sahip elemanlardır. Elektronik sektöründe bir çok lojik kapının ve mevcut tümdevrelerin açık kollektörlü çıkışa sahip olan türleri de bulunmaktadır. Bu türden elemanların kullanımında çıkış uçlarına *pull-up* direnci adı verilen dirençler bağlanmalıdır. Bir ortak yolun kullanımında açık kollektörlü uçlar bir ortak hatta bağlanır ve bu ortak hat bir *pull-up* direnci ile sonlandırılır. Devrenin bu yapısı 3-durumlu elemanların kullanımıyla bir benzerlik gösterir.

Yandaki şekilde bir ortak yola bağlanmış açık-kollektörlü (*open-collector*) TÜMLEME kapıları görülmektedir. Burada her üç girişe de 0 uygulanması durumunda tümleyici çıkışlarında lojik 1 oluşur, bu durumda y çıkışı lojik 0'a çekilir. Çıkış fonksiyonu ise $\overline{y}=\overline{(x_1+x_2+x_3)}$ şeklinde ifade edilebilir; böylece çıkış ifadesinin VEYA işleminden ibaret olduğu görülebilir.

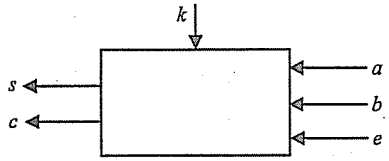


7.6. Özet

Belirli bir anda, çıkışları doğrudan o andaki girişlere bağlı olan devreler kombinezonsal devre olarak adlandırılır; bu tür devrelerde geçmiş bilgilerin tutulması gerekmediğinden saklayıcı, bellek gibi elemanlar kullanılmaz. En temel kombinezonsal devre lojik kapılardır. Ancak kapıların biraraya getirilmesiyle belirli bir işe odaklanmış toplayıcı, çıkarıcı, kodlayıcı, kod çözücü, seçici, dağıtıcı gibi birimler oluşturulur. Lojik tasarımlarda bu birimler oldukça fazla kullanılır. Bu birimler hem tümdevre olarak üretilmişlerdir, hem de ASIC yardımcı araçlarında hazır olarak kütüphanelerde vardır.

7.7. Sorular

1. Aşağıdaki şekilde 4 giriş (*a, b, e, k*) ve 2 çıkışlı (*c, s*) devre, k simgeli girişi 0 ise tam toplayıcı olarak, 1 ise tam çıkarıcı olarak çalışmaktadır. Bu devreye ilişkin fonksiyonun:
   a) Probleme ilişkin doğruluk tablosunu oluşturunuz.
   b) Çıkış fonksiyonlarını *Karnaugh* Diyagramları yöntemiyle bulunuz;
   c) Devreyi sadece TVE kapıları kullanarak tasarlayınız ve lojik şemayı çiziniz.



2. Bir Hava alanından şehir merkezine 20 dakikada bir helikopter seferi düzenlenmektedir. Helikopterlerde bulunan bir sayıcı doğal ikili kodda helikoptere binen yolcuları saymaktadır. Helikopterler 7 kişiliktir. Helikopterlerin kalkması için;
   - Kalkış zamanı henüz gelmediyse, helikopterin dolmuş olması
   - Kalkış zamanı geldiyse içinde en az iki yolcunun bulunması gereklidir.

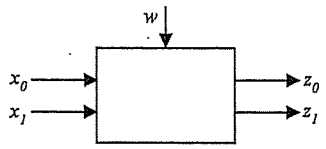
Kalkış zamanı geldiğinde, helikopterdeki zaman saatinin T çıkışı lojik 1 olmakta, aksi durumda T lojik 0 olmaktadır.

Yukarıda verilen bilgiler uyarınca, pilota, kalkışa hazır işareti verecek bir lojik devreyi tasarlayınız. Anlaşılacak şekilde açıklamalar eklemeyi unutmayınız.

3. Tek hatlı bir tren yolunun karayolu ile kavşak oluşturduğu yerde karayolu trafiğini düzenlemek için kırmızı ve yeşil trafik işaretleri yerleştirilecektir. Tren yolu üzerinde kavşaktan her iki yönde 2 km. uzaklıkta iki duyarga bulunmaktadır. Bu duyargalar, katar üzerinden geçerken lojik 1, diğer durumda lojik 0 işaretlerini üretmektedirler. Katarın kavşağa en yakın ucunun kavşağa 2 Km.nin altında olduğu sürece karayolunun trafiği kapanması (kırmızı lamba yanarak), aksi durumda karayolunun açık tutulması (yeşil lamba yanarak) istenmektedir.

   Katarın uzunluğu hiçbir zaman 4 km.ye ulaşmadığına göre trafik ışıklarının kontrolünü gerçekleyen devreyi tasarlayınız.

4. Aşağıda blok çizimi verilmiş devre aşağıda ifade edildiği şekilde çalışmaktadır.



$w=0$ ise	$z_0=x_0$	$z_1=x_1$
$w=1$ ise	$z_0=x_1$	$z_1=x_0$

   a) Bu devreye ilişkin çıkış fonksiyonun lojik ifadelerini en yalın bir şekilde yazın,
   b) Devreyi TVEYA bağlaçları kullanarak gerçekleştirin.
   c) Devreyi veri-seçici devrelerle (MUX) gerçekleştirin.

5. Bir koridor aydınlatma lambası iki anahtarla kumanda edilecektir. Anahtarlardan biri konum değiştirdiğinde lambanın bulunduğu durumdan diğer duruma geçmesi istenmektedir.

   a) İki anahtar ve bir lamba bulunan bu devreyi çizin. (anahtarlardan ikisi de "1" ve ikisi de "0" konumda olduğunda lamba yansın.)
   b) Oluşturulan devre hangi fonksiyonu gerçekleştirmektedir? 4 adet TVEYA bağlayıcı kullanarak bu fonksiyona ilişkin devrenin lojik çizimini yapın.

6. 4x1 bir veri seçiciyi TVE kapı elemanları ve tümleyicilerle oluşturunuz.

7. İki BCD sayının toplamını yapan ve sonucu BCD olarak veren bir devre tasarlanacaktır.

   a) İki BCD sayı 4 bitlik bir ikili toplayıcının girişlerine uygulanırsa elde edilebilecek farklı sonuçları bir tablo halinde yazınız.
   b) Bu tablonun hangi sonuçlarına belirleyeceğiniz sabit bir sayı eklenirse sonuçlar BCD olarak düzeltilmiş olur?
   c) b- şıkkında elde edilen tablonun hangi sonuçları bir elde çıkışı oluşturmalıdır? Bu elde çıkışının cebirsel ifadesini ikili toplayıcının çıkışları cinsinden ifade ediniz.
   d) İki adet 4 bitlik ikili toplayıcı ve gerekli TVE kapılarını kullanarak BCD toplayıcı devresi tasarlayınız.

8) Aşağıda lojik fonksiyonlar tanımlanmıştır.

$$F_1 = \bar{x}\cdot\bar{y} + x\cdot y\cdot\bar{z}$$
$$F_2 = \bar{x} + y$$
$$F_3 = x\cdot y + \bar{x}\cdot\bar{y}$$

Bu fonksiyonları,

   a) Bir kod çözücü (decoder) ve TVE (NAND) kapılarıyla gerçekleyin.
   b) Bir dağıtıcı (demultiplexer) ve TVE (NAND) kapılarıyla gerçekleyin.

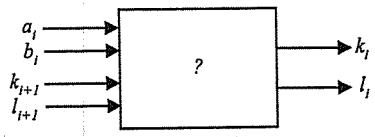
9. 3 bitlik xyz bilgisine bir de p eşlik (parity) biti $p = x + y\ z$ kuralıyla eklenerek bir bilgi iletim ortamından bir alıcı cihaza gönderilmektedir. Alıcı cihazda gelen bilgilerin iletim ortamında bozulup,bozulmadığını test eden bir Eşlik Biti Kontrolörü vardır. Eşlik biti kontrolörüne ilişkin lojik fonksiyonu bulun, uygun bulduğunuz tek tip bağlaçlarla lojik çizimi yapın.

10. Aşağıda lojik ifadesi verilen genel fonksiyon 4 girişli 2 çıkışlı bir devreye aittir. Girişlerine sadece doğal BCD kodundaki sayılar uygulanan bu devrenin aşağıda verilen maliyet ölçütüne göre en düşük maliyetle tasarlanması istenmektedir.

$$f_1(a, b, c, d) = \bar{c}\cdot\bar{d} + b\cdot d + \bar{a}\cdot\bar{b}\cdot c$$
$$f_2(a, b, c, d) = \bar{b}\cdot\bar{d} + b\cdot d + b\cdot c + a\cdot c + c\cdot\bar{d}$$

   a) f_1, f_2 fonksiyonlarını *Karnaugh* Diyagramları ile çarpımlar toplamı ve toplamlar çarpımı terimleri bulacak biçimde indirgeyiniz.
   b) Her iki indirgenmiş fonksiyonu TVEYA kapıları ile gerçekleyiniz. Hangi tasarımın karmaşıklık açısından daha uygun olduğunu belirleyiniz.

11. Aşağıda solda yer alan doğruluk tablosunda k_i ve l_i, iki değişkenin (a_i ve b_i) aldığı değerlerin eşit ya da birinin diğerinden büyük ya da küçük olduğunu göstermektedir.



a_i	b_i	k_i	l_i
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

Bu tablonun mantığını kullanarak, 2 tabanında yazılmış n basamaklı iki işaretsiz sayı karşılaştırılacaktır. Bu karşılaştırmada,yukarıda sağda blok diyagrami verilmiş olan devreden her basamak için bir adet olmak üzere, n adet kullanılacaktır. (k_{i+1}, l_{i+1}) bir üst basamağa ilişkin karşılaştırma sonuçlarıdır.

   a) $k_i = f_1(a_i, b_i, k_{i+1}, l_{i+1})$ ve $l_i = f_2(a_i, b_i, k_{i+1}, l_{i+1})$ fonksiyonlarının doğruluk tablosunu oluşturun.
   b) Bu fonksiyonları gerçekleştiren devreyi TVE kapılarıyla tasarlayın.
   c) Aynı devreyi veri seçiciler ile tasarlayın.

d) $n = 4$ için , iki sayının karşılaştırılmasında kullanılabilecek genel devreyi yukarıda tasarladığınız bloklar yardımı ile oluşturun. (Sadece blok çizimi yapılacaktır.)

12. Sayısal ifadesi $F = \Sigma_1 (0,2,3,4,5,7,8,13,14,15) + \Sigma_\emptyset (6,10,11)$ olan fonksiyonun,

a) *Karnaugh* diyagramını çiziniz.

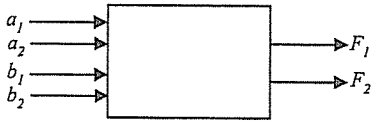
b) *Karnaugh* diyagramını çarpımlar toplamı terimleri bulacak biçimde indirgeyiniz.

b) Bu fonksiyonun lojik çizimini önce VE, VEYA, TÜMLEME kapıları ile, sonra da TVE kapıları ile gerçekleyin.

13. İki tabanında ikişer bitlik iki sayı

$A = a_1 a_2$
$B = b_1 b_2$

sayılarını karşılaştıran devrenin iki çıkışı F_1 ve F_2 dir,



Bu devrede

$A < B$ ise $F_1 = 0$ ve $F2 = 0$
$A > B$ ise $F_1 = 1$ ve $F2 = 0$
$A = B$ ise $F_2 = 1$

değerlerini alacaklardır.

a) F_1 fonksiyonuna ilişkin doğruluk tablosunu oluşturunuz ve Karnaugh diyagramı ile fonksiyonu çarpımlar toplamı şeklinde indirgeyiniz.

b) F_1 e ilişkin devrenin lojik çizimini TVEYA kapıları ile gerçekleyin

c) F_2 fonksiyonuna ilişkin doğruluk tablosunu oluşturunuz ve *Karnaugh* diyagramı ile fonksiyonu çarpımlar toplamı şeklinde indirgeyiniz.

d) F_2 ye ilişkin devreyi iki tane EŞDEĞERLİK (TYA DA) ve bir tane TVEYA kapısı ile gerçekleyin.

14. 4 girişli 1 çıkışlı lojik devre girişine uygulanan doğal BCD kodundaki sayı 2 veya 3'e tam bölünebiliyorsa çıkışından lojik "1" işareti üretmektedir.

a) Devrenin doğruluk tablosunu oluşturunuz.

b) Çıkış ifadesiniz indirgeyiniz.

c) Devreyi TVE bağlaçlarıyla gerçekleştiriniz.

15. Bir oylamaya katılan 4 kişi kararlarını önlerinde bulunan butonlara basarak belirtiyorlar. Teklifi kabul eden kişi butona basmakta aksi halde basmamaktadır. Salonda bulunan yeşil ve kırmızı renkli 2 lamba ile oylama sonucu aşağıdaki gibi belirlenmektedir;

Kabul oyları 0 veya 1 ise sadece kırmızı lamba yanıyor.

Kabul oyları 2 ise her iki lamba da yanıyor.

Kabul oyları 3 veya 4 ise sadece yeşil lamba yanıyor.

Not: Oylamaya geçilmeden önce lambaların yanmaması için oturum başkanının önünde lambalara gerilim gelmesini engelleyen bir anahtar bulunmaktadır.

a) Bu düzeni sağlayacak devreye ilişkin doğruluk tablosunu oluşturunuz

b) Devrenin tasarımı için kendinize göre en uygun çözümü öneriniz.

16. Bir tam toplayıcının veri girişleri A_i ve B_i , elde girişi E_i , sonuç T_i , elde çıkışı da $Ç_{i+1}$ ile simgelendirilmiştir.

a) $P_i = A_i + B_i$ ile $G_i = A_i B_i$ olarak verilirse,bu tam toplayıcının giriş-çıkış bağıntılarını aşağıdaki gibi belirleyen f_1 ve f_2 fonksiyonlarının cebirsel ifadelerini yazınız

$T_i = f_1 (P , E)$ ile $Ç_{i+1} = f_2 (G_i , P_i , E_i)$

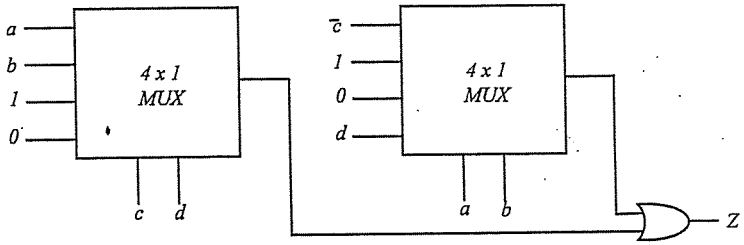
Devrenin lojik çizimini YA DA, VE, VEYA bağlaçları kullanarak gerçekleştiriniz.

17. 2 tane Seçici (MUX) ve bir VEYA kapısı içeren devreye ilişkin devre şeması aşağıda verilmiştir.

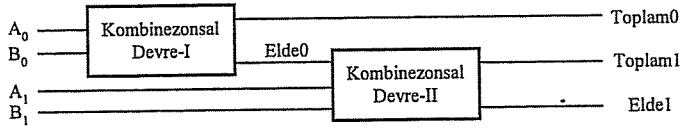
a) Devreye ilişkin lojik fonksiyonu belirleyiniz.

b) Fonksiyona ilişkin doğruluk tablosunu oluşturunuz.

c) *a, b* ve *c*'yi seçme girişleri alarak devreyi 8x1 Seçici (MUX) ile gerçekleyiniz.



18. A_1A_0 ve B_1B_0 şeklinde verilen iki bitlik iki tabanındaki iki sayıyı toplama işlemi gerçeklenecektir. Bu amaçla aşağıda verilen iki adet kombinezonsal devre kullanılacaktır.



a) Kombinezonsal Devre-I, $(A_0)(B_0)$ girişlerini toplayıp, (Elde0)(Toplam0) çıkışlarını verecek bir devredir. Girişlerle, (Elde0) ve (Toplam0) çıkışlarını göz önünde bulundurarak, devreye ilişkin doğruluk tablosunu oluşturunuz ve devreyi lojik kapı elemanları ile gerçekleştiriniz, devre şemasını çiziniz.

b) Kombinezonsal Devre-II, (Elde0)(A_1)(B_1) girişlerini toplayıp, (Elde1)(Toplam1) çıkışlarını verecek bir devredir. Girişlerle, (Elde1) ve (Toplam1) çıkışlarını göz önünde bulundurarak, devreye ilişkin doğruluk tablosunu oluşturunuz ve devreyi lojik kapı elemanları ile gerçekleştiriniz, devre şemasını çiziniz.

19. Üç katlı bir apartmanda merdiven aydınlatması şu şekilde yapılmaktadır: Her katta bir anahtar bulunmaktadır. Bu anahtarların konumuna göre katlarda bulunan lambaların hepsi birden yanmakta veya sönmektedir. Bu anahtarlar x_1, x_2, x_3 ile lambalar ise L ile gösterilmektedir. Anahtarın kapalı durumu 1, açık durumu 0, lambanın yanma konumu 1, sönme konumu 0'dır. Çalışma düzeni şöyledir:

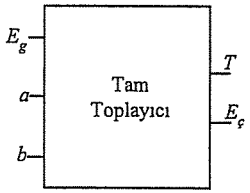
Her üç anahtar da devreyi açık bırakan konumdayken lamba yanmamaktadır. Anahtarlardan herhangi biri devreyi kapatırsa lamba yanacaktır. İki anahtar birden devreyi kapatırsa lamba yanmamaktadır. Her üç anahtar da kapalıysa lamba yanacaktır.

a) Bu devreye ilişkin doğruluk tablosunu oluşturunuz.

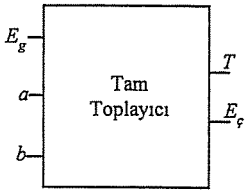
b) Bu tabloyu kullanarak devre çıkışına ilişkin çarpımlar toplamı kanonik biçimini elde ediniz.

c) x_1 ve x_2'yi seçme girişleri alarak, devreyi MUX elemanı kullanarak gerçekleyiniz.

20. İki tabanında verilmiş üç bitlik $(x_1, x_2, x_3)_2$ sayısı ile $(y_1, y_2, y_3)_2$ sayısı yanda lojik şeması verilen 1-bitlik Tam Toplayıcılarla (TT) toplanmak istenmektedir. TT'ları yanda verildiği biçimde kullanarak tasarımı gerçekleştiriniz. Devreye ilişkin giriş ve çıkış değişkenlerini göstererek devre bağlantı şemasını çiziniz.



21. İki tabanında verilmiş dört bitlik $(x_4, x_3, x_2, x_1)_2$ sayısı ile $(y_4, y_3, y_2, y_1)_2$ sayısı yanda lojik şeması verilen Tam Toplayıcılarla toplanmak istenmektedir.



a) 1-bitlik Tam Toplayıcı devreyi lojik kapı elemanlarıyla tasarlayınız.

b) Tam Toplayıcıları (TT) yanda verildiği biçimde kullanarak bütün devreye ilişkin giriş ve çıkış değişkenlerini ifade ederek devre bağlantı şemasını çiziniz.

22. 16 Kombinasyonlu bir kombinezonsal devreyi göz önünde bulundurunuz. Bu devrede ilk dört ve son dört kombinasyon kesinlikle değerlendirilememektedir. En az iki giriş Lojik 1 olduğunda devrenin çıkışında Lojik 1, aksi taktirde çıkışında Lojik 0 veren devre tasarlanacaktır. Bu amaçla,

a) Probleme ilişkin doğruluk tablosunu oluşturunuz.

b) Çarpımlar toplamı ifadesini elde edecek biçimde ifadeyi *Karnaugh* Diyagramı ile indirgeyiniz.

c) İndirgenmiş ifadeye ilişkin devreyi temel kapı elemanlarıyla gerçekleyiniz.

d) İndirgenmiş ifadeye ilişkin devreyi sadece TVE kapılarıyla gerçekleyiniz.

e) İndirgenmiş ifadeye ilişkin devreyi sadece TVEYA kapılarıyla gerçekleyiniz.

f) Tasarım kriterleri açısından c), d) ve e) tasarımlarını karşılaştırınız. Hangi tasarımın neden daha iyi bir seçim olacağını açıklayınız.

23. Üç girişi ve bir çıkışı bulunan bir Kombinezonsal Devre (Çoğunluk Detektör Devresi) girişlerin çoğunda Lojik 0 işareti varsa, çıkışında Lojik 0, girişlerin çoğunda Lojik 1 işareti varsa çıkışında Lojik 1 üretmektedir.

a) Bu devreye ilişkin doğruluk tablosunu oluşturunuz.

b) Doğruluk tablosunu kullanarak lojik fonksiyonu *Karnaugh* diyagramıyla çarpımlar toplamı ifadesini bulacak biçimde indirgeyiniz.

c) İndirgenmiş ifadeye ilişkin devreyi sadece iki girişli TVE kapılarıyla gerçekleyiniz.

24. Dört araçlık bir otopark aydınlatması şu şekilde yapılmaktadır:Park yerlerinde A_1, A_2, A_3, A_4 şeklinde adlandırılan algılayıcılar bulunmaktadır. Bu algılayıcılardan alınan işaretler değerlendirilmekte ve otoparkın üstünde bulunan lambanın yanması veya sönmesi sağlanmaktadır. Algılayıcıların park yerinde araç olduğunu belirten durumu Lojik 1, araç olmadığını belirten durumu Lojik 0, lambanın yanık olma durumu Lojik 1, sönük konumu ise Lojik 0'dır. Devrenin çalışma düzeni şöyledir: Otoparktaki araç sayısı üç ve üzerinde ise lamba yanacaktır. Birinci ve üçüncü araç sahiplerinin ayrıcalıkları vardır. Bu araçlar tek başlarına bile olsalar otopark ışığı yanacaktır.

a) Bu devreye ilişkin doğruluk tablosunu oluşturunuz.

b) Lojik fonksiyonu *Karnaugh* diyagramıyla çarpımlar toplamı ifadesini biçiminde indirgeyiniz.

c) İndirgenmiş ifadeye ilişkin devreyi temel kapı elemanlarıyla gerçekleyiniz.

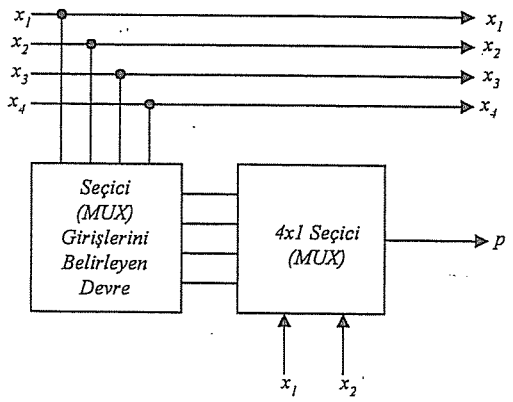
d) İndirgenmiş ifadeye ilişkin devreyi sadece TVE kapılarıyla gerçekleyiniz.

e) İndirgenmiş ifadeye ilişkin devreyi sadece TVEYA kapılarıyla gerçekleyiniz.

f) Tasarım kriterleri açısından c), d) ve e) tasarımlarını karşılaştırınız. Hangi tasarımın neden daha uygun bir seçim olacağını açıklayınız.

25. BCD kodunda (0-9 arası) verilmiş olan bir sayı bir sistemden uzakta bulunan bir başka sisteme aktarılacaktır. Herhangi bir hataya sebebiyet vermemek için bu bilgi ile bir de eşlik biti gönderilecektir. Şekilde verildiği üzere, BCD kodunda 1'lerin sayısını tek yapan eşlik (parity) biti üretecini x_1 ve x_2'yi seçme girişleri alarak Seçici (MUX) elemanı ile gerçekleyiniz.

Yol Gösterme: Doğruluk tablosunu dört giriş-bir çıkışlı devre için çıkışta 1'lerin sayısı tek olacak biçimde oluşturunuz. Tabloyu Çarpımlar Toplamı Kanonik biçiminde ifade ediniz. x_1 ve x_2 kontrol girişleri olacağından çarpımlar toplamı ifadesini x_1x_2'nin lojik bileşenlerine göre terimlere ayırarak Seçici (MUX) girişlerini belirleyiniz ve MUX girişlerini çiziniz.



8.

Maliyet Faktörü ve Karmaşıklık Hesabı

Bir sayısal sistem tasarlanırken en önemli unsurlardan iki tanesi *maliyet* ve *devrenin çalışma* hızıdır. Bu ikisine ek olarak harcadığı güç miktarı, tasarım süresi, uyumluluğu, transistör sayısı, gerçeklenebilirliği gibi faktörler de tasarım süresince göz önünde bulundurulması gereken etkenlerdir. *Maliyet*, sistemin tasarlanması ve fiziksel olarak gerçeklenmesi için yapılması gereken harcamalar ve geçen süreyi gösterir. *Çalışma hızı* ise, belirlenen işin ne kadar birim zamanda yapılabildiği gösterir. Çünkü, belirli bir iş, belirli bir zaman dilimi içerisinde yapılmalıdır; aksi halde o işin yapılmasının bir anlamı olmayabilir. Bu bölümde, lojik devre tasarımı yapılırken gözönüne alınması gereken maliyet unsurları ve çalışma hızının hesaplanmasında da kullanılan karmaşıklık hesabı konusu aşağıdaki başlıklar altında ele alınmıştır:

8.1. Maliyet Hesabı
8.2. Karmaşıklık Hesabı
Zaman ve Eleman Karmaşıklığı
8.3. Büyük O Gösterilimi ve Karmaşıklık
8.4. Özet
8.5. Sorular

Maliyet hesabı, tasarım maliyeti ve gerçekleştirme maliyeti olarak iki açıdan ele alınır. Eğer, tasarlanan devre milyonlarca üretilecekse tasarım maliyeti ihmal edilebilir; ancak az sayıda üretilecekse, sonuçta, elde edilecek ürünün/devrenin maliyetini belirleyecek unsurdur. Gerçekleştirme maliyeti, tasarlandıktan sonra devrenin fiziksel olarak bir kart üzerinde yapılması veya ayrık bir tümdevre olarak üretilmesi için harcanması gereken miktardır; burada, devrenin ne kadar lojik eleman kullanılarak gerçekleştirildiğini gösteren eleman karmaşıklığı önemli bir etmendir.

Çalışma hızı ise, devrenin, yapması için tasarlanmış olduğu işi ne kadar birim zamanda yaptığını gösteren bir sayıdır; bu, kombinezonsal devreler veya asenkron ardışıl devreler için toplam propagasyon gecikmesi iken, senkron ardışıl devrelerde saat çevrimi sayısıdır. Örneğin, bir işlemci, genel olarak senkron ardışıl olarak tasarlanır ve toplama, öteleme, çarpma gibi işlemleri kaç saat çevrimde yaptığından

söz edilir; çünkü, oradan da süre hesaplanabilir. Lojik devre tasarlanarak gerçeklenmesi istenen bir sistemin tasarlanması için birçok yol/yöntem vardır; bazıları, devrenin çok hızlı çalışmasını sağlayacak bir mimariye sahip iken, bazıları da devrenin çalışma hızından bir miktar fedakarlık yapılması durumunda daha az lojik eleman kullanılarak tasarlanmasını sağlar. Devrenin çalışma hızı ve lojik eleman miktarı karmaşıklık (*complexity*) hesabı konusuna girer; eleman karmaşıklığı (*component complexity*) ve zaman/hız karmaşıklığı (*time complexity*) olarak adlandırılır. Eleman karmaşıklığı, devrenin temel tasarım birimlerinden ne kadar kullanıldığı veya kullanılacağını gösteren bir bağıntıdır; bağıntı içerisine parametrelerin gerçek değerleri koyulduğunda toplam temel eleman sayısı bulunur. Zaman karmaşıklığını, temel lojik elemanın propagasyon gecikmesine bağlı bir bağıntıdır.

- Maliyet
  - tasarım; ne kadar gün/adam'da tasarlandığı
  - üretim; birim maliyeti/harcaması (eleman karmaşıklığı etkin)
- Karmaşıklık
  - eleman karmaşıklığı; kullanılan temel lojik birim/eleman sayısı
  - zaman (hız) karmaşıklığı; girişle çıkış arasındaki gecikme süresi

Karmaşıklık hesabı ise, daha çok tasarım aşamasında karşımıza çıkar; eleman karmaşıklığıyla zaman karmaşıklığının ters orantılı olduğu söylenebilir. Tasarlanan devrenin çok hızlı olması istenirse, çok fazla lojik eleman kullanılmaya dayanan tasarım yöntemleri kullanılmak zorunda kalınabilinir. Tabi ki, uygulamaya en uygun mimarinin seçilmesi de oldukça önemlidir.

8.1. Maliyet Hesabı

Bir lojik devrenin maliyeti, onu tasarlayıp üretmek için yapılması gereken harcamadır. Bu, hem tasarım hem de üretim aşamasını kapsar. Tasarlanan lojik devre bir tümdevre olarak üretilecekse bir ASIC yardımcı aracı kullanılır; bunun aracılığıyla lojik devre en üst seviyeden başlanarak tasarlanır, örnek giriş değerlerine göre doğru çalışıp çalışmadığı sınanır ve davranışı gözlenir; herşey tasarım doğrultusunda sonuç vermişse tümdevrenin fabrika ortamında üretilmesi için gerekli katman yüzey haritaları elde edilir. Bu aşamada tasarım bitmiştir denilebilir. Artık, tümdevrenin fabrika ortamında üretilmesi ve fiziksel sınamasının yapılması aşamasına gelinir. Bu durumda lojik devrenin tasarım maliyeti,

$$M_{tasarım} = M_{insan} + M_{yardımcı\ Araç} \qquad (8.1.)$$

şeklinde hesaplanır. Toplam maliyet ise,

$$M = M_{tasarım} + M_{üretim} \qquad (8.2.)$$

bağıntısından hesaplanır. Tasarlanan lojik devre bir kart üzerinde gerçekleştirilecekse, yine bir yardımcı araç yardımıyla tasarlanıp test edilebilir; ardından kart üzerine yerleştirilmesi için gerekli baskılı devre oluşturulur ve tasarımda kullanılan elemanlar kart üzerine yerleştirildikten sonra lehimleme işi yapılarak devreye ait kart elde edilir; tabi ki, kartın fiziksel testi de yapılmalıdır. Kart tasarımında yardımcı araç maliyeti ihmal edilebilir[1].

$$M_{tasarım} = M_{insan} \qquad (8.3.)$$

$$M = M_{tasarım} + M_{üretim} \qquad (8.4.)$$

Sonuç olarak bir lojik devrenin maliyeti insan kaynaklarına, sahip olunan tasarım araçlarına (yazılım ve donanım) ve fabrika ortamında üretim giderlerine bağlıdır. Fabrika ortamında üretim maliyetini de, hem devrenin eleman karmaşıklığı hem de seçilecek mikroelektronik teknolojisi yakından etkiler!

Örnek-8.1.

Bir proje ekibine, 1000 adet üretilmesi düşünülen mikroişlemci tasarımı yaptırılması istenmektedir; 5 kişiden oluşan proje ekibindeki bir üyenin ortalama aylık gideri 1.000 YTL civarındadır. Mikroişlemcinin fabrika ortamında üretim maliyeti 100adet/2000 YTL tutarındadır. Proje tasarımına Ocak ayında başlayıp Haziran sonunda bitirecek olan ekip mikroişlemci üretildikten sonra da 2 ay fiziksel test yapacaktır. Bu koşullar altında bir adet tümdevrenin maliyeti ne kadar olur? Tasarım aracı maliyetiyse aylık 500 YTL'dir.

Bir adet mikroişlemcinin maliyeti için önce 8.1. ve 8.2. bağıntıları uyarınca toplam maliyet bulunmalıdır:

$$M_{insan}=(5\cdot(6+2))= 40.000\ YTL$$

$$M_{yardımcı\ Araç}=(500\cdot 6)=3.000\ YTL$$

$$M_{üretim}=1000\cdot(2000/100)= 20.000\ YTL$$

$$M= M_{insan}+ M_{yardımcı\ Araç}+M_{üretim}$$

$$M_{toplam}= 40.000 + 3.000 + 20.000 = 63.000\ YTL$$

$$M_{birim}= 63.000\ YTL\ /1000= 63\ YTL$$

Verilen koşullar altında bir tümdevre 63 YTL maliyetle üretilmektedir. Bu örnek için tasarım maliyeti daha baskındır. Eğer adet sayısı artırılmış olsaydı, örneğin 10 bin adet üretilmiş olsaydı birim maliyet 14,3 YTL olurdu.

[1] Kart düzeyinde tasarım yapacak yazılımların maliyetleri ASIC tasarımı yapılacak yazılımların maliyetinden oldukça azdır. Kart tasarımı yapılan bu tür yazılımlar standart PC üzerinde çalışırken ASIC tasarımı yardımcı araçları, daha çok, UNIX tabanlı güçlü sistemlere ihtiyaç duyarlar. Böyle bir sistem, Türkiye'de ETA laboratuvarlarında vardır; ihtiyaç duyan firmalar için özel amaçlı tümdevreler tasarlanıp ürettirilmektedir.

8.2. Karmaşıklık Hesabı

Karmaşıklık hesabı[2], devrenin ne kadar temel lojik birimi kullanılarak gerçekleştirileceğini ve fonksiyonunu yerine getirebilmesi için ne kadar süre geçeceği belirlemekte veya hesaplamakta kullanılır; ilkine *eleman karmaşıklığı*, ikincisine de *zaman karmaşıklığı* denilir. Öyle ya, devre tasarımı için sonsuz kaynak ve sonsuz süre yoktur; gerçek dünyada sınırlı miktarda kaynak (ki olabildiğince az kullanılması istenir) ve sınırlı süre vardır. Tüm bunlar tasarım öncesi veya aşamasında gözönünde bulundurulmalıdır.

- Eleman Karmaşıklığı (Component Complexity)
- Zaman Karmaşıklığı (Time Complexity)

Karmaşık sayısal sistem tasarlanırken tüm parametreleri de gözönüne alarak tasarım yapmak da oldukça karmaşık bir hal alır. Çünkü, genelde uygulamada, bir taraf iyileştirilirken diğer taraflar olumsuz etkilenir Ancak, tasarımda gözönüne alınması gereken en önemli iki unsur; maliyet ve hızdır. Tanımlanan işin kabul edilebilecek hızda optimum maliyetle tasarlanması beklenir. Tasarım maliyeti, zamanla ilgilidir; ne kadar kısa zamanda bitirilirse o kadar verimli çalışılmış olunur; gerçekleştirme maliyeti, tasarlanacak sistemde kullanılacak teknoloji ve en temel tasarım birimi, ki bu çoğu zaman lojik kapı veya transistör olarak ele alınır, sayısıdır. Aynı işi daha az sayıda transistör veya kapı içeren tasarımla yapmak daha iyidir ve aynı zamanda maliyeti de düşürür.

Eğer tasarım bir tümdevre olarak gerçeklenecekse eleman karmaşıklığı transistör sayısı olarak düşünülür. Eğer baskılı bir devre üzerinde bir kart şeklinde yapılacaksa temel/standart tümdevre veya lojik kapı sayısı açısından bakılır.

- **Tümdevre Sayısı**

  Sayısının az olması güç sarfiyatının az olmasını sağlar; tümdevreler arası bağlantı yolu az olur; kart tasarımı ve baskılı devre çizimi kolay olur.

- **Farklı Türde Tümdevrelerin Sayısı**

  Tasarımda aynı tür tümdevreler veya kapılar kullanılırsa, tümdevrelerin sahip olduğu tüm kısımlar kullanılabilir; boş kısım az kalır. Örneğin bir tane VEYA kapısına için 4 tane VEYA kapısı içeren 74LS32 tümdevresi kullanılırsa 3 tanesi boşa gitmiş olur. Bu nedenle lojik ifadelerin indirgenmesinde çoğunlukla aynı tip kapı kullanılmasına çalışılır; örneğin NAND kapısı gibi.

- **Tümdevre Uç Sayısı**

  Uç sayısı fazla olan tümdevreleri üretmek daha pahalı olur ve tümdevreler arası bağlantıyı karmaşıklaştır. Eğer zaman karmaşıklığı imkan veriyorsa uç sayısı az olan tümdevreler elaman karmaşıklığını azaltır.

[2] Bazı kaynaklarda karmaşıklık sözcüğü yerine maliyet kullanılmaktadır; bu kitapta maliyet sözcüğü farklı anlamda kullanılmıştır. Karmaşıklığı azaltmak maliyeti de etkileyen bir unsurdur. Ancak, karmaşıklık tasarımcı açısından, maliyet ise ticari açıdan anlamlı sözcüklerdir denilebilir.

- **Tümdevrenin Çalışma Hızı**

  Kendisinden beklenen hızı en kötü durumda sağlamalıdır; bunun için çalışma sıcaklığı koşullarına dikkat edilmelidir. Aşırı sıcakta veya soğukta çalışan tümdevreler zaman davranışlarından sapmalar gösterirler; bu nedenle, gerekirse fan benzeri bir sistemle soğutma yapılmalıdır.

- **Çıkış Uçlarının Yüklenme Kapasitesi**

  Çıkış uçlarının besleyeceği, yani bağlanacağı giriş uçları sayısını (*fanout*) gösterir. Bir çıkış ucu ancak belirli sayıda giriş ucunu sürebilecek güce sahiptir. Bu sayı TTL ve CMOS'lar için yaklaşık 10 verilebilir; yani bir TTL çıkışı 10 taneye kadar TTL girişe bağlanabilir. Benzer durum CMOS için de geçerlidir. Daha fazla girişi sürmesi için araya kuvvetlendirici anlamında kapılar koyulur.

- **Teknolojisi**

  Seçilebilecek birçok mikroelektronik teknolojisi vardır. Bunlar hız, güç harcaması ve üretim maliyeti açısından farklılık gösterir. Eğer devreden yüksek hızda çalışma beklenmiyorsa ve güç kaynağı büyüklüğü sorunu yoksa en az maliyetle gerçekleştirilecek teknoloji seçilir. Ancak, yüksek hızlara ve güç tasarrufuna ihtiyaç varsa seçilecek teknoloji daha maliyetli devre oraya çıkmasına neden olacaktır. Değişik teknolojiler ve özellikleri için bkz. Ek-A.

Zaman karmaşıklığı doğrudan girişle çıkış arasında bulunan lojik birimlerin gecikmelerinin toplamından bulunur. Birden çok girişli veya birden çok çıkışlı devrelerde en uzun gecikmeye sebep olan kısım alınır. Çünkü, en kötü durumda tüm girişlerin çıkışa yansıması gereklidir.

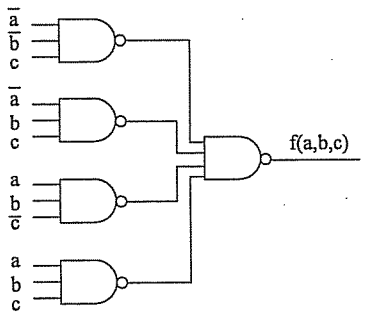
Örnek-8.2.

Yanda lojik şeması verilen ve TVE kapılarıyla gerçekleştirilmiş olan kombinezonsal devrenin toplam gecikmesi ve en fazla hangi hızda çalışabileceğini aşağıda belirtilen koşular altında hesaplayınız.

TVE kapısının gecikme süreleri:

0→1 : 9 ns (tipik)
15 ns (azami)
1→0 : 10 ns (tipik)
15 ns (azami)

(bu değerler T74LS10 tümdevresinden alınmıştır)



Şekil-8.1. TVE kapılarıyla oluşturulmuş bir devre.

Şekil-8.1 den de görüleceği gibi giriş ve çıkışlar arasında ikişer tane TVE kapısı vardır; yani tüm girişlerin çıkışa yansıması 2 tane TVE kapısının gecikmesiyle yansımaktadır. Hesaplama biri tipik diğeri en kötü durum olmak üzere iki açıdan yapılabilir: Şekil-8.1 de verilen devrenin gecikmesi tipik değerler için *20 ns*, en kötü durum için *30 ns* olur. Bu sonuçlara göre, devre girişleri, garantili çalışma için, *30 ns*'den daha kısa süre içerisinde değiştirilmemelidir. Aksi durumda anlamlı çıkışlar elde edilemez.

Devrenin çalışma frekansı toplam gecikmeyle elde edilen sürenin terslenmesiyle bulunur; yani, $frekans=1/gecikme_{toplam}$ bağıntısından bulunur.

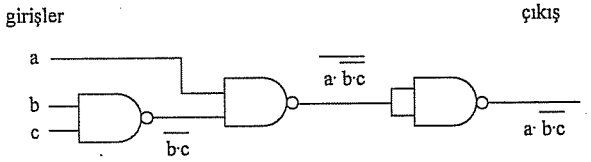
$$f= 1/30\ ns=1\ /\ (30\cdot 10^{-9}) \approx 33\ 000\ 000\ Hz \approx 33\ Mhz$$

Bu şekilde hesaplanan frekans değeri, *33 MHz* hiçbir zaman uygulanmaması gereken frekanstır; yani en büyük sınır değerdir ve girişlerdeki değişiklik bu değerin altında olmalıdır.

Örnek-8.3.

Şekil-8.2 de çizimle verilen ve TVE kapısıyla gerçekleştirilen lojik devrenin en kötü durumda toplam gecikmesinin Örnek-8.2 de verilen koşullar altında tasarlayınız. Herhangi bir anda giriş değerlerinin çıkışa yansıması için girişlerindeki değişkenlerin değiştirilme hızı /frekansı ne olmalıdır?

Bu devrede, giriş değişkenleriyle çıkış arasında homojen olmayan geçiş düzeyi sayısı vardır. Örneğin *a* giriş değişkeninin çıkışa yansıması için 2 TVE kapısında geçilirken *b* ve *c* değişkenlerinin çıkışa yansıması için 3 TVE kapısından geçilir. Böyle durumlarda en büyük gecikmeye neden olan geçiş düzeyi sayısı alınmalıdır. Şekil-8.2 deki devre için *b* veya *c* değişkenleri 3 TVE kapısı gecikmesiyle çıkışa yansırlar.



Şekil-8.2. TVE kapılarıyla oluşturulmuş bir devre (Örnek-8.3).

$$t_{gecikme}=maksimum(t_{a\text{-}çıkış},\ t_{b\text{-}çıkış},\ t_{c\text{-}çıkış}) \qquad (8.5.)$$

$$t_{gecikme}=maksimum(30,\ 45,\ 45)$$

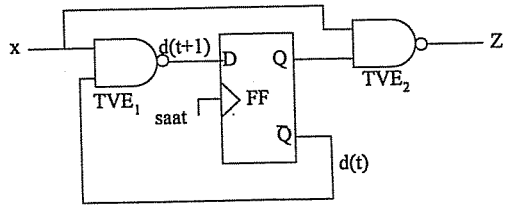
$$t_{gecikme}=45\ ns$$

bulunur. Frekans ise $f=1/\ t_{gecikme}$ bağıntısında $f=1/45 \approx 22\ Mhz$ olarak bulunur.

Örnek-8.4.

Şekil-8.3 de basit bir ardışıl devre verilmiştir; bu devreye ilişkin zaman karmaşıklığını hesaplayınız ve devreye uygulanabilecek en yüksek saat işareti frekansını aşağıdaki koşullar altında belirleyiniz.

TVE kapısının gecikme süreleri:	D Flip-Flop'un zaman parametreleri :
0→1 : 9 ns (tipik)	t_y : 20 ns
15 ns (azami)	t_t : 30 ns
1→0 : 10 ns (tipik)	t_g : 50 ns
15 ns (azami)	



Şekil-8.3. TVE kapılarıyla oluşturulmuş ardışıl devre (Örnek-8.4).

Ardışıl devrelere uygulanabilecek en yüksek saat işareti frekansı doğrudan devrenin toplam gecikmesine bağlıdır. Bu nedenle söz konusu devrenin, en kötü durumda, girişle çıkış arasındaki toplam gecikmesi hesaplanmalıdır; bu değerin tersi saat işaretinin alabileceği en yüksek frekansı gösterir.

Ardışıl devrelerde toplam gecikme iki ana parametreye bağlıdır; biri kombinezonsal devre, diğeriyse durum saklama biriminin gecikmesidir. Şekil-8.3 de verilen devrede TVE kapılarından oluşan iki tane kombinezonsal devre ve durum saklama birimi olarak da bir tane D türü flip-flop vardır. Saat işaretinin senkronlama anında $d(t+1)$ durumunu flip-flop'a yüklemek ve o andaki yeni çıkışı elde etmek şekilden de görüleceği gibi flip-flop zaman parametrelerine ve TVE_2 kapısının gecikmesine bağlıdır.

Ardışıl devrelerde toplam gecikme, senkronlama anından itibaren çıkışı ya da bir sonraki durum bilgisini elde etmek için oluşan gecikme belirler; hangisi daha büyükse o alınır. Yani, tam senkronlama anından sonra çıkışı (*Z*) elde etmek için ne kadar süre gerektiği ve bir sonraki durum bilgisini ($d(t+1)$) elde etmek için ne kadar süre gerektiği hesaplanır ve büyük olanı devrenin gecikmesi olarak kabul edilir.

Çıkış elde etmek için gecikme t_g(*çıkış*), durum bilgisini elde etmek için de t_g(durum) olarak adlandırılırsa aşağıdaki bağıntılar tanımlanabilir:

$$t_{gecikme}=maksimum(t_g(çıkış),\ t_g(durum)) \quad (8.6.)$$

$$t_g(çıkış)=t_g(FF)+t_g(TVE_2) \quad (8.7.)$$

$$t_g(durum)=t_g(TVE1)+t_y(FF)+t_g(FF) \quad (8.8.)$$

Bu bağıntılar uyarınca ilgili sayılar yerine koyulursa Şekil-8.3 de verilen ardışıl devrenin gecikmesi aşağıdaki gibi hesaplanır:

$$t_g(çıkış)=50+15=65\ ns$$

$$t_g(durum)=15+20+50=85\ ns$$

$$t_{gecikme}=maksimum(65,\ 85)=85\ ns$$

Devrenin gecikmesi 85 ns bulunur;frekansı ise aşağıdaki gibi bulunur:

$$f=1/t_{gecikme}$$

$$f=1/85\ ns=1/(85\cdot10^{-9})=10^9/85\approx 11{,}8\ MHz$$

Bu değer, yani 11,8 MHz üst sınırdır; uygulanacak saat işareti frekansı bu değerden küçük olmalıdır.

Flip-flop'un t_y, t_t ve t_g gecikme parametrelerinin açıklaması ve ayrıntılı bilgi için bakınız Ayrıt 10.1.6. Dikkat edilirse t_t gecikmesi hesaplamada kullanılmamıştır; bu değerin yalnızca t_g'den küçük olması yeterlidir!

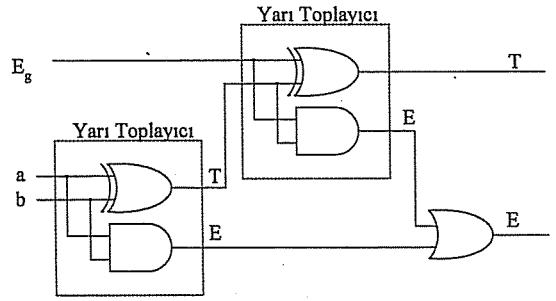
Eleman Karmaşıklığı

Eskiden sayısal devre tasarımında kullanılan temel tasarım birimleri/elemanları VE, VEYA, TÜMLEME gibi kapılardı. İster elle olsun ister VLSI tasarım araçlarıyla olsun tüm sistemler kapılara dayanılarak gerçekleştirildi. Ancak, günümüzde çok çeşitli ve tasarım kolaylığı sağlayacak temel birim olarak kabul edilebilecek birçok birim var; sayıcı, bellek, saklayıcı, toplayıcı, ALU gibi... Örneğin, bir mikroişlemci tasarımı bu temel birimler kullanılarak çok zorlanmadan tasarlanabilir. Eğer tasarımda kapılar kullanılıyorsa, devrenin tamamen aynı cins kapılarla tasarlanması maliyet açısında kazanç sağlayabilir. Dolayısıyla, lojik ifadelerin indirgenmesinde çoğunlukla tek tür kapı kullanılmasına uğraşılır; örneğin TVE kapısı gibi.

Eleman karmaşıklığı (*component complexity*), tasarlanan lojik devrenin temel olarak ele alınan lojik birinden kaç tane kullanıldığı gösteren bir bağıntıdır; bir veya birkaç lojik eleman temel alınabilir. Bu durumda herbiri bağıntıda yer almalıdır. Örneğin, temel lojik elemanları TVE kapısı ve 4 bitlik bir saklayıcı olan bir lojik devrenin eleman karmaşıklığını gösteren bağıntı, *EK= 33·TVE+ 7·Saklayıcı* ise bu devrenin kart şeklinde tasarlanırsa kaç tane tümdevreden oluşacağı veya ASIC olarak tasarlanırsa kaç tane transistör içereceği belirlenebilir.

Örnek-8.5.

Aşağıda 1 bitlik iki tane yarı toplayıcı kullanılarak gerçekleştirilmiş olan yine 1 bitlik bir tam toplayıcı görülmektedir (Şekil-7.11 den alınmıştır). Bu tam toplayıcının eleman karmaşıklığı temel birim yarı toplayıcı ise *EK=2·yarı toplayıcı*'dır. Eğer temel tasarım birimi kapılar ise eleman karmaşıklığını hesaplayınız. Eğer 4 bitlik tam toplayıcı tasarlanmak istenirse eleman karmaşıklığı ne olur?



Temel tasarımı birimi kapılar ise, şekilden görüleceği üzere tam toplayıcıda üç değişik kapı (VE, VEYA ve DVEYA) kullanılmıştır. Buna göre,

$$EK_{1\ bitlik\ tam\ toplayıcı}=2\cdot VE+1\cdot VEYA+2\cdot DVEYA$$

olarak bulunur. Bu 1 bitlik tam toplayıcı için eleman karmaşıklığıdır; 4 bitlik paralel tam toplayıcı için,

$$EK_{4\ bitlik\ tam\ toplayıcı}=8\cdot VE+4\cdot VEYA+8\cdot DVEYA$$

gibi bulunur.

Örnek-8.6.

Bir lojik devrenin cebirsel ifadesi/lojik fonksiyonu verildiğinde, devrenin tasarımı yapılmadan eleman karmaşıklığı hesaplanabilir mi? Örneğin, aşağıdaki cebirsel ifadeleri gerçekleştirebilmek için gerekli eleman karmaşıklığı 2 girişli VEYA kapısı hesaplayınız.

$$f(x,y,z)=x+y+z$$

$$f(a,b,c,d)=\bar{a}.b.\bar{c}+a.d+b.\bar{c}.\bar{d}$$

$$f(a,b,c)=[a+b+c].[a+\bar{b}+c].[\bar{a}+b+c].[\bar{a}+b+\bar{c}]$$

İlk cebirsel ifade olan $f(x,y,z)=x+y+z$ oldukça yalın bir ifadedir. Bunun, 2 girişli VEYA kapısı açısından elaman karmaşıklığı,

$$EK=3 \cdot VEYA$$

olur. Eğer 3 girişli VEYA kapısı temel alınarak hesaplanmış olsaydı, $EK=2 \cdot VEYA$ olurdu. İkinci ve üçüncü denklem olan $f(a,b,c,d)=\bar{a}.\bar{b}.\bar{c}+a.d+\bar{b}.\bar{c}.\bar{d}$ ve $f(a,b,c)=[a+b+c].[a+\bar{b}+c].[\bar{a}+b+c].[\bar{a}+b+\bar{c}]$ için de benzer karmaşıklık hesabı tasarım yapılmadan gözle bakılarak yapılabilir. Ancak, ifade, değişken ve terimler açısından büyüdükçe gözle bakılarak elaman karmaşıklığı yapılması güç olur. O-nun yerine teorem geliştirilmelidir[3].

8.3. Büyük O Gösterilimi ve Karmaşıklık

Program yazılarak gerçekleştirilen algoritmaların karmaşıklık mertebesini göstermek için kullanılan notasyonlardan biri O' dur (büyük o diye okunur); *O(n)* şeklinde bir yazım, algoritma işlevini en kötü durumda *n* çevrim süresi mertebe yapar anlamına gelir. Büyük o notasyonu sayısal sistem tasarımında da zaman ve eleman karmaşıklığını göstermek için kullanılabilir.

Zaman ve eleman karmaşıklığı doğrudan bağıntısal düzeyde gerçek rakamları elde etmek için kullanılırken büyük o (O) notasyonu birkaç bitlik gerçekleştirilen lojik devrelerin daha çok bitlik uygulamalarında karmaşıklığın nasıl değiştiğini / mertebesin gösterir. Örneğin, 8 bitlik bir sayıcı için flip-flop sayısı *O(N)* bulunmuşsa, sayıcının 32 bitlik yapılması durumunda flip-flop gereksinimi nasıl artar denilirse, doğrusal arttığı söylenebilir. Diğer bir örnek şöyle verilebilir: bir lojik devrenin 1 bitlik tasarlanmasında $O(N^2)$ mertebesinde 2 girişli TVE kapısı gerekmektedir. Devre 10 bitlik yapılmak istenirse kapı gereksinimi nasıl olur? N^2'den karasel olarak değiştiği söylenir.

Karmaşıklık denklemleriyse, doğrudan sonucu verecek denklemlerdir; parametrelerin değerleri yerine koyulduğunda gerçek rakamlar bulunur.

Örnek-8.5.

Bir lojik devrenin 1 bitlik tasarımı için üç ayrı mimari şekli vardır ve bunların TVE kapılarıyla gerçekleştirilmesi için sırasıyla $O(N^{1,7})$, $O(N^2)$ ve $O(N^{2,1})$ mertebesinde kapı gerekmektedir. Bu devre 10 bitlik yapılmak istendiğinde eleman karmaşıklığı açısından en ekonomik olan hangisidir.

Tabi ki $O(N^{1,7})$ olanıdır!

[3] Yazarlarımızın bu konudaki akademik çalışmaları devam etmektedir; genel olarak bir lojik ifadeden eleman karmaşıklığının bulunması ifadedeki terim sayısına, terimlerdeki değişken sayısına ve terimlerdeki değişkenlerin tümlenme sayısına bağlıdır denilebilir.

8.4. Özet

Lojik tasarım yapılırken üç önemli kriter karşımıza çıkar; ilki devrenin çalışma hızı, ikincisi ne kadar kapı vs. gibi birim kullanılacağı ve diğeri de maliyetidir. Çalışma hızı ve kullanılacak birim sayısı devrenin karmaşıklığı olarak adlandırılır; zaman karmaşıklığı ve eleman karmaşıklığı şeklinde. Devrenin çalışma hızı, her ne kadar tümdevrelerin üretim teknolojisine bağlı olsa da tasarım aşamasında tutulan yol/kullanılan mimari yapı hızı etkileyebilir. Eleman karmaşıklığı, devrenin kapı, flip-flop vs. gibi içerdiği temel/standart lojik birimlerinin sayısını gösterir. Tasarım mimarisinde çalışma hızıyla eleman karmaşıklığının doğru orantılı olduğu söylenebilir; yani, daha hızlı mimariye sahip devre tasarlanmak istenirse eleman karmaşıklığı da artar. Ancak önemli olan bilinçsizce artması yerine gerektiği kadar artmasıdır.

Çalışma hızı veya toplam gecikme doğrudan girişlerde olacak bir değişikliğin çıkışa ne kadar süre sonra yansıtılacağını gösterir. Dolayısıyla girişle çıkış arasındaki lojik birimlerin toplam gecikmelerinden hesaplanır.

Eleman karmaşıklığıysa, lojik devrenin temel olarak kabul edilen lojik birimden ne kadar kullanılarak tasarlanacağını gösteren bir ifadedir; cebirsel ifadelerdeki terim sayısı, herbir terimdeki değişken sayısı vs. gibi parametrelerden bulunabilir. Büyük O notasyonu, karmaşıklığı mertebesel göstermek için kullanılan bir ifade şeklidir; küçük birimler için hesaplanmış karmaşıklığın büyük birimler tasarımında nasıl değiştiğini veya birkaç bitlik için hesaplanmış karmaşıklığın daha çok bit için, örneğin 32 , 64 bit için yapıldığında nasıl değişeceğini gösterir.

8.5. Sorular

1. Bir lojik devrenin maliyet analizinde, ayrık elemanlarla kart şeklinde tasarlanmasıyla bir ASIC olarak üretilmesi arasındaki farklılıklar nelerdir? Birim maliyette ağırlığı olan parametrelerin ağırlıklarının azaltılması nelere bağlıdır?
2. Tümdevrenin üretim teknolojisi çalışma hızını doğrudan etkiler. Hangi nedenlerden dolayı çalışma hızı üretim teknolojisine de bağlıdır?
3. Tasarlanan bir lojik devrenin bir tümdevre olarak üretilmesi istenirse, devre ASIC yardımcı aracıyla tasarlanmalı ve sınanmalıdır. Devrenin tümdevre olarak üretilmesi durumunda seçilecek hangi teknolojiler vardır? Bunlardan hangisinin seçilmesi gerektiği hangi unsurlara dayanılarak yapılır? Araştırınız.
4. 4:2 seçiciyi TVE kapılarıyla tasarlayınız ve aşağıdaki koşullar altında zaman karmaşıklığını hesaplayınız ve seçinin en yüksek hangi frekansta çalıştırılabileceğini belirleyiniz.

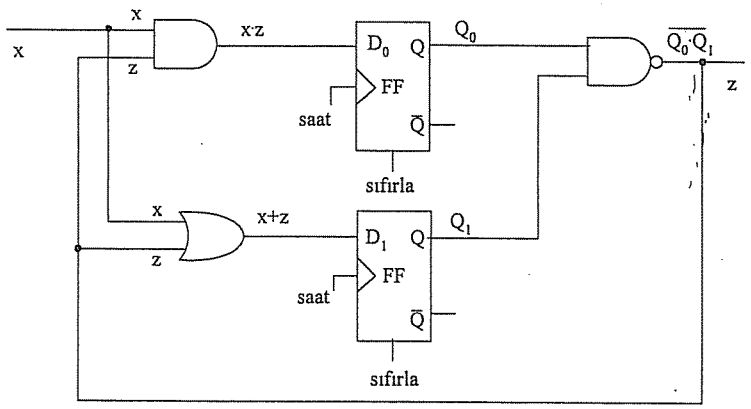
   TVE kapısının gecikme süreleri:

   0→1 : 9 ns (tipik), 15 ns (azami)

   1→0 : 10 ns (tipik), 15 ns (azami)

5. Aşağıda verilmiş olan ardışıl devreye ilişkin zaman karmaşıklığını hesaplayınız, ve devreye uygulanabilecek en yüksek saat işareti frekansını aşağıdaki koşullar altında belirleyiniz.

Kapıların gecikme süreleri:	Flip-Flop'un zaman parametrekeri :
0→1 : 9 ns (tipik)	t_y : 20 ns
15 ns (azami)	t_t : 30 ns
1→0 : 10 ns (tipik)	t_g : 50 ns
15 ns (azami)	



6. Eleman karmaşıklığı hesabı hangi lojik eleman temel alınarak yapılır? Neden?
7. Zaman karmaşıklığı hesabı, kombinezonsal devreler için ayrı ardışıl devreler için ayrı yapılır. Heriki yöntem içinde nasıl yapılacağını açıklayınız.
8. Aşağıda büyük O notasyonları verilen çözümler aynı devrenin farklı mimarisel tasarımları için bulunmuştur. Devre 1000 kat büyük kapasitede yapılmak istenirse, eleman karmaşıklığı açısından hangisi seçilmelidir? Neden?

   $O(\log_2 N)$, $O(N)$, $O(N^2)$

9.

Programlanabilir Kombinezonsal Devreler

Programlanabilir kombinezonsal devreler, fiziksel olarak tek bir tümdevre içerisinde birçok VE-VEYA kapıları bulunan ve bunlar arasındaki bağlantıların uygun şekilde programlanmasıyla karmaşık bir lojik ifadenin tek bir tümdevreyle gerçekleştirilmesini sağlayan yarı hazır elemanlar/birimlerdir. Bu tür devreler özellikle kart üzerine tasarımı yapılacak lojik devrenin, daha az tümdevre kullanılarak gerçekleştirilmesini sağlarlar; çünkü, birçok kapı tümdevresi kullanılması gerektiren lojik ifadeler bir veya iki tane programlanabilir kombinezonsal devre kullanılarak yapılabilmektedir. Bu bölümde programlanabilir kombinezonsal lojik devrelerin türleri, özellikleri ve kullanımına ilişkin örnekler aşağıdaki başlıklara altında ele alınmıştır:

9.1. Prog. Kombinezonsal Devreler
9.2. Prog.Kombinezonsal Devre Elemanları
PROM-Prog. Yalnızca Okunabilir Bellek
PAL-Programlanabilir Diziler
PLA-Programlanabilir Lojik Diziler
9.3. Özet
9.4. Sorular

Lojik devre tasarımında karmaşıklık ve maliyet önemli kriterlerdir (bkz. Bölüm 8); tasarlanacak devrenin aynı fonksiyonu yerine getirebilecek şekilde daha az elemanla tasarlanması hedeflenir. Bu amaçla, tasarlanması istenen kombinezonsal devreye ait lojik ifade, herşeyden önce, indirgenir; bir başka deyişle en az sayıda temel lojik birimi kullanılarak gerçeklenmesini sağlayacak fazlalıklar atılır. Daha sonra, lojik ifadelerin indirgenmesi sonucu elde edilen yeni ifadeler kapılar veya standart lojik birimler kullanılarak gerçekleştirilmelidir. Bu aşamada, ya herbiri 4, 6 veya 8 gibi belirli sayıda kapı içeren tümdevreler ya da özel olarak tasarlanmış ve

imal edilmiş orta ölçekli tümdevreler ile programlanma özelliği olan bazı PLD'ler kullanılır. İlk yaklaşım, yalın lojik ifadelerin gerçekleştirilmesinde tutulan yoldur; çünkü devre zaten karmaşık değildir ve birkaç kapı tümdevresi kullanılarak elde edilebilir. Ancak lojik ifadenin karmaşıklığı arttıkça kapı tümdevreleri yerine ikinci yaklaşım olan, orta ölçekli tümdevreler ya da Programlanabilir kombinezonsal lojik devrelerin kullanılması yararlı olur. Çünkü, tasarımda kullanılacak fiziksel tümdevre sayısı daha az olur ve kart tasarımı yalınlaşır. ROM gibi bazı PLD elemanlarında ise minimum terimler kanonik biçimine ilişkin ifade doğrudan kullanılarak devre tasarımı gerçeklenmiş olur.

Programlanabilir kombinezonsal devreler, hem yalın kombinezonsal devrelerin tasarımında hem de ardışıl devrelerin çıkış ve geçiş işaretlerini üreten kombinezonsal devre kısmında kullanılır. Farklı gereksinimleri karşılamak için çeşitli türde Programlanabilir kombinezonsal devre yapıları vardır; bir bellek elemanı olarak üretilen PROM ve türevleri de (PROM, EPROM, E^2PROM) birer Programlanabilir kombinezonsal devredir. Ancak PAL ve PLA olarak adlandırılan Programlanabilir kombinezonsal devreler, özel olarak bu amaçla üretilmişlerdir. İçerisinde saklama özelliği olan Programlanabilir kombinezonsal devreler de vardır; bunlar programlanabilir ardışıl diziler (PSA) olarak adlandırılır (ayrıntı için bkz. Ayrıt 11.4 e). Programlanabilir kombinezonsal devreler aşağıdaki gibi sınıflanmaktadır:

- PROM ve Türevleri
- PAL (Programmable Array Logic)
- PLA (Programmable Logic Array)
- PSA (Programmable Squential Array) – bkz. Ayrıt 11.4.

İlerleyen kısımlarda PROM, PAL, PLA gibi Programlanabilir kombinezonsal devreler ele alınmıştır; saklama özelliği olan ve ardışıl devre tasarımında kullanılan PSA Bölüm 11, Ayrıt 11.4 de ele alınmıştır.

9.1. Programlanabilir Kombinezonsal Devreler - PLD

Programlanabilir Kombinezonsal Devreler, PLD (*Programmable Logic Devices*) genel adıyla anılırlar. Bu devrelerde amaç, karmaşık lojik işlevleri tek bir veya birkaç devre elemanı kullanarak gerçeklemektir; buradaki programlanabilir olma özelliği, lojik kapılar bağlamında istenilen işlevin sağlanması anlamına gelir. Programlanabilir bir kombinezonsal devre, kendi bünyesinde programlanabilir özellikte VE-VEYA kapıları ve programlama işinin yapılmasını sağlayan sigortalar barındırır. Programlama işlemi, giriş değişkenleriyle kapı girişlerine bağlantısını sağlayan sigortaların olduğu gibi bırakılmasıyla gerçeklenir. Dolayısıyla, bir değişkenin kapı girişinde etkimesi isteniyorsa sigorta olduğu gibi bırakılır, eğer değişken bağlantısı istenmiyorsa o değişkene ilişkin sigorta attırılır.

Programlanmış bazı PLD elemanlarını tekrar programlamak mümkündür. Ancak bazı PLD elemanları ise tekrar programlanamazlar. PLD elemanları programlanabilirlik özelliklerine göre, bir kez programlanabilenler ve birçok kez (defalarca) programlanabilenler olmak üzere iki sınıfa ayrılırlar:

- Bir kez programlanabilen : PROM, PAL, PLA
- Defalarca Programlanabilen: EPROM, E^2PROM ve LCA/LGA

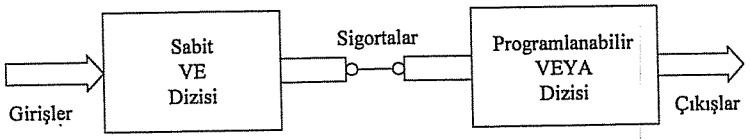
Bir kez programlanabilen PLD elemanları için PROM (*Programmable Read Only Memory*), PAL (*Programmable Array Logic*) ve PLA (*Programmable Logic Array*), çok kez programlanabilen PLD elemanları içinde ise EPROM (*Erasable Programmable Read Only Memory*), E^2PROM (*Electrically Erasable Programmable Read Only Memory*), LCA/LGA (*Logic Cell Arrays/Logic Gate Arrays*) sayılabilir.

9.2. Programlanabilir Kombinezonsal Devre Elemanları

Bir programlanabilir kombinezonsal devre, VE kapı dizisi, VEYA kapı dizisiyle bunların programlanmasına imkan veren sigortalar içerir. Programlanabilen kapı dizilerine, bir başka deyişle sigortaların bulundukları yerlere göre üç farklı PLD elemanı bulunmaktadır. Sigorta, VE kapı dizisiyle VEYA kapı dizisi arasındaysa PROM, giriş işaretiyle VE kapı dizisi arasındaysa PAL, hem giriş işareti ile VE kapı dizisi hem de VE kapı dizisiyle VEYA kapı dizisi arasında varsa PLA olarak adlandırılır. PLA'larda 2 farklı yerde sigorta bulunduğu için daha esnek programlama imkanı vardır denilebilir. Sigortanın bulunduğu konum neyin programlanabileceğini gösterir; örneğin VE kapısı önüne bulunuyorsa VE kapı dizisi, VEYA kapısı önüne bulunuyorsa VEYA kapı dizisi programlanabilir anlamına gelir.

- **PROM** (*Programmable Read Only Memory*):

PROM, sabit VE kapı dizisi ve sigortalarla programlanabilen VEYA kapı dizisi şeklinde bir mimariye sahip tümleşik bir elemandır. PROM'un mimari gösterilimi aşağıdaki gibi verilebilir. Görüldüğü gibi girişler önce VE kapılarına uygulanmakta ve bunların çıkışları VEYA kapılarına programlanarak verilmektedir.

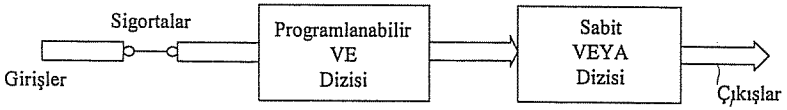


PROM programlanmadan önce VE kapı dizisindeki tüm çıkışlar VEYA kapısı girişlerine bağlıdırlar. Başka bir deyişle, tüm çıkışlar tüm girişlere bağlıdır. Programlama, bağlı olmayacak VE kapısı çıkışıyla VEYA kapısı girişi arasındaki ilgili

sigortayı attırmaktır. Tüm sigortalar attırılırsa, VE kapılarına ait tüm çıkışların VEYA kapılarına bağlanmaması anlamına gelir ki, bu anlamsız olur. (ROM hakkında daha ayrıntılı bilgi için bkz. Ayrıt 11.3.) PROM, minimum terimler kanonik biçiminde verilmiş olan lojik fonksiyonların gerçekleştirilmesi için uygunudur.

- **PAL** (*Programmable Array Logic*):

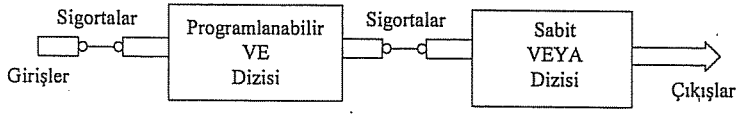
PAL, programlanabilen VE kapı dizisiyle sabit VEYA kapı dizisi şeklinde bir mimariye sahiptir: PAL'in mimari gösterilimi aşağıdaki gibi verilebilir. Görüldüğü gibi girişler daha VE kapısına verilmeden programlanmaktadır.



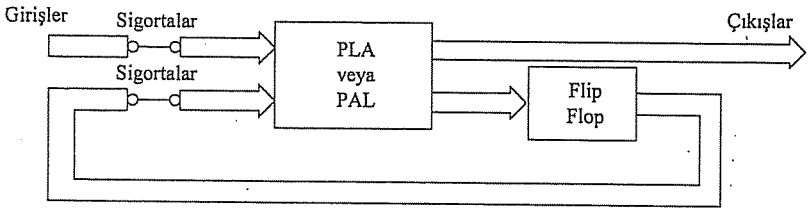
PAL, çarpımlar toplamı şeklinde indirgenmiş lojik ifadelerin gerçekleştirilmesi için uygundur; sanki bunun için tasarlanmış denilebilir.

- **PLA** (*Programmable Logic Array*):

PLA, programlanabilen VE kapı dizisiyle programlanabilen VEYA kapı dizisi içeren bir mimariye sahiptir; gösterilimi aşağıdaki gibidir. Görüldüğü gibi hem VE dizisi hem de VEYA dizisi öncesi programlamak için sigortalar vardır.



PAL veya PLA elemanları ve Flip-Flop'lar kullanarak bir programlanabilir ardışıl devre tasarımını gerçekleştirmek de mümkündür. Bu şekilde oluşturulan devrenin blok gösterilimi aşağıdaki gibi olur:

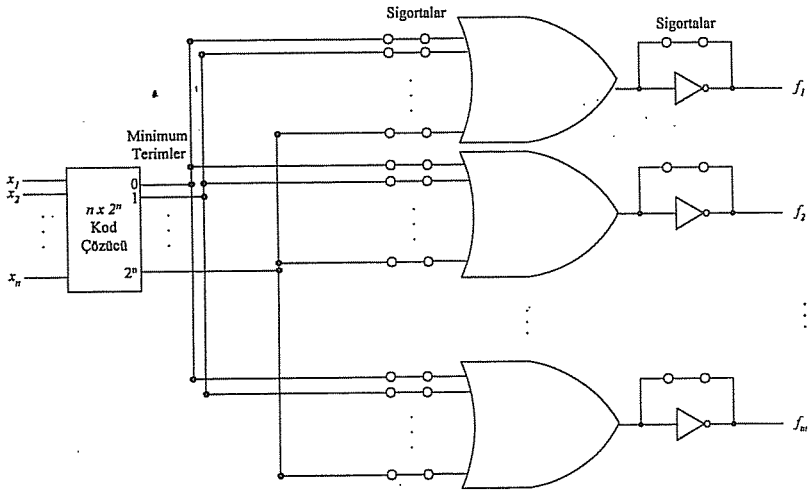


9.2.1. PROM'un İç Yapısı

Sadece VEYA dizisinin programlanabildiği programlanabilir kombinezonsal devreye kısaca PROM denir; veya Programlanabilir Yalnızca Okunabilir Bellek denilir. PROM'un mimarisine ilişkin mimari gösterilimin iç yapısı ve sigortaların bulunduğu yerler Şekil-9.1 de gösterildiği gibidir; şekildeki Kod Çözücünün VE kapılarıyla gerçekleştirildiği varsayılırsa sigortaların VE kapılarıyla VEYA kapıları arasında olduğu görülmektedir. PROM'u programlamak, istenmeyen bağlantılara ait sigortaları attırmak demektir; dolayısıyla lojik ifadelerden attırılacak sigortaların hangileri olduğu belirlenmelidir.

PROM elemanlarıyla minimum terimler kanonik biçiminde (*mintermlerin* toplamı şeklinde) verilmiş ifadeler gerçeklenir. Mimari yapıdan de görüleceği üzere, PROM devresi ***n*** giriş, ***m*** çıkışlı kombinezonsal bir devredir. Bir PROM elemanı, içerdiği toplam bit sayısına göre sınıflandırılır. Örneğin *n* giriş, *m* çıkışlı PROM elemanı, 2^n *x m* bitlik PROM olarak isimlendirilir. Sayısal bir örnek vermek gerekirse, 13 giriş, 8 çıkışlı bir PROM, $2^{13}x8= 8192x8= 8Kx8bit= 8Kbyte$'lık PROM olarak yorumlanır.

PROM, aslında, bir bellek türüdür; ROM ailesinin bir üyesidir. Bilgisayar ve sayısal sistemler üzerinde kalıcı verilerin veya program kodlarının tutulması saklanması için kullanılır; bir programlanabilir kombinezonsal devre olarak da kullanılmaktadır. ROM ailesi ile ilgili ayrıntılı bilgi Ayrıt 11.3 te verilmiştir.



Şekil-9.1. PROM elemanına ilişkin mimari iç yapı.

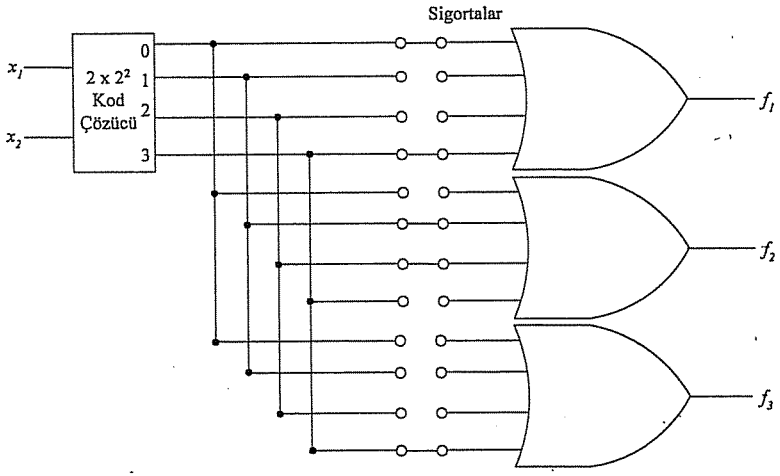
Örnek-9.1.

Lojik ifadesi aşağıdaki gibi doğruluk tablosuyla verilen fonksiyonları gerçekleyen devreyi PROM elemanını uygun şekilde programlayarak gerçekleştiriniz. Devre için kaç bitlik PROM elamanına gereksinim vardır?

x_1	x_2	f_1	f_2	f_3
0	0	1	0	0
0	1	0	1	0
1	0	0	1	0
1	1	1	0	1

Bu sorunun çözümü için gerekli en önemli özellik fonksiyonların aldıkları değerleri belirten doğruluk tablosudur ve bu da soruyla birlikte verilmiştir. Yukarıdaki doğruluk tablosuna göre devre 2 girişli 3 çıkışlıdır; bu durumda 2^2x3 bitlik PROM elemanına ihtiyaç vardır. Bundan sonra yapılması gereken 2 giriş, 3 çıkışlı PROM'u programlamak için gerekli sigortaların attırılmasıdır.

Şekil-9.2 de istenen lojik fonksiyonları gerçekleştirmek için attırılması gereken sigortalar gösterilmiştir; örneğin f_1 fonksiyonu için 0 ve 3. *mintermlerin* kalması, buna karşın 1 ve 2. *mintermlerin* attırılması gerekir. Çünkü 0. ve 3. *mintermler* lojik 1; 2 ve 3. *mintermler* lojik 0 değerindedir. Aynı işlemler f_2 ve f_3 fonksiyonları için yapılırsa yukarıdaki şekil elde edilir.



Şekil-9.2. Tasarlanan PROM elemanına ilişkin mimari iç yapısı.

9.2.2. PAL'in İç Yapısı

Sadece VE dizisinin programlanabildiği programlanabilir kombinezonsal devreye kısaca PAL denir; veya, Programlanabilir Dizi adı verilir. PAL'e ilişkin mimari yapı ve sigortaların bulunduğu yerler Şekil-9.3 de gösterilmiştir. PAL elemanıyla minimum terimler kanonik biçiminde (*mintermlerin* toplamı şeklinde) verilmiş ifadelerin indirgenmiş şekilleri gerçeklenir. Örneğin aşağıdaki lojik fonksiyon böyle ifadeyle gösterilmiştir.

$$f(a,b,c,d)=\overline{b\cdot d}+\overline{b}\cdot\overline{c}+\overline{a}\cdot c\cdot\overline{d}$$

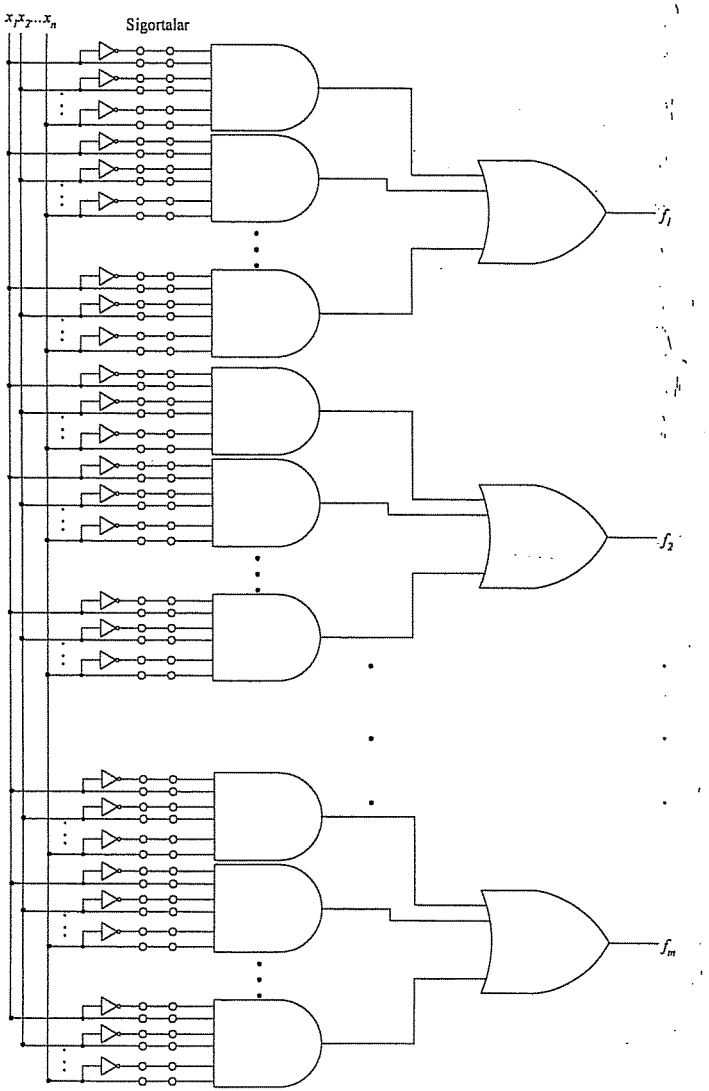
VEYA kapısının sabit olması, sadece VE kapılarının programlanabilmesinden dolayı PAL elemanlarının programlanması oldukça basittir. Ancak PROM'da olduğu gibi tek tip kapı dizisi programlanabildiği için tasarımcıya kısıtlama getirmektedir denilebilir. Bu sebepten dolayı hem VE hem de VEYA kapılarının programlanabildiği PLA elemanları kullanıcılara daha esnek çözümler sunmaktadırlar. PLA elemanına geçilmeden önce PAL tasarımına ilişkin bir örnekle PAL'lerin programlanmasını görmek yararlı olacaktır.

Örnek-9.2.

Aşağıda minimum terimlerin toplamı şeklinde verilen fonksiyonları gerçekleyen devreyi PAL elemanıyla gerçekleştirmek için gerekli programlama bilgilerini hesaplayınız ve PAL'in programlanmış şeklini çiziniz.

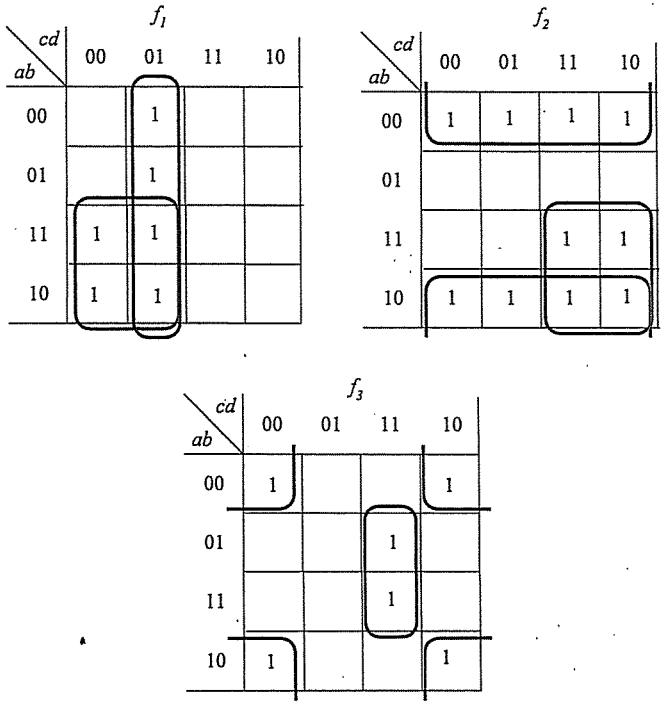
$$f_1(a,b,c,d)=\sum(1,5,8,9,12,13)\ ,\ f_2(a,b,c,d)=\sum(0,1,2,3,8,9,10,11,14,15)$$
$$f_3(a,b,c,d)=\sum(0,2,7,8,10,15)$$

Bu sorunun çözümü için ilk önce minimum terimlerin indirgenmiş ifadelerinin elde edilmesi gerekir. Bu amaçla *Karnaugh* diyagramları yardımıyla çıkış fonksiyonları bulunur ve Şekil-9.3 te iç yapısı verilen PAL'in üzerine işlenir.



Şekil-9.3. PAL'in mimari iç yapısı.

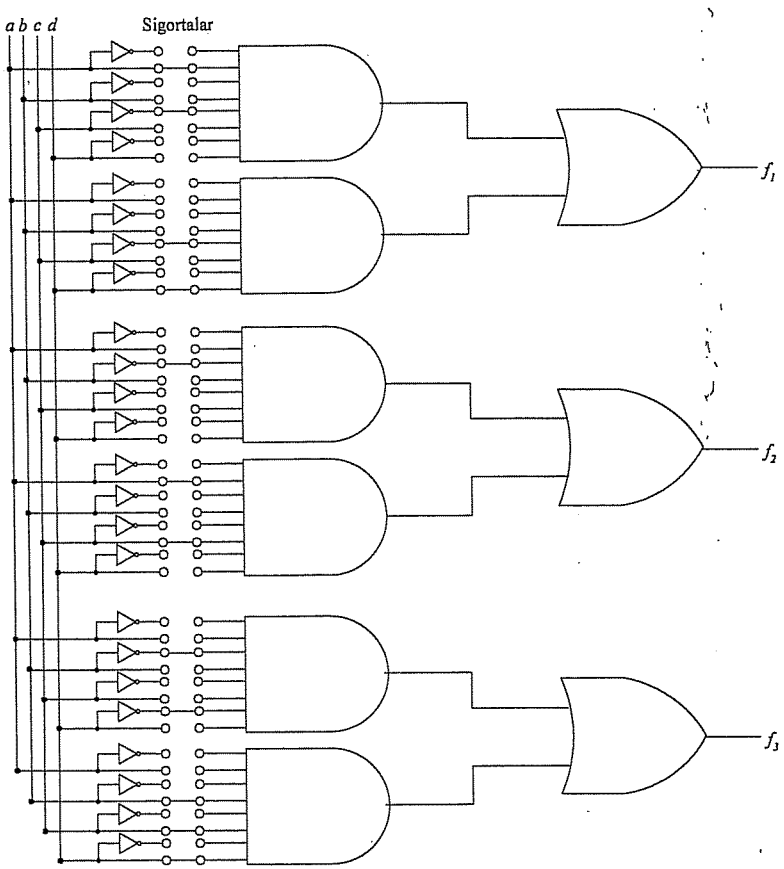
Verilen problemin *Karnaugh* diyagramlarıyla çıkış fonksiyonlarının bulunması;



şeklinde indirgenirse aşağıdaki ifade elde edilir:

$$f_1(a,b,c,d)=a\cdot\overline{c}+\overline{c}\cdot d,\quad f_2(a,b,c,d)=\overline{b}+a\cdot c,\quad f_3(a,b,c,d)=\overline{b}\cdot\overline{d}+b\cdot c\cdot d$$

Bu indirgenmiş ifadeden, tasarım PAL elemanıyla Şekil-9.4 teki gibi gerçekleştirilir. Örneğin f_1 fonksiyonunda iki tane çarpımlar toplamı şeklinde terim bulunmaktadır. Birinci terim $a\cdot\overline{c}$, ikinci terim ise $\overline{c}\cdot d$ biçimindedir. Herbir terim için birer VE kapısı kullanılarak programlama yapılır. Bunlar çarpımlardan elde edilen terimlere karşılık düşer; çarpımların toplamı ise VEYA kapısıyla elde edilir. Şekil-9.4 de görüldüğü gibi gereksiz olan bağlantılara ait sigortalar attırılmıştır. Diğer fonksiyonlar içinde benzer işlemler yapılarak hangi sigortaların kalacağı hangilerinin attırılacağı belirlenir.



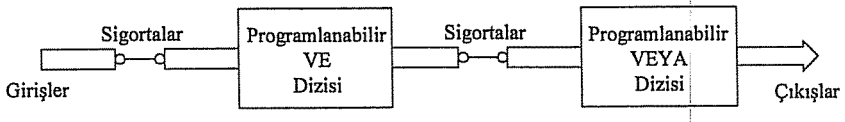
Şekil-9.4. Örnek-9.2 nin çözümü (PAL örneği).

PAL programlanmasında, yukarıdaki şekilde görüldüğü gibi *a*, *b*, *c* ve *d* olarak adlandırılan herbir dış girişin hem kendisi hem de tümleyeni için sigorta vardır. VE kapılarının girişine uygulanacak dış girişler için sigortalar olduğu gibi bırakılır; uygulanmayacak olanların da sigortası attırılır.

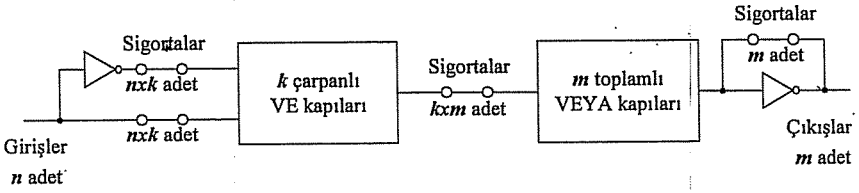
9.2.3. *PLA'nın İç Yapısı*

Hem VE hem de VEYA dizilerinin programlanabildiği programlanabilir kombinezonsal devreye PLA denir; veya, Programlanabilir Lojik Diziler (PLA) olarak söylenirler. PLA'ya ilişkin mimari yapı ve sigortaların bulunduğu yerler Şekil-9.5 te gösterildiği gibidir:

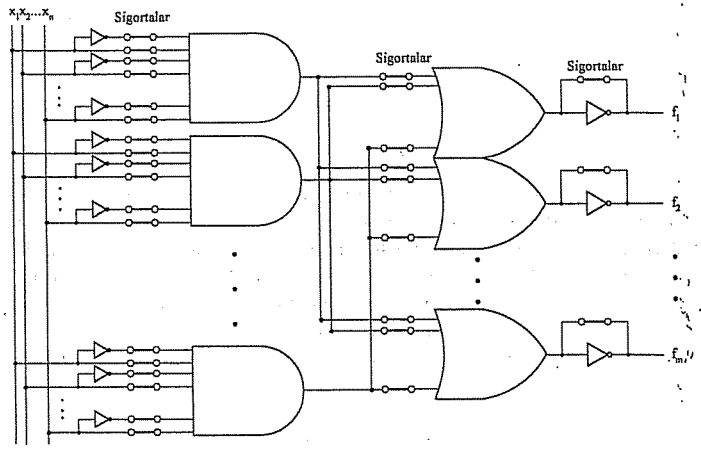
PLA elemanıyla çarpımlar toplamı şeklindeki ifadelerin indirgenmiş biçimleri gerçeklenir. Şekil-9.6 dan görüleceği gibi, PLA devresi ***n*** giriş, ***m*** çıkışlı bir devredir. VE kapı dizisi girişlerinde *n* değişkenin hem kendisi hem de tümleyeni bulunmaktadır. Benzer biçimde PLA'nın çıkışında fonksiyonun hem kendisini hem de tümleyenini gerçekleştirmek olanaklıdır; programlamada hangisi daha az terimle gerçekleştirilebiliyorsa o seçilir. Fonksiyonun tümleyeni gerçeklenirse, çıkıştaki sigortalar attırılarak tümleyeni alınır ve böylece fonksiyonun kendisi elde edilir. PLA elemanının lojik kapı ve sigortalarla gösterilimi Şekil-9.7 deki gibidir.



Şekil-9.5. PLA elemanına ilişkin mimari yapı.



Şekil-9.6. PLA elemanının giriş/çıkış ve programlanabilir eleman boyutları.



Şekil-9.7. PLA elemanının iç yapısı.

Örnek-9.3.

Aşağıda *Karnaugh* diyagramları üzerinde verilen fonksiyonları PLA elemanı kullanarak gerçekleştirmek için gerekli programlama bilgilerini belirleyiniz.

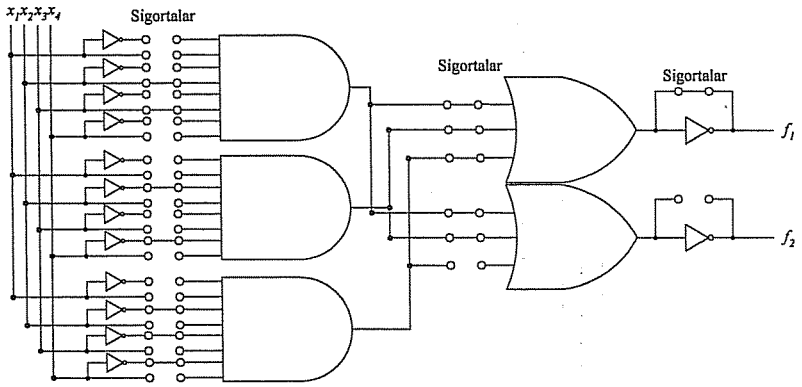
x_1x_2 \ x_3x_4	00	01	11	10
00	0	1	0	0
01	1	0	1	1
11	1	0	1	1
10	0	1	0	0

x_1x_2 \ x_3x_4	00	01	11	10
00	1	1	1	1
01	0	1	0	0
11	0	1	0	0
10	1	1	1	1

PLA ile tasarımda ilk adım, *Karnaugh* diyagramlarıyla verilen fonksiyonları çarpımlar toplamı şeklinde indirgemektir. Dolayısıyla, diyagramlara ilişkin indirgenmiş fonksiyonlar $f_1, \overline{f_1}$ ile $f_2, \overline{f_2}$ şeklinde aşağıdaki gibi elde edilir.

$$f_1 = x_2x_3 + x_2\overline{x}_4 + \overline{x}_2\overline{x}_3x_4 \qquad f_2 = \overline{x}_2 + \overline{x}_3x_4$$
$$\overline{f_1} = \overline{x}_2x_3 + x_2\overline{x}_3x_4 + \overline{x}_2\overline{x}_4 \qquad \overline{f_2} = x_2x_3 + x_2\overline{x}_4$$

PLA ile tasarımda ikinci adım, hangi fonksiyonların gerçekleştirileceğine karar verme aşamasıdır. Fonksiyonlar dikkatlice incelenirse, f_1 ile $\overline{f_2}$'ye ait ilk iki terimin birbirinin aynı olduğu görülür. Bir başka deyişle, f_1 ile $\overline{f_2}$'nin ilk iki terimi ortak kullanılabilir. Bu durumda bu iki fonksiyonun (f_1 ile $\overline{f_2}$) gerçeklenmesi halinde sadece 3 terimin gerçeklenmesi gerekecektir; böylece maliyeti daha düşük bir çözüm elde edilmiş olunur.



Şekil-9.7. Örnek-9.3 ün çözümü (PLA örneği).

Bulunan 3 terim, gerçeklenebilecek en düşük terim sayısıdır ve PLA tasarımında seçilmesi gereken fonksiyonlar da bu yüzden terimler olmalıdır. Çıkışta dikkat edilmesi gereken nokta gerçeklenen fonksiyonlardan f_1'in olduğu gibi, $\overline{f_2}$ fonksiyonun da tümleyeni alınarak f_2'nin elde edilmesidir. PLA tasarımında üçüncü ve son adım fonksiyonların programlanması aşamasıdır; programlanmış PLA'nın iç mimari yapısı Şekil-9.7 de gösterildiği gibi olur:

9.3. Özet

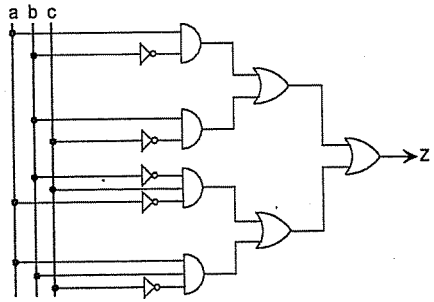
Programlanabilir kombinezonsal devreler, eleman karmaşıklığını azaltmak ve tasarıma kolaylık getirmek amacıyla tasarımcılar tarafından tercih edilirler. Genel olarak VE ve VEYA kapılarında oluşan tümleşik kombinezonsal devreler, programlanmak için birçok sigortaya sahiptirler; programlanma sürecinde bağlantısı kalması istenen girişlerin sigortası olduğu gibi bırakılırken, bağlantısının olmaması istenen girişlere ait sigortalar attırılır. Farklı gereksinimleri karşılamak amacıyla PROM, PAL, PLA gibi çeşitli türde programlanabilir kombinezonsal devreler geliştirilmiştir. aralarındaki fark sigortaların bulunduğu yerdir. Sigortalar, PROM'da VE ile VEYA, PAL'de giriş ile VE, PAL'de ise hem girişle VE hem de VE ile

VEYA arasındadırlar. Bu nedenle PROM minimum terimler kanonik biçimde verilmiş lojik ifadeler için uygunken, PAL indirgenmiş lojik ifadeler için uygundur. PLA ise, iki farklı yerde sigorta bulundurduğundan dolayı daha esnek bir çözüm sunar: Hem indirgenmiş fonksiyonlar için çözüm sunarken hem de terim karmaşıklığı en az olan çözümün yapılmasına olanak verir.

9.4. Sorular

1. Aşağıdaki şekilde bir devre şeması verilmiştir.

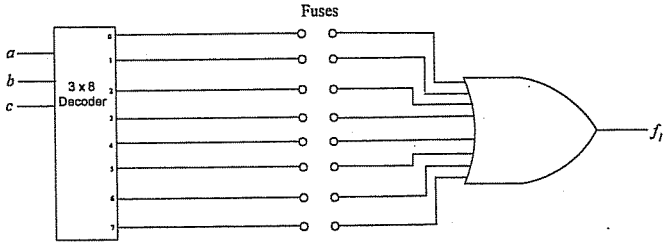
   a) Devrenin çıkış fonksiyonuna ilişkin lojik ifadeyi elde ediniz.



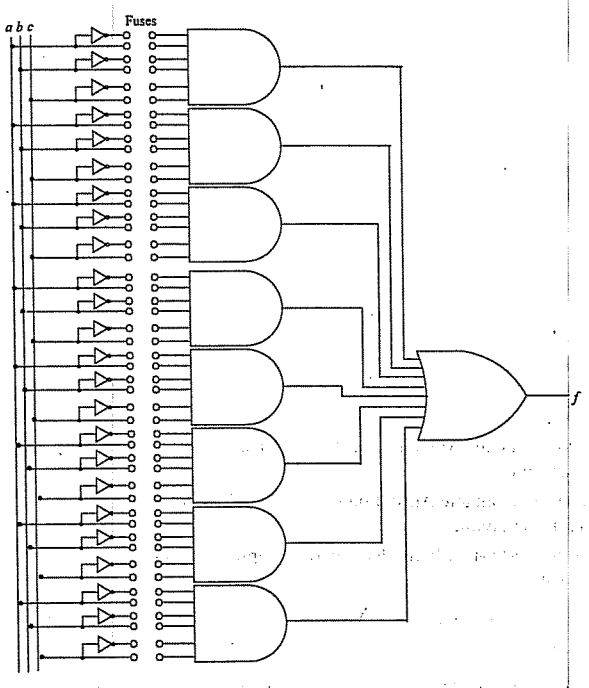
   b) Bulduğunuz fonksiyona ilişkin minimum terimler kanonik biçimini elde ediniz.

   c) Yukarıdaki devre yerine 3'e8 kod çözücü devre kullanarak aynı işlevi yerine getirecek devreyi tasarlayınız.

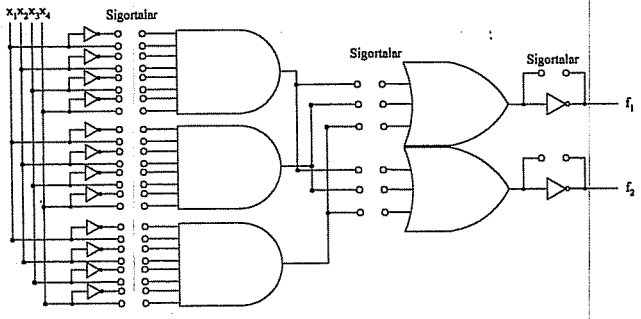
   d) Bir sonraki sayfada lojik şeması verilen önceden sigortaları attırılmış ROM (Read Only Memory) elemanını kullanarak aynı işlevi yerine getirecek devreyi tasarlayınız. Hangi sigortaların attırılmaması gerektiğini şekil üzerine çizerek gösteriniz.



   f) Aşağıda lojik şeması verilen önceden sigortaları attırılmış PAL (Programmable Array Logic) elemanını kullanarak aynı işlevi yerine getirecek devreyi tasarlayınız. Hangi sigortaların attırılmaması gerektiğini şekil üzerine çizerek gösteriniz.



2. Bir kombinezonsal devre; $f_1 = \sum(3,5,6,7)$, $f_2 = \sum(0,2,4,7)$ fonksiyonları ile tanımlanmıştır. Bu lojik fonksiyonları sağlayan devreyi aşağıda verilen önceden sigortaları attırılmış PLA (Programmable Array Logic) elemanı kullanarak tasarlayınız. Hangi sigortaların attırılmaması gerektiğini şekil üzerine çizerek gösteriniz.



3. İkili/GRAY kodu dönüştürme işlemini yapacak bir devre, PROM kullanılarak tasarlanmak istenmektedir. 4 giriş ikili sayıları, 10 çıkış GRAY kodunu ifade edecek biçimde doğruluk tablosunu oluşturunuz ve gerekli işlemleri yaparak PROM ile tasarımı gerçekleştiriniz, devre şemasını çiziniz.

4. Bir kombinezonsal devre; $f = \sum(0,3,4,7)$, $f_2 = \sum(0,1,2,4,7)$ fonksiyonları ile tanımlanmıştır.
   a) ROM (Read Only Memory) elemanını kullanarak aynı işlevi yerine getirecek devreyi tasarlayınız.
   b) PLA (Programmable Logic Array) elemanını kullanarak aynı işlevi yerine getirecek devreyi tasarlayınız.
   c) Tasarımcı ROM ve PLA elemanlarıyla yapılan tasarımlardan hangisini tercih etmelidir? Neden?

5. Bir kombinezonsal devre; $f_1 = \sum(1,3,5,7)$, $f_2 = \sum(0,2,4,6)$ fonksiyonları ile tanımlanmıştır.
   a) ROM (Read Only Memory) elemanını kullanarak aynı işlevi yerine getirecek devreyi tasarlayınız.
   b) PAL (Programmable Array Logic) elemanını kullanarak aynı işlevi yerine getirecek devreyi tasarlayınız.
   c) Tasarımcı, ROM ve PAL elemanlarıyla yapılan tasarımlardan hangisini tercih etmelidir? Neden?

6. Bir kombinezonsal devre; $f_1 = \sum(0,2,3,5,6)$, $f_2 = \sum(0,2,4,6)$ fonksiyonları ile tanımlanmıştır.
   a) ROM (Read Only Memory) elemanını kullanarak aynı işlevi yerine getirecek devreyi tasarlayınız.
   b) PAL (Programmable Array Logic) elemanını kullanarak aynı işlevi yerine getirecek devreyi tasarlayınız.
   c) PLA (Programmable Logic Array) elemanını kullanarak aynı işlevi yerine getirecek devreyi tasarlayınız.
   d) Tasarımcı, ROM, PAL ve PLA elemanlarıyla yapılan tasarımlardan hangisini tercih etmelidir? Neden?

10.

Ardışıl Devre Temelleri

Ardışıl devreler lojik devre tasarımında önemli bir yer tutar. Kombinezonsal devrelerden farklı olarak, ardışıl devrelerde çıkış geçmiş durumlara da bağlıdır. Bilindiği gibi kombinezonsal devrelerde herhangi bir andaki çıkış değeri yine o andaki giriş değerlerinden üretiliyor. Ardışıl devrelerde ise, çıkış değeri o andaki giriş ve geçmişteki durum bilgilerinden üretilir. Yani çıkış, daha önceki girişler ve geçmiş bilgilere dayanılarak elde edilir. Dolayısıyla ardışıl devrelerde geçmiş bilgileri tutmak için saklayıcı ve bellek benzeri birimlere ihtiyaç duyulur; kombinezonsal devreler de ardışıl devrelerin bir alt birimi olarak kullanılır. Bu bölümde ardışıl devrelerin genel yapısı, sınıflanması, tasarımda kullanılan temel elemanlar, tasarımda yaklaşım yöntemleri aşağıdaki başlıklar altında ele alınmıştır:

10.1. Ardışıl Devrelerin Genel Yapısı
10.1.1. *Mealy* ve *Moore* Makinalar
10.1.2. Asenkron&Senkron Ardışıl Devreler
10.1.3. Durumlar ve Durum Diyagramları
Durum Tablosu
Durum Diyagramı
Sonlu Durum Makinaları
10.1.4. Standart Tasarım Birimleri
Kombinezonsal Devreler
İkililer (Flip-Flop'lar), Tutucular
Saklayıcılar, Sayıcılar
Bellekler, Program. Ardışıl Diziler
10.1.5. Kenar ve Düzey Tetikleme
10.2. Flip-Flop'lar
JK, D,
SR, T
10.3. Tutucular (Latches)
SR, D
10.4. Ardışıl Devre Analiz Yöntemi
10.5. Ardışıl Devre Tasarım Yöntemi
10.6. Özet
10.7. Sorular

Ardışıl devrelerle lojik devre tasarımı bu işin doğasıdır denilebilir. Çünkü uygulamada gereksinim duyulan sayısal devrelerin büyük bir çoğunluğu ya zamana ya da önceki durumlarına bağlıdır. Dolayısıyla bu tür devrelerin tasarımı ardışıl devreler ile gerçekleştirilir. Örneğin 1, 2, 3, 4... şeklinde her seferinde birer artımla sayma

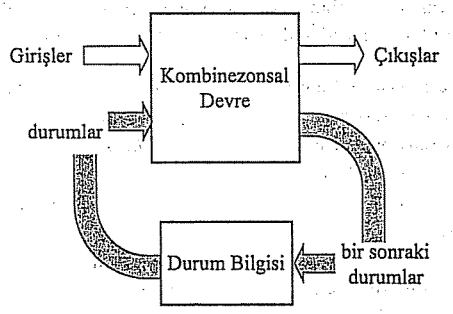
işlemi yapan bir lojik devrede, bir sonraki sayının ne olacağı o andaki sayının ne olduğuna bağlıdır; o andaki sayı 3 ise bir sonraki 4, 6 ise bir sonraki 7 olur. Burada o andaki sayının ne olduğu saklayıcı ve bellek benzeri bir birimde tutulmalıdır.

10.1. Ardışıl Devrelerin Genel Yapısı

Ardışıl devrelerde adı üzerinde ard arda gelişen olaylar vardır; sonuca ulaşmak için belirli işler belirli bir sıra ile yapılmalıdır. Bir ardışıl devrenin işlevini yerine getirmesi için, herhangi bir anda, o andaki durum ve bir sonraki durum söz konusudur. Çünkü bir sonraki durum o andaki duruma ve giriş değerlerine bağlıdır. Örneğin bir kapıdan girenleri sayan bir lojik devre ardışıl devre yaklaşımıyla gerçekleştirilebilir; kaç kişinin girdiği bir yerlerde saklanmalı ve bu değer her giriş için bir arttırılmalıdır.

Ardışıl devreler geçmiş bilgileri tutan bellek ve saklayıcı gibi birimler ile kombinezonsal devrelerin birleştirilmesinden oluşturulur. Saklayıcı ve bellek, durum bilgilerini tutarken; kombinezonsal devre ise, o anki durum bilgileri ve giriş değerlerinden çıkış değerini ve bir sonraki durum bilgilerini üretir. Aşağıdaki şekilde görüldüğü gibi ardışıl devrelerin herhangi bir andaki çıkış değeri Z(x, y), o andaki giriş değerleri G(x) ve durum bilgisi D(y)'ye bağlıdır:

$$Z(x, y) = f(G(x), D(y))$$



Şekil-10.1. Ardışıl devrelerin genel yapısı.

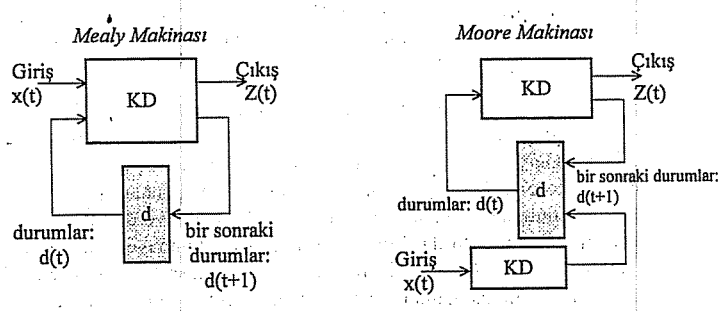
Durum bilgisi, devrenin karmaşıklığına göre, bir bitlik olabileceği gibi, birçok bit'ten de oluşabilir; hatta, durum bilgisi çok fazlaysa bellek birimi bile kullanılabilir. Bir veya birkaç bitlik durum bilgileri için flip-flop'lar, birçok bit'ten oluşan bit dizileri için saklayıcılar/sayıcılar, bitler ile ifade edilemeyecek kadar fazla olan durum bilgileri için ROM, PROM, RAM gibi bellek birimleri kullanılır.

Kombinezonsal devre, o andaki girişlerden ve durum bilgilerinden çıkışları üretir; ve aynı zamanda, sonraki durum bilgilerini üretir. Dolayısıyla, girişlerin çokluğu kombinezonsal devrenin, durumların çokluğu da bellek veya saklayıcı birimlerin karmaşıklığını arttırır.

10.1.1. Mealy ve Moore Makinaları

Ardışıl devre tasarımı yapılırken birkaç yaklaşım şekli kullanılması alışıla gelmiştir. Bunlardan biri kanonik yaklaşımdır; diğerleri, *Mealy* ve *Moore* makinaları olarak adlandırılır. Bu yaklaşımlar, aslında birbirine benzer; aralarındaki temel fark, çıkışı ve bir sonraki durum bilgilerini üretiş şeklinde saklıdır. *Mealy* makinalarında *t* anındaki çıkış değeri *t* anındaki durumlara ve *t* anındaki girişlere bağlıdır. *Moore* makinalarda ise, *t* anındaki çıkış değeri yalnızca *t* anındaki durumlara bağlıdır; *t* anındaki giriş değerleri, yalnızca bir sonraki durum bilgilerini üretmek için kullanılır; o andaki çıkış değerlerine bir etkisi yoktur. Şekil-10.2 de bu iki makinanın mimarisi gösterilmiştir. Aşağıda ise fonksiyonel bağıntısı görülmektedir. Burada, *t*:şimdiki anı, *t+1*: bir sonraki anı; *Z*: çıkışı; *d*: durumu, *x*: giriş fonksiyonunu göstermektedir.

- ***Mealy* makinası :** $Z(t)=f(d(t), x(t))$
  $d(t+1)=f(d(t), x(t))$
- ***Moore* makinası :** $Z(t)=f(d(t))$
  $d(t+1)=f(d(t), x(t))$



Şekil-10.2. *Mealy* ve *Moore* makinalar.

Yukarıdaki şekilde görüldüğü gibi *Mealy* makinada herhangi bir anda çıkış, yine o andaki girişlere ve durumlara bağlıdır; kombinezonsal devre, çıkışı üretmek ve bir sonraki durumları elde etmek için kullanılır. *Moore* makinada ise iki tane kombinezonsal devre vardır; biri, çıkışı ve bir sonraki durumların bir kısmını üretirken,

diğeri giriş değerlerinden bir sonraki durum bilgisi değerlerini elde etmek için kullanılır. Uygulamada bazı örnekler doğası gereği *Mealy*, bazılarıysa *Moore* makinası yaklaşımına uygundur. Tasarımda uygun yaklaşım yöntemi seçilirse hem tasarım süreci kolaylaşır hem de optimum çözüm elde edilmiş olunur.

Diğer bir yaklaşım şekliyse kanonik olarak adlandırılan yaklaşımdır. Kanonik yaklaşımın, mimarisel gösterimi, aslında, *Mealy* makine gibidir. *Moore* makinasına da benzer... Kanonik yaklaşım, daha çok, ardışıl devre tasarımı ve analizi sürecinde geçen kavramları, yapıları ve teknikleri göstermek için kullanılır.

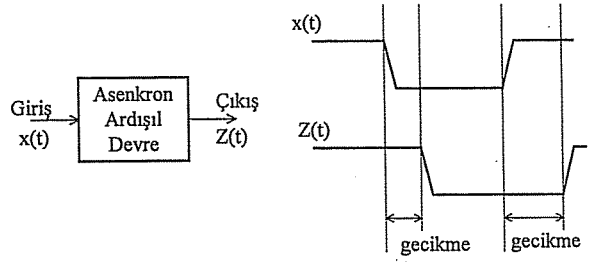
Kanonik yaklaşım, ASIC[1] tasarım araçlarında kullanılan çok düzeyli modüler tasarımda, yalnızca bir düzey içerisindeki belirli bir ardışıl devrenin/standart birimin tasarlanmasında kullanılır. Uygulamada,bir sayısal sistemin tasarım ve analiz süreçlerini kolaylaştırmak standart olarak kabul görmüş birimler kullanılması alışagelmiştir. Bu gibi birimler ASIC tasarım araçlarında bolca vardır. Üstelik bu standart birimlerin birçoğu ayrık tümdevre olarak da üretilmektedir: aritmetik işlem tümdevresi, ALU, sayıcı vs. gibi.

10.1.2. Asenkron & Senkron Ardışıl Devreler

Ardışıl devreler, girişlerinde olan değişikliğin çıkışlarına yansıtılma zamanları açısından iki şekilde sınıflanır. Birinde, girişlerinde olan değişiklik, beklemeksizin, yalnızca kullanılan kapı vs gibi birimlerin gecikme süresi kadar sonra çıkışa yansıtılır. Burada herhangi bir senkronlama işareti yoktur. Dolayısıyla bu çalışma şekli *asenkron* olarak adlandırılır. Diğerinde ise, girişlerde olan değişiklik çıkışlara hemen yansıtılmaz; değişikliğin yansıtılması için bir senkronlama işareti kullanılır. Bu işaret aktif olduğunda girişlerde olan değişiklik çıkışlara yansıtılır; burada da, tabi ki, devrede kullanılan kapı vs gibi birimlerin gecikmesi söz konusudur. Bu şekilde çalışma şekliyse *senkron* olarak adlandırılır.

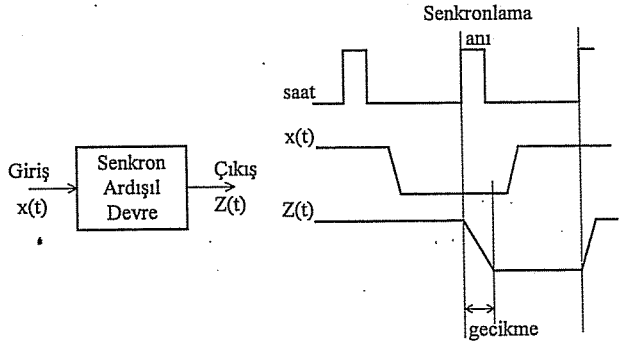
Ardışıl devre, asenkron çalışma şekline göre davranış gösteriyorsa *asenkron ardışıl* devre, senkron çalışma şekline göre davranış gösteriyorsa *senkron ardışıl* devre olarak adlandırılır. Gerçekte, asenkron çalışma işin doğasına en yakın olanıdır denilebilir; girişlerinde olan bir değişiklik doğrudan çıkışlara yansıtılmakta, ancak, bu tür çalışmada, umulmadık zamanlama hataları olabilmekte ve devre istenmeyen durumlara geçebilmektedir. Bu da, lojik tasarımda pek istenmeyen bir durumdur; lojik tasarımda güvenli ve güvenilir davranışı olan devre tasarlanması istenir. Büyükçe bir ardışıl devrenin, zamanlama hatası üretmeyecek şekilde asenkron olarak tasarlanması güçtür ve tasarım sorunları da fazladır. Ancak, tabi ki tasarlanabilir. Bu sorunların üstesinden gelebilmek için senkron çalışma şekli tercih edilir.

[1] ASIC (Application Specific Integrated Circuit): Belirli bir işi yapmak için özel tasarlanmış tümdevre.



Şekil-10.3. Asenkron devrenin zaman davranışı.

Senkron ardışıl devrelerde, bütün durum geçişleri *saat* işareti olarak adlandırılan senkronlama işaretine bağlıdır; ancak, saat işaretinin senkronlama anında, giriş işaretindeki değişiklik çıkışa yansıtılır (bkz. Şekil-10.4). Onun dışında, giriş işaretindeki değişiklik çıkışa yansıtılmaz.



Şekil-10.4. Senkron devrenin zaman davranışı.

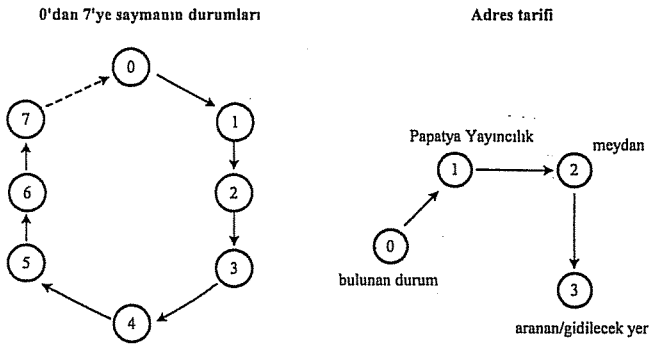
Eğer giriş işaretinin değişme sıklığı saat işaretinin değişme sıklığından büyükse, girişteki bazı değişiklikler hiç algılanmayabilir. Bu nedenle saat işareti değişme sıklığı, ki bu frekans olarak bilinir, giriş işaretinin değişme sıklığından büyük seçilir. Saat işareti, genel olarak, kare dalga veya Şekil-10.4 deki gibi darbe şeklinde olabilir; kare dalgada lojik 0'a karşılık düşen boşluk ve lojik 1'e karşılık düşen darbe oranları eşittir. Darbe şeklinde işaretlerdeyse, lojik 1'e karşı düşen darbe oranı oldukça küçüktür; idealde ne kadar kısa süreli olursa o kadar iyi olur.

Saat işareti senkronlama anı değişik şekillerde seçilebilir; bu durumda tasarımda kullanılacak temel birimler de buna uygun seçilmelidirler. Şekil-10.4 de verilen zaman davranışında saat işaretinin yükselen kenarı senkronlama anı olarak kullanılmıştır; bu,

kısaca, *yükselen kenarda tetikleme* olarak adlandırılır. Senkronlama anı düşen kenar (lojik 1'den lojik 0'a geçiş) olursa *düşen kenarda tetikleme*, kenar değil de işaretin lojik 1 veya 0 olması kullanılırsa *düzey tetikleme* olarak adlandırılır.

10.1.3. Durumlar ve Durum Diyagramı

Durum, ardışıl devrenin kendisine atanan fonksiyonu yerine getirebilmesi için bulunacağı herhangi bir konumdur denilebilir; ardışıl yaklaşımla yapılacak bir iş, herşeyden önce durumlarla ifade edilmelidir. En basit bir ardışıl devre, biri başlangıç diğeri sonuç (çıkış) olmak üzere iki duruma sahiptir; birçok tasarımda ara durumlarda olacağından gerçekte durum sayısı ikiden fazla olur. Örneğin 0'dan 7'ye kadar sayma işlemi yapan bir ardışıl devrede 8 tane durum vardır (bkz. Şekil-10.5); devre, başlangıçta 0 durumundadır ve bir sonraki durumu 1, daha sonra 2, 3, 4, 5, 6, 7 olacaktır. Görüleceği gibi devrenin 7 durumuna geçmesi için 6 durumunda; 6 durumuna geçmesi için 5 durumunda bulunması gerekir...



Şekil-10.5. Durumlara örnekler (0'dan-7'ye sayma ve adres tarifi örneği).

Ardışıl yaklaşıma günlük yaşamdan da örnek verilebilir; örneğin bir yer tarifi yapılırken şöyle denilebilir: İleriden sola dönünüz, bir süre ilerledikten sonra orada Papatya Yayıncılığı göreceksiniz, oradan sağa dönünüz ve karşınıza meydan çıkar; oradan doğru gidiniz gibi... Burada da durumlar söz konusudur; bulunulan konum, sola dönülünce Papatya Yayıncılığın görülmesi ve sağa dönülünce meydan çıkması söz konusu durumlara karşılık düşer...

Ardışıl devre tasarımında durumların belirlenmesi sistemin gerçeklenmesi için temel adımdır denilebilir. Çünkü sistemin davranışı ancak durumlar ve durumlar arası geçiş koşullarından belirlenebilir. Sistemin kaç tane duruma sahip olacağı net olarak belirlenmelidir. Durumlar ve durumlar arası geçiş koşullarını belirli düzende göstermek için durum tabloları ve durum diyagramları kullanılır.

Durum Tablosu

Bir ardışıl devreye ait durumları ve durumlar arası geçişi göstermek için kullanılır. Durum tablosunda, devrenin alabileceği bütün olası durumlar, onlara karşılık gelen bir sonraki durumlar ve her durum için çıkış değerleri açık ve net olarak gösterilir. Bu tablo tasarım veya analiz aşamasında oldukça yarar sağlar.

Örnek-10.1.

0'dan 7'ye sayan ve çıkışını tek sayılarda lojik 1 yapan bir ardışıl devrenin durum tablosunu oluşturunuz.

Burada 8 ayrı durum vardır; herbir duruma ait sayının ikili olarak kodlandığı düşülürse, durum tablosu aşağıdaki gibi oluşturulur. Çıkış sütunu bir sonraki durumlar sütunlarından çıkarılır. Tablo-10.1 de görüleceği üzere bir sonraki durumlar sütununda sayının değeri tek ise çıkış 1, çift ise çıkış 0 yapılmıştır.

Tablo-10.1. Durum tablosu örneği: 0'dan 7'ye sayma

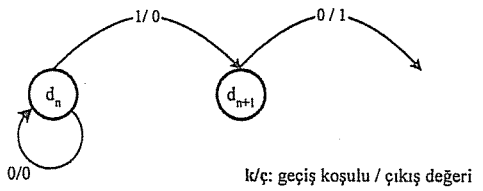
Şimdiki durumlar			Bir sonraki durumlar			Çıkış
A	B	C	A	B	C	Z
0	0	0	0	0	1	1
0	0	1	0	1	0	0
0	1	0	0	1	1	1
0	1	1	1	0	0	0
1	0	0	1	0	1	1
1	0	1	1	1	0	0
1	1	0	1	1	1	1
1	1	1	0	0	0	0

Ardışıl devreye ait durum tablosu birkez oluşturulduktan sonra, ilgili devrenin lojik elemanlarla tasarlanması ve gerçekleştirilmesi oldukça kolaydır; durum tablosunda yapılacak bir hata tasarlanan devrenin hatalı olmasına neden olur. Ardışıl devre analizi süreci, çoğu zaman devrenin durum tablosunun bulunmasıyla son bulur.

Durum Diyagramı

Durum diyagramı bir ardışıl devrenin durumlarını ve durumlar arası geçişi gösteren bir graftır; durum tablosundan farkı gösterimin grafiksel olarak yapılmasıdır. Şekil-10.6 de görüldüğü gibi durumlar birer dairesel düğüm, durumlar arası geçişse

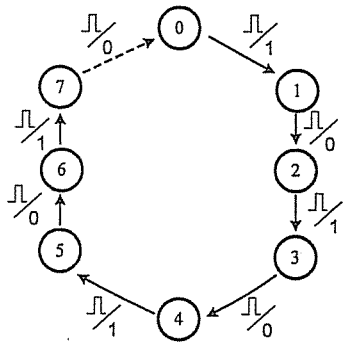
birer çizgiyle gösterilir. Durum düğümlerine anlamlı bir isim verilirken, geçiş çizgilerine geçiş koşulu ve o andaki çıkış değeri yazılır. Geçiş çizgileri yönlendirilmiş olmalıdır; yani, nereden çıkıp nereye gittiği açık olarak belirlenmelidir. Ayrıca, geçiş çizgisinin hemen üstüne geçiş koşulu ve çıkış değeri k/ç notasyonunda yazılır. Örneğin 1/0 yazılırsa, geçiş koşulu girişin lojik 1 olmasını gerektiğini belirtirken çıkış 0 olduğunu gösterir; ilk verilen geçiş koşulu ikinci verilen çıkış değeri olarak yorumlanır.



Örnek-10.2.

0'dan 7'ye sayan ve çıkışını tek sayılarda lojik 1 yapan bir ardışıl devrenin durum diyagramını çiziniz.

Burada, durum değişikliği için özel bir kontrol işareti kullanılmamaktadır; saat işaretinin senkronlama anı geçiş için yeterli olmaktadır; çıkışın tek sayı olduğu durumlara geçilirken çıkış değeri lojik 1, çift olurken lojik 0 yapılmıştır.



Şekil-10.6. Durum ve geçiş çizgisi örneği.

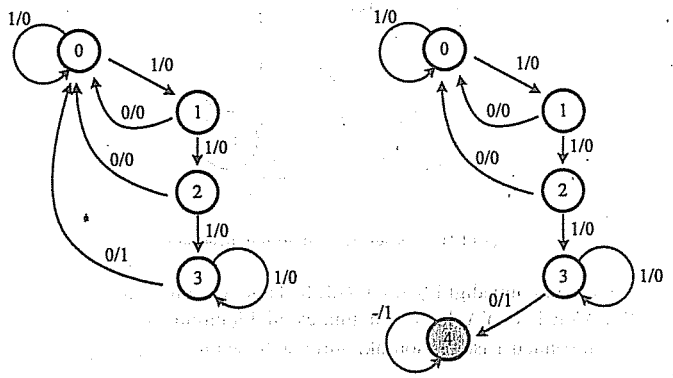
Durum diyagramları çizilerken, genel olarak, saat işareti gösterilmez; çünkü aksi söylenmediği sürece tasarlanan devrenin senkron ardışıl devre olduğu varsayılmaktadır ve her senkron ardışıl devrede bir saat işareti olmalıdır.

Örnek-10.3.

Basit bir şifreleme devresi yapılmak istenmektedir; devrenin 1 girişi ve 1 çıkışı vardır ve kendisine girilen şifre koduna göre çıkışı lojik 1 yapması istenmektedir. Şifrenin lojik değerde girildiği ve 1110 olduğu varsayılmaktadır. Yani, ard arda 3 tane lojik 1 ve daha sonra lojik 0 girildiğinde çıkışın 1; aksi durumda çıkışın 0 olması istenmektedir. Devrenin olabilecek durumlarını belirleyiniz ve durum diyagramını çiziniz.

Şifre : 1110

Durum diyagramını çizebilmek herşeyden önce durumların sayısı belirlenmelidir. Bu şifre örneği, 1110 bit deseni, için 4 veya 5 durumlu çizilebilir. Eğer başlangıç durumuyla sonuç durumu aynı düğüm olursa 4 düğümlü olabilir; bu durumda şifrenin her tespit edilmesinden sonra başlangıç durumuna dönülür ve şifre tespit süreci yeniden başlar (bkz. Şekil-10.7.a). Eğer, şifrenin bir kez tespit edilmesi ve şifre tespit edilir edilmez sistemin kilitlenmesi istenirse 5 durum gerekir (bkz. Şekil-10.7.b).



a) Başlangıç durumuna dönülmesi b) Çıkışın kilitlenmesi

Şekil-10.7. Şifre örneği (1110 şifresi).

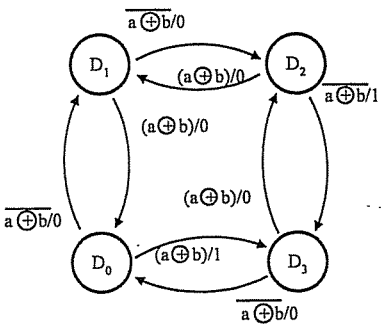
Başlangıç durumu, henüz hiçbir bilginin gelmediği veya henüz istenmeyen değerlerin geldiği durumdur; gelen bilgi 1 ise, bir sonraki duruma geçer; 0 ise, bulunduğu durumda kalır. Yani, sanki hiçbir bilgi gelmediği varsayılır. Bir sonraki durumda iken (1 nolu düğüm), yine 1 geldiğinde bir sonraki duruma geçer; 0 gelirse başlangıç durumuna döner; şifrenin yakalanması için ilk önce ard arda 1 1 1 bit dizisi gelmelidir. Bu dizi geldiğinde 3 numaralı duruma geçilir ve bu durumda iken 1

gelirse olduğu yerde kalınır; 0 gelirse şifrenin doğru girildiği tespit edilmiş olunur. İçi taralı düğüm kilitlenen düğümdür. Bu düğüme bir kez girildiğinde orada kalınır.

Örnek-10.4.

Bir ardışıl devrenin girişleri G={a, b}, çıkışları Ç={w, z} ve durumları D={D_0, D_1, D_2, D_3 } kümeleriyle temsil edilmektedir. Bu tanımlamaya göre girişlerin herikisi aynı olduğunda bir sonraki, farklı olduğunda bir önceki duruma geçen ardışıl devrenin durum diyagramını tasarlayınız. Çıkışlar D_3 durumuna geçerken lojik 1 olmaktadır.

Bu örnekte durumların sayısı ve adları verilmektedir; D_0, D_1, D_2, D_3 olarak adlandırılan 4 durum vardır. Buna göre durum diyagramı aşağıdaki gibi olur.



Şekil-10.8. Örnek-10.4 ün durum diyagramı.

İki girişin aynı olup olmadığı EŞDEĞERLİLİK kapısıyla anlaşılabilir; bu, ⊕ ile ifade edilen DIŞLAYICI VEYA kapısının tümleyeni biçiminde gösterilir. Dolayısıyla a⊕b işleminin sonucu 1 ise bir sonraki duruma, 0 ise bir önceki duruma geçilir.

Sonlu Durum Makinaları

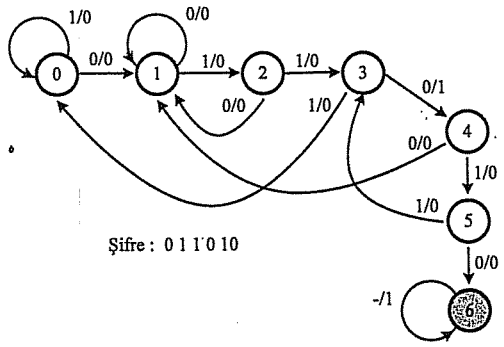
Sonlu durum makinası, çıkışı elde etmek için belirli sayıda geçmiş bilgisine ihtiyaç duyan bir ardışıl devre uygulamasıdır. Bazı ardışıl devre uygulamalarında çıkış hemen o andaki durum bilgilerinden üretilirken bazı uygulamalarda çıkış belirli sayıda geçmişte kalan durum bilgilerine dayanılarak üretilir. Örneğin şifre çözme devreleri tipik bir sonlu durum makinasıdır. Şifre bulunana kadar durumlar üzerinde hareket edilir ve şifre karakterleri doğru sırada alınmışsa çıkış üretilir. Buradan görüleceği gibi şifrede kullanılan karakter sayısı kadar geçmiş durum bilgilerine ihtiyaç vardır. Her ardışıl devre sonlu durum makinası özelliğine sahip değildir; birçok uygulama için yalnızca bir önceki durum bilgisi yeterlidir. Örneğin kendisine

ard arda gelen bit dizisi içerisinde birlerin sayının tek mi çift mi olduğunu bulan bir devrede, yalnızca o ana kadar gelenlerin tek mi çift mi olduğunun bilinmesi yeterlidir.

Örnek-10.5.

Kendisine, 011010 şeklinde bit dizisi olarak gelen şifreyi çözen ve şifre bulunduğunda çıkışını lojik 1 yapan bir girişli bir çıkışlı sonlu durum makinasına ait durum diyagramını çiziniz ve durum tablosunu oluşturunuz.

Şifreyi çözecek durum diyagramı için herşeyden önce gerekli durumların sayısı belirlenmelidir. Birinci durum başlangıç durumudur; yani, henüz hiçbir bit'in gelmediği veya istenmeyen bit'in geldiği durumdur. Kaç durum gerektiği doğrudan şifredeki karakter sayısından ve başlangıç/çıkış durumlarında belirlenir. Eğer başlangıç ve çıkış durumları aynıysa, gerekli durum sayısı doğrudan şifredeki karakter sayısı kadardır; başlangıç ve çıkış durumları farklı seçilirse, devre için gerekli durum sayısı şifredeki karakter sayısına 1 eklenerek bulunur. Bu örnek için başlangıç ve çıkış durumlarının ayrı olduğu varsayılmıştır. Buna göre devre için gerekli durum sayısı, şifre 6 tane karakter içerdiğinden bir fazlası olan 7'dir. Bu 7 durum arasındaki geçişler aşağıdaki şekilde gösterildiği gibi olur:



Şekil-10.9. Şifre örneği (011010 şifresi)

011010 şifresini çözecek bir ardışıl devrenin durum diyagramı yukarıdaki gibi olur; etiketleri 0'dan başlayıp 6'ya kadar olan durumlardan 0 başlangıç, 6 çıkış durumunu gösterir. Başlangıç durumunda iken, 1 geldiği sürece orada kalır; 0 gelirse bir sonraki duruma geçer. Bir sonraki durumunda iken, 0 gelirse orada kalır; 1 geldiğinde bir sonraki duruma, yani 2 nolu duruma geçer; 2 nolu duruma ard arda 0 ve 1'in gelmesiyle geçilir. Sırasıyla 3 nolu duruma 011; 4 nolu duruma 0110; 5 nolu duruma 01101 ve 6 nolu duruma 011010 gelmesiyle geçilir.

Dikkat edilirse, 5 nolu durumdayken, yani 01101 gelmişken, 0 gelirse çıkış durumuna geçilmektedir; ancak, 1 gelirse şifre doğru sırada gelmiş olmaz. Çünkü 011011 bit dizisi gelmiş olur. Bu doğru şifre değildir ancak, son üç bit (011011) doğru şifrenin ilk üç bitiyle aynıdır. Bu nedenle şifre doğru gelmemiş olsa bile ilgili ara durum olan 3 nolu duruma dönülür. Çünkü 3 nolu duruma gelinmesi için ard arda 011 gelmesi gerekir. Şekil-10.9 daki diyagrama ait durum tablosu Tablo-10.2 de verilmiştir.

Tablo-10.2. Şifre örneği durum tablosu (Örnek-10.5).

durum etiketi	durum $d_2d_1d_0$	giriş x	bir sonraki durum $d_2d_1d_0$	çıkış z
0	000	0	001	0
0	000	1	000	0
1	001	0	001	0
1	001	1	010	0
2	010	0	001	0
2	010	1	011	0
3	011	0	000	0
3	011	1	100	0
4	100	0	001	0
4	100	1	101	0
5	101	0	110	0
5	101	1	011	0
6	110	0	110	1
6	110	1	110	1

Yukarıdaki tablo, Şekil-10.9 da verilen durum diyagramın tabloya dökülmüş halidir; görüleceği gibi 0 etiketli durumda iken x=0 ise 1 etiketli duruma geçmekte, x=1 için olduğu durumda kalmaktadır. Tüm durumlar için geçiş şekli incelenirse durum diyagramını sağladığı görülür. Bu şekilde durum tablosu belirlenmiş bir ardışıl devrenin tasarımı oldukça kolaylaşır. Yapılması gereken flip-flop türünün seçilmesi ve flip-flop'un girişlerine ait durum geçiş fonksiyonlarının belirlenmesidir.

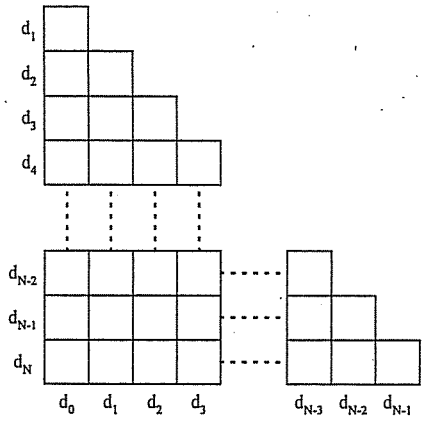
Durum İndirgeme ve Durum Atama

Ardışıl devre tasarımı sırasında oluşturulan durum tablosunda bazı durumlar aynı giriş değerleri için farklı durumlara gidiyormuş gibi gözükebilir. Ancak bunun tasarımcı tarafından araştırılması ve devrenin farklı çıkış değerlerini verip vermediğinin belirlenmesi gerekir. Eğer devre aynı giriş değerleri için aynı çıkışları veriyor ve sonraki durumlar da aynı oluyorsa, durum tablosunda birbirinin aynısı olan durumlar farklı durum değişkenleriyle temsil ediliyor anlamına gelir; bu nedenle, birbirinin aynı durumları ayıklamak için durum indirgemesinden yararlanılır. Durum indirgeme işleminden sonra farklı durum değişkenleriyle gösterilen durumlar ortadan kaldırılmış, sadece tek bir durum geçişiyle ifade edilmiş olunur.

Durum indirgeme için görüşe dayanan ve durum tablosuna bakılarak gerçeklenen yöntemler bulunmakla birlikte karmaşık devrelerde durum indirgemesine imkan tanıyan sistematik yöntemler de mevcuttur. Eşdeğerlik tablosu bunlardan en çok kullanılanıdır ve burada ayrıntılı olarak irdelenmiştir.

Eşdeğerlik Tablosu ile Durum İndirgeme

Burada eşdeğerlik tablosu oluşturularak durum indirgemeye sistematik bir yaklaşım getirilmiştir. N durumlu bir ardışıl devreye ilişkin eşdeğerlik tablosu Şekil-10.10 da verilmiştir.



Şekil-10.10. Eşdeğerlik tablosu genel gösterilimi

Şekilden görülebileceği gibi dikey sıralamada ilk durum, yatay sıralamada ise son durum yer almamaktadır. Yapılacak işlem, herbir durumunun bir diğeriyle eşdeğer olup olmadığını belirlemek ya da koşullara bağlı ise bu koşulları ortaya çıkartmaktır.

Tablonun kesişim noktalarında; örneğin d_2, d_{N-2}, eğer durumlar eşdeğer değilse X, eşdeğer ise ✓, eğer koşullara bağlı ise bu durumlar koşul olarak belirtilir; örneğin d_0, d_2. İşleme, önce tamamen bağımsız eşdeğer olmayan durumlardan başlanır ve bunların kesişim noktalarına X konulur. Sonra d_0 durumuyla en üstte yer alan d_1 durumundan başlanır ve aşağıya doğru inilir; d_N durumu da değerlendirildikten sonra işleme d_2 durumuyla devam edilir ve aynı sırayla tablonun sonuna kadar sürdürülür. Sonuçta sağlanan ve sağlanmayan koşullara bakılarak eşdeğer durumlar belirlenir. Eşdeğer durumlardan indirgeme yapılır, eğer gerekirse yeniden adlandırma yapılarak durum atama şartları yerine getirilir.

Örnek-10.6.

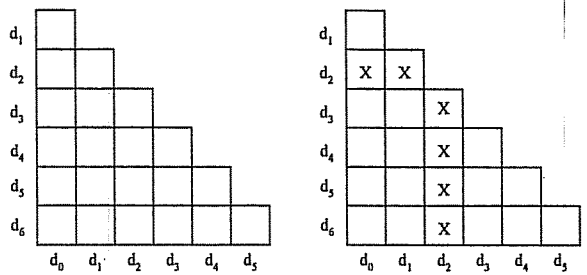
Aşağıdaki durum tablosunu inceleyiniz. Eşdeğerlik tablosu oluşturarak durum indirgemesi yapılıp yapılamayacağını belirleyiniz.

Şimdiki durumlar	Bir sonraki durumlar		Çıkışlar	
	x=0	x=1	x=0	x=1
d_0	d_1	d_0	0	0
d_1	d_2	d_4	1	0
d_2	d_5	d_2	1	1
d_3	d_4	d_0	0	0
d_4	d_2	d_4	1	0
d_5	d_4	d_3	0	0
d_6	d_4	d_3	0	0

Eşdeğerlik tablosu yöntemiyle indirgeme yapmak için yapılması gerekenler adım adım şöyledir:

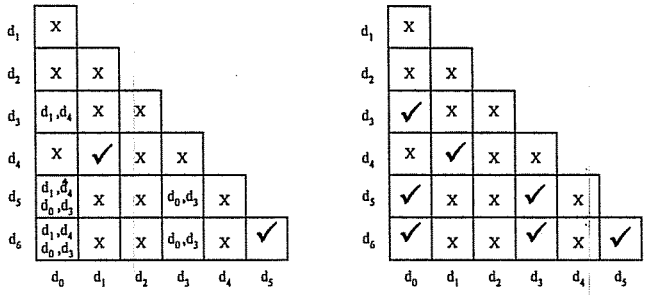
1. adım:

Dikey sıralamada ilk durum d_0, yatay sıralamada ise son durum d_6 çıkarılarak yukarıda solda verilen eşdeğerlik tablosu oluşturulur ve sonra tamamen bağımsız durumlar belirlenerek, ki burada d_2 dir, yukarıda sağda verilen eşdeğerlilik tablosunda d_2'nin bulunduğu tüm satır ve sütunlara X konulur.



2. adım:

Sütunlarda yer alan d_0 durumuyla en üst satırda yer alan d_1 durumundan başlanır; eğer durumlar eşdeğer değilse X, eşdeğer ise ✓, eğer koşullara bağlı ise bu durumlar koşul olarak belirtilir ve aşağıya doğru inilir; d_6 durumu da değerlendirildikten sonra işlem bir sonraki sütundaki d_1 durumuyla ve ikinci satırdaki d_2 durumuyla devam eder ve aynı sırayla tablonun sonuna kadar sürdürülür. Sonuçta aşağıda sol tarafta verilen eşdeğerlik tablosu elde edilmiş olur.



Bu eşdeğerlik tablosunda ilk olarak d_1 durumuyla d_0 durumu durum tablosuna bakılarak karşılaştırılır. Her iki durum aynı giriş değerleri için farklı çıkış değerlerine gitmektedir. Dolayısıyla bir sonraki durumlarda farklı olacaktır. Bu durumda iki durum eşdeğer değildir ve kesişim hanesi X yerleştirilir. Sonra d_2 durumuyla d_0 durumu durum tablosuna bakılarak karşılaştırılır, ancak d_2 burada tamamen farklı bir çıkış olduğu için önceden belirlendiği gibi hiç bir durumla eşdeğerliği söz konusu değildir. Bir sonraki karşılaştırma için $d_3 - d_0$ durumlarına geçilir. Bu durumlar ayı giriş değerleri için aynı çıkış değerlerini sağlamaktadırlar. Ancak durumlarda farklılıklar bulunmaktadır. *x* girişi 1 olduğundan her iki durum da d_0'a gitmesine rağmen *x=0* olduğunda d_3, d_4'e, d_0 ise d_1'e gider. Bu durumda d_1 ile d_4 durumlarının eşdeğerliği sorgulanmalıdır. Bu yüzden d_1, d_4 koşulu kesişim hanesine yazılır. İşlem bu şekilde sürdürülerek eşdeğerlik tablosu tamamlanır. Bu arada eşdeğer çıkan

durumlara ✓ yazılarak eşdeğerlikleri gösterilir. Örneğin d_4 ve d_1 durumları eşdeğer durumlardır. Çünkü aynı giriş değerleri için çıkış değerleri aynı olduğu gibi bir sonraki durumlar da birbirine eşittir. Bu durumda kesişim hanesine ✓ yazılır. Bu eşdeğerlik koşullu eşdeğerlikleri de sağlar. Örneğin d_3-d_0'ın eşdeğerliği d_1-d_4 eşdeğerliğine bağlı çıkmıştı. Bu durumda d_1, d_4'e eşdeğer olduğu için d_3-d_0 durumları da birbirine eşdeğer olur. Bu durumda d_3-d_0 ve d_4-d_1 durumlarının eşdeğerlik koşuluna bağlı bütün durumlar eşdeğer çıkar ve bunlar ✓ ile gösterilirler. Sonuç olarak yukarıda sağ tarafta verilen eşdeğerlik tablosu elde edilir.

Sonuç olarak d_6, d_5'e, d_5, d_3'e d_3'de d_0'a eşdeğer, bir başka deyişle bütün bu durumlar birbirine ve dolayısıyla tanımlanmış ilk durum olan d_0'a eşdeğer çıkar. Bu durumda d_6, d_5, d_3 durumları yerine d_0 durumu kullanılabilir. Ayrıca d_4 durumu da d_1 durumuna eşdeğer çıkmaktadır. Bu değerlendirmeler neticesinde geriye sadece üç durum kalır: d_0,d_1 ve d_2.

3. adım:

Sonuçta sağlanan ve sağlanmayan koşullara bakılarak eşdeğer durumlar belirlenmiş oldu. Son olarak ilk durum tablosuna bakılarak ve eşdeğer durumlar değerlendirilerek indirgenmiş durum tablosu oluşturulur.

Şimdiki durumlar	Bir sonraki durumlar		Çıkışlar	
	x=0	x=1	x=0	x=1
d_0	d_1	d_0	0	0
d_1	d_2	d_4	1	0
d_2	d_5	d_2	1	1
d_3	d_4	d_0	0	0
d_4	d_2	d_4	1	0
d_5	d_4	d_3	0	0
d_6	d_4	d_3	0	0

Ayrıca d_6=d_0, d_5=d_0, d_3=d_0 ve d_4=d_1 alınarak,

Şimdiki durumlar	Bir sonraki durumlar		Çıkışlar	
	x=0	x=1	x=0	x=1
d_0	d_1	d_0	0	0
d_1	d_2	d_1	1	0
d_2	d_0	d_2	1	1
d_0	d_1	d_0	0	0
d_1	d_2	d_1	1	0
d_0	d_1	d_0	0	0
d_0	d_1	d_0	0	0

Yazılır ve indirgenerek aşağıdaki sonuç elde edilir.

Şimdiki durumlar	Bir sonraki durumlar		Çıkışlar	
	x=0	x=1	x=0	x=1
d_0	d_1	d_0	0	0
d_1	d_2	d_1	1	0
d_2	d_0	d_2	1	1

10.1.4. Standart Tasarım Birimleri

Ardışıl devre tasarımında kullanılan lojik elemanlar, genel olarak, kombinezonsal devreler, flip-flop'lar, tutucular, saklayıcılar, sayıcılar ve çeşitli bellek elemanları olarak sıralanabilir. Temel yaklaşım, durumları tutmak için saklayıcı ve bellek elemanları, geçişleri belirlemek için de kombinezonsal devreler kullanılır:

Standart tasarım birimleri aşağıdaki gibi sınıflanabilir:

- Kombinezonsal Devre
- Flip-Flop (FF)
- Tutucu (Latch)
- Saklayıcı (Register)
- Sayıcı (Counter)
- RAM, ROM gibi bellekler
- Programlanabilir Ardışıl Diziler (PSA).

Uygulamaya göre tüm bu elemanlar bir devrede bulunabilir; ancak, en yalın bir ardışıl devrede bulunması gereken elamanlar FF ve kombinezonsal devredir. Örneğin 1 tane FF ve 1 tane lojik kapı ile ardışıl lojik devre tasarlanabilir. Uygulama büyüdükçe, tasarım esnekliği ve maliyet açısından saklayıcı, sayıcı ve bellek gibi birimlerinin kullanılması kaçınılmaz olur.

- **Kombinezonsal Devreler**

Kombinezonsal devreler yalnızca lojik kapılara dayalı devrelerdir; devrenin herhangi bir andaki çıkış değeri yine o andaki giriş uçlarındaki değerlere bağlıdır. Giriş uçlarında olan bir değişiklik belirli bir gecikme sonrası çıkışa yansır; gecikme, doğrudan giriş ile çıkış arasındaki kapıların gecikmelerinin toplamıdır. Ayrıntı için bkz. Bölüm 7.

Kombinezonsal devreler ya ayrık şekilde üretilmiş kapılar kullanılmasına dayanılarak veya içerisinde birçok kapı olan PLA veya PAL'lar kullanılarak tasarlanır. Büyükçe karmaşık bir devre tasarımında PLA veya PAL kullanılması olası tümdevre sayısını azaltacaktır. Ayrıntı için bkz. Bölüm 9.

- **Flip-Flop (FF)**

Flip-flop'lar temel saklama birimi olarak ardışıl devre tasarımında bolca kullanılır; 1 bitlik durum bilgisi saklanması için kullanılır; yanyana dizilerek *n* bitlik durum bilgisi saklayabilecek bir FF dizisi oluşturulabilir. Tasarım aşamasında esneklik sağlanması ve tasarlanan devreye ait geçiş fonksiyonlarının basit çıkması için D, JK, SR ve T olarak adlandırılan birçok FF türü vardır. Ayrıca, flip-floplar saat işaretiyle tetiklenme özelliklerine kenar ve düzey tetiklemeli olarak da adlandırılır. Ayrıntı için bkz. Ayrıt 10.2.

- **Tutucular**

Tutucular, flip-flop'lar gibi 1 bitlik veri saklama için kullanılan en temel saklama birimleridir. Flip-flop'ların tasarımı da tutuculara dayanmaktadır denilebilir. İki tane TVE (NAND) veya TVEYA (NOR) kapısıyla yalın bir tutucu tasarlanabilir; bu, SR tutucu olarak adlandırılır. S ve R girişlerindeki değerlere göre çıkışları asenkron olarak değişir. Bir saat işaretine bağlı olarak senkron davranması için 2 tane VE (AND) kapısı kullanılarak girişler saat işaretiyle maskelenebilir. Ayrıntı için bkz Ayrıt 10.3.

- **Saklayıcılar**

Saklayıcı, yanyana dizilmiş FF kümesidir. FF'un tek bitlik bilgi tutmasına karşılık saklayıcılar birden çok bitlik bilgilerin tutulması/saklanması için kullanılır; 8 bitlik bilgi tutan bir saklayıcı 8 bitlik, 16 bitlik bilgi tutan saklayıcılar 16 bitlik saklayıcı olarak adlandırılırlar. Saklayıcılar bilginin giriş ve çıkış şekline göre paralel/paralel, paralel/seri, seri/paralel ve seri/seri olarak sınıflanırlar. Buradaki paralelin anlamı tüm bitlerin aynı anda gireceği ve çıkacağını; seri ise, bitlerin tek tek gireceğini ve çıkacağını gösterir. Üzerinde tuttuğu bit dizisini sağa ve sola öteleyen saklayıcılara ötelemeli saklayıcı denir. Ayrıntı için bkz. Bölüm 11 ve Ayrıt 11.1.

- **Sayıcılar**

Sayıcılar, saklayıcılarda olduğu gibi *n* bitlik bilgi tutmanın yanısıra, her saat çevriminde çıkış uçlarındaki değeri belirli bir sırada değiştiren elemanlardır. Örneğin, çıkışlarındaki değeri, 0'dan başlayıp birer artımla 15 kadar değiştiren bir saklayıcı 4 bitlik yukarı sayıcı olarak adlandırılır. Sayıcı 4 bitliktir. Çünkü 4 bit ile 16 değişik durum gösterilebilir. Bir sayıcı aksi söylenmediği sürece ikili ve yukarı sayıcı olarak varsayılır. İkili sayıcı (*binary counter*) demek, *0*'dan başlayıp 2^n-1 "e kadar birer artımla (0,1... 2^n-2, 2^n-1) sayması anlamına gelir. Sayıcılar, özel amaçlı gereksinimleri karşılamak için değişik şekillerde tasarlanabilirler. Genel olarak, ikili sayıcıya ek olarak çokça kullanılan sayıcılar şunlardır: Ondalık, Halka, Modulo, *Ripple* sayıcı...

Sayıcılar sayma şekillerine göre, ileri (up), geri (down), ileri/geri (up/down), belirli bir değerden başlayan; ve, saat işareti uygulanmasına göre senkron ve asenkron olarak sınıflandırılırlar. Ayrıntı için bkz. Bölüm 11 ve Ayrıt 11.2.

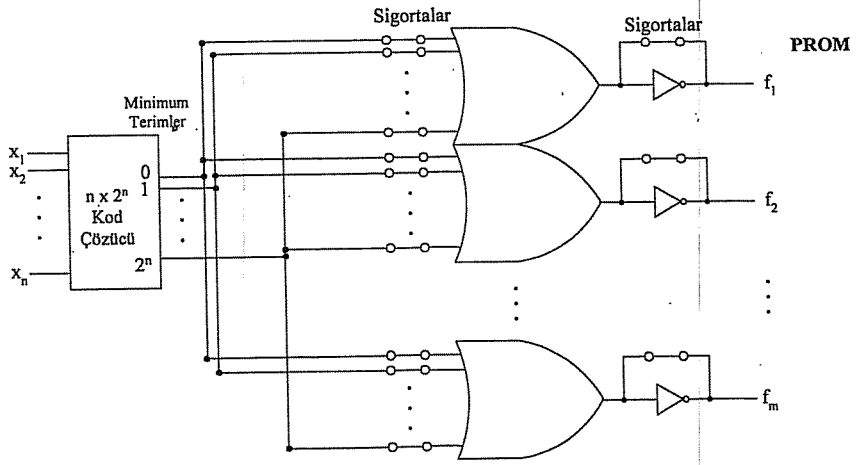
- **RAM, ROM gibi Bellekler**

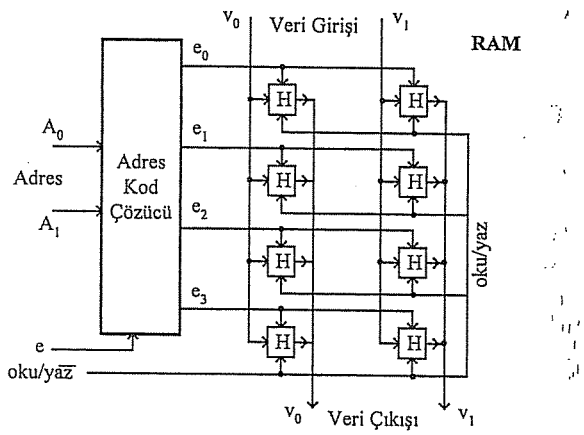
Ardışıl devre tasarımında durum bilgilerinin saklanması için bellek birimleri de kullanılabilir. Küçük boyutlu devrelerin tasarımı FF, saklayıcı ve sayıcılar ile kotarılabilirken, durum diyagramındaki durumların sayısı çok arttığında veya saklanması gereken bilgini boyutu çoğaldığında bellek birimlerinin kullanılması daha verimli ve etkin olur. Bellekler çeşitli türlerde üretilirler. Ayrıntı için bkz. Bölüm 11 ve Ayrıt 11.3.

- **RAM** (*Random Access Memories*)
- **ROM** (*Read Only Memories*)
- **PROM** (*Programable ROM*)
- **EPROM** (*Erasable Programmable ROM*)
- **E²PROM** (*Electrically Erasable Programmable ROM*)
- **CAM** (*Content Addressable Memories*)

- **Programlanabilir Ardışıl Dizileri**

Programlanabilir ardışıl diziler (*Programmable Sequential Arrays*, PSA), kombinezonsal devrelerde kullanılan PLA'lara benzer; ancak burada PLA'dan farkı dizinin içerisinde kapılara ek olarak durum bilgisini tutan FF'lar veya saklayıcılar olmasıdır.

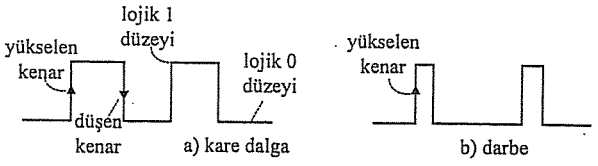




PROM (Şekil-9.1 den alınmıştır) ve RAM'e (Şekil-11.13 den alınmıştır) ait iç mimariler

10.1.5. Kenar ve Düzey Tetikleme

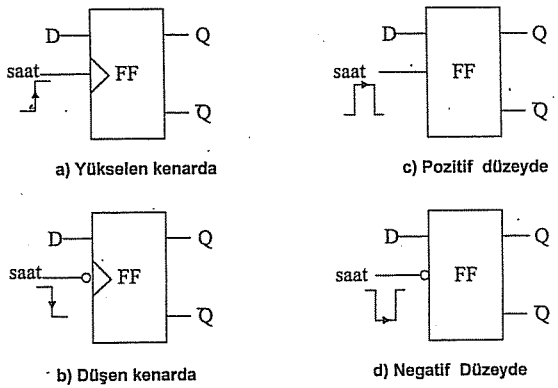
Tetikleme, bir ardışıl devrenin veya flip-flop, saklayıcı ve sayıcı gibi birimlerin o andaki durumunu değiştirip bir sonraki duruma geçmesi için dışarıdan uygulanan uyarma işaretidir. Böyle bir işarete senkron ardışıl devrelerde ihtiyaç duyulur. Çünkü, asenkron ardışıl devrelerde girişlerde oluşan değişiklik, bir uyarma işaretine gerek duyulmaksızın bir miktar gecikmeyle durumları ve çıkışları etkiler. Ardışıl devrelerde senkronlama anı (bkz. Ayrıt 10.1.2) saat işareti denilen tetikleme işaretiyle bildirilir; saat işareti genel bir isimlendirmedir: Tetikleme işareti ise saat işaretinin belirli bir bölgesi ve anıdır denilebilir. Örneğin saat işareti aşağıda görüldüğü gibi kare dalga ve darbe şeklindeyse, tetikleme işareti kare dalganın veya darbenin 0'dan 1'e geçiş anı, kara dalganın 1 veya 0 olması ve 1'den 0'a geçiş anı olabilir. Geçiş anlarına, yani 0'dan 1'e veya 1'den 0'a geçişlere <u>kenar tetikleme</u> denilirken; işaretin 1 veya 0 olması <u>düzey tetikleme</u> olarak adlandırılır:



Senkron ardışıl devrelerde tetikleme, biri kenar diğeri düzey olmak üzere iki türlü yapılabilmektedir. Kenar tetiklemede işaretin yükselen ya da düşen kenarında durum değişikliği yapılır; düzey tetiklemede ise işaretin lojik 1 veya lojik 0 olmasına göre tetikleme yapılır:

- yükselen kenar; 0'dan 1'e geçiş
- düşen kenar; 1'den 0'a geçiş
- pozitif düzey; lojik 1 olması
- negatif düzey; lojik 0 olması

Şekil-10.11 da tetikleme şekillerinin bir flip-flop örneğinde nasıl gösterildiği verilmiştir. Şekilden görüldüğü gibi kenar tetiklemeli olan birimlerin saat işareti girişi '>' simgesi koyularak belirgin hale getirilmektedir; girişten hemen önceki küçük bir daire işaretin (–o) tümleyeni anlamına gelir. Yani, yükselen ise düşen, pozitif ise negatif yapılmış olunur. Kenar tetiklemede, işaretin durum değiştirmesi yeterlidir; durum değiştirdiği algılanınca ilgili ardışıl devre veya birimde bir sonraki duruma geçiş sürecine girer. Dolayısıyla saat işaretinin kare dalga olması gerekmez; kısa süreli darbeler olması yeterlidir. Uygulamada, bu durum, çoğu zaman istenen bir özelliktir. Düzey tetiklemede ise saat işaretinin istenen düzeye gelmesi ve orada çok kısada olsa belirli süre kalması gerekir...



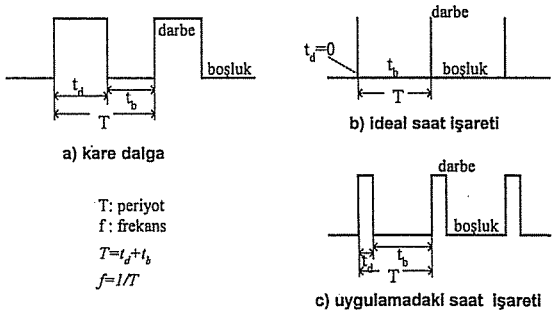
Şekil-10.11. D türü FF üzerinde değişik tetikleme biçimleri.

10.1.6. Saat İşareti ve Senkron Hücrelerin Zaman Davranışı

Senkron ardışıl devrelerde en önemli noktalardan biri saat işareti ve devrenin zamanlama davranışıdır. İdeal durumda saat işaretinin sıfır darbe genişliğinde ve birimlerin/elemanların sıfır gecikmeyle çalıştığı varsayılır. Ancak, uygulamada durum farklıdır; her zaman için gecikme söz konusu olur ve saat işareti sıfır darbe genişliğinde olamaz. Dolayısıyla bir devre tasarlanırken fonksiyonel davranışına ek olarak zamanlama davranışı da göz önüne alınmalıdır. Tasarımında tutulan yaklaşım ve kullanılan lojik birimler devrenin zamanlama davranışını da belirler. Bir devrenin hangi hızlarda çalıştırılabileceği doğrudan devrenin zamanlama davranışından çıkar. Zamanlama davranışı doğru belirlenmeyen veya tasarlanırken açık noktalar bırakılan devrelerde umulmadık davranışlar gözlenebilir; bu da, bazı kritik uygulamalarda kabul edilemeyecek sonuçlara neden olabilir.

Zamanlama davranışı en net görülebilen ardışıl devreler saat işareti engellenmemiş kenar tetiklemeli birimlere sahip olan devrelerdir. Bu tür devrelerde, sisteme uygulanabilecek saat işareti frekansı kolayca hesaplanabilir (bkz. Bölüm 8, Ayrıt 8.2.) ve öngörülmeyen zamanlama davranışları yok edilebilir. Düzey tetiklemeli ve asenkron ardışıl devrelerde zamanlama davranışına çok dikkat edilmeli ve girişlerin olası durumları karşısında devrenin zamanlama davranışı kestirilebilmelidir. İlerleyen paragraflarda kenar tetiklemeli ardışıl devrelere has zamanlama davranışları ele alınmıştır.

Bir ikili işaret, lojik olarak 1 ve 0'lardan oluşur; 1 anı darbe, 0 anı boşluk olarak adlandırılır. Kare dalga bir işarette, Şekil-10.12.a da görüldüğü gibi, darbe boşluk oranı eşittir. Ancak, ardışıl devrelere uygulanacak saat işaretinin, idealde, Şekil-10.12.b de görüldüğü gibi darbe süresinin sıfır olması istenir; ancak, uygulamada çok kısa da olsa darbe anının Şekil-10.12.c deki gibi süresi olur. İşaretin darbe süresi t_d, boşluk süresi de t_b olarak simgelendirilir.



Şekil-10.12. Çeşitli saat işareti biçimleri.

Saat işaretiyle ilgili periyot, darbe oranı, boşluk oranı ve frekans değerleri tanımı ve birimleri şöyledir:

- **Periyot -*T***

İşaretin başladığı noktaya gelene kadar geçen süre; periyodik işaretler belirli aralıklarla kendilerini şeklen tekrarlar. İşaretin, şeklen, başladığı noktaya tekrar gelmesine kadar geçen süre periyot olarak adlandırılır ve birimi saniyedir. Çok hızlı tekrarlamalarda milisaniye (ms), mikrosaniye (μs), nanosaniye (ns) ve pikosaniye (ps) olan alt katları da kullanılır. Alt katlar 10^{-3} mertebesinde azalırlar. Periyot, biri darbe diğeri boşluk olarak adlandırılan iki kısımdan oluşur.

- **Darbe oranı -t_d**

Bir ikili işaretin lojik 1 olduğu an veya değer darbe oranı olarak adlandırılır; kenar tetiklemeli devreler için, idealde, sıfır olması istenir.

- **Boşluk oranı – t_b**

Bir ikili işaretin lojik 0 olduğu an veya değer boşluk oranı olarak adlandırılır; kenar tetiklemeli devreler için, idealde, T'ye eşit olması istenir.

- **Frekans -f**

İşaretin birim zamanda kendisini tekrarlama sayısıdır; Periyodun tersi olarak değerlendirilir ve birim zaman 1 saniye ise birimi Hertz (Hz) 'dir. Üst katları sırasıyla KHz, MHz, GHz ve THz olarak gider; artımlar 10^3 mertebesindedir.

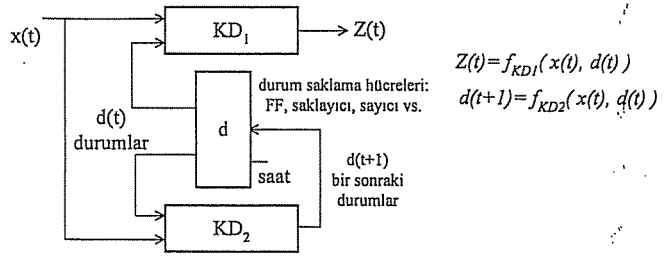
- **Genlik -V**

İşaretin lojik 1 veya 0 olması durumlarının analog değeri olarak tanımlanabilir; birimi Volt (V) olup TTL ve CMOS teknolojisinde lojik 1 idealde +5 V, lojik 0 da 0 V olarak değerlendirilir; uygulamada nasıl olduğu konusunda ayrıntılı bilgi için bakınız Ek-A: TTL ve CMOS Tümdevre Özellikleri.

Senkron Ardışıl Devrelerde Zamanlama Parametreleri

Ardışıl devreler temelde iki kısımdan oluşur; biri FF, saklayıcı, sayıcı gibi saklama özelliği olan birimler, diğeri de kombinezonsal kısımdır. Bir ardışıl devrenin genelleştirilmiş mimarisi aşağıdaki şekildeki gibi gösterilebilir; özelleştirilmiş mimarisinde ise birimler bazıları olmayabilir. Ardışıl devrelerde en önemli iki unsur çıkış fonksiyonu ve bir sonraki durum bilgisini üretecek olan durum geçiş fonksiyondur.

Genelleştirilmiş ardışıl devre mimarisinde çıkış fonksiyonu girişlere ve bir sonraki durum bilgisine bağlıdır; benzer şekilde, durum geçiş fonksiyonu da hem girişlere hem de bir sonraki durum bilgilerinden üretilir. Ancak, çıkış fonksiyonu ile geçiş fonksiyonunu için elde etmek için kullanılan kombinezonsal devre Şekil-10.13 te gösterildiği gibi farklı olabilirler; KD_1 çıkış, KD_2 geçiş fonksiyonu üretmek için kullanılan kombinezonsal devreler olsun. Bu koşullar altında çıkış üretmek için toplam gecikme, durumları saklayan birimler ile KD_1'in gecikmesinden hesaplanırken; yeni durum bilgisini elde etmek için toplam gecikme yine durumları saklayan birimlerle KD_2'nin gecikmesinden hesaplanır.



Şekil-10.13. Ardışıl devredeki birimler, çıkış ve durum geçiş fonksiyonları.

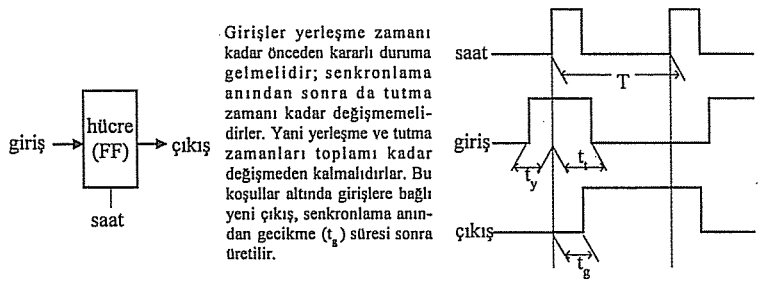
Kombinezonsal devrelerin gecikmesi, eğer kapılarla gerçekleştirilmişse doğrudan giriş ile çıkış arasında bulunan kapıların gecikmelerinin toplamından bulunur. Ardışıl devrelerde ise, gecikme, kombinezonsal devrenin ve durum saklayıcılarının gecikmelerine bağlıdır. Örneğin çıkış fonksiyonu gecikmesi KD_1 ile durum saklanması için kullanılan birimin gecikmelerinin toplamıdır; benzer şekilde geçiş fonksiyonu gecikmesi KD_2 ile durum saklanmasında kullanılan birimlerin gecikmelerinin toplamında bulunur.

$$T_t = maksimum(T_{çıkış}, T_{geçiş})$$
$$T_{çıkış} = KD_1 + d$$
$$T_{geçiş} = KD_2 + d$$

Senkron ardışıl devreye uygulanacak saat işareti frekansının ne olacağı doğrudan devrenin gecikmesine bağlıdır; toplam gecikme T_t ile gösterilirse $f=1/T_t$. bağıntısıyla elde edilen frekans değeri devreye uygulanabilecek en yüksek frekansı gösterir. Devreye uygulanacak frekans bu değerden küçük olmalıdır. İlerleyen paragrafta durum saklanması için kullanılan FF, saklayıcı, sayıcı gibi birimlerin gecikme parametreleri verilmiştir.

Senkron Hücrenin Zaman Diyagramı

Senkron hücre flip-flop, saklayıcı, sayıcı gibi birimlerin temel yapı taşıdır; bu birimlerin hepsi senkron hücrelerin inşasına dayanır. Bir senkron hücrenin şekli ve zaman davranışı Şekil-10.14 de gösterildiği gibidir; örnekler kenar tetikleme ve yükselen kenar açısından verilmiştir.



Şekil-10.14. Senkron hücre şekli ve zaman diyagramı.

Bir senkron hücre için yerleşme (*setup*), tutma (*hold*) ve geçiş gecikmesi (*transition delay*) olarak adlandırılan üç temel gecikme parametresi tanımlanmıştır. Bunların anlamı şöyledir:

- **Yerleşme zamanı** -t_y (SetUp time -t_{su})

Senkronlama anı öncesinde giriş değişkenlerinin sabit kalması gereken süre; bu süre içerisinde giriş değişkenleri sahip olduğu değerleri korumalıdır.

- **Tutma zamanı** -t_t (Hold time – t_h)

Senkronlama anı sonrası giriş değişkenlerinin sabit kalması gereken süre; bu süre içerisinde giriş değişkenleri sahip olduğu değerleri korumalıdır. Bu süre sonrası, giriş değişkenlerinin değerleri bir sonraki çıkış için değiştirilebilir.

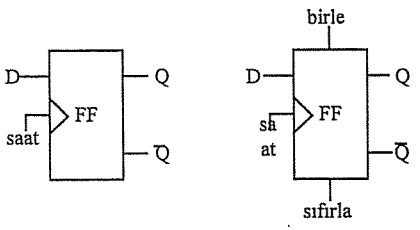
- **Geçiş geçikmesi** -t_g (Transition delay –t_d)

Senkronlama anı ile yeni durumun kararlı olmasına kadar geçen süre; yani, senkronlama anından sonra giriş değerlerinin çıkışa yansıtılması için geçen süre.

10.2. Flip-Flop'lar

Flip-flop'lar 1 bitlik saklama birimleridir; en yalın saklama elemanı olarak tek başına kullanıldığı gibi birden çok flip-flop biraraya getirilip saklayıcı, sayıcı, bellek veya özel amaçlı birimler oluşturulabilir. Mikroişlemcilerin içerisinde veya kartların üzerinde bayrak olarak adlandırılan değişikler de flip-flop'lar üzerinde tutulurlar.

Bir flip-flop'un, temel olarak iki tane çıkışı, bir tane saat işareti girişi ve bir veya iki tane de veri giriş ucu vardır. Çıkış uçları birbirlerinin tümleyenidir; birisi değeri lojik 1 ise, diğeri lojik 0'dır. Farklı türde flip-flop'lar tasarlanmıştır; herbirinin özellikleri farklıdır ve tasarım esnekliği sağlar. Uygun flip-floplar seçilirse, tasarımda ortaya çıkan kombinezonsal devrenin karmaşıklığı azaltılabilir; çünkü devreye ait geçiş fonksiyonları sadeleşir.



Şekil-10.15. Flip-flop'un genel gösterilişi.

Yukarıdaki şekilde, solda, en yalın bir flip-flop'un şekli görülmektedir; Q asıl çıkışı gösterirken $\overline{Q}$ onun tümleyenidir; D ise veri girişidir. Bu örnekte D girişindeki değer, her saat darbesinden Q çıkışına aktarılır; zaman diyagramı Şekil-10.15 de görülmektedir. Sağda ise, giriş çıkışlara ek olarak birle ve sıfırla uçları olan bir flip-flop şekli görülmektedir; birle ucu aktif olduğunda Q çıkışı lojik 1 değerini alırken, sıfırla ucu aktif olduğunda Q çıkışı lojik 0 değerini alır.

Tasarım esnekliği ve geçiş fonksiyonu sadeliği için D, JK, SR ve T olarak adlandırılan birçok flip-flop şekli ortaya atılmıştır. Herbirinin en uygun ve esnek olduğu uygulama türü vardır.

- D flip-flop
- JK flip-flop
- SR flip-flop
- T flip-flop

Flip-flop'ların girişlerindeki değerlerin çıkışlarına yansıtılması için saat işareti kullanılır; senkronlama anında, girişlerdeki değerler ve flip-flop geçiş fonksiyonu kullanılarak çıkış üretilir. Eğer senkronlama anı, saat işaretinin yükselen kenarı ise, yükselen kenarda tetiklenen, düşen kenarda ise düşen kenarda tetiklenen, lojik düzeye göre oluyorsa düzey tetiklemeli flip-flop olara anılır. Aksi söylenmediği sürece örnekler yükselen kenar için verilecektir. Şekil-10.15 de verilen flip-flop tipik bir D flip-flop'udur; D ve JK flip-flop'ları en çok kullanılan flip-flop türleridir.

- **D Flip Flop**

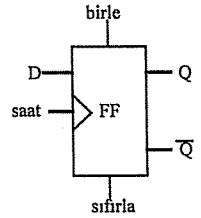
D tipi flip-flop'un D olarak adlandırılan 1 tane girişi vardır. D uçundaki giriş değeri bir sonraki çıkış bilgisidir. Giriş çıkış arasındaki bağıntı ve uyarma tablosu şöyle olur:

$$Q(t+1) = D(t)$$

$Q \backslash Q_{(t+1)}$	00	01	11	10
D	0	1	1	0

Tablo-10.3. D flip flop doğruluk tablosu.

giriş D	giriş saat	çıkışlar Q $\overline{Q}$
0	↑[2]	0 1
1	↑	1 0



- **SR Flip Flop**

SR flip-flop'u S (*Set*) ve R (*Reset*) olarak adlandırılan iki girişe sahiptir; $t+1$ anındaki çıkış değeri SR girişlerinin t anındaki giriş değerlerine bağlıdır. Giriş çıkış arasındaki bağıntı ve uyarma tablosu aşağıdaki gibidir (heriki girişin 1 olması durumunda geçiş olmaz):

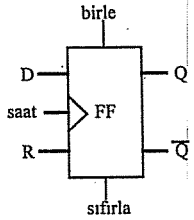
$$Q(t+1) = Q(t)\overline{R(t)} + S(t)$$

$R(t)S(t) = 0$ durumları için
$R(t)S(t) = 1$ için tanımsız

$Q \backslash Q_{(t+1)}$	00	01	11	10
S R	0 ϕ	1 0	ϕ 0	0 1

Tablo-10.4. SR flip flop doğruluk tablosu.

giriş S R	giriş saat	çıkışlar Q $\overline{Q}$
0 0	↑	Q $\overline{Q}$
0 1	↑	0 1
1 0	↑	1 0
1 1	↑	tanımsız



- **JK Flip Flop**

JK flip-flop'u SR flip-flop'a benzer; iki girişi vardır. SR'den farkı girişlerin her ikisinin 1 olması durumunda da durum geçişi olur. Giriş çıkış arasındaki bağıntı aşağıdaki gibidir:

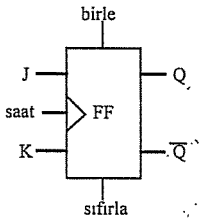
$$Q(t+1) = Q(t)\overline{K}(t) + \overline{Q}(t)J(t)$$

$Q \backslash Q_{(t+1)}$	00	01	11	10
J K	0 ϕ	1 ϕ	ϕ 0	ϕ 1

[2] ↑ : saat işaretinin etkin olmasını gösterir; yerine göre, yükselen kenarı, düşen kenarı, negatif düzey veya pozitif düzeydir.

Tablo-10.5. JK flip flop doğruluk tablosu

giriş J K	giriş saat	çıkışlar Q $\overline{Q}$
0 0	↑	Q $\overline{Q}$
0 1	↑	0 1
1 0	↑	1 0
1 1	↑	$\overline{Q}$ Q



- **T Flip Flop**

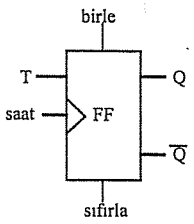
T (*Toggle*) olarak adlandırılan bir girişi vardır. Diğerlerinden biraz farklı bir flip-flop'dur: T=0 kaldığı sürece durum değiştirmez; ne zaman T=1 olursa durum değiştirir. Giriş çıkış arasındaki bağıntı ve uyarma tablosu aşağıdaki gibidir:

$$Q(t+1) = Q(t) \oplus T(t)$$

T \ $Q\,Q_{(t+1)}$	00	01	11	10
	0	1	1	0

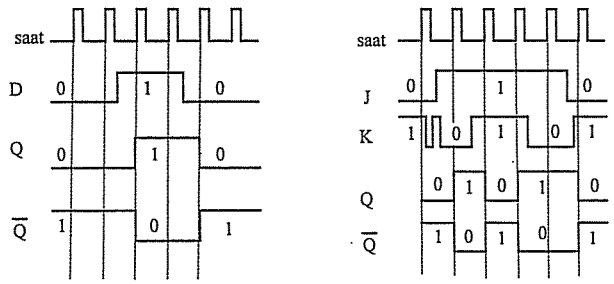
Tablo-10.6. T flip flop'u doğruluk tablosu

giriş T	giriş saat	çıkışlar Q $\overline{Q}$
0	↑	Q $\overline{Q}$
1	↑	$\overline{Q}$ Q



T flip-flop'u JK flip-flop'u kullanılarak gerçekleştirilebilir; bunun için JK girişleri birleştirilip tek bir giriş yapılırsa T flip-flop'u olur.

Uygulamada, en çok D ve JK flip-flop'ları kullanılmaktadır; birer tümdevre olarak D veya JK flip-flop'ları kolayca bulunabilir ve satın alınabilir: Bu nedenle, özellikle ayrık elemanlarda tasarımda bu flip-flop'lara dayanılarak tasarım yapılır. Ancak, ASIC tasarımında flip-flop seçiminde daha geniş bir seçenek vardır denilebilir. D ve JK flip-flop'larının zaman diyagramları Şekil-10.15 de verilmiştir; tetikleme olarak yükselen kenar ele alınmıştır.



a) D flip-flop b) JK flip-flop

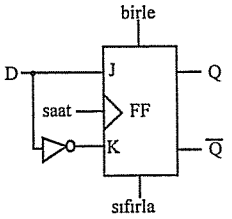
Şekil-10.15. D ve JK flip-flop'larının zaman diyagramı.

Örnek-10.5.

JK flip-flop'u kullanarak T ve D tipi flip-flop tasarlanabileceğini ayrı ayrı gösteriniz.

JK flip-flop'undan T ve D tipinde flip-flop'ların nasıl elde edileceği doğrudan ilgili flip-flop'ların doğruluk tablolarından görülebilir. İlk önce T tipini ele alalım. T flip-flop'da T girişi 0 ise çıkış değişmezken T=1 ise çıkış bir önceki çıkış değerinin tümleyenidir. JK flip-flop'un durum tablosuna (bkz. tablo-10.5.) bakıldığında, bu durum, J ve K girişlerine aynı değerler uygulandığında elde edilmektedir. Yani JK flip-flop'unda JK girişleri 00 olduğunda Q çıkışı değişmeden bir önceki durumunu korumaktadır; 11 olduğunda ise Q çıkışı bir önceki değerini tersi/tümleyeni olmaktadır. Dolayısıyla JK flip-flop'unun J ve K girişleri kısa devre edilip tek bir giriş haline getirilirse T flip-flop'u elde edilir.

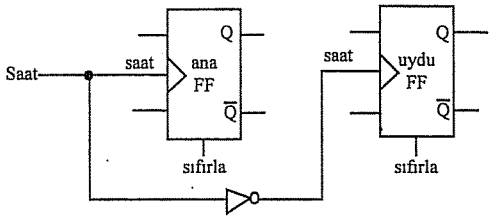
JK flip-flop'undan D tipi flip-flop elde etmek için, ilgili flip-flop'ların Tablo-10.5 ve 10.3 deki durum tablolarına bakıldığında K girişine, J girişine uygulanan değerin tümleyeni uygulandığında, sanki J girişindeki değerin flip-flop'a yüklendiği gibi bir durum oluşmaktadır. Dolayısıyla JK flip-flop'una aşağıdaki gibi bir tümleme kapısı eklenerek D flip-flop elde edilir.



Ana-Uydu Flip Flop

Ana-uydu (*master-slave*) flip-flop iki tane flip-flop'un bağlanmasıyla elde edilen ve tetikleme periyodu içerisinde girişindeki yeni değeri alırken çıkışındaki bir önceki değeri bir süre tutan flip-flop düzeneğidir. Şöyle ki, bir flip-flop'un çıkışları, girişlerindeki değerlere bağlı olarak, tetikleme anında, yeni durumunu alır. Ana-uydu flip-flop'larda ise, iki tane flip-flop'dan oluştuğu için, girişlere bağlı yeni çıkış değerleri tetikleme anında çıkışlara hemen yansıtılmaz; bir süre gecikmeyle yansıtılır.

Bu gecikme kara dalga şeklindeki bir saat işareti için ½ periyot'dur. Şekil-10.16 da tipik bir ana-uydu flip-flop'un mimarisi görülmektedir; saat işareti ana olarak adlandırılan birinci flip-flop'a doğrudan uygulanırken uydu (*slave*) olarak adlandırılan ikinci flip-flop'a tümlenerek uygulanmaktadır.



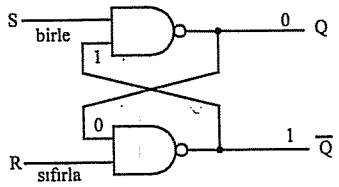
Şekil-10.16. Ana-uydu flip-flop mimarisi.

Flip-flop'ların yükselen kenarda tetiklendiği varsayılırsa (Şekil-10.3 de de öyle çizilmiştir), ana flip-flop, saatin yükselen kenarında girişlerdeki değerlere karşılık gelen yeni durumu içerisinde tutar; bu süreçte uydu flip-flop durumunu değiştirmez. Çünkü, saat işareti uyduya tümlenerek verilmiştir. Dolayısıyla uydu flip-flop, saatin düşen kenarına kadar çıkışlarını değiştirmez. Saatin düşen kenarında, tümleme kapısı çıkışı yükselen kenar anlamında çıkış ürettiği için ana flip-flop'un çıkışları uydu flip-flop'a aktarılır. Bu aralıkta, yani saatin yükselen kenarı ile düşen kenarı arasında geçen süre kadar ana-uydu flip-flop'un çıkışları değişmez.

Uygulamada, zamanlama davranışı açısından, ana-uydu flip-flop gibi davranış gösteren saklama birimlerine ihtiyaç duyulmaktadır. Böylece bir saat çevriminde iki iş yapılabilmekte ve lojik devre tasarımında çıkan zamanlama sorunlarının üstesinden gelinebilinmektedir. Örneğin, lojik tasarımcı, zaman zaman, içeriği değişse de bir önceki çıkışlarını bir süre tutan birimlere ihtiyaç duyarlar.

10.3. Tutucular (Latches)

Tutucular (*latches*), ardışıl lojik devre tasarımında en temel saklama biridir; flip-flop'lar da tutuculara birkaç tane kapı eklenerek yapılmaktadır. Bir tutucu 1 bitlik veriyi kendi üzerinde tutar; birden çok bitlik veriyi tutmak için yanyana birden çok tutucu koymak gerekir. Tutucular, kombinezonsal devre tasarımının yapı taşları olan VE, VEYA, TVE, TVEYA kapılarıyla kolayca gerçekleştirilebilir. Örneğin, aşağıdaki şekilde (Şekil-10.17) bir tutucunun TVE kapısıyla gerçekleştirilmiş yalın iç mimarisi görülmektedir; bu SR tutucu olarak adlandırılır. Çünkü biri *S* (set, birle) diğeri *R* (reset, sıfırla) olan 2 tane girişi vardır. Çıkışları da 2 tanedir ve bunlar flip-flop'larda olduğu gibi birbirlerinin tümleyenidir.



$$Q=\overline{S\cdot\overline{Q}} \qquad \overline{Q}=\overline{R\cdot Q} \quad \Rightarrow \quad Q=\overline{S\cdot\overline{R\cdot\overline{Q}}}$$

Şekil-10.17. T-VE kapısıyla SR tutucu mimarisi.

TVE kapısıyla gerçekleştirilen bir tutucuda *Q* çıkışı, şekilden görüleceği üzere, *S* girişiyle $\overline{Q}$ çıkışının TVE'lenmesiyle üretilmektedir; $\overline{Q}$ çıkışı da *R* girişiyle *Q* çıkışının TVE'lenmesiyle üretilir. Sonuç olarak, herhangi bir andaki *Q* çıkışı hem bir önceki *Q* çıkışına hem o andaki *S* ve *R* girişlerine bağlıdır. Örneğin, herhangi bir anda *Q(t)=1* olsun, bu durumda bir sonraki çıkış olan *Q(t+1)*, hem *S* hem de *R* girişlerine bağlıdır. Eğer, *Q(t)=1* ise,

SR=00 ise, Q(t+1)= ⊠ istenmeyen durum;
SR=01 ise, Q(t+1)= 0;
SR=10 ise, Q(t+1)= 1
SR=11 ise, Q(t+1)= 1= Q(t) ;

olur; Q(t)=0 ise,

SR=00 ise, Q(t+1)= ⊠ istenmeyen durum;
SR=01 ise, Q(t+1)= 0;
SR=10 ise, Q(t+1)= 1
SR=11 ise, Q(t+1)= 0= Q(t) ;

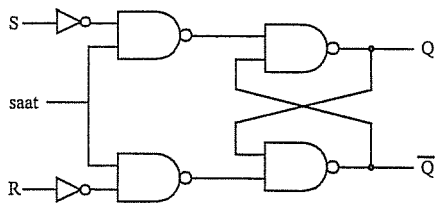
olur. *S=1, R=1* ise, *Q(t+1)= Q(t)* olur; yani, çıkışlardaki değer korunur. Görüldüğü gibi tutucunun *(t+1)* anındaki çıkışları *t* anındaki çıkışlara da bağlıdır. Bu özellik ardışıl devre tasarımında kolaylık ve esneklik sağlar.

Şekil-10.17 de verilen SR tutucunun durum tablosu Tablo-10.6 daki gibi olur:

Tablo-10.7. SR tutucunun durum tablosu ve geçişler.

S	R	Q(t)	Q(t+1)	Açıklama	Q(t+1)
0	0	0	-	istenmeyen durum	-
0	1	0	0	çıkış sıfırlanır	0
1	0	0	1	çıkış birlenir	1
1	1	0, 1	0, 1	değişiklik olmaz	Q(t)

Şekil-10.17 de verilen SR tutucu asenkron çalışacak mimariye sahiptir; durum değiştirmesi doğrudan girişlerin değişmesine bağlı olup dışarıdan bir saat işareti uygulanmasına gerek duyulmamaktadır. Benzer bir tutucunun senkron hali aşağıda, Şekil-10.18 deki gibi verilebilir:



Şekil-10.18. Senkron SR tutucu mimarisi.

Yukarıda senkronlanmış SR tutucunun saat girişi düzey tetiklemelidir; saat girişi 0 olursa, çıkışlar değişmeden kalırken 1 olursa durum tablosundaki geçişler geçerli olur. Q ve $\overline{Q}$ çıkışlarına ait bağıntılar şöyle hesaplanabilir:

$$Q = \overline{\overline{(\overline{S} \cdot saat) \cdot \overline{Q}}}$$

$$\overline{Q} = \overline{\overline{(\overline{R} \cdot saat) \cdot Q}} \Rightarrow Q = \overline{\overline{(\overline{S} \cdot saat)} \cdot \overline{(\overline{R} \cdot saat) \cdot Q}}$$

Görüldüğü gibi Q çıkışı saat işaretinin düzeyine bağlı duruma geldi. Eğer, *saat=0* ise, ki pasif durumdur, Q çıkışı eski halini korur; *saat=1* ise, ki etkin durumdur, çıkışlar yukarıdaki bağıntılara dayanılarak elde edilir.

Eğer, SR girişlerine birer tümleme kapısı bağlanmasa idi, ne olurdu? Bu durumda, *saat=0* olursa, girişlerdeki TVE kapılarının her ikisi de lojik 0 çıkışı üretirdi; bu da, SR tutucu ve flip-flop'ları için istenmeyen durumdur.

10.4. Ardışıl Devre Analiz Yöntemi

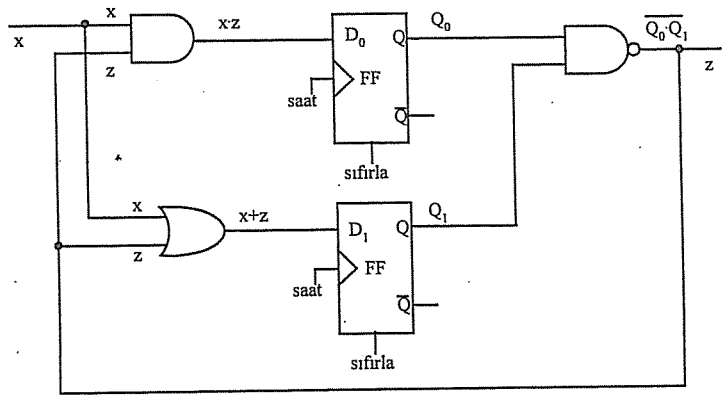
Ardışıl devre analizi, tasarlanmış olan bir devrenin ne iş yaptığı/fonksiyonunu çıkarmak için yapılan inceleme sürecidir. Bu amaçla, devrenin çizilmiş halinden/lojik şemasından geçiş fonksiyonları, durum tablosu ve durum diyagramı elde edilir; ve, devrenin genel davranışı ortaya çıkarılır. Bu ayrıtta, senkron ardışıl devrelerin analiz yöntemi ve izlenmesi gereken adımlar verilmiştir. Analiz sürecinde izlenmesi gereken adımlar Tablo-10.8 de verildiği gibi özetlenebilir.

Tablo-10.8. Ardışıl devre analiz yöntemi adımları.

Adım	Yapılması Gerekenler
1	FF (veya durum saklama birimlerinin) girişlerine ait ve devrenin çıkışlarına ilişkin Boole fonksiyonları devreye bakılarak yazılır.
2	FF'un (veya durum saklama birimlerinin) geçiş fonksiyonları kullanılarak bir sonraki durum fonksiyonları elde edilir.
3	Durum tablosu oluşturulur.
4	Durum diyagramı çizilir.

Örnek-10.6.

Şekil-10.19 da lojik çizimi verilen ardışıl devrenin analizini Tablo-10.8 uyarınca adım adım yaparak durum tablosunu elde ediniz ve durum diyagramını çiziniz. Devrenin geçiş yapmadığı bir durum var mıdır? Durum diyagramını yorumlayınız.



Şekil-10.19. Örnek-10.6 nın ardışıl devresi.

Analizi yapılması istenen devre bir girişli bir çıkışlıdır ve durum saklanması için D türü flip-flop'lar kullanılmıştır; 2 tane flip-flop olduğuna göre 4 değişik durum vardır. Tablo-10.8 uyarınca ilk yapılması gereken adım, flip-flop'ların girişlerine ait D_1, D_0 ve çıkışa ait z lojik ifadelerinin belirlenmesidir. Bunlar da doğrudan verilen devreye bakılarak yazılır:

$$D_0 = x \cdot z = x \cdot \overline{(Q_1 \cdot Q_0)}$$
$$D_1 = x + z = x + \overline{(Q_1 \cdot Q_0)}$$
$$z = \overline{(Q_1 \cdot Q_0)}$$

Yukarıda verilen D_1, D_0 *ve* z ifadeleri o andaki giriş, çıkış ve durum değişkenlerine bağlıdır; burada Q_1 ve Q_0 durum değişkenlerine karşılık düşerken; x, giriş ve z çıkış değişkenleridir. Bu aşamadan sonra ikinci adıma geçilir; ikinci adım üçüncü adımla içice yapılabilir. Yani bir sonraki durumları elde etmek ve durum tablosu oluşturmak birarada gerçekleştirilebilir. İki tane flip-flop olduğuna göre 2^2'den 4 değişik durum vardır. Bu dört değişik durum d_3, d_2, d_1, d_0 olarak isimlendirilirse, durum tablosu boş iken aşağıdaki gibidir:

simgesel durum adları	şimdiki durumlar		flip-flop girişleri		bir sonraki durumlar		Çıkış
			x=0	x=1	x=0	x=1	
	Q_1	Q_0	D_1 D_0	D_1 D_0	Q_1 Q_0	Q_1 Q_0	z
d_0	0	0					
d_1	0	1					
d_2	1	0					
d_3	1	1					

Yukarıdaki boş durum tablosu D_1, D_0 *ve* z ifadelerine ve D türü flip-flop'un geçiş fonksiyonuna dayanılarak doldurulmalıdır. Örneğin $x=0$ iken D_1'in değerleri $D_1 = x + \overline{(Q_1 \cdot Q_0)}$ bağıntısı uyarınca sırasıyla 1, 1, 1 ve 0 olarak bulunur; $x=1$ için tüm değerleri 1 olur. Benzer işlem D_0 için de yapılıp sonuçlar yerlerine yerleştirilirse tablo aşağıdaki gibi olur:

simgesel durum adları	şimdiki durumlar		flip-flop girişleri		bir sonraki durumlar		Çıkış
			x=0	x=1	x=0	x=1	
	Q_1	Q_0	D_1 D_0	D_1 D_0	Q_1 Q_0	Q_1 Q_0	z
d_0	0	0	1 0	1 1			
d_1	0	1	1 0	1 1			
d_2	1	0	1 0	1 1			
d_3	1	1	0 0	1 0			

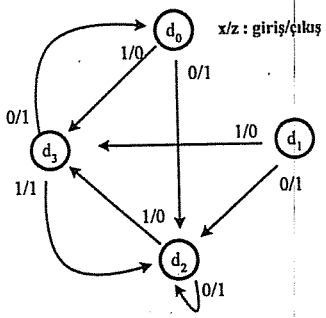
Bir sonraki durumlar flip-flop girişlerine ve devrede kullanılan flip-flop'un davranışına (geçiş fonksiyonuna) bağlıdır. Şekil-10.19 da verilen devrede D türü flip-flop kullanılmıştır; bu tür flip-flop'larda bir sonraki çıkışlar o andaki girişlerin aynısıdır. D türü flip-flop'lar geçiş fonksiyonu en açık görülen flip-flop türüdür; dolayısıyla

bir sonraki durum bilgisini elde etmek oldukça yalındır. JK türü flip-flop kullanılsaydı bir sonraki durumlar daha dikkatli davranılarak çıkarılmaydı! Bu durumda bir sonraki durumlar doğrudan flip-flop'ların girişleriyle aynıdır. Çıkış değerleri de, $z = \overline{(Q_1 \cdot Q_0)}$ bağıntısı uyarınca belirlenip tabloya yerleştirilirse durum tablosu aşağıdaki gibi olur:

simgesel durum adları	şimdiki durumlar		flip-flop girişleri		bir sonraki durumlar		Çıkış
			x=0	x=1	x=0	x=1	
	Q_1	Q_0	D_1 D_0	D_1 D_0	Q_1 Q_0	Q_1 Q_0	z
d_0	0	0	1 0	1 1	1 0	1 1	1
d_1	0	1	1 0	1 1	1 0	1 1	1
d_2	1	0	1 0	1 1	1 0	1 1	1
d_3	1	1	0 0	1 0	0 0	1 0	0

Verilen devrenin durum diyagramı yukarıdaki tablodan çizebilir; ancak durum tablosu üzerindeki geçişleri simgesel durum adlarıyla sadeleştirmek durum tablosu çizimini kolaylaştırır ve olabilecek hatayı ortadan kaldırır. Bu nedenle yukarıdaki durum tablosu aşağıdaki gibi sadeleştirilebilir. Durum diyagramı da Şekil-10.20 deki gibi çizilir.

şimdiki durum	bir sonraki durumlar		Çıkış
	x=0	x=1	z
d_0	d_2	d_3	1
d_1	d_2	d_3	1
d_2	d_2	d_3	1
d_3	d_0	d_2	0

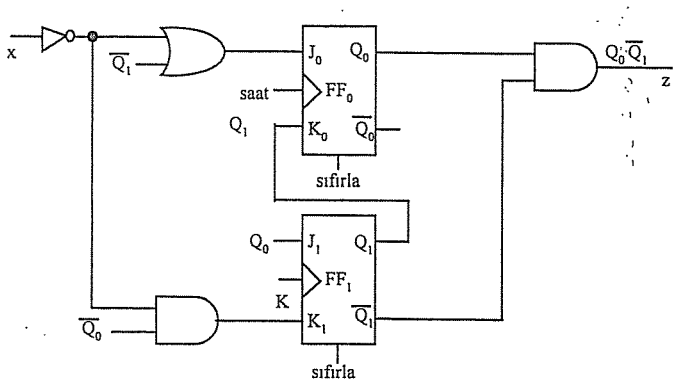


Şekil-10.20. Örnek-10.6 nın durum diyagramı.

Şekil-10.19 da verilen devrenin durum diyagramı Şekil-10.20 de görüldüğü gibi elde edilmiştir; dikkatli bakılırsa d_1 durumuna geçiş yoktur. Devrenin davranışı genel $x=1$ iken olarak d_3 durumuna yönelmektedir.

Örnek-10.7.

Şekil-10.21 de lojik çizimi verilen senkron ardışıl devrenin analizini Tablo-10.8 uyarınca yaparak durum tablosunu elde ediniz ve durum diyagramını çiziniz. Devrenin davranışında kilitlendiği bir durum var mıdır?



Şekil-10.21. Örnek-10.7 nın ardışıl devresi.

Devre bir girişli bir çıkışlıdır; durum bilgisinin saklanması için JK flip-flop'ları kullanılmıştır. İki tane flip-flop olduğu için 2 bitlik durum bilgisi saklanmaktadır; dolayısıyla 2^2'den 4 tane değişik durum vardır. Devre herhangi bir anda bu 4 değişik durumdan birinde bulunur; durumlar d_3, d_2, d_1, d_0 olarak adlandırılabilir. Eğer sıfırla girişi etkin (sıfırla=1) yapılırsa ilk durum olan d_0'a geçer. Devrenin analizi için herşeyden önce flip-flop'ların girişleri olan J_0, K_0, J_1 ve K_1 girişlerine ait lojik ifadeler yazılır; yukarıdaki çizime göre bu girişlerin ve çıkışın lojik ifadeleri şöyledir:

$$J_0 = \overline{x} + \overline{Q_1} \qquad K_0 = Q_1$$

$$J_1 = Q_0 \qquad K_1 = \overline{x} \cdot \overline{Q_0} \qquad z = Q_0 \cdot \overline{Q_1}$$

Bu bağıntılara ve d_3, d_2, d_1, d_0 durumlarının sırasıyla 00, 01, 10, 11 olarak kodlanmasıyla flip-flop'ların giriş değerleri ve devrenin çıkış değeri aşağıdaki gibi elde edilir:

simgesel durum adları	şimdiki durumlar		flip-flop girişleri x=0		flip-flop girişleri x=1		bir sonraki durumlar x=0	bir sonraki durumlar x=1	Çıkış
	Q_1	Q_0	$J_1 K_1$	$J_0 K_0$	$J_1 K_1$	$J_0 K_0$	$Q_1 Q_0$	$Q_1 Q_0$	z
d_0	0	0	0 1	1 0	0 0	1 0			0
d_1	0	1	1 0	1 0	1 0	1 0			1
d_2	1	0	0 1	1 1	0 0	0 1			0
d_3	1	1	1 0	1 1	1 0	0 1			0

Bu aşamada yapılması gereken, bir sonraki durumların elde edilmesidir; bu amaçla flip-flop'ların JK girişlerinin aldığı değerler (bir önceki tabloda elde edilmişti) ve JK flip-flop'unun geçiş fonksiyonundan (bkz. Tablo-10.4 JK flip-flop'un doğruluk tablosu) yararlanılır.

JK flip-flop'un geçiş fonksiyonu
(Tablo-10.4. JK flip flop'u doğruluk tablosu)

giriş J K	giriş saat	çıkışlar Q Q'
0 0	↑	Q Q'
0 1	↑	0 1
1 0	↑	1 0
1 1	↑	Q' Q

$$Q^+ = J \cdot Q' + K' \cdot Q$$

simgesel durum adları	şimdiki durumlar		flip-flop girişleri x=0		flip-flop girişleri x=1		bir sonraki durumlar x=0	bir sonraki durumlar x=1	Çıkış
	Q_1	Q_0	$J_1 K_1$	$J_0 K_0$	$J_1 K_1$	$J_0 K_0$	$Q_1 Q_0$	$Q_1 Q_0$	z
d_0	0	0	0 1	1 0	0 0	1 0	**0 1**	**0 1**	0
d_1	0	1	1 0	1 0	1 0	1 0	**1 1**	**1 1**	1
d_2	1	0	0 1	1 1	0 0	0 1	**0 1**	**1 0**	0
d_3	1	1	1 0	1 1	1 0	0 1	**1 0**	**1 0**	0

Yukarıdaki tabloda koyu yazılan değerler bir sonraki durumlara ait değerleri göstermektedir. Bu değerler hemen solunda bulunan JK değerlerinden elde edilmiştir. Bunların nasıl elde edildiğini göstermek amacıyla x=1 ve J_0, K_0 girişleri ele alınarak şöyle açıklanabilir:

x=1 ve J_0, K_0 girişleriyle ⇨ x=1 için Q_0 durum çıkışı belirlenir.

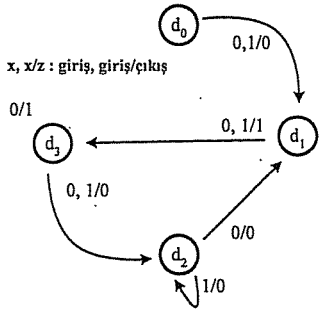
x=1 için J_0 ve K_0 girişlerinin aldığı değerler sırasıyla 10, 10, 01, 01 şeklindedir; bu giriş kombinezonları bir JK flip-flop'un uçlarına uygulandığında bir sonraki durum, yani Q_0 çıkışı sırasıyla 1, 1, 0, 0 olur. Çünkü JK girişleri 10 uygulanırsa flip-flop'un çıkışı 1, 01 uygulanırsa 0 olmaktadır.

x=1 için	J_0	K_0	Q_0
	1	0	1
	1	0	1
	0	1	0
	0	1	0

Bkz. Tablo-10.4.

Bu aşamada yapılması gereken elde edilen durum tablosunun simgesel durum adları kullanılarak sadeleştirilmesi ve durum diyagramının elde edilmesidir:

şimdiki durum	bir sonraki durumlar		Çıkış
	x=0	x=1	z
d_0	d_1	d_1	0
d_1	d_3	d_3	1
d_2	d_1	d_2	0
d_3	d_2	d_2	0



Şekil-10.22. Örnek-10.7 nın durum diyagramı.

Şekil-10.21 de verilen lojik devrenin analizi sonucu elde edilen durum diyagramı Şekil-10.22 de gösterildiği gibi elde edilmiştir. Devrenin kilitlenen durumu yoktur. Ancak, d_0 durumuna başlangıç durumu dışında gelmediği görülmektedir.

10.5. Ardışıl Devre Tasarım Yöntemi

Ardışıl devre tasarımı (diğer bir deyişle sentezi), durum diyagramı veya fonksiyonel ifadesi verilen ve bir iş için gerekli lojik devrenin tasarlanması sürecidir. Bu süreçte devrenin açık şeması kapı, flip-flop, saklayıcı, sayıcı gibi standart lojik birimler düzeyinde ortaya çıkarılır ve aralarındaki bağlantılar belirlenir.

Ardışıl devre tasarımında ilk elde edilmesi gereken unsur kaç tane durum olacağıdır. Durum sayısı doğrudan kaç tane flip-flop olacağını veya daha genel bir ifadeyle durum bilgisinin kaç bit olacağını belirler. Durum sayısı k olan herhangi bir ardışıl lojik devrede durum bilgisinin kaç bit olacağı $d= log_2k$ bağıntısı uyarınca hesaplanır; sonuç kesirli sayı çıkarsa bir üst tamsayı alınır. Örneğin 2, 3 veya 4 durum varsa 2 bitlik durum bilgisi; 5, 6, 7 veya 8 durum bilgisi varsa 3 bitlik durum bilgisine ihtiyaç vardır; 33 durum bilgisi varsa d= $\log_2 33$ bağıntısından 6 bitlik durum bilgisine ihtiyaç olduğu bulunur. Eğer 32 tane durum olsaydı d= $\log_2 32$=5 sonucu uyarınca 5 bitlik durum bilgisine ihtiyaç olacaktı!

Senkron bir ardışıl devre tasarımında izlenmesi gereken adımlar Tablo-10.9 da listelenmiştir; yapılması gereken ilk adım devrenin fonksiyonunu açıkça belirlemektir. Daha sonra durum sayısının belirlenmesi, flip-flop türünün seçilmesi gelir. Durum sayısı belirlenirken, bazen bir tane durum azaltılması devrenin karmaşıklığını azaltır. Çünkü bir tane durum azaltılması demek durum bilgisinin saklandığı flip-flop'lardan bir tane daha az kullanılması anlamına gelir; bir tane daha az flip-flop kullanılınca da onunla ilgili kombinezonsal devre de sadeleşebilir. Aşağıdaki tabloda ardışıl devre tasarımında yapılması gerekenler adım adım verilmiştir. Herşeyden önce devrenin işlevi açıkça tanımlanmalıdır; tanımlama ya sözle, ya durum diyagramı verilerek ya da algoritmik olarak tanımlanabilir.

Tablo-10.9. Ardışıl devre tasarım yöntemi adımları.

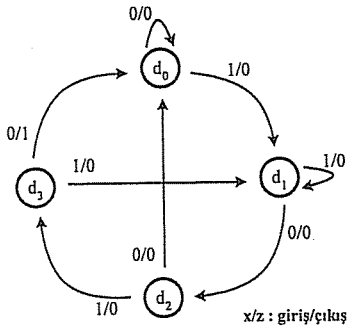
Adım	Yapılması Gerekenler
1	Devrenin ne iş yapacağı/fonksiyonu açıkça tanımlanır.
2	Durum tablosunun elde edilmesi: verilen tanımdan durum tablosu çıkarılamıyorsa, önce durum diyagramı elde edilir; daha sonra durum tablosu oluşturulur.
3	Durum sayısından durum değişkeni sayısı belirlenir ve durumların saklanması için kullanılacak FF, saklayıcı, bellek gibi birimler seçilir. Durum kodlaması yapılır.
4	Flip-flop'ların uyarma tablolarında yararlanarak flip-flop'ların girişlerine ait uyarma bilgiler elde edilir ve durum tablosuna eklenir. Yani, çıkış fonksiyonları Z_i ve durum saklama birimleri girişlerine ait Boole fonksiyonları tablo biçiminde elde edilerek belirlenir. Gerekirse indirgeme işlemi yapılarak fonksiyonlar sadeleştirilir; bu amaçla *Karnaugh* diyagramı, *Quine-Mc Cluskey* yöntemi kullanılabilir.

Ardışıl devre tasarımında flip-flop türünün seçilmesi önemli unsurdur; JK flip-flop kullanılması devre davranışına daha uygun iken D türü flip-flop kullanılması kombinezonsal devre karmaşıklığını arttırır veya tersi durumda D türü kullanılması daha uygun iken JK veya diğer türde flip-flop kullanılması daha fazla kapı vs. gibi elemanlar kullanılmasını gerektirir.

Örnek-10.8.

Girişi x, çıkışı ise z ile ifade edilen bir ardışıl devrenin girişine 1010 bit dizisi geldiğinde çıkışını lojik 1, aksi durumda lojik 0 yapan ardışıl devreyi D-tipi flip-flop'lar kullanarak tasarlayınız.

Tasarımında ilk yapılması gereken devrenin durum diyagramı ve durum tablosunu elde etmektir; buradan, durum geçişleri ve bu geçişlere göre devre çıkışı z'nin değişimini elde edilmiş olunur (Bkz. Şekil-10.23). Durum diyagramı, 4 veya 5 durumla ifade edilebilir. Eğer başlangıç durumuyla sonuç durumu aynı olarak kabul edilirse 4, farklı olarak kabul edilirse 5 durum yeter (bu, Örnek-10.3 de açıklanmıştır). Şekil-10.23 de 4 durum için gerekli durum diyagramı görülmektedir.



Şekil-10.23. Örnek-10.8 in durum diyagramı.

Devre, başlangıçta d_0 durumundadır; girişe lojik 0 geldiği sürece de orada kalır. Ancak, girişe lojik 1 gelirse bir sonraki durum olan d_1'e geçilir; burada iken, lojik 0 gelirse 10 bit dizisi gelmiş olacağından bir sonraki durum olan d_2'ye, lojik 1 gelirse olduğu yerde kalır. 10 bit dizisi alındığında, artık beklenen önce 1 sonra 0 değerleridir; d_2 durumunda iken lojik 1 gelirse d_3'e, lojik 0 gelirse başlangıç durumu olan d_0'a dönülür. Çünkü 10 gelmişken üçüncü olarak lojik 0 gelirse 100 gelmiş olur. 100 üçlü bit dizisi beklenen 1010 bit dizisiyle hiç uyuşmadığı başlangıç durumuna dönülür. Son olarak, 101 bit dizisi gelmiş iken (bu sırada d_3 durumunda bulunulur), lojik 0 gelirse beklenen bit dizisi tamamlanacağı için çıkış lojik 1 yapılır ve aynı zamanda yeni 1010 bit dizileri için başlangıç durumuna dönülür; lojik 1 gelirse, 1011 gelmiş olacağından d_1'e dönülür. Çünkü istenen bit dizisinin ilk üç biti doğru sırada gelmiştir; ancak dördüncü bit için 0 yerine 1 gelmiştir. Yani, 1010 beklenirken 1011 gelmiş oldu; burada 1 bitlik uyuşma vardır. Dolayısıyla durum diyagramında başlangıca göre 1 bitlik ilerisi d_1 durumuna geçilir; 1 bitlik uyuşma şöyle bulunur: gelen bit dizisi içerisindeki bitlerle beklenen bit dizisi içerisindeki

bitler ters sırada karşılaştırılır. Yani son gelen lojik değer bit dizisi içerisindeki ilk bit ile, sondan bir önceki ikinci bit ile, sondan iki önceki üçüncü bit ile... Örneğimize göre, son gelen lojik değer beklenen bit dizisinin ilk biti ile aynıdır; diğerleri farklıdır.

Şekil-10.23 deki durum diyagramına göre aşağıdaki durum tablosu elde edilir:

şimdiki durum	bir sonraki durumlar x=0	x=1	Çıkış x=0	x=1
d_0	d_0	d_1	0	0
d_1	d_2	d_1	0	0
d_2	d_0	d_3	0	0
d_3	d_0	d_1	1	0

Geçişleri gösteren durum tablosu elde edildikten flip-flop'lara ait uyarma değerleri tabloya yerleştirilir. D flip-flop'ları kullanılarak tasarım yapılacağı için D flip-flop'unun durum geçiş tablosunda dayanılarak aşağıdaki durum tablosu oluşturulur:

simgesel durum adları d_i	şimdiki durum $Q_1\ Q_0$	giriş x	saat	bir sonraki durum $Q_1^+\ Q_0^+$	$FF1_1$ D_1	FF_0 D_0	çıkış z
d_0	0 0	0	↑	0 0	0	0	0
d_0	0 0	1	↑	0 1	0	1	0
d_1	0 1	0	↑	1 0	1	0	0
d_1	0 1	1	↑	0 1	0	1	0
d_2	1 0	0	↑	0 0	0	0	0
d_2	1 0	1	↑	1 1	1	1	0
d_3	1 1	0	↑	0 0	0	0	1
d_3	1 1	1	↑	0 1	0	1	0

D \ QQ^+	00	01	11	10
0	0	1	1	0

Flip-flop'ların girişlerine ait uyarma bilgileri ve çıkış bilgileri, yukarıdaki tabloda görüldüğü gibi elde edildikten sonra, D_1, D_0 ve z'nin lojik ifadesi aşağıdaki gibi bulunur ve oradan lojik şema çizilir: (Q: şimdiki, Q^+: bir sonraki değer için kullanılmıştır)

x \ Q_1Q_0	00	01	11	10
0	0	1	0	0
1	0	0	0	1

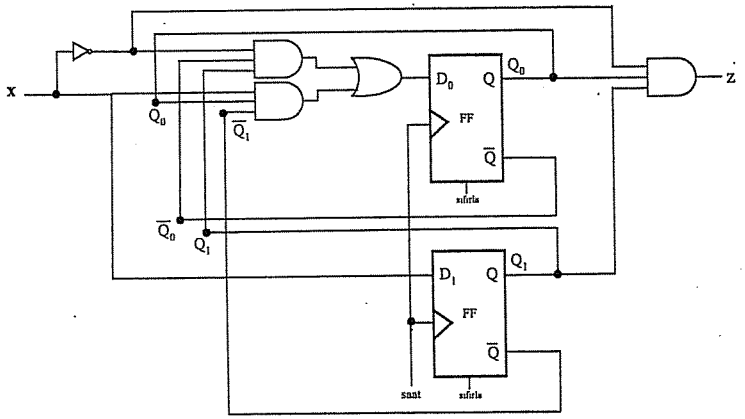
$D_1 = \overline{x}\,\overline{Q}_0 Q_1 + x Q_0 \overline{Q}_1$

x \ Q_1Q_0	00	01	11	10
0	0	0	0	0
1	1	1	1	1

$D_0 = x$

x \ Q_1Q_0	00	01	11	10
0	0	0	1	0
1	0	0	0	0

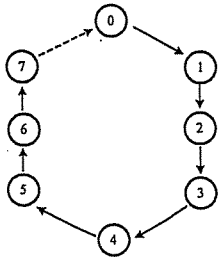
$z = \overline{x}\, Q_1 Q_0$



Örnek-10.9.

Örnek-10.2 de durum diyagramı verilen bir sayıcının JK flip-flop'ları kullanılarak tasarımı yapılması istenmektedir; sayıcı 0'dan başlayıp 7'ye kadar birer artımla sayacak ve senkron olacaktır.

Herşeyden devrenin kaç bitlik durum bilgisi olduğu belirlenmeli ve ardından durum tablosu çıkarılmalıdır. Durum diyagramı Örnek-10.2 de çıkarılmıştı; bkz. Şekil-10.6. Durum sayısı k=8 olduğundan $d=\log_2 8$ bağıntısı uyarınca 3 bitlik durum bilgisi vardır; dolayısıyla 3 tane flip-flop kullanılmalıdır. Durumlar Q_2, Q_1 ve Q_0 olarak adlandırılırsa durum tablosunun başlangıçtaki durumu aşağıdaki gibi olur:



simgesel durum adları	şimdiki durum	kontrol girişi	saat	bir sonraki durum	FF_2	FF_1	FF_0
d_i	$Q_2\ Q_1\ Q_0$			$Q_2^+\ Q_1^+\ Q_0^+$	$J_2\ K_2$	$J_1\ K_1$	$J_0\ K_0$
d_0	0 0 0	-	♪	0 0 1			
d_1	0 0 1	-	♪	0 1 0			
d_2	0 1 0	-	♪	0 1 1			
d_3	0 1 1	-	♪	1 0 0			
d_4	1 0 0	-	♪	1 0 1			
d_5	1 0 1	-	♪	1 1 0			
d_6	1 1 0	-	♪	1 1 1			
d_7	1 1 1	-	♪	0 0 0			

İstenen sayıcı devrenin 3 bitlik çıkışı vardır; giriş işareti olarak yalnızca saat işareti olup onun dışında herhangi bir kontrol girişi yoktur: ♪: saat işaretinin senkronlama anını; - : yok veya değerlendirme dışı anlamında kullanılır.

Bu aşamada yapılması gereken durum tablosundaki şimdiki durumlara ve bir sonraki durumlara bakılarak flip-flop'ların J ve K girişlerine ait lojik ifadelerin belirlenmesidir. Bunun için şimdiki durumdan bir sonraki duruma geçmesi için JK değeri ne olmalıdır soruları yanıtlanır. Örneğin d_0 durumu için Q_2'nin şimdiki değeri 0, bir sonraki değeri de 0 veya d_7 durumu için Q_2'nin şimdiki değeri 1, bir sonraki değeri 0, bu geçişlerin olması için ilgili flip-flop'ların girişleri ne olmalıdır. Bu aşamada yapılması gereken bu geçişleri sağlayacak JK ifadelerini tespit edilmesidir.

JK flip-flop şimdiki durumdan bir sonraki duruma geçiş koşulları	$Q \backslash Q_{(t+1)}$	00	01	11	10
		$0\,\phi$	$1\,\phi$	$\phi\,0$	$\phi\,1$

Aşağıda JK flip-flop'unun geçiş tablosu farklı bir açıdan verilmiştir; *Karnaugh* diyagramına benzetilmiştir. Böyle bir diyagrama bakılarak durum tablosundaki JK geçiş değerleri daha kolay görülebilir.

Yukarıdaki diyagramda şimdiki durum Q, bir sonraki durumda Q_{t+1} ile gösterilmiştir. O zaman, şimdiki durumu 0, bir sonraki durumu 1 olan geçişler için JK girişleri 1ϕ (J=1, K=ϕ), şimdiki durumu 0, bir sonraki durumu 0 olan geçişler için de JK girişleri 0ϕ (J=0, K=ϕ) olmalıdır. Burada, ϕ, keyfi değer olup hem 1 hem de 0 değerini alabilir demektir; yani, önemi yoktur anlamına gelir.

Belirtilen koşullar altında sayıcının durum tablosundaki geçişler aşağıdaki gibi olur:

simgesel durum adları	simdiki durum	kontrol girişi	saat	bir sonraki durum	FF_2	FF_1	FF_0
d_i	$Q_2\ Q_1\ Q_0$			$Q_2^+\ Q_1^+\ Q_0^+$	$J_2\ K_2$	$J_1\ K_1$	$J_0\ K_0$
d_0	0 0 0	-	↑	0 0 1	0 Φ	0 Φ	1 Φ
d_1	0 0 1	-	↑	0 1 0	0 Φ	1 Φ	Φ 1
d_2	0 1 0	-	↑	0 1 1	0 Φ	Φ 0	1 Φ
d_3	0 1 1	-	↑	1 0 0	1 Φ	Φ 1	Φ 1
d_4	1 0 0	-	↑	1 0 1	Φ 0	0 Φ	1 Φ
d_5	1 0 1	-	↑	1 1 0	Φ 0	1 Φ	Φ 1
d_6	1 1 0	-	↑	1 1 1	Φ 0	Φ 0	1 Φ
d_7	1 1 1	-	↑	0 0 0	Φ 1	Φ 1	Φ 1

(Φ: keyfi; ne olursa olsun)

Durum diyagramında bir sütuna ait ϕ keyfi değerleri arttıkça daha yalın lojik ifade elde etme olasılığı artar. Böylece eleman karmaşıklığı az olan tasarım yapılabilir. Çünkü ϕ yerine, tasarımcı, istediği değeri yerleştirebilir. Bu durum bu örnekte ilerleyen paragrafta verilen *Karnaugh* diyagramında da görülebilir.

Devreye ait durum diyagramı oluşturulduktan sonra herbir flip-flop'un J ve K girişlerine ait lojik ifadeler bulunur; bu amaçla *Karnaugh* diyagramı yöntemi kullanılabilir. Flip-flop'ların herbir J ve K girişleri için tek tek *Karnaugh* diyagramı oluşturularak indirgenmiş lojik ifadeler bulunur. Flip-flop'lara ait bu lojik ifadeler bir sonraki durum bilgileri üretmek için kullanılır. Aşağıda sırasıyla J_2, K_2 ; J_1, K_1 ve J_0, K_0 girişlerine ait lojik ifadelerin bulunması görülmektedir. Herbir *Karnaugh* diyagramındaki değerler durum tablosundaki ilgili sütundan alınmıştır.

J_2 için

Q_0 \ Q_2Q_1	00	01	11	10
0	0	0	ϕ	ϕ
1	0	1	ϕ	ϕ

$J_2 = Q_1Q_0$

K_2 için

Q_0 \ Q_2Q_1	00	01	11	10
0	ϕ	ϕ	0	0
1	ϕ	ϕ	1	0

$K_2 = Q_1Q_0$

J_1 için

Q_0 \ Q_2Q_1	00	01	11	10
0	0	ϕ	ϕ	0
1	1	ϕ	ϕ	1

$J_1 = Q_0$

K_1 için

Q_0 \ Q_2Q_1	00	01	11	10
0	ϕ	0	0	ϕ
1	ϕ	1	1	ϕ

$K_1 = Q_0$

J_0 için

Q_0 \ Q_2Q_1	00	01	11	10
0	1	1	1	1
1	ϕ	ϕ	ϕ	ϕ

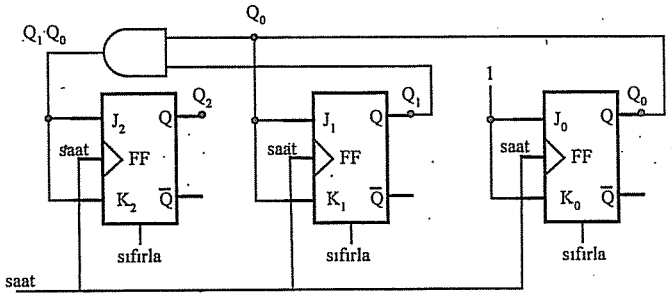
$J_0 = 1$

K_0 için

Q_0 \ Q_2Q_1	00	01	11	10
0	ϕ	ϕ	ϕ	ϕ
1	1	1	1	1

$K_0 = 1$

Böylece flip-flop'ların JK girişlerine ait lojik ifadeler bulunmuş oldu; bu ifadelere göre sayıcının lojik şeması aşağıdaki gibi (Şekil-10.24) olur:



Şekil-10.24. Örnek-10.9 un sonucu.

10.6. Özet

Ardışıl devrelerde herhangi bir andaki çıkış o andaki girişlerden ve daha önceki girişlerin belirlemiş olduğu durum bilgilerinden üretilir; yani, daha önceki giriş bilgilerinin de fonksiyonudur. Ardışıl devreler lojik tasarım konusunda önemli bir konuma sahiptir. Çünkü, uygulamada gereksinimlerin çoğu ardışıl davranış göstermektedir. Kombinezonsal devreler, genel olarak ardışıl devre tasarımında birer yan birimdir denilebilir. Ardışıl devrelerde sistemin davranışı durum tablosu veya durum diyagramıyla ifade edilir; bu şekilde ifade edilmiş bir ardışıl devre kolayca tasarlanabilir. Ardışıl devre analizi, kapı, flip-flop, saklayıcı, sayıcı gibi lojik elemanlar düzeyinde çizimi verilen devrenin davranışını gösterecek durum tablosu veya durum diyagramını elde etme sürecidir. Tasarım ise, tam tersi olarak durum tablosu ve diyagramından lojik elemanlar düzeyinden devreyi belirlemektir.

Ardışıl devreler çeşitli şekillerde sınıflanmaktadır: biri, çıkış ve geçiş işaretlerinin üretilmesine göre *Mealy* ve *Moore* makinalar; diğeri, saat işaretinin uygulanma şekline göre senkron ve asenkron devreler; bir diğeri de durum değiştirme şekline göre düzey ve kenar tetiklemeli olarak sınıflanmaktadır.

10.7. Sorular

1. İki adet JK flip-flop'u ve gerektiği kadar kapı kullanarak a ve b şıklarında istenen tümleyici devreleri tasarlayınız.
   a) bire tümleyici devresi
   b) ikiye tümleyici devresi
2. İki adet JK flip-flop'u ve gerektiği kadar kapı kullanarak 2 bitlik doğal ikili sayıcı tasarlayınız.
3. K adlı bir kontrol girişi olan ve K=0 iken ileri sayıcı, K=1 iken bire tümleyici olarak çalışan bir 2 bitlik ardışıl devre tasarlayınız. Tasarımda JK flip-flop'ları kullanılacaktır, gerektiği kadar kapı ve seçici devreleri de kullanılabilir.
4. Birden başlayıp ikişer artımla yediye kadar (1, 3, 5, 7, 1 şeklinde) sayan bir ikili sayıcı tasarlanmak istenmektedir.
   a) Devrenin kaç tane flip-flop'la gerçeklenebileceğini belirleyiniz ve durum tablosuyla durum diyagramını oluşturunuz.
   b) Sayıcıyı a şıkkında yaptıklarınıza dayanarak JK flip-flop'ları ve gerektiği kadar kapı kullanarak tasarlayınız ve şemasını çiziniz.
   c) Tasarladığınız sayıcı, başlangıçta, 0, 2, 4, 6 gibi sayma düzeninde bulunmayan gibi bir sayıya konumlanmışsa nasıl davranır. Sayıcının, başlangıçta, 1'e konumlanması için devreye nasıl bir ekleme yapılmalıdır.

5. Aşağıda verilen sayma düzeninde sayan bir sayıcı tasarlanacaktır.

   0 ⇨ 8 ⇨ C ⇨ E ⇨ 7 ⇨ 3 ⇨ 9 ⇨ 4 ⇨ 2 ⇨ 1 ⇨ 0

   a) Sayıcının kaç tane flip-flop'la tasarlanabileceğini belirleyiniz ve durum tablosuyla durum diyagramını oluşturunuz.
   b) *k* sayıcının dışarıdan kontrol edilmesi için bir giriş değişkeni olduğuna göre, *k=1* iken yukarıda verilen sayma düzeninde sayan, *k=0* olduğunda ise çıkışlar 0 (sıfır) olana kadar sayıp bekleme moduna geçen sayıcıyı T flip-flop'u kullanarak tasarlayıp şemasını çiziniz.
   c) Tasarlanan sayıcı herhangi bir nedenle istenmeyen durumdan birinde bulunursa, tetikleme işareti geldiğinde geçeceği yeni durumu belirleyiniz.
   d) Sayıcıyı aynı koşullarda D flip-flop'ları ve TVE kapıları kullanarak tasarlayınız.
   e) b ve d şıklarında yaptığınız tasarımı karşılaştırınız. Hangisi daha az maliyetlidir?
6. FF'lardan oluşan ve flip-flop'ların giriş bağıntıları aşağıdaki gibi olan sayıcının sayma düzeninin d_0, d_1, d_2, d_3 sırasına uygun olduğunu gösteriniz. Sayıcı istenmeyen durumlardan birine geçerse nasıl davranır?

   $T_0 = Q_0 Q_3 + Q_1 Q_2 Q_3 \qquad T_1 = Q_2 Q_3 \qquad T_2 = \overline{Q}_1 \cdot Q_3 \qquad T_3 = 1$
7. D türü flip-flop'larla tasarlanmış bir Mealy makinaya ilişkin flip-flop'ların giriş bağıntıları aşağıda verilmiştir: (x: giriş, Z: çıkış, Q_0 ve Q_1 durum değişkenleridir)

   $$D_0 = Q_0 \cdot \overline{Q}_1 \cdot \overline{x} + Q_1 \cdot x + \overline{Q}_0 \cdot x \qquad D_1 = Q_0 \cdot Q_1 \cdot \overline{x} + \overline{Q}_1 \cdot x$$
   $$Z = Q_0 \cdot Q_1 \cdot \overline{x} + Q_1 \cdot x$$

   Bu devrenin *Mealy* makinası ve *Moore* makinası modellerine göre geçiş/çıkış tablosunu belirleyiniz.
   a) Devreyi *Moore* makinası olarak JK flip-flop'ları kullanarak tasarlayınız.
   b) Tasarlanan devrenin artık durumları var mıdır? Açıklayınız.
8. Bir girişli bir çıkışlı bir devrenin girişine sırasıyla 0110 bit dizisi geldiğinde çıkışını 1 yapması istemektedir.
   a) Devreyi *Mealy* makinası olarak JK flip-flop'u kullanarak tasarlayınız. Tasarımda herşeyden önce durum diyagramı ve durum tablosunu çıkarınız.
   b) Devreyi *Moore* makinası olarak JK flip-flop'u kullanarak tasarlayınız. Tasarımda herşeyden önce durum diyagramı ve durum tablosunu çıkarınız.
9. Bir oylamaya katılan 4 kişi, kararlarını önlerinde bulunan butonlara basarak belirtiyorlar. Teklifi kabul eden kişi butona basmakta, aksi halde basmamaktadır: Teklifin kabul edilmesi için en az 2 kişinin kabul etmesi ve bir önceki teklifin kabul edilmemiş olması gerekmektedir. Başlangıçta bir önceki teklifin kabul edilmediği varsayılarak bu işi kotaracak devreyi tasarlayınız ve şemasını çiziniz.
10. Gerektiği kadar JK ikilisi ve kapılar kullanarak kendi öğrenci numaranız sırasında sayan bir sayıcı tasarlayınız. Öğrenci numaranızdaki alfanumerik karakterleri yok varsayınız.

11. A={a, b, c} kümesi veriliyor. A kümesinden elemanlar kullanılarak türetilen katarlardan içerisinde "yalnızca iki tane b'yi izleyen a" bulunan katarları kabul eden bir sonlu durum makinası tasarlayınız ve çiziniz. Örneğin ...bba... kabul edilecek, ancak ...bbba... gibi bir katar kabul edilmeyecektir.

12. JK flip-flop'ları kullanılarak 0, 1, 2...14, 15, 0, 1 şeklinde 4 bitlik bir sayıcı tasarlanmak istenmektedir. Sayıcının durum tablosunu oluşturunuz ve flip-flop'ların JK girişlerine ait lojik ifadeleri bulunuz. Sayıcının çıkışları sırasıyla Q_3 Q_2 Q_1 ve Q_0 olarak simgelendiğine göre (Q_3 en anlamlı; Q_1 en anlamsız hane) J_3 K_3, J_2 K_2, J_1 K_1 ve J_0 K_0 girişlerine ait lojik ifadelerin aşağıdaki gibi olduğunu gösteriniz.

$J_3 = Q_2 Q_1 Q_0$ $\quad K_3 = Q_2 Q_1 Q_0$

$J_2 = Q_1 Q_0$ $\quad K_2 = Q_1 Q_0$

$J_1 = Q_0$ $\quad K_1 = Q_0$

$J_0 = 1$ $\quad K_0 = 1$

13. JK flip-flop'u kullanılarak 11. sorudakine benzer, ancak 4 bit yerine 6 bitlik bir sayıcı tasarlanmak istenmektedir. Böyle bir sayıcının 3 bitlik örneği de ardışıl devre tasarım yöntemi ayrıntıda yapılmıştı. Tasarım yapılmadan flip-flop'lara ait J_5 K_5, J_4 K_4, J_3 K_3, J_2 K_2, J_1 K_1 ve J_0 K_0 girişlerine ait lojik ifadeleri öngörülebilir mi? Nasıl?

11.

Saklayıcı, Sayıcı ve Bellek Elemanları

Saklayıcı, sayıcı ve bellek elemanları ardışıl lojik devre tasarımı ve sayısal devre uygulamalarında çok kullanılan birimlerdir. Verinin belirli bir süre saklanması, üzerinde işlem yapılıp yeniden saklanması ve aktarılması sürekli gereksinim duyulan işlerdir; çok küçük olmadığı sürece her lojik devrede bu gibi elemanlar kullanılması kaçınılmazdır. Flip-flop, farklı bir işlevi olsa da, en küçük saklama birimidir denilebilir; üzerinde 1 bitlik veriyi tutar. Birden çok, örneğin *n* tane, flip-flop yanyana getirilerek *n* bitlik veri tutulabilir/saklanabilir; bir lojik devrede her zaman flip-flop'a ihtiyaç duyulur. Ancak tasarımı yapılacak devrenin karmaşıklığı arttıkça, yani giriş çıkış parametreleri ve fonksiyonel davranışı genişledikçe, saklayıcı, sayıcı ve bellek gibi elemanların kullanılması tasarım esnekliği getirir ve devrenin gerçekleştirilmesini basitleştirir. Bu bölümde saklayıcı, sayıcı, bellek elemanlarının genel yapısı, türleri ve fonksiyonel davranışları aşağıdaki başlıklar altında ele alınmıştır:

11.1. Saklayıcılar
- *PIPO, PISO,*
- *SIPO, SISO*
- *Universal*

11.2. Sayıcılar
- *İkili Sayıcı, Ondalık Sayıcı*
- *Gray-Kod Sayıcı*
- *Halka Sayıcı*
- *Modulo Sayıcı, Ripple Sayıcı*

11.3. Bellekler
- *RAM, ROM,*
- *PROM, EPROM, E²PROM,*
- *CAM (İçeriğiyle Erişilebilen Bellek)*

11.4. Programlanabilir Ardışıl Devreler

11.5. Özet

11.6. Sorular

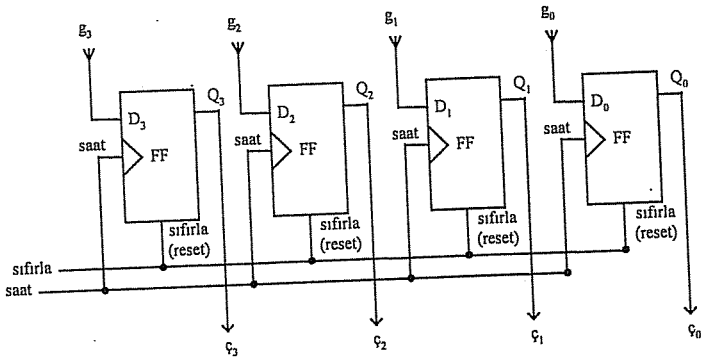
Mikroelektronik teknolojisindeki gelişmeler, her geçen gün, daha özellikli ASIC yardımcı tasarım araçlarının geliştirilmesine ve birim alana yerleştirilebilen transistör sayının arttırılmasına olanak vermektedir. Dolayısıyla, daha özellikli

tümdevreler, mikroişlemciler geliştirilmesi mümkün olmaktadır. Mikroişlemci tasarımında hemen hemen herşey saklayıcı, sayıcı ve bellek üzerine kurulmuştur denilebilir. Bu nedenledir ki, mikroişlemciyi öğrenmek, herşeyden önce, onun iç mimarisini öğrenmekten geçer; iç mimarisi de saklayıcı, sayıcı ve bellek elemanları üzerine kuruludur.

11.1. Saklayıcılar

Saklayıcı, verinin geçici olarak saklandığı elemanlardır; bit düzeyinde birimle ifade edilirler. Örneğin 4 bitlik, 8 bitlik, 32 bitlik saklayıcı gibi. Tuttuğu veri üzerinde öteleme, sıfırlama, birleme gibi yalın işlemler dışında bir işlem yapılmaz. Dolayısıyla bir aritmetik veya mantıksal işlemde parametrelerin saklanmasında kullanılır. Yalın bir saklayıcı, lojik kapı vs. kullanmaksızın sadece flip-flop'lar kullanılarak tasarlanabilir. Örneğin 4 bitlik yalın bir saklayıcı Şekil-11.1 de görüldüğü gibi 4 tane D tipi flip-flop yanyana getirilerek gerçekleştirilebilir; girişleri ve çıkışları arasındaki bağıntı aşağıdaki gibi olur:

$$Q_3(t+1)=D_3(t)=g_3 \qquad Q_3^+=D_3^+$$
$$Q_2(t+1)=D_2(t)=g_2 \qquad Q_2^+=D_2^+$$
$$Q_1(t+1)=D_1(t)=g_1 \quad \textit{veya} \quad Q_1^+=D_1^+$$
$$Q_0(t+1)=D_0(t)=g_0 \qquad Q_0^+=D_0^+$$



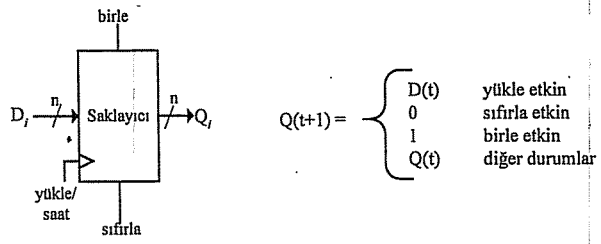
Şekil-11.1. D flip-flop' ile gerçekleştirilmiş 4-bitlik yalın saklayıcı.

Yukarıda görülen 4 bitlik saklayıcıda $g_3g_2g_1g_0$ giriş, $ç_3ç_2ç_1ç_0$ de çıkışlardır; *saat* ucu, yükle anlamında kullanılır. Şekildeki flip-flop'ların çizimine göre saat işaretinin yükselen kenarı senkronlama anıdır; dolayısıyla saat işaretinin yükselen kenarında girişlerdeki değerler saklayıcı içerisine alınır. Saklayıcı çıkışları doğrudan flip-flop

çıkışları olduğu için de tutulan değer doğrudan çıkışlara yansıtılır. Dolayısıyla, ayrı bir *göster* ucuna gerek yoktur. *sıfırla* (*reset*) ucuysa, saklayıcı içeriğini gerektiği anda 0000 yapmak için kullanılır.

Şekil-11.1 de D flip-flop'larıyla gerçekleştirilmiş saklayıcı en yalın durumdadır; bir saklayıcıda temelde giriş çıkış uçlarına ek olarak *yükle* (*load*) ve sıfırla (*reset*) uçları bulunur. Saklayıcı, genel olarak, tuttuğu/sakladığı veri üzerinde işlem yapmaz; işlevi, veriyi geçici olarak tutmak ve işlem yapacak birimlere sunmaktır. Ancak, tuttuğu veriyi bit düzeyinde sağa sola öteleyen saklayıcı türleri vardır. Bu tür saklayıcılara ötelemeli saklayıcı (*shift register*) denir. İkili tabanda bir sayının 1 bit sola ötelenmesi, sayının 2 ile çarpılmasına; 1 bit sağa ötelenmesi 2'ye bölünmesi anlamına gelir.

Saklayıcılar verinin giriş ve çıkış şekline göre paralel/paralel (PIPO), paralel/seri, (PISO) seri/paralel (SIPO) ve seri/seri (SISO) olarak sınıflanırlar. Paralelin anlamı tüm bitlerin aynı anda gireceği ve çıkacağını; seri ise, bitlerin tek tek girip çıkacağını gösterir. Seri giriş çıkışı olan saklayıcılar aynı zamanda ötelemeli saklayıcıdır; ötelemeli saklayıcılarda bir de *ötele* (*shift*) ucu bulunur. Bazı saklayıcılarda sıfırla ucunun karşıtı olan ve saklayıcının tüm bitlerini 0 yerine 1 yapan uç da bulunabilir; bu uç *birle* (*set*) olarak adlandırılır. Aşağıda, Şekil-11.2 de yalın bir saklayıcıda olabilen uçlar ve saklayıcının simgesel gösterilimi görülmektedir.



Şekil-11.2. Saklayıcı uçları: G/Ç, yükle, sıfırla, birle.

Yükle ucu etkin olduğu zaman giriş uçlarındaki (D_i) veri saklayıcı üzerinde tutulur ve aynı zamanda çıkış uçlarına (Q_i) yansıtılır; bir başka yükleme, sıfırlama veya birleme olana kadar orada tutulur. *Sıfırla* etkin olduğu zaman tüm çıkışlar lojik 0 değerini alır; *birle* ucu sıfırla ucunun tam tersini yapar, tüm çıkışların lojik 1 olmasını sağlar. *n* saklayıcının kaç bitlik olduğunu gösterir.

Ötelemeli Saklayıcı (*Shift Register*)

Ötelemeli saklayıcıda bilgi bitleri sağ veya sola ötelenerek kaydırılabilir. Yani, 1 bitlik bilginin tutulduğu hücrenin içeriği hemen sağında veya solunda komşu olan hücreye kaydırılabilir; komşu olan hücredeki değerde ona komşu olana kaydırılır...

Kaydırma işleminin etkisi aşağıdaki gibi açıklanabilir; verinin 8 bit ve 11110000 olduğu varsayımıyla kaydırma işleminin etkisi şöyle olur:

veri: 11110000, 1 bit sağa öteleme ⇒ 01111000

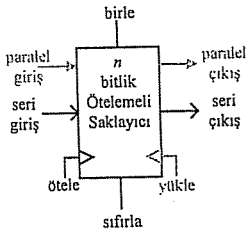
veri: 11110000, 1 bit sola öteleme ⇒ 11100001

veri: 11110000, 4 bit sağa öteleme ⇒ 00001111

veri: 11110000, 8 bit sağa veya sola öteleme ⇒ 11110000 (aynısı)

(Sağdan/soldan boşa düşen bitlerin diğer taraftan içeri girdiği varsayılmıştır! Bu şekilde yapılan öteleme döndürme olarak adlandırılır.)

Ötelemeli saklayıcının genel simgesel gösterilimi Şekil-11.3. ve D flip-flop'larıyla gerçekleştirilmiş 4 bitlik ötelemeli saklayıcının açık şeması Şekil-11.4 de görülmektedir. Ötelemeli saklayıcılarda paralel giriş veya paralel çıkışlar da olabilir; aşağıdaki şekilde gri çizgiler paralel giriş/çıkışlar olması durumda geçerli uçlardır.



Şekil-11.3. Ötelemeli saklayıcı simgesel gösterilimi.

Yukarıda şekilde bir saklayıcıda bulunabilecek olası giriş/çıkış ve denetim uçları gösterilmiştir. Uygulamada bunların hepsinin birarada bulunması gerekmez. Örneğin paralel giriş, seri çıkış özelliğinde bir saklayıcı gereksinimi varsa, yalnızca bu uçları ve gerekli denetim uçları olan saklayıcı yeterli olur.

PIPO, PISO, SIPO, SISO ve Üniversal Saklayıcı

Saklayıcılar giriş ve çıkışlarının seri veya paralel olmasına göre sınıflanırlar; yalın bir saklayıcı, yalnızca, örneğin Şekil-11.1 deki gibi paralel girişli paralel çıkışlı veya örneğin Şekil-11.4 deki gibi seri girişli seri çıkışlı saklayıcılardır. Bu tür saklayıcılar bit sayısı kadar D türü flip-flop'lar kullanılarak yalın bir şekilde gerçekleştirilebilir. Diğer kombinezonlar ve ek denetim uçları söz konusu olduğunda ise saklayıcının iç yapısı karmaşıklaşır.

Paralel giriş PI (*Parallel In*), çıkışsa PO (*Parallel Out*) ile gösterilirken seri giriş SI (*Serial In*), seri çıkışsa SO (*Serial Out*) ile simgelenirler; bu dört giriş çıkış şekliyle

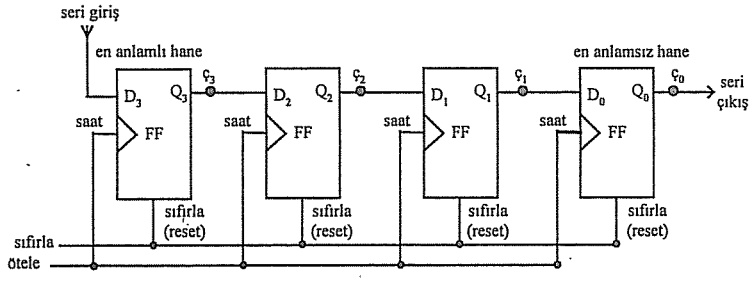
saklayıcılar PIPO, PISO, SIPO ve SISO olarak sınıflanırlar. Seri giriş veya çıkış özelliği olan saklayıcılara, aynı zamanda, ötelemeli saklayıcı (*shift register*) denir. Tüm dört sınıfın özelliklerine sahip saklayıcılara da üniversal saklayıcı denir; dolayısıyla üniversal saklayıcıların giriş/çıkış ve denetim uçları fazla olur. Şekil-11.3 de simgesel olarak gösterilen saklayıcı tipik bir üniversal saklayıcıya aittir.

Tasarlanan lojik devre ASIC olarak üretilecekse, tasarımcı, kendi gereksinime uygun saklayıcıyı flip-flop ve kapılar kullanarak yenibaştan geliştirebilir. Ancak, devre bir kart üzerine yerleştirilecekse, hazır saklayıcı tümdevreleri kullanılır; bu amaçla çok değişik özellikte saklayıcılar tümdevre olarak bulunmaktadır. Örneğin, 74164 tümdevresi 8 bitlik SIPO (seri giriş paralel çıkışlı); 74165 ise 8 bit PISO (paralel giriş seri çıkışlı) özellikte birer saklayıcıdırlar. Tümdevre olarak üretilmiş olan saklayıcılar, genel olarak, 4 veya 8 bitlik olarak üretilirler; 4 bitlik saklayıcılara örnek olarak 74194 (4 bitlik iki yönlü), 74195 (4 bitlik paralel erişimli) verilebilir. Birkaç bitlik yalın saklayıcı için D flip-flop tümdevreleri kullanılabilir; örneğin 7474 (2 tane D türü FF; birle, sıfırla uçları var ve yükselen kenarda tetikleniyor) veya 74173 (4 tane D türü FF; sıfırla ucu var, yükselen kenarda tetikleniyor ve çıkışlar 3-durumlu) tümdevresi kullanılabilir.

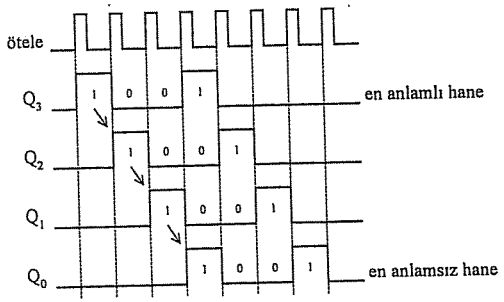
Örnek-11.1.

D flip-flop'ları kullanarak 4 bitlik seri girişli ve seri çıkışlı ötelemeli bir saklayıcının lojik şemasını çiziniz. Saklayıcıda tutulan başlangıç değerin 0000 olduğu varsayımlıyla 1001 sayısının saklayıcı içerisine ötelenmesini şekillerle çizerek gösteriniz.

İstenen saklayıcının şeması aşağıdaki gibi olur; 1001 verisinin saklayıcı içerisinde ilerlemesi ise Şekil-11.5 de görülmektedir.



Şekil-11.4. Seri giriş/çıkışlı 4 bitlik ötelemeli saklayıcı (Örnek-11.1).



Şekil-11.5. 1001 sayısının çeşitli şekillerde ötelenmesi sonuçları (Örnek-11.1).

Örnek-11.2.

01001100 sayısının 1 bit sağa ötelendiğinde ikiye, 2 bit sağa ötelendiğinde dörde bölme işlemi yaptığı gösteriniz. Aynı sayı 3 bit sağa ve 1 sola ötelendiğinde ne olur? Sağdan veya soldan boşa düşen bitlerin kaybolduğu ve diğer taraftan içeri 0 girdiği varsayılmıştır!

Herşeyden önce sayıyı ve ötelenmiş durumlarını belirlemek gerekir. İstenen öteleme işlemleri sonucu sayı aşağıdaki gibi olur. Parantez içlerinde sayıların 10 tabanındaki karşılıkları yazılmıştır.

sayı:	01001100 (76)
1 bit sağa öteleme:	00100110 (38)
2 bit sağa öteleme:	00010011 (19)
3 bit sağa öteleme:	00001001 (9)
1 bit sola öteleme:	10011000 (152)

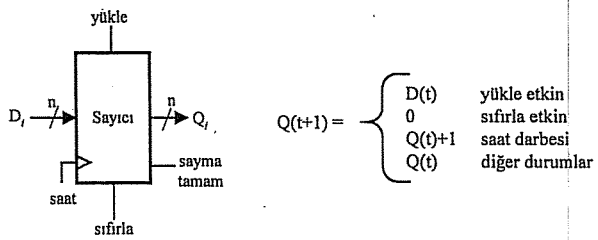
Görüleceği üzere sayı 1 bit sağa ötelendiğinde 2'ye, 2 bit sağa ötelendiğinde 4'e bölünmüş gibi oldu; 3 bit sağa ötelendiğinde ise 8'e bölünmüş gibi oldu. Çünkü 76 sayısının 8'e bölünmesiyle sonuç 9,5 çıkması gerekirdi; 9 çıktı. Yani kesirli kısım kaybolmuş. Kısaca bir tamsayının ikili tabandaki değeri *n* bit sağa ötelenirse sayı 2^n'e bölünmüş olunur; sola ötelendiğindeyse çarpma yapılmış olunur. Ancak, öteleme yapılmalıdır; döndürme değil!

11.2. Sayıcılar

Sayıcılar, saklayıcılarda olduğu gibi *n* bitlik bilgi tutmanın yanısıra, her saat çevriminde tuttukları değeri belirli bir sırada değiştiren elemanlardır; arttırır veya azaltır. Örneğin, 4 bitlik yukarı sayıcı denildiğinde çıkışlarındaki değeri, 0'dan başlayıp birer artımla 15 kadar arttıran sayıcı akla gelir. Bir sayıcı aksi söylenmediği sürece *ikili* (*binary*) ve *yukarı sayıcı* (*up counter*) olarak varsayılır. İkili sayıcı, *0*'dan başlayıp 2^n-1'e kadar birer artımla (0,1... 2^n-2, 2^n-1) sayılması anlamına gelir.

Sayıcılar, esnek bir tasarıma imkan vermesi ve tasarım kolaylığı sağlaması nedeniyle birçok türde geliştirilmişlerdir. Bir sayıcı, genel olarak aşağıdaki 5 özellik uyarınca sınıflanmaktadır:

- Bit sayısı göre: *4 bitlik, 8 bitlik, 32 bitlik gibi.*
- Sayma sırası ve şekline göre: *yukarı, aşağıya, tek sayılar, özel sırada sayma gibi.*
- Çalışma moduna göre: *tek modlu, çok modlu gibi.*
- Durumların sayısına göre: *modülo sayıcı, 5 durumlu sayıcı gibi.*
- Geçiş fonksiyonlarının zamanlanmasına göre: *senkron, asenkron gibi.*



Şekil-11.6. Sayıcı uçları: G/Ç, yükle, sıfırla, saat işareti, sayma tamam.

Tek modlu sayıcılarda yalnızca bir çeşit sayma şekli vardır; ya yukarıya doğru, ya aşağıya doğru ya da herhangi bir şekilde sayar. Çok modlu sayıcılarda ise birden çok sayma şekli vardır; bir tane sayıcıyla istenirse yukarı, istenirse aşağıya doğru sayma işlemi yaptırılabilir. Bunun için sayma şeklini gösteren ek uç vardır: *yukarı/aşağı* gibi. Şekil-11.6 da bir sayıcının genel olarak giriş ve çıkış uçları gösterilmiştir. Burada görülen uçlara ek olarak *sayma etkin, yukarı/aşağıya* gibi uçlar da bulunabilir. *Sayma tamam* ucu, saymanın bir çevrim tamamlandığını gösterir; bu çıkış sayıcıların ard arda bağlanıp daha çok bitlik sayıcılar elde edilmesinde kullanılır.

Lojik tasarımda değişik türde sayıcılara gereksinim duyulur; bunu daha çok uygulama belirler. Örneğin bir kapıdan girenleri sayıp sonucu 7-parçalı göstergede gösteren bir devre yapmak için yalın bir yukarı sayıcı yeterlidir; her giriş için bir saat işareti üretilirse sayıcı tuttuğu değerin içeriğini bir arttır. Başka bir uygulama içinse başka bir sayıcı türü uygun olabilir. Bu nedenle birçok sayıcı türü standart olmuştur denilebilir. Bunlar dışında bir sayıcıya ihtiyaç olduğunda ise o sayıcı özel olarak tasarlanabilir. Yeni bir sayıcı tasarlamak için ya standart sayıcılardan biri ya da doğrudan flip-flop'lar kullanılır. Standart olmuş sayıcılar ve özellikleri şu şekilde özetlenebilir:

İkili Sayıcı (*Binary Counter*): *n* bitlik çıkış ucu ve 2^n tane değişik durum vardır; aksi belirtilmediği sürece *0*'dan 2^n-1'e kadar birer artımla saydığı varsayılır.

Ondalık/BCD Sayıcı (*Decimal/BCD Counter*): Bir tane ondalık hane için *0*'dan başlayıp *9*'a kadar sayan sayıcılardır; 10 değişik durum vardır. Aksi belirtilmediği sürece *0, 1, 2...9* sırasında sayar.

Gray-Kod Sayıcı (*Gray-Code Counter*): İkili sayıcıya benzer; *n* bitlik çıkış ucu ve 2^n tane değişik durum vardır. Ancak sayma sırası ikili sayıcıda olduğu gibi $0,1,2...2^n-1$ olarak yapılmaz; sayma sırası öyle düzenlenir ki, ard arda iki durum arasında yalnızca 1 bitlik değişiklik yapılır.

Halka Sayıcı (*Ring Counter*): Halka sayıcı, ötelemeli saklayıcı gibi içerisindeki değeri lojik olarak sağa veya sola öteleyerek çalışır; *n* bitlik halka sayıcıda *n* adet durum vardır.

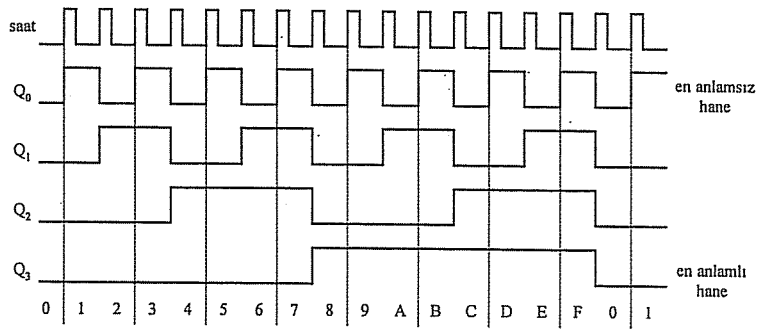
Modulo Sayıcı (*Modulo Counter*): Modulo sayıcılar, sayma sırasını modulo aritmetiğine göre belirli bir yerde kesip yeniden başlangıç noktasına dönmesini sağlar. Örneğin Modulo 10 sayıcı denildiğinde, 4 bitlik sayıcı akla gelir: ancak, 4 bit ile 0'dan başlayıp 15'e kadar sayma yapılmaz; çıkışların değeri 9'u gösterdiği zaman bir sonraki çıkışı başlangıç durumu olur.

Ripple Sayıcı: Ripple sayıcılarda asenkron davranış özelliği vardır. Senkron sayıcılarda tüm flip-flop'lara uygulanan ortak saat işareti ripple sayıcılarda yalnızca en anlamsız haneye gelen flip-flop'a uygulanır; diğer flip-flop'lara bir önceki flip-flop'un çıkışı uygulanır. Böylece asenkron yapıda bir sayıcı elde edilmiş olunur.

Tablo-11.1. Sayıcıların sayma düzenleri.

Durum	İkili	Ondalık/ BCD	Gray-Kodu	Halka	Modulo-12	Ripple
0	0000	0000	0000	0000000001	0000	0000
1	0001	0001	0001	0000000010	0001	0001
2	0010	0010	0011	0000000100	0010	0010
3	0011	0011	0010	0000001000	0011	0011
4	0100	0100	0110	0000010000	0100	0100
5	0101	0101	0111	0000100000	0101	0101
6	0110	0110	0101	0001000000	0110	0110
7	0111	0111	0100	0010000000	0111	0111
8	1000	1000	1100	0100000000	1000	1000
9	1001	1001	1000	1000000000	1001	1001
10	1010	-	1001	-	1010	1010
11	1011	-	1011	-	1011	1011
12	1100	-	1010	-	-	1100
13	1101	-	1110	-	-	1101
14	1110	-	1111	-	-	1110
15	1111	-	1101	-	-	1111

Tablo-11.1 de çeşitli sayıcıların sayma düzenleri görülmektedir. Doğal sayıcı olarak da adlandırılan ikili sayıcının saat işaretine göre çıkış işaretlerini gösteren zamanlama diyagramı ise aşağıdaki şekilde (Şekil-11.7) görülmektedir. Dikkat edilirse en anlamsız hane olan Q_0'ın periyodu saat işareti periyodunun iki katıdır; Q_1, dört; Q_2, sekiz; Q_3 ise onaltı katıdır. Sayıcı, saat işaretinin yükselen kenarında geçiş yapmaktadır.



(saat işaretinin yükselen kenarına göre)

Şekil-11.7. Dört bitlik senkron ikili sayıcının zaman diyagramı.

Tasarlanan lojik devre ASIC olarak üretilecekse, tasarımcı, kendi gereksinime uygun sayıcıyı flip-flop ve kapılar kullanarak yenibaştan geliştirebilir; veya ASIC tasarım aracının kütüphanesinden kullanabilir. Ancak, devre bir kart üzerine yerleştirilecekse, hazır sayıcı tümdevreleri kullanılır; bu amaçla çok değişik özellikte sayıcılar tümdevre olarak üretilmişlerdir. Örneğin, 74190 tümdevresi BCD yukarı/aşağı; 74191, 4 bit yukarı/aşağıya ikili; 40102, 8 bit BCD aşağıya; 40104, 8 bit aşağıya; 4030, 12 durumlu *ripple* sayıcılardır. Bunlara ek olarak TTL (74 serisi) ve CMOS (40 serisi) birçok sayıcı tümdevresi üretilmiştir. Tasarım kolaylığı açısından en uygun sayıcının seçilmesi, tasarımı karmaşıklığını büyük ölçüde azaltabilir. Tümdevreler hakkında daha fazla bilgi için bkz. Ek-A, Ek-B; ayrıntı için tümdevre kataloglarına bakılmalıdır!

Örnek-11.3.

İkili sayıcı ile BCD sayıcı arasında ne fark vardır? Uygulamaya bağlı olarak hangisinin seçilmesi gerektiğine nasıl karar verilir?

BCD sayıcı 10 tabanındaki rakamlar düzeninde sayma yapar. eğer sayma işlemi 0 ile 9 arasında olacaksa bir tane BCD sayıcı yeterlidir; 0 ile 99 arasında sayma yapılacaksa iki tane BCD sayıcı gerekir. Kısacası, 10 tabanındaki her hane için bir BCD sayıcı gereklidir; sayıcılar ard arda bağlanır. İkili sayıcı, örneğin 4 bitlik ikili

sayıcı, 0 ile 15 arasında sayar; 16 değişik durum vardır. BCD sayıcılar 10 tabanındaki rakamları sayabildiği için 10 değişik durum vardır. Uygulamada BCD sayıcı seçilmesinin nedeni, eğer sayma sonucu 10 tabanında değerlendirilecekse her haneye ait çıkışlar doğrudan sayıcı çıkışlarından alınır. Eğer doğal ikili sayıcı kullanılmış olsaydı, hanelere ait rakamlar ek kodlama yapılması gerekirdi. Ancak, BCD sayıcılarda her hane için var olduğu halde 6 durum kullanılmaz. Bu da, durum israfına neden olur denilebilir. Şöyle ki, 2 haneli BCD sayıcının karşılığı 8 bitlik (herbir BCD için 4 flip-flop kullanılır) bir ikili sayıcıdır; 2 haneli BCD sayıcıyla 0'dan 99'a kadar sayılabilir (100 değişik sayı vardır), 8 bitlik ikili sayıcıyla ise 0-127 arasında sayılabilir (128 değişik sayı vardır). Dolayısıyla 128 sayıyı sayılabilecek bir donanımla 100 sayı sayılmış olunur.

Örnek-11.4.

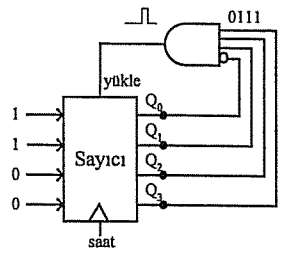
Hazır 4 bitlik paralel yükleme yapabilme özelliği olan modülo-16 sayıcı kullanılarak 3 ile 14 arasında bir sayıcı tasarlanabilir mi? Cevap *evet* ise, nasıl yapılabileceğini açıklayınız ve lojik şemasını çiziniz.

Evet, tasarlanabilir. Hazır sayıcının paralel yükleme özelliği olduğuna göre girişlerine 3_{10} sayısının ikili karşılığı 0011 sabit olarak uygulanır. Sayarken de, 14_{10} sayısının ikili karşılığı olan 1110 bit dizisine ulaşıldığında bir sonraki saat çevriminde bu bit dizisinden bir tetikleme işareti üretilir ve sayıcının yükle girişine bağlanır. Sayıcının durum tablosu Tablo-11.2 de, lojik şeması da Şekil-11.8 de görülmektedir.

Tablo-11.3. *3-14* arası sayan sayıcı durum tablosu (örnek-11.4).

Simdiki durum					bir sonraki durum				tetikleme
Q_3	Q_2	Q_1	Q_0		Q_3	Q_2	Q_1	Q_0	
0	0	1	1	3	0	1	0	0	0
0	1	0	0	4	0	1	0	1	0
0	1	0	1	5	0	1	1	0	0
0	1	1	0	6	0	1	1	1	0
0	1	1	1	7	1	0	0	0	0
1	0	0	0	8	1	0	0	1	0
1	0	0	1	9	1	0	1	0	0
1	0	1	0	10	1	0	1	1	0
1	0	1	1	11	1	1	0	0	0
1	1	0	0	12	1	1	0	1	0
1	1	0	1	13	1	1	1	0	0
1	1	1	0	14	0	0	1	1	1

Sayma düzeninde son durum olan 1110'a gelindiğinde tetikleme işareti üretilir ve sayıcının başlangıç sayısı olan ve paralel girişlerde sabit olarak bulunan 0011 değeri sayıcıya yüklenir.



Şekil-11.7. *3-14* arası sayıcı lojik şeması (Örnek-11.4).

Örnek-11.5.

Flip-flop kullanılarak 3 bitlik bir ikili sayıcı tasarlayınız. Lojik şemasını çiziniz.

Böyle bir sayıcı, ardışıl devre tasarımının verildiği Bölüm 10'da ayrıntılı olarak ele alınmıştır. Sayıcının durum diyagramı, durum tablosu ve lojik şeması için bkz. Örnek-10.9.

11.3. Bellekler: RAM, ROM, PROM, EPROM, E²PROM ve CAM

Bellekler, *sözcük* olarak adlandırılan bit gruplarını topluca tutan saklama birimleridir; üzerlerinde birçok bellek gözü vardır ve herbir bellek gözü aynı miktarda bir grup bit dizisini saklarlar. Bir bellek gözünde saklanan bir grup bit dizisi sözcük (*word*) olarak adlandırılır; bir sözcük, yerine göre birkaç bit'ten oluşabileceği gibi 8, 16, 32, 64 ve daha fazla bit'ten oluşabilir. Bir bilgisayarın üç temel[1] parçasından biri olan bellek birimleri sayısal tabanlı sistemlerde vazgeçilmez birimlerdir. Lojik devre tasarımında da değişik amaçlar için kullanılır; doğrudan, bir kombinezonsal devre oluşturmak için kullanılabileceği gibi ardışıl devre tasarımında durum bilgisi tutmak veya geçiş işaretlerini üretmek için de kullanılabilir. Ardışıl devrelerde, özellikle, durum sayısı fazlaysa bellek kullanılması esneklik sağlar ve tasarım daha az karmaşık devreyle gerçekleştirilebilir. Çünkü bir bellek biriminde onlarca, yüzlerce, hatta binlerce saklayıcının tutabildiği veri saklanabilir.

En küçük bellek birimi bit'tir; bu, ikili tabanda 1 hanelik sayı demektir. Bellekler-de kullanılan birim ise, genel olarak sekizli (*Byte*) veya sözcüktür (*word*); 1 sekizli 8 tane bit içerir, 1 sözcük ise yerine 8'den başlayıp 128, ve hatta 256 bit içerebilir. Eğer bir bellek gözü 8 bit ise, o gözde 0 ile 255 arasında bir sayı veya bu sayılara karşılık düşürülen karakterler saklanabilir. Bir sözcüğün kaç bit olduğu değişkendir; aksi söylenmediği sürece 16, 32 veya 64 bit varsayılabilir; ancak bu değerlerde

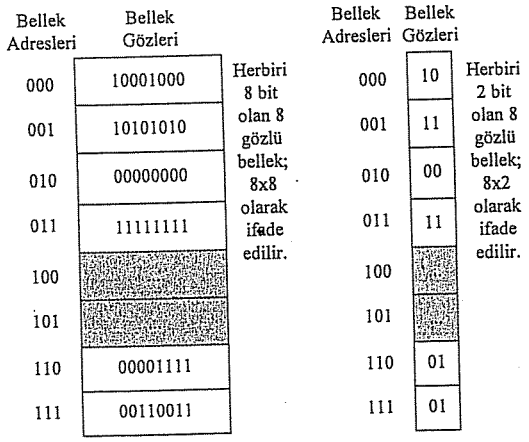
[1] Bir bilgisayarın temel birimleri, mikroişlemci, giriş-çıkış birimleri ve belleklerdir. Tüm aritmetik ve mantıksal işlemler program kodlarının yürütüldüğü mikroişlemci içerisinde yapılırken, giriş-çıkış birimleri dış birimlerle veri alış verişinde kullanılır. Bellekler ise program kodlarının ve verilerin saklandığı birimlerdir.

sabit değildir. Her sistemin sözcük boyu farklı olabilmektedir. Örneğin Pentium III işlemcisinin sözcük boyu 32 bit iken 8086 işlemcisinin sözcük boyu 16 bit'tir. Dolayısıyla bir sayısal sistemden bahsedilirken sözcük boyunun kaç bit olduğu en baştan söylenir.

Bellek birimlerinde herbir bellek gözünün kendisine ait özel bir bellek adresi vardır. Bir bellek gözüne erişip oradaki veriyi okumak veya oraya veri saklamak için o gözün adresi kullanılır. Bir bellek ifade edilirken önce belleğin içerdiği göz sayısı ve ardından herbir gözün kaç bit'lik veri tuttuğu belirtilir. Örneğin 64x4 ifadesi belleğin herbiri 4 bit olan 64 gözü olduğunu söyler; benzer şekilde 1024x8 ifadesi herbir gözü 8 bit olan 1024 gözü olduğunu belirtir. Aşağıdaki belleklerin ifade edilmesine birkaç örnek daha verilmiştir:

32x8	:	8 bitlik 32 gözlü bellek
1Kx8	:	8 bitlik 1024 (1K=1024) gözlü bellek
512x4	:	4 bitlik 512 gözlü bellek
512x16	:	16 bitlik 512 gözlü bellek
1Mx1	:	1 bitlik 1048576 gözlü bellek
1Mx8	:	8 bitlik 1048576 gözlü bellek

Genel olarak tüm bellek gözlerindeki verilere aynı anda erişilemez; ancak, bazı özel amaçlı tasarlanmış bellek birimlerinde 2 göze aynı anda erişilebilmektedir.



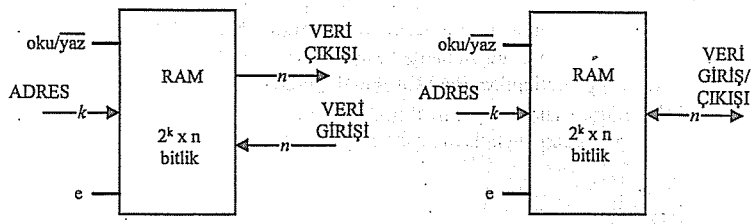
Şekil-11.8. Bellek gözleri ve adresleri.

Bellekler, farklı gereksinimleri karşılamak amacıyla çeşitli şekillerde üretilmektedir. Hem okuma hem de yazma yapılabilen bellek türü RAM olarak adlandırılır; yalnızca okuma yapılabilen bellek türü ise ROM'dur. RAM üzerindeki veriler, elektrik enerjisine ihtiyaç duyarlar; eğer elektrik enerjisi kesilirse RAM üzerindeki veriler kaybolur; ROM üzerindeki verilerin saklanması için ayrıca bir elektrik enerjisine ihtiyaç duyulmaz; veriler, ROM'un üzerine üretim aşamasında[2] kazınır. Böylece, ROM'a kazınan veriler sürekli orada kalırlar ve değiştirilemezler. Bellek türleri kısaca şöyle özetlenebilir:

- RAM *(Random Access Memory)*
- ROM *(Read Only Memory)*
- PROM *(Programmable ROM)*
- EPROM *(Erasable Programmable ROM)*
- E^2PROM *(Electrically Erasable Programmable ROM)*
- Asosiyatif Bellek/CAM *(Content Addressable Memory)*

- **RAM** (*Random Access Memory*)

RAM, bilgisayar ve diğer sayısal tabanlı sistemlerde yoğun olarak kullanılan bir bellek türüdür; koşturulan program kodları ve üzerinde işlem yapılan veriler RAM üzerinde tutulur. RAM üzerindeki bir adresten veri okunması için, o verinin daha önce yazılmış/saklanmış olması gerekir. RAM üzerine veri saklama *yazma* (*write*), RAM'den veri alma da *okuma* (read) işlemi olarak adlandırılır. Şekil-11.9 da bir RAM biriminin G/Ç (Giriş/Çıkış) ve kontrol uçlari görülmektedir; adres ve veri uçlarının kaç bit olduğu, doğrudan RAM'in boyuna (kaç gözlü olduğuna) ve herbir gözün kaç bit olduğuna bağlıdır. Şekilde adres giriş uçları sayısı k, veri G/Ç uçları sayısı da n ile gösterilmiştir. Buna göre şekildeki RAM $2^k xn$ bitlik bir RAM birimidir; eğer $k=10$, $n=8$ ise söz konusu RAM *1024x8* (1Kx8) büyüklüğünde bir bellektir.



a) Veri giriş çıkış uçları farklı b) Veri giriş çıkış uçları aynı

Şekil-11.9. RAM'in G/Ç ve kontrol uçları.

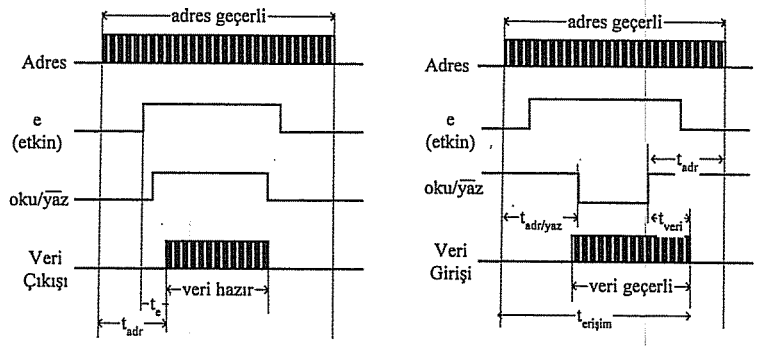
[2] Dolayısıyla ROM'a yazılacak/kazınacak veriler üretim öncesi fabrikaya verilir; üretim sonrası içinde bir değişiklik yapılması mümkün değildir. Küçük ölçekli projelerde veya araştırma amaçlı tasarımlarda ROM gerekirse, fabrika ortamında üretim yaptırılamayacağına göre, ROM'un bir çeşidi olan PROM kullanılır.

Şekil-11.9 de G/Ç ve kontrol uçları açısından iki farklı RAM gerçekleştirimi gösterilmiştir; birinde veri uçları okuma ve yazma için ayrı, diğerinde aynıdır. İkinci durum, RAM'in tümdevre olarak gerçekleştirilmesi durumunda daha az uç sayısıyla tümdevre paketlenmesi imkanı verir. RAM tümdevreleri G/Ç uçlarına ek olarak, en azından, bir tane *e* (etkin, enable), bir tane *oku/yaz'* kontrol uçlarına sahiptir; RAM, okuma veya yazma yapılması amacıyla erişilirken *e* ucu etkin yapılmalıdır. O anda okuma veya yazma yapılacağını ise oku/yaz' kontrol ucu belirler; genel olarak, 1 ise okuma, 0 ise yazma yapılır veya terside olabilir...

RAM'ler, iç yapılarında 1 bitlik verinin tutulması için kullanılan temel saklama birimleri açısından biri statik, diğeri dinamik olarak adlandırılan iki farklı şekilde üretilirler. Statik RAM'lerde en temel saklama birimi olarak flip-flop'lar veya tutucular kullanılır; dinamik bellekler de ise temel saklama birimi olarak şarj olma ve boşalma özelliğinden yararlanılan kapasitör/transistör elemanları kullanır. Statik bellekler, temelde flip-flop veya tutuculardan oluştuğu için üzerinde saklanan bilgi, tümdevreye elektrik enerjisi uygulandığı sürece orada tutulur; dinamik belleklerde ise, tümdevreye elektrik enerjisi uygulansa bile, gözlerde saklanan bilgi belirli bir süre sonra kaybolur. Çünkü, dinamik belleklerde bilgi kapasitörün şarj olması ilkesiyle saklanıyor; belirli bir sonra boşalacağından tutulan bilgi yitirilecektir. Bu nedenle, dinamik belleklerde belirli aralıklarla bilginin tazelenmesi işlemi (*refreshment*) yapılmalıdır. Bu amaçla, dinamik bellek kullanılan sistemlerde ayrıca bellek tazeleme mekanizmaları vardır.

Bellek tazeleme gerektirmediğinden, sayısal sistem tasarımında statik bellek kullanmak daha kolaydır denilebilir; aynı zamanda statik belleklere erişim daha hızlı yapılır. Ancak, statik bellekler daha maliyetlidir. Bu nedenler çok büyük bellek gereksinimler için, genelde tutulan yol, dinamik bellek kullanmaktır. Örneğin standart PC'lerdeki RAM'ler dinamik olup cep (*cache*) bellek olarak kullanılan RAM'ler statik yapıdadır.

RAM birimlerine erişip okuma veya yazma yapmak için ilgili RAM'in G/Ç ve kontrol uçlarının zamanlama işaretlerine uyulmalıdır. Şekil-11.10 da tipik olarak bir RAM'in okuma ve yazma zamanlama işaretleri/çevrimleri gösterilmiştir; bu tipik bir örnek olup, kullanılan RAM'in katalogundan bu zamanlama işareti incelenmelidir!

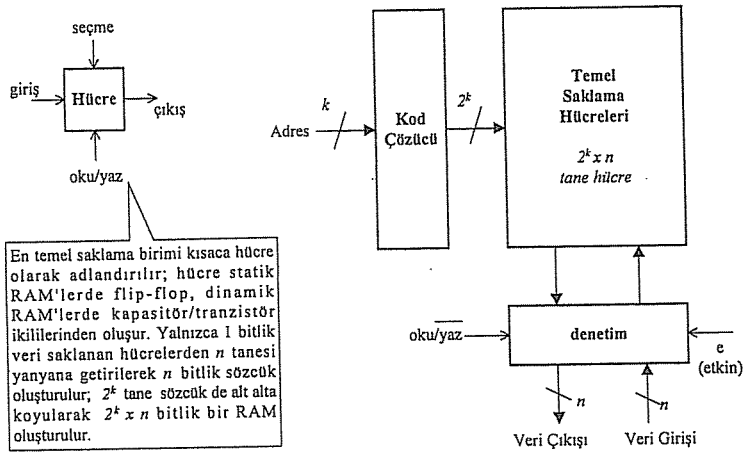


a) Okuma çevirimi b) Yazma çevirimi

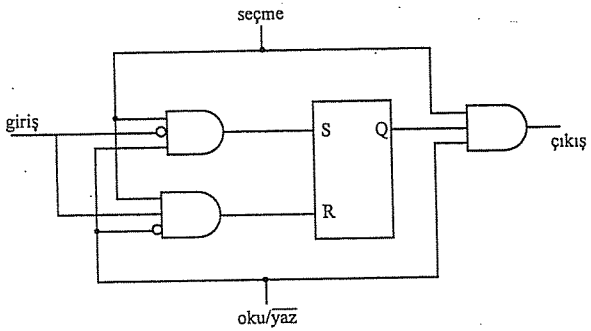
Şekil-11.10. RAM okuma ve yazma zaman diyagramları.

Yukarıdaki şekilde a)'da RAM içerisinden okuma yapılması için G/Ç ve kontrol işaretlerinin tipik zamanlama şekiller gösterilmiştir; bu işlem okuma çevrimi (*read cycle*) olarak adlandırılır; t_{adr} ve t_e zaman değerleri RAM'e okuma için erişmede propagasyon gecikmelerini gösterir. Verinin çıkış ucunda görülmesi için toplam t_{adr} süresi kadar gecikme olur; adres değişmediği sürece ve *e* (etkin) ucu aktif olduğu sürece veri çıkışta tutulur. b)'de ise yazma çevrimi (*write cycle*) olarak adlandırılan ve RAM'e yazma işleminin yapılabilmesi için gerekli zamanlamayı gösteren işaretleşme görülmektedir. Görüleceği gibi adres değişmeden sabit kaldıktan sonra *e* (etkin) aktif yapılır, veri RAM girişine uygulanır ve *oku/yaz'* ucu yazma etkin olacak şekilde aktif yapılır ve bir süre beklenir; aktif olma süresi bittikten sonra da adres bilgisi t_{adr} süresi kadar değişmemelidir. Buradaki zamanlama bilgilerine ait gerçek değerlere ilgili RAM'in katalogundan bakılmalıdır. Burada verilen G/Ç ve kontrol işaretlerinin davranışsal şeklidir.

Şekil-11.11 de RAM içerisinde 1 bitlik veri saklayan hücrenin simgesel şekli ve RAM'in iç mimarisi görülmektedir. Hücrenin iç yapısı, statik RAM'lerde flip-flop'lardan, dinamik RAM'lerde ise kapasitör/transistör ikilisinden oluşur; Şekil-11.12 de statik RAM'lerde hücre yapılarının tipik iç yapısı görülmektedir. Statik RAM'ler hem hızlı hem de kullanımı yalındır; ancak daha pahalıdır.



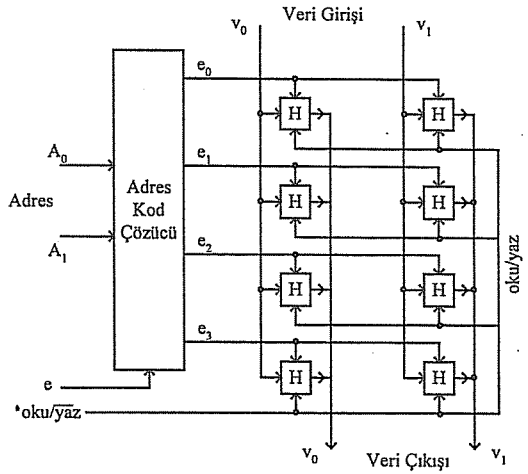
Şekil-11.11. Bellek hücresi ve RAM'in iç mimarisi/düzenlenmesi.



Şekil-11.12. Statik RAM hücrelerinin iç yapısı.

Statik RAM hücresi, tipik olarak, 1 tane flip-flop, 3 tane üç girişli VE kapısı ve 2 tane tümleme kapısıyla gerçekleştirilir. Şekil-11.12.a) daki şekil incelenirse, *oku / yaz* ve *seçme* uçları lojik 1 değerinde iseler flip-flop'un *Q* çıkışı doğrudan *çıkış* ucuna yansır. Ancak *oku / yaz* ucu lojik 0 değerinde ise *Q*'nun *çıkış* ucuna yansıması engellenir. Yazma işlemi için *oku / yaz* ucunun lojik 0 ve *seçme* ucunun lojik 1 değerinde olması gerekir. Bu koşullar uyarınca *giriş* ucundaki değer flip-flop'a yüklenir.

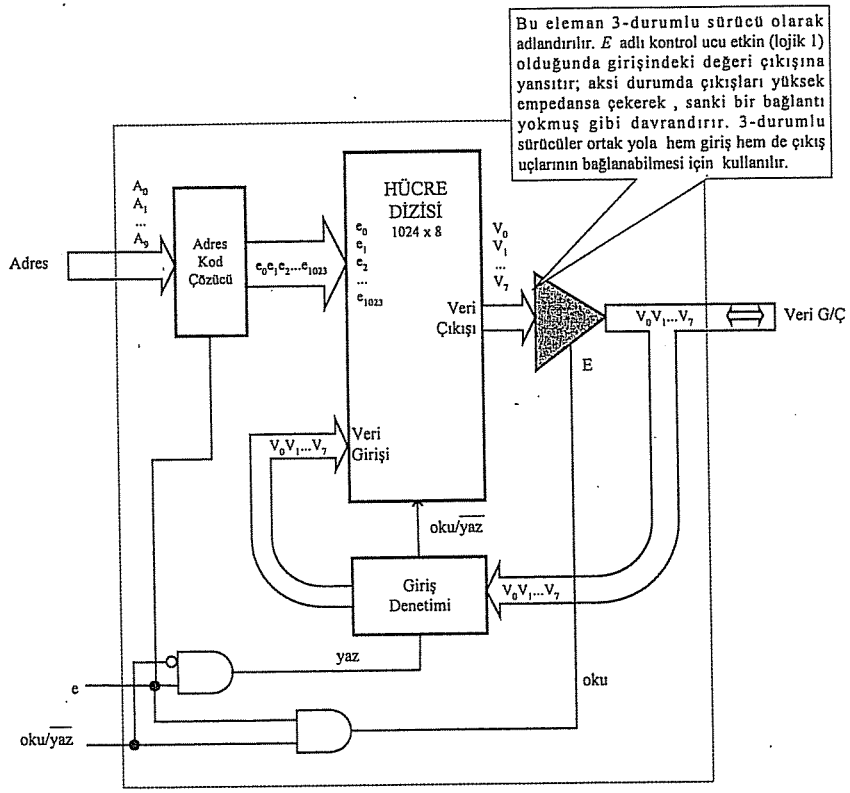
Şekil-11.13 de herbiri 2 bit olan 4 gözlü (4x2) RAM'in hücre düzeyindeki mimarisi görülmektedir. İki bit olan adres girişleri doğrudan bir kod çözücüye uygulanır; 2x4'lük bir kod çözücüye. Kod çözücü çıkışları, aynı anda, bir tanesi etkin diğerleri pasif haldedir; dolayısıyla A_0 ve A_1 girişlerinde olan adres bilgisine göre e_0, e_1, e_2, ve e_3 çıkışlarından yalnızca bir tanesi etkin olur. Etkin çıkış hangisi ise, o uca bağlı hücreler üzerinde *oku / yaz* ucundaki lojik değere göre işlem yapılır; eğer, *oku / yaz* lojik 1 ise okuma, lojik 0 ise yazma işlemi yapılır. Kod çözücünün kendisine atanmış olan işlevini yerine getirmesi için *e* (etkin) ucunun aktif olması gerektiği de unutulmamalıdır!



Şekil-11.13. RAM'in mimarisi.

Şekil-11.14 de 1 KByte büyüklüğünde bir RAM belleğin mimarisi/iç yapısı görülmektedir; 1 KByte belleğin diğer bir gösterilimi 1024x8 şeklindedir. Belleğin 10 adres ucu ve 8 veri ucu vardır; bu uçlar adres için $A_9A_8...A_0$, veri için $V_7V_6...V_0$ şeklinde simgelenmiştir. Şekilde görülen hücre dizisi içerisinde 1024x8'den toplam 8192 tane hücre vardır ve sekizerli gruplar halinde Şekil-11.13 deki benzer yapıda sıralanmıştır. Hücre dizisinin veri giriş ve çıkış uçları farklıdır; ancak RAM'in veri G/Ç uçları aynıdır. Bu amaçla 3-durumlu sürücü olarak adlandırılan eleman kullanılmıştır. Şekildeki RAM'in çalışması adres girişinde *0000000000*, *e=1*, *oku / yaz =1* için şöyle açıklanabilir: Kod çözücünün e_0 ucu etkin olur ve hücre dizisinin ilk sırasındaki 8 tane hücre etkinleştirilmiş olunur; *oku/yaz'=1* olması da oku anlamına geldiği için 3-durumlu sürücünün *E* (etkin) girişi lojik 1 değerini alarak etkin olur ve girişindeki 8 bitlik lojik değeri çıkışına yansıtır. Böylece okuma işlemi

yapılmış olunur. Aynı işlem *oku / $\overline{yaz}$ =0* için yapılacak olursa, bu sefer, 3-durumlu sürücünü çıkışı yüksek empedansa geçer ve *Veri G/Ç* uçundaki 8 bitlik veri *Giriş Denetimi* üzerinden hücre dizisinin *Veri Girişine* uygulanır. Böylece yazma işlemi gerçekleştirilmiş olunur.

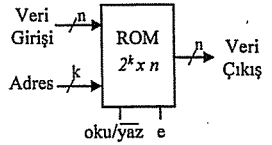


Şekil-11.14. Örnek 1 Kbyte'lık RAM'in yapısı.

- **ROM** (*Read Only Memory*)

ROM, verinin bir kez yazıldıktan sonra bir daha silinemediği veya değiştirilemediği bir bellek türüdür. Bilgisayar dahil hemen her türlü sayısal sistemde ROM kullanılması kaçınılmazdır denilebilir; genelde, sistemin ilk açılmasında yapılması gereken işlere ait program kodları veya değiştirilmeyecek olan parametreler ROM üzerinde saklanır. ROM üzerine bilgiler üretim aşamasında fabrika ortamında kazınır.

Şekil-11.15 de bir ROM biriminin G/Ç (Giriş/Çıkış) ve kontrol uçları görülmektedir; ROM veri yazılamayacağı için, dikkat edilirse veri uçları dışarıya doğrudur.



Şekil-11.15. ROM'un G/Ç ve kontrol uçları.

ROM'larda okuma zamanı üretim teknolojisine göre 40 ile 200 ns arasındadır/mertebesindedir; 40 ns olanlar Bipolar, 150-200 ns olanlarda MOS (NMOS, CMOS) teknolojisiyle üretilmişlerdir.

- **PROM** (*Programmable ROM*)

PROM türü bellekler ROM ailesinin bir üyesidir; bellek üzerine yazılacak veri fabrika ortamında kazınmaz; tasarımcı tarafından bir kez programlanacak şekilde üretilirler. Bu nedenle PROM, programlanabilir ROM olarak adlandırılır. İlk üretildiğinde PROM gözlerindeki tüm bitler lojik 1 düzeyindedir; daha sonra, programlama aşamasında lojik 0 olması gereken her bellek hücresi yakılır; yakma işlemi, ilgili bellek hücrelerine belirli düzeyde bir akım belirli bir süre uygulanarak yapılır. Bu akım düzeyi ve uygulandığı süre, PROM normal çalışmasında çektiği akım/süre'den farklıdır. PROM birkez programlandıktan[3] sonra ROM olur.

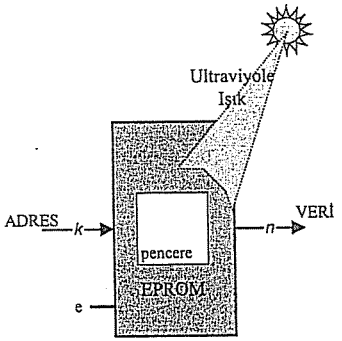
PROM türü bellekler, orta ve küçük ölçekli lojik devre tasarım ve imalatında çokça kullanılır. Ardışıl devrelerde kombinezonsal devrenin görevini görecek biçimde de kullanılabilir. İşlemci tabanlı sayısal sistemlerde ise (örneğin bilgisayar), sistemin ilk açılmasında gerekli program kodları ve değişmeyen bazı parametrelerin tutulmasında kullanılır; veya sistemin her açılışında diske ihtiyaç duyulması istenmeyen program kodlarının saklanması ve böylece sistemin ROM üzerinden açılması gerektiği durumlarda kullanılır.

PROM'larda erişim zamanı 40 ns civarındadır; bu, tabi ki *okuma* zamanıdır, *yazma* zamanı çok daha fazladır. Ancak, yazma zamanı sistemin çalışmasını etkilemediği için gözönüne alınmaz. PROM'un iç yapısı/mimarisi için Ayrıt-9.2.1 e bakınız.

[3] PROM'u programlamak için hazır düzenekler vardır; bilgisayarın ilgili portuna bağlanarak PROM içine yazılması gereken veriler bilgisayar üzerinden düzeneğe gönderilir. Bunun için veriler, bir editörde satır düzeninde yazılırlar ve yalın metin formatında saklanır. Programlama düzeneği tasarımcı tarafından da yapılabilir. Bu durumda lojik 1 olan hücreleri lojik 0 yapacak akım düzeyine ve uygulanacak süreye uyulmalıdır!

- **EPROM** (*Erasable Programmable ROM*)

EPROM türü bellekler tasarım aşamasında ve çok fazla sayıda olmayan lojik devre üretimlerinde kullanılır; PROM gibi ROM ailesinin bir üyesidir. ROM'dan farkı hem programlanabilir hem de silinebilir özelliği olmasıdır. PROM gibi programlanabilir; ancak, EPROM programlandıktan sonra içerisindeki veriler ultraviyole ışıkla silinip başlangıç haline dönüştürülebilir. Bu nedenle, EPROM tümdevrelerinin üzerinde bir pencere vardır; silme işlemi bu pencere üzerine ultraviyole ışık uygulanarak yapılır. Bu pencere normal zamanda siyah bir band ile ultraviyole ışık geçirmeyecek şekilde kapatılmalıdır; aksi durumda EPROM üzerindeki veriler kaybolabilir.



Şekil-11.16. EPROM'un G/Ç ve kontrol uçları.

EPROM'larda okuma hızı tipik olarak 150-200 ns mertebesindedir; silme zamanı ise yoğun ultraviyole ışık altında 10-30 dk (ışığın şidetine göre değişir) civarındadır.

- **E^2PROM veya EEPROM** (*Electrically Erasable Programmable ROM*)

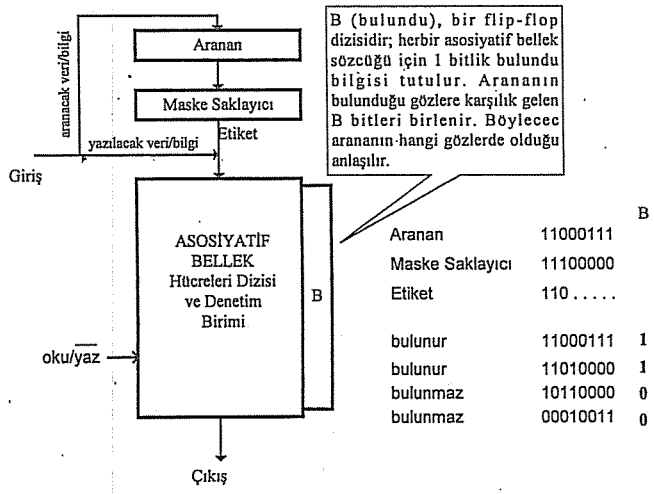
E^2PROM, EPROM'un farklı bir türüdür; ROM ailesinin bir üyesidir. E^2PROM'larda yazma ve silme işlemler elektriksel olarak yapılır; EPROM'larda olduğu gibi ultraviyole ışığa ihtiyaç yoktur; belirli düzeyde gerilim belirli süre uygulanarak hem yazma hem silme işlemleri kotarılır. Okuma süresi 200-250 ns; yazma ve silme milisaniyeler mertebesindedir.

- **Asosiyatif Bellek veya CAM** (*Content Addressable Memories*)

Asosiyatif bellek, bellek gözlerine erişmek için adres bilgisine ihtiyaç olmayan bir bellek türüdür; içeriğiyle erişilebilen bellek (Content Addressable Memory, CAM) olarak da adlandırılan asosiyatif bellekler, bellek içerisinde bir bilginin varlığını hızlı bir şekilde arayıp erişmek için kullanılır. RAM'de olduğu gibi herbir gözün

bellek adresi yoktur; ancak, RAM gibi hem okuma hem yazma yapılabilir. RAM, ROM gibi belleklerin içerdikleri veriler üzerinde bir bilginin varlığını arayıp bilgiye erişmek için onun bellekte saklı olduğu bellek gözlerine ait adreslerin bilinmesi gerekir. Eğer adresler bilinmiyorsa en küçük adresten başlanıp aranan bilgi bulunana kadar bellek gözlerine bakılması gerekir; yani, bilginin hangi bellek gözünde olduğu aranmalıdır. Bu süreç, tek tek tüm bellek gözlerine bakılmasını gerektirebileceğinden zaman alır ve adresi bilinmeyen veriye erişme süresi uzun olur. Asosiyatif belleklerde, erişilmek istenen gözün içerisindeki verinin bir kısmı, ki bu etiket (*tag*) olarak adlandırılır, sanki adres bilgisiymiş gibi verilir ve verinin geri kalan kısmına erişilir. Yani, erişilmek istenen gözün içeriği az da olsa verilerek bellek gözündeki tüm bilgilere hızlı bir şekilde ulaşılır. Eğer etiket bilgisine uyuşan birden çok veriyle karşılaşılırsa onlar arasında arama yapılır.

Asosiyatif bellekler aramanın çokça yapıldığı bellekli uygulamalarında devrenin zaman karmaşıklığını oldukça iyileştirir; yani, devrenin çalışmasını hızlandırır. Ancak bir bellek gözüne ait hücre yapısının eleman karmaşıklığı RAM'in hücre yapısına göre daha fazladır ve denetim devresi daha fazla temel lojik eleman kullanılarak gerçekleştirilebilir. Bu nedenle, asosiyatif bellekler maliyeti oldukça yüksek olur. Ancak, bilgi arama süresinin kritik olduğu tasarımlarda kullanılır.



Şekil-11.17. Asosiyatif bellek iç mimarisi.

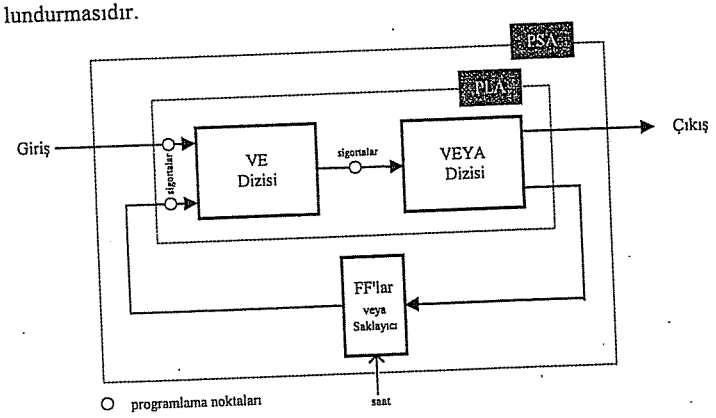
Asosiyatif bellekler çeşitli şekillerde tasarlanabilir; eğer, bellek gözlerinin tüm bitleri üzerinde arama yapılıyorsa tam asosiyatif, bir kısmı üzerinde yapılıyorsa yarı asosiyatif olarak adlandırılır. Şekil-11.17 de hem tam hem de yarı olarak kullanılabilen

asosiyatif belleğin iç yapısı verilmiştir. Eğer Maske saklayıcısındaki tüm bitler lojik 1 yapılırsa tam asosiyatif bellek gibi çalışır ve Aranan ile Maske Saklayıcısının VE'lenmesiyle elde edilen Etiket değeri aranan bilgiyle aynı olur. Dolayısıyla bellek gözlerinin tüm bitleri üzerinde arama yapılır. Maske Saklayıcısı yarı asosiyatif bellek olarak kullanmak için koyulmuştur; böylece tüm bitler üzerinde arama yapılmayıp, sadece belirli bir kısmı üzerinde arama yapılır: Ancak, bu durumda, Şekil-11.17 deki örnek aramalarda görüleceği gibi birden çok uyuşan çıkma olasılığı yüksektir. Çünkü, gözlerde birkaç bit'i aynı olan birçok veri olacaktır. B, bulundu anlamda kullanılan flip-flop dizisidir; aranan hangi bellek gözünde bulunmuşsa ona ait B bayrağı/bit'i lojik 1 yapılır.

Uygulamada asosiyatif ve RAM, ROM gibi bellek türleri bir arada kullanılabilir. Örneğin 1024x128 boyutunda bir belleğin 8 bitlik kısmı asosiyatif, kalan 120 bitlik kısmı da RAM, ROM gibi bellekler üzerinde tutulabilir. Böylece, herbiri gözü 128 bit olan bellek üzerinde 8 bitlik etiketlerle arama yapılır ve bilginin 8 bitlik kısmıyla tamamı olan 128 bitlik bilgiye erişilebilir.

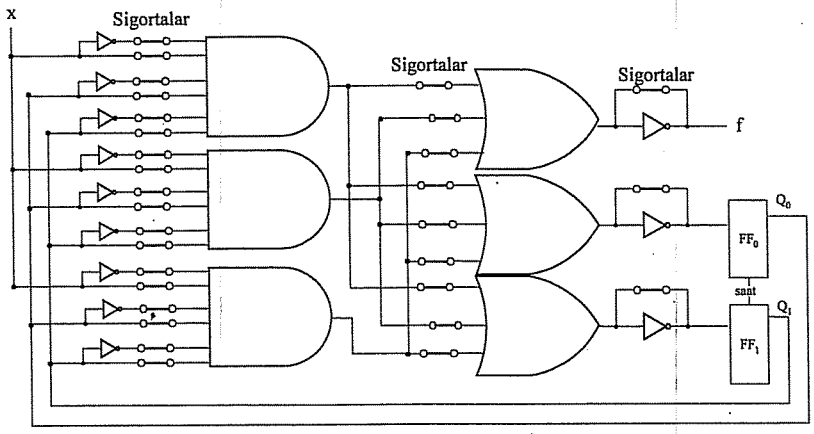
11.4. Programlanabilir Ardışıl Devreler

Programlanabilir Ardışıl Devreler (*Programmable Sequential Arrays*, kısaca PSA), tek bir tümdevre kullanarak ardışıl devrelerin gerçekleştirilmesinde kullanılır; kısaca PSA olarak adlandırılan bu tür elemanların iç yapılarında VE, VEYA dizileri, programlamada kullanılan sigortalar ve durum saklanması için FF'lar veya saklayıcı bulunur. Şekil-11.18 de PSA'ın iç yapısı görülmektedir. VE, VEYA ve sigortalar, programlanabilir tümleşik kombinezonsal devrelerdeki PLA gibidir. PSA'nın PLA'dan farkı, durum saklanması için FF'lar veya saklayıcıyı da bünyesinde bulundurmasıdır.



Şekil-11.18. Programlanabilir ardışıl devrenin genel yapısı.

Ardışıl lojik devre tasarımında PSA kullanılacağı zaman, durum bilgileri PSA içerisindeki FF'lar veya saklayıcıda tutulur; geçiş ve çıkış fonksiyonları ise sigortalar attırılarak programlanabilen VE/VEYA dizileriyle üretilir. VE/VEYA dizileri bir programlanabilen tümleşik kombinezonsal devre PLA gibidir (bkz. Ayrıt-9.2.3.); Tüm girişlerin kendileri ve tümleyenleri VE dizisindeki herbir VE kapısına sigorta üzerinden bağlıdır. VE kapılarının çıkışları da VEYA dizisindeki VEYA kapılarına sigorta üzerinden bağlıdır. Programlanmadan önce tüm sigortalar sağlamdır; bağlantı olması istenmeyen iki nokta arasındaki sigorta attırılarak programlama yapılır. Dolayısıyla programlama işlemi bağlantısı olmayacak giriş/çıkış noktaları arasındaki sigortaların attırılmasıdır denilebilir. PLA içerisinde önce VE daha sonra VEYA kapıları olduğu için geçiş ve çıkış fonksiyonların ait lojik ifadeler çarpımların toplamı şeklinde olmalıdırlar. Şekil-11.19 da 1 girişli ve 1 çıkışlı, dört durumlu bir ardışıl devrenin gerçekleştirilebileceği örnek PSA'ın iç yapısı verilmiştir.



Şekil-11.19. PSA iç yapısı için tipik bir örnek (4 durumlu, 1giriş ve 1 çıkışlı).

Örnek-11.6.

Aşağıda çıkış ve geçiş ifadeleri verilen ardışıl lojik devreyi Şekil-11.19 da verilen örnek PSA tümdevresiyle tasarlayacak biçimde attırılması gereken sigortaları belirleyiniz ve devreyi programlanmış (gerekli sigortalar attırılmış) olarak çiziniz.

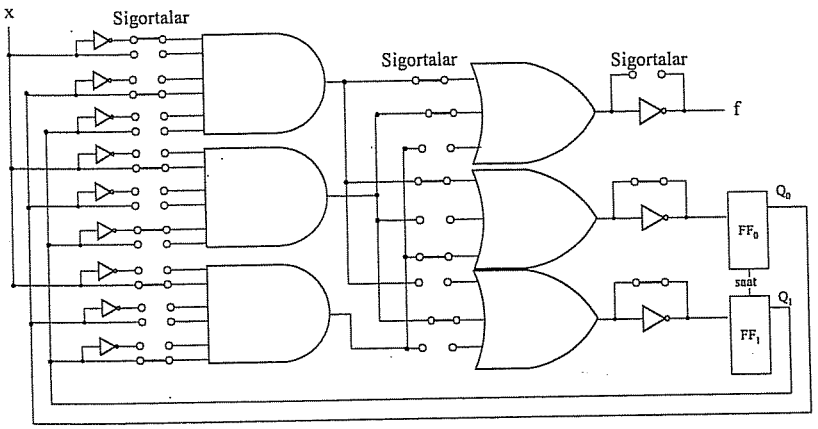
$$\overline{f} = Q_0 \cdot \overline{x} + \overline{Q}_1 \cdot x$$

$$FF_0\text{-}Giriş = Q_0 \cdot \overline{x} + Q_1 \cdot x$$

$$FF_0\text{-}Giriş = Q_1 \cdot x$$

Burada yapılması gereken, çıkış (*f*) ve geçiş (*FF_0-Giriş, FF_1-Giriş*) fonksiyonlarını sağlayacak şekilde gerekli sigortaların bırakılması, diğerlerini attırılmasıdır.

Şekil-11.20 de gerekli bağlantıyı sağlayacak şekilde sigortaların attırılmış durumu görülmektedir. Şekil dikkatlice incelenirse çıkış ve geçiş fonksiyonlarını sağlayacak sigortaların olduğu gibi bırakıldığı görülür.



Şekil-11.20. Örnek.11.6'nın PSA ile çözümü.

11.5. Özet

Saklayıcı, sayıcı ve bellek birimleri tüm sayısal devrelerin temel birimleridir; hemen herşey kapı ve flip-flop'larla birlikte bunlarla tasarlanır. Bunlar lojik devre tasarımının temel yapı taşları gibidir; türleri, özellikleri ve davranışlarının bilinmesi tasarımcı için oldukça yararlıdır. Flip-flop üzerinde 1 bitlik veri saklayan birimdir; birden çok bitlik veri saklanması için saklayıcı veya sayıcılar kullanılır.

Saklayıcıların üzerindeki veriyi sağa sola öteleme yetenekleri vardır; sayıcılar ise, tuttukları veri belirli bir sıra içerisinde arttırabilirler ve azaltabilirler. Bu şekilde öteleme yeteneği olan saklayıcılar ötelemeli saklayıcı olarak adlandırılır. Bellek birimleri ise, sözcük olarak adlandırılan bit bloklarını toplu olarak saklarlar; herbir bellek gözünde aynı sayıda bitlik veri saklanır. Sayıcılar,saat işaretinin uygulanmasına göre asenkron ve senkron; sayma düzenlerine göre ikili, BCD, halka, modülo sayıcı olarak sınıflanırlar. Bellek birimleri de üretim teknolojisi ve saklama şekillerine göre RAM, ROM; PROM; EPROM; E^2PROM, CAM olarak sınıflanmıştır.

11.6. Sorular

1. a) Flip-flop'lar kullanarak 3 bitlik *Gray* kodunda sayan bir sayıcı tasarlanmak istenmektedir. Sayıcının başlangıç anında çıkışların 000 olduğu varsayımıyla seçtiğiniz bir flip-flop türüyle sayıcıyı tasarlayınız ve şemasını çiziniz.

   b) a şıkkında tasarladığınız 3 bitlik *Gray* kodu sayıcısından iki tane kullanılarak 6 bitlik *Gray* kodu sayıcısı yapılabilir mi? Yapılabilmesi için neler eklenmesi gerekir ? Açıklayınız.

2. Saat işaretinin düşen kenarında tetiklenen 4 bitlik bir senkron ikili sayıcının zamanlama diyagramını (saat işaretine göre çıkış işaretlerinin şeklini) çiziniz.

3. Hazır 4 bitlik paralel yükleme yapabilme özelliği olan modülo-16 sayıcı kullanılarak 6 ile 13 arasında bir sayıcı tasarlayınız ve lojik şemasını çiziniz.

4. Örnek-11.6 da kullanılan Programlanabilir Ardışıl Devreyi (PSA) kullanarak aşağıda çıkış ve geçiş ifadeleri verilen ardışıl devre tasarlayınız. Devrenin çizimini sigortalara attırılmış olarak çiziniz.

   FF_0-*Giriş*= $\overline{Q}_0 Q_1$

   FF_0-*Giriş*= $Q_0 \cdot \overline{Q}_1$

   $\overline{f} = x \cdot Q_0 \cdot Q_1 + \overline{Q}_0 Q_1 + Q_0 \overline{Q}_1$

5. Ardışıl devrelerin Programlanabilir Ardışıl Devre (PSA) kullanılarak gerçekleştirilmesi durumunda, kullanılacak PSA'nın kaç tane VE, VEYA kapısı içermesi gerektiği nasıl hesaplanabilir? Açıklamaya çalışınız.

6. Bir bellek gözünde saklanan veri sözcük olarak adlandırılır ve belleğin bütün gözlerinde aynı uzunlukta veri saklanabilir. Bir sözcük yerine göre 8, 16, 32, 64 veya daha büyük bit'ten oluşabilir. Bellekleri yanyana ekleyip sözcük boyunu arttırmaya enine genişleme; alt alta koyup bellek gözü sayısını arttırmaya boyuna genişleme denir. 1 Kbyte (1024x8) kapasiteli bellek birimlerinden ikişer tane kullanarak 2048x8 ve 1024x16 boyutunda iki tane bellek birimi tasarlayınız. Belleklerin adres ve veri uçlarının nasıl bağlanacağını çizerek gösteriniz.

7. RAM'lerin üretim şekillerine göre biri dinamik, diğeri statik olmak üzere iki farklı türde üretilirler. Bilgisayarlarda ana bellek olarak kullanılan RAM'ler genel olarak dinamik, cep (cache) bellek olarak kullanılanlar ise statiktirler. Dinamik RAM'lerin tuttukları verileri kaybetmemesi için belirli aralıklarla verinin tazelenmesi olarak adlandırılan bir iş yapılır. Bunun neden yapıldığını açıklayınız. Bilgisayarlarda bu işlem nasıl yapılır?

8. Asosiyatif bellekle RAM arasındaki benzerlikleri ve farklılıkları açıklayınız. Uygulamada ikisi birlikte kullanılabilir mi? Ne tür uygulamalarda kullanılması uygundur?

9. Bir tane *yön* adlı denetim girişi olan ve *yön=1* ise yukarıya, *yön=0* ise aşağıya sayan 2 bitlik bir sayıcıyı JK flip-flop'ları kullanarak tasarlayınız ve devrenin lojik şemasını çiziniz.

10. Bir önceki (Soru 9) soruda JK flip-flop'larıyla tasarlanan sayıcıyı D flip-flop'larıyla gerçekleştirecek biçimde tasarlayarak, hangisinin daha az maliyetle gerçekleştirilebileceğini nedenleri ile açıklayarak tartışınız.

11. Statik RAM , Dinamik RAM, ROM, PROM, EPROM, E^2PROM birer bellek türüdür. Bellek ürerindeki veriyi okuma veya belleğe veri saklama işlemi erişim zamanı (access time) olarak adlandırılır. Bu bellek türlerini, erişim zamanı en küçük (en hızlı) olacak şekilde sıralayınız.

12. Neden statik RAM'e erişim zamanı dinamik RAM'e göre daha kısadır; yani, daha hızlıdır? Bir lojik tasarımda RAM seçilirken statik mi, dinamik mi olacağı nelere göre belirlenir?

12.

Ardışıl Devre Tasarım Yöntemleri

Ardışıl devre tasarımı sayısal sistemlerde önemli bir konuma sahiptir; sistemlerin pek çoğu ardışıl devre mimarisine dayanır. En basitinden en karmaşığına kadar hemen hemen tüm sayısal devreler belirli bir anda çıkış üretebilmeleri için belirli sayıda önceki çıkışlara veya durumlara ihtiyaç duyarlar. Çünkü, genelde, doğada ve uygulamada o andaki çıkış değerlerinin ne olacağı önceki durumlara yakından bağlıdır. Ardışıl devrelerin analiz ve tasarım yöntemleri Bölüm 10, ardışıl devrelerde kullanılan temel standart elemanlar ve davranışları Bölüm 11'de verilmiştir. Bu bölümde, ilk önce, Bölüm 10'da da değinilen ardışıl devre tasarımının standart yöntemi ve ardından modüler tasarım olarak adlandırılan çeşitli yaklaşım yöntemleri aşağıdaki başlıklar altında gösterilmeye çalışılacaktır:

12.1. Kanonik Yaklaşım/Tasarım
12.2. Modüler Yaklaşım/Tasarım
Veri Altyapısı
Kontrol Sistemi
Saklayıcı Aktarım Lojik (RTL)
12.3. Algoritmik Yaklaşım/Tasarım
12.4. Özet
12.5. Sorular

Ardışıl devre tasarımı çok geniş bir yelpazeye sahiptir; bir led'i sürekli olarak yakıp söndüren bir sistem, bir kapıdan içeri girenleri bir sayıcı ile sayabilen bir sistem, bir mikroişlemcinin iç tasarımı, bir evin veya mekanın güvenliğini sağlayan özel amaçlı sistem, herbiri birer ardışıl devre yapısındadır.

İlk örnek, sadece bir flip-flop'la gerçekleştirilebilecek kadar yalın ve basit iken, mikroişlemcinin iç tasarımı oldukça karmaşık ve flip-flop, saklayıcı, sayıcı gibi onlarca, yüzlerce standart lojik birimler içerebilmektedir. Doğal olarak basit denilebilecek bir ardışıl devrenin tasarımıyla karmaşık denilebilecek bir ardışıl devrenin tasarımında farklı yaklaşımlar kullanılmaktadır.

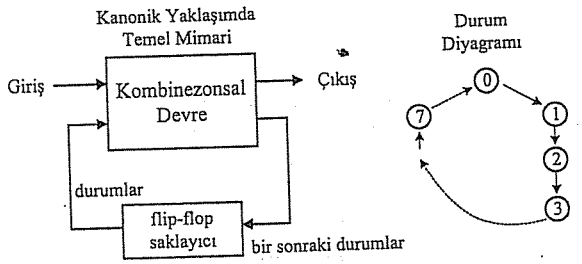
Basit denilebilecek tasarımlarda kullanılan yöntem kanonik yaklaşımdır; bu, durum sayısı fazla olmayan, girişleri ve çıkışları belirli bir sayının üstünde çıkmayan ardışıl devrelerin tasarımı için iyi bir yoldur denilebilir. Tasarlanacak devrenin karmaşıklığı arttıkça kanonik yaklaşım yerine modüler veya algoritmik yaklaşım şekli kullanılır[1].

- **Kanonik Yaklaşım**
- **Modüler Yaklaşım**
- **Algoritmik Yaklaşım**

Modüler yaklaşımda, tasarımın temeli halıhazırda var olan saklayıcı, sayıcı, PROM vs. gibi birimler üzerine kurulur; sistemin davranışı bu gibi standart birimlerin davranışıyla yakınlaştırılır ve en uygun hazır birim kullanılarak tasarlanır. Algoritmik yaklaşımda ise, tüm sistem, herşeyden önce, iki parça olarak düşünülür; biri veri altyapısı diğeri kontrol sistemi olarak adlandırılır. Tasarım yapılırken, önce, veri altyapısı belirlenir ve ardından kontrol sistemi algoritmik yaklaşımla çıkarılır.

12.1. Kanonik Yaklaşım/Tasarım Yöntemi

Kanonik yaklaşım, ardışıl devre tasarımında en temel yöntemdir; tasarlanacak devre durumlarla ifade edilir; durumlar arası geçiş ve herbir durumda çıkış değerleri durum tablosu veya durum diyagramıyla gösterilir. Tasarımda, durumları tutmak için flip-flop veya saklayıcı, durum geçişlerini ve çıkışları üretmek için de kombinezonsal devre kullanılır. Aşağıda, Şekil-12.1 de kanonik yaklaşımın mimarisi görülmektedir. Bu yaklaşımda modüler bir mimari şekli yoktur; kombinezonsal devre bilinen indirgeme yöntemleriyle kapılar kullanılarak gerçekleştirilir.



Şekil-12.1. Kanonik yaklaşım mimarisi.

[1] Benzer isimde yaklaşım şekilleri lojik tasarım literatüründe çokça geçmektedir . Buradaki sınıflama yazarlarımız Dr. Çölkesen ve Dr. Arsan tarafından önerilmektedir. Birçok kaynakta da buna benzer sınıflamalarla karşılaşılır. Kavram birlikteliği açısından aynı tanıma aynı adlandırmanın yapılıp yapılmadığına dikkat edilmelidir.

Kanonik yaklaşımla ardışıl bir lojik devre tasarımı yöntemi Ayrıt 10.4 de ele alınmış ve birkaç tane örnek çözüm yapılmıştır (Bu ayrıtın örnekleri için bkz. Örnek-10.8 ve Örnek-10.9) Bu ayrıtta da genel bir bakış yapılıp bir örnek verilmektedir. Kanonik yaklaşımla tasarım Tablo 12.1 de verildiği adımlar izlenerek yapılır.

Tablo-12.1. Kanonik yaklaşımla ardışıl devre tasarım adımları (Tablo-10.3 ün benzeridir).

Adım	Yapılması Gerekenler
1	Devrenin ne iş yapacağı/fonksiyonu açıkça tanımlanır.
2	Durum tablosunun elde edilmesi; tanımlama durum diyagramı olarak verilmişse, oradan durum tablosu oluşturulur.
3	Durum sayısından durum değişkeni sayısı belirlenir ve durumların saklanması için kullanılacak FF, saklayıcı, bellek gibi birimler seçilir. Durum kodlaması yapılır.
4	Kodlanmış durum diyagramından ve ters tanım bağıntılarından yararlanarak uyarma tablosu oluşturulur. Yani, çıkış fonksiyonları Z_i ve durum saklama birimleri girişlerine ilişkin Boole fonksiyonları tablo biçiminde elde edilerek bu fonksiyonlar belirlenir. Gerekirse indirgeme işlemi yapılarak fonksiyonlar sadeleştirilir.

Örnek 12.1.

Seri iletişimde, veri, bit düzeyinde aktarılır; yani, veriye ait bitler tek tek gönderilir veya alınır. Böyle bir aktarımda veri içerisindeki 1'lerin tek veya çift sayıda olduğunu belirleyen bir devre tasarlanmak istenmektedir. Öyle bir ardışıl devre tasarlayınız ki, bir girişli (x) ve bir çıkışlı (z) olsun ve girişine gelen sonsuz sayıda bitler içerisinde tek sayıda 1 varsa çıkışını 1, aksi durumunda çıkışı 0 olsun. Çıkış, başlangıçta henüz hiçbir bit gelmediği için çift sayıda 1 olduğunu gösteren 0 değerine sahip olacaktır. Devreyi D flip-flop'u ve gerektiği kadar kapı elemanları kullanarak kanonik yaklaşıma göre tasarlayınız. Çeşitli bit desenleri için devrenin çıkışı aşağıdaki gibi olmaktadır:

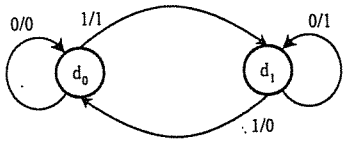
0001 0101 0011 ⇒ 1

1111 ⇒ 0

1000 0000 0000 0001 ⇒ 0

1010 0001 ⇒ 1

İstenen devre oldukça yalın bir devredir. Devrenin yalnızca 2 durumu vardır. Dolayısıyla 1 tane flip-flop ve birkaç kapı kullanılarak tasarlanabilir. Devrenin durum diyagramı aşağıdaki gibi olur:



x/z: geçiş koşulu / çıkış değeri

Bu durum diyagramına ait durum tablosuysa aşağıdaki verilebilir. Durum diyagramı ve tablosuna dikkatlice bakılırsa, x girişine 0 geldiğinde devre durumunu değiştirmiyor; ancak, 1 geldiğinde diğer duruma geçiyor. Durumlar arası geçişte de çıkış değeri tümleniyor. Bu işlem aslında bir flip-flop çeşidinin geçiş tablosuyla aynı; dolayısıyla o flip-flop kullanılırsa başka bir işleme gerek olmadan sağlanır. Ancak, bu devreyi D flip-flop'uyla gerçekleştirelim. Devrenin çıkışı flip-flop'un çıkışı olarak varsayılırsa D flip-flop'un girişine uygulanacak geçiş fonksiyonuna ait *Karnaugh* diyagramı oluşturulur ve indirgenmiş lojik ifade bulunur.

şimdiki durum	bir sonraki durum		Çıkış	
	x=0	x=1	x=0	x=1
Q	Q(t+1)			
d_0	d_0	d_1	0	1
d_1	d_1	d_0	1	0

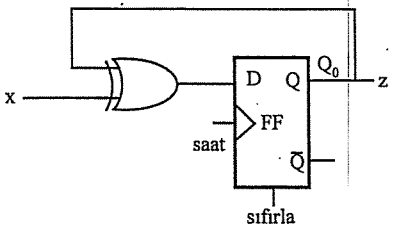
Yukarıdaki tablo D flip-flop için, geçiş özelliği de gözönünde bulundurularak aşağıdaki gibi çizilebilir. Buradan *Karnaugh* diyagramı aracılığıyla bir sonraki durumu gösteren D girişine ait lojik ifade bulunur.

	şimdiki durum	bir sonraki durum	
x	Q	Q(t+1)	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

x \ Q	0	1
0	0	1
1	1	0

$D=\overline{x}Q+x\overline{Q}$

$D=x\oplus Q$



Şekil-12.2. Örnek-12.1'in çözümüne ait lojik devre.

12.2. Modüler Yaklaşım/Tasarım Yöntemi

Modüler yaklaşımda temel tasarım birimleri olarak, kapılar ve flip-flop'lara ek olarak, sonradan yine temelde kapı ve flip-flop'larla gerçekleştirilmiş olanı saklayıcı, sayıcı, programlanabilir lojik elemanlar, bellek birimleri ve bunlara benzer modüller kullanmaya dayanır. Uygulamada, ister kart şeklinde üretilecek lojik devre tasarımı olsun, isterse bir ASIC yardımcı tasarım aracıyla bir tümdevre tasarımı için olsun halihazırda birçok modül vardır. Kart şeklinde tasarım için bu modüllerin pekçoğu birer tümdevre olarak da üretilmiştir. Örneğin, 4 bitlik sayıcı, 4 bitlik ötelemeli saklayıcı, 4 bitlik ALU, birçok boyutta RAM, ROM gibi bellekler birer tümdevre olarak üretilmiştir ve Karaköy'de[2] bolca bulunur; ASIC yardımcı tasarım araçları içinse hazır kütüphaneler içerisinde birer modül olarak bulunmaktadır.

Modüler tasarım, birazcık da olsa, tasarımcının birikimine ve öngörüsüne dayanır. Çünkü tasarlanacak devrenin davranışı, hangi standart modülün davranışına yakınlık gösteriyorsa o modül seçilmelidir. Bu da, var olan modüllerin davranışlarının bilinmesine bağlıdır; modüllerin giriş çıkış uçları, kontrol uçları iyi bilinmeli ve onların zaman diyagramları yorumlanabilmelidir.

Bilindiği gibi saklayıcıların, sayıcıların, bellek birimlerinin birçok türü vardır ve herbirinin özellikleri bir diğerine göre farklılık gösterir. Örneğin ikili sayıcının davranışı halka sayıcıdan farklıdır; veya, paralel girişli seri çıkışlı bir saklayıcının davranışı seri girişli seri çıkışlı bir saklayıcıdan farklıdır; ya da RAM ile ROM'un davranışları birbirlerinden farklıdır (Ayrıntı için bkz. Bölüm 11). Modüler tasarımda standart modüllerin bu farklılıkları, doğru öngörü açısından iyi yorumlanmış olmalıdır.

Modüler yaklaşım yöntemi, her ne kadar tasarımcının bilgi birikimine ve öngörüsüne dayalı olsa da, Tablo-12.2 de sıralanan yöntemler tasarımcının ufkunu genişletip tasarım aşamasını bir disiplin altına sokar. Böylece, her düzeyden tasarımcı, modüler yaklaşımı uygulayabilir ve onun avantajlarından yararlanabilir.

[2] Karaköy, İstanbul'un merkezi semtlerinden biridir ve burada elektronik devre elemanları, transistörler, lojik tümdevreleri gibi birçok malzeme pasajlar (Selanik, Karaköy vb.) içerisindeki işyerlerinde satılmaktadır.

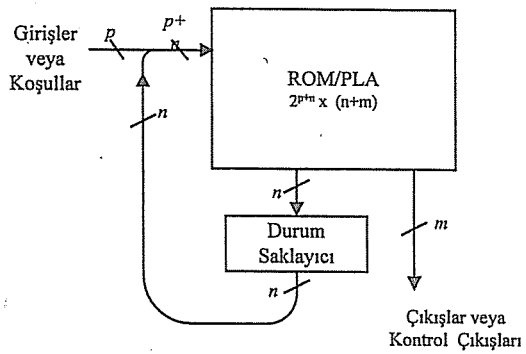
Uygulamada birçok tasarım, standart modüller ile yapılmaya çok yatkındır; bu yıllarca tasarım yapmış uzman kişiler ve tasarımcıların söyledikleri sözdür[3]. Bu nedenle tasarımın herşeyden önce standart bir modülle yapılıp yapılamayacağı düşünülmelidir. Belki de, standart bir modüle birkaç kapı eklenerek istenen devre sağlanabilir. Örneğin, bir yazıyı kayan bir şekilde alfasayısal ekranda gösteren bir ardışıl devre sayıcı ve bellek kullanılarak kolayca tasarlanabilir. Bellek gözlerine yazıya ait karakterler yerleştirilir; sayıcı da, her saat darbesinde ardışıl olarak bir sonraki bellek adresini üretirse sistem tasarlanmış olur. Çünkü sayıcının çıkışları doğrudan belleğin adreslerin uygulanır. Burada dikkat edilmesi gereken nokta belleği okuma için gerekli zaman diyagramına uyulmasıdır.

Tablo-12.2. Modüler yaklaşımda disipline yöntemler

Kullanılan Standart Modüller	Açıklama
Saklayıcı + kapılar	Durumları saklamak için saklayıcı; geçiş ve çıkış işaretlerini üretmek için kapılar kullanılır. Kanonik yaklaşımla benzer sonucu verebilir.
Saklayıcı + MUX	Durumları saklamak için saklayıcı; geçiş ve çıkış işaretlerini üretmek için MUX'un işlevi, MUX kullanılır; azda olsa kapı kullanılabilir.
Saklayıcı + Bellek	Durumları saklamak için saklayıcı; geçiş ve çıkış işaretlerini üretmek için bellek kullanılır; herbir duruma ait geçiş ve çıkış değerleri bellekte tutulur.
Sayıcı + kapılar	Durumları saklamak için sayıcı; geçiş işaretlerini üretmek için sayıcının işlevi ve kapılar kullanılır.
Sayıcı + MUX	Durumları saklamak için sayıcı; geçiş işaretlerini üretmek için sayıcının işlevi, MUX veya azda olsa kapı kullanılır.
Sayıcı + Bellekler	Durumları saklamak için sayıcı; geçiş ve çıkış işaretlerini üretmek için sayıcının işlevi ve bellek kullanılır; sayıcı belleği adreslemek için de kullanılır.
Programlanabilir Ardışıl Diziler (PSA)	Durumları saklamak, geçiş ve çıkış işaretlerini üretmek için Programlanabilir Ardışıl Devre içerisindeki Flip-flop'lar ve kapı diziler kullanılır.
Programlanabilir Sistemler	Ardışıl devrenin davranışı ve durumlar, genel amaçlı mikroişlemci içeremeyen programlanabilir sistemler aracılığıyla kotarılır.
Mikroişlemciler	Ardışıl devre, mikroişlemcili sistem üzerinde benzetimi yapılacak şekilde programlanır.

[3] Yazarlarımız da aynı sözleri söylemektedirler.

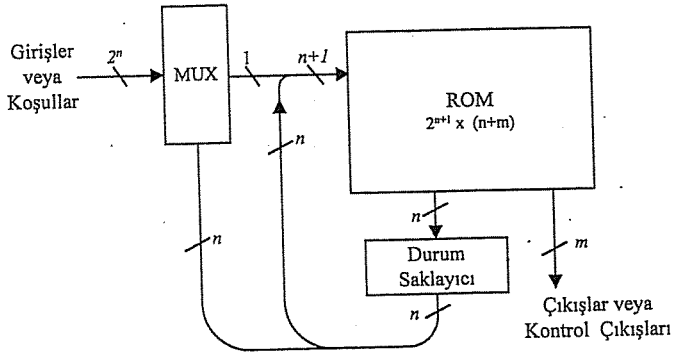
Tablo-12.2 den görüleceği gibi modüler yaklaşımda birçok yol/yöntem vardır. Bir devre bu yolların yalnızca birinin kullanılmasıyla tasarlanabileceği gibi birkaçının biraraya getirilmesiyle de tasarlanabilir. Devrenin karmaşıklığı arttıkça; yani, durum sayısı, çıkış ve geçiş fonksiyonlarının sayısı fazlalaştıkça belleğe dayalı tasarım oldukça kolaylık ve esneklik sağlar. Algoritmik yaklaşımdaki kontrol sisteminin tasarımı da, çoğu zaman modüler tasarımdaki bellekli tasarıma dayanır. Aşağıdaki şekilde (Şekil-12.3) saklayıcı (veya saklayıcı da olabilir) ve bellek kullanılarak yapılan modüler tasarımın genel mimarisi görülmektedir.



Şekil-12.3. Saklayıcı+Bellek yaklaşım mimarisi.

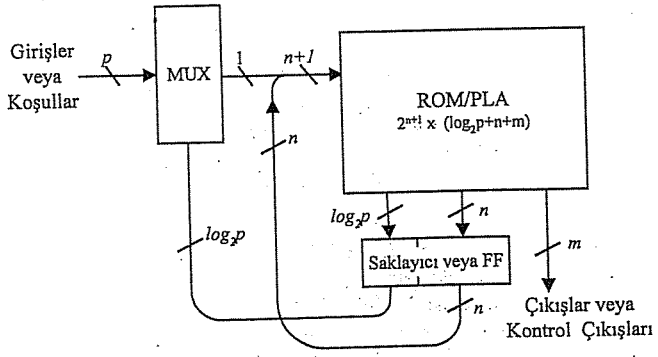
Şekil-12.3 deki mimarisel gösterimde p giriş, n durum ve m de çıkış değişkenleridir; yani, p girişli, 2^n durumlu ve m çıkışlı bir devrenin modüler tasarımına örnektir. Böyle bir tasarımda kullanılacak belleğin kapasitesi/boyu $2^{p+n}x(n+m)$ olmalıdır. Örneğin 4 girişli, 128 durumlu ve 8 çıkışlı bir ardışıl devre bu yöntemle tasarlanmak istenirse, 128 durum 7 durum değişkeniyle $(log_2 128=7)$ gösterilebileceğinden $2^{4+7}x(7+8)= 2^{11}x15= 2048x15$ boyutunda bir belleğe gereksinim vardır. Bu tür yaklaşımda kullanılması gereken bellek boyu büyük çıkabilir. Eğer giriş ve durum değişkenlerinde bazı kısıtlamalar yapılabilirse kullanılacak bellek boyutu azaltılabilir. Örneğin, Şekil-12.4 deki gibi MUX olan bir yöntem kullanılırsa, aynı, yani 4 girişli, 128 durumlu ve 8 çıkışlı bir ardışıl devre için gerekli bellek boyutu 2^{1+7} x $(7+8)= 2^8$ x $15= 256x15$ olarak çıkar.

Görüldüğü gibi giriş değişkeni, durum sayısı ve çıkış sayısı aynı olmasına karşın Şekil-12.3 deki mimaride $2048x15$, Şekil-12.4 deki mimaride $256x15$ boyunda bellek gereksinimi ortaya çıkmıştır. Üstelik, ikincisinde, bellek büyüklüğü değişmeksizin giriş değişkeni sayısı $2^4=16$'ya kadar çıkartılabilir; ancak bir kısıtlaması vardır. O da, durum geçişleri sırasında yalnızca bir tane giriş değişkeninin değerlendirmeye alınmasıdır. Tabi ki bir de MUX gereksinimi vardır.



Şekil-12.4. Saklayıcı+Bellek+MUX yaklaşım mimarisi

Şekil-12.4 deki yaklaşım yöntemi bir adım daha ileriye götürülerek Şekil-12.5 deki duruma getirilebilir. Burada, durumlar için ayrı, MUX girişleri için ayrı saklayıcı (veya FF) kullanılmıştır. Bu tür yaklaşım yöntemi daha esnek bir çözüm ortaya koyar ve birçok uygulama için çözüm olabilir. Şekilden (Şekil-12.5) görüldüğü gibi p giriş, n durum, m çıkış değişkenleridir. Buna göre, böyle bir yöntemde gerekli bellek boyu $(2^{n+1})x(log_2p+n+m)$'den hesaplanır. Örneğin 4 girişli, 128 durumlu ve 8 çıkışlı bir devre için gerekli bellek boyu, $2^{7+1}x(\log_2 4+7+8)= 2^8x(2+7+8)= 256x17$ çıkar. Aynı rakamlarla Şekil-12.4 deki mimari için gerekli bellek boyu 256x15 bulunmuştu; ikinci yöntem bir ileri adım olarak söylenmesine karşın daha büyük bellek gereksinimin ortaya çıktığı görülmektedir. Öyleyse, hangi açıdan bir adım ilerisidir? MUX açısından...



Şekil-12.5. Saklayıcı+Bellek+MUX yaklaşımı için ikinci mimari

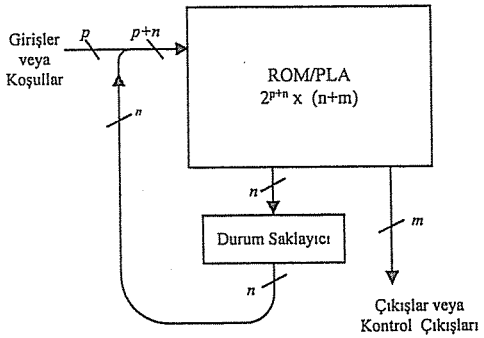
Örnek-12.2.

İki giriş üç çıkışlı bir ardışıl devrenin durum geçişlerini ve çıkışlarını gösteren durum tablosu aşağıdaki gibi elde edilmiştir. Bu devre saklayıcı-bellek yöntemine göre gerçekleştirilmek istenmektedir. Buna göre, önce devrenin genel mimarisini çiziniz ve ardından bellek birimi içerisine yazılacak bilgileri belirleyiniz.

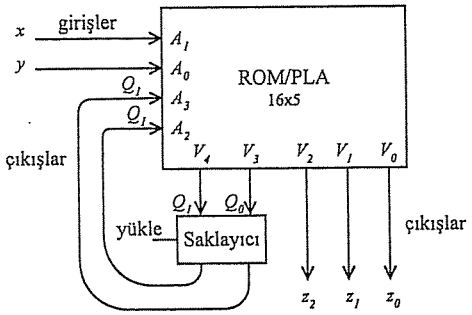
Örnek-12.2'nin durum tablosu

simgesel durum adları	simdiki durumlar		girişler		bir sonraki durumlar		Çıkış		
	Q_1	Q_0	x	y	Q_1	Q_0	z_2	z_1	z_0
d_0	0	0	0	0	0	0	1	1	1
			0	1	1	1	1	0	1
			1	0	0	1	0	0	1
			1	1	1	0	0	0	0
d_1	0	1	0	0	0	0	1	1	1
			0	1	1	0	1	0	0
			1	0	1	1	0	1	0
			1	1	0	1	0	0	1
d_2	1	0	0	0	0	0	1	1	1
			0	1	0	0	0	0	1
			1	0	1	1	1	1	0
			1	1	0	1	1	0	0
d_3	1	1	0	0	0	0	1	1	1
			0	1	1	1	0	0	0
			1	0	0	1	1	0	0
			1	1	1	0	1	0	0

Devrenin d_0, d_1, d_2 ve d_3 olarak adlandırılan 4 durumu vardır; durumlar arasındaki geçişler ve çıkış değerleri x ve y adlı girişlerine bağlı olarak yukarıdaki tabloda gösterildiği gibi olmaktadır.



Devrenin genel mimarisi Şekil-12.3 deki gibi olur; bir örneği için bakınız sağ üst tarafa. Tabloda, şimdiki durumlar ve girişler belleğin adreslerine, bir sonraki durumlar ve çıkışlar da bellek gözlerine yazılacak verilere karşılık düşer. Dolayısıyla gerekli bellek $2^{(2+2)}x(2+3)= 16x5$ boyunda olmalıdır; yani herbiri 5 bit olan 16 gözlü bellek gereklidir. Çünkü, 4 durum $log_2 4$ bağıntısından 2 durum değişkeniyle ifade edilir ve 2 tane de giriş değişkeni olduğu için giriş 4 bitliktir ve 2^4'den 16 adres vardır. Bellek gözleri 5 bitlik olmalıdır; 2 tanesi bir sonraki durumları tutmak için kullanılırken 3 tanesi de çıkışları üretmek için kullanılır. Bu çözüm şekli, saklayıcı bellek tabanlı modüler tasarımda savurgan bir bellek kullanımı gerektirir. Özellikle, durum ve girişlerin sayısı fazla olduğunda gerekli bellek boyu çok büyüyebilir. Bu durumda, eğer uygulama kısıtlama getirmiyorsa Şekil-12.4. ve Şekil-12.5 deki yöntemlerin kullanılması daha iyi sonuç verebilir. Aşağıda, Şekil-12.6 da, devrenin lojik şeması görülmektedir. Burada kullanılan bellek ROM türevi veya PLA olabilir.



Şekil-12.6. Örnek-12.2'nin çözümü (devrenin lojik şeması).

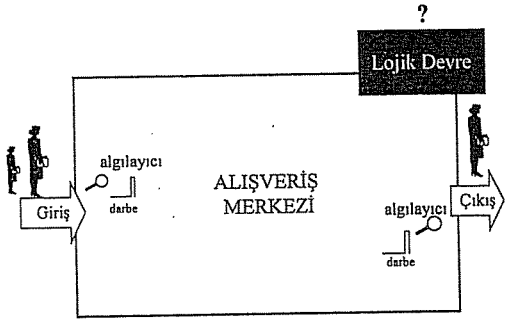
Bellek gözlerine yazılacak olan veriler doğrudan tablodan elde edilir: iki sütunlu bir tablo oluşturulur; sol tarafa adresler, sağ tarafa ise adreslere karşılık düşen veriler yazılır. Aşağıda, verilen tablodan elde dilmiş olan bellek içeriği görülmektedir:

Adres				Veri				
A_3	A_2	A_1	A_0	V_4	V_3	V_2	V_1	V_0
0	0	0	0	0	0	1	1	1
0	0	0	1	1	1	1	0	1
0	0	1	0	0	1	0	0	1
0	0	1	1	1	0	0	0	0
0	1	0	0	0	0	1	1	1
0	1	0	1	1	0	1	0	0
0	1	1	0	1	1	0	1	0
0	1	1	1	0	1	0	0	1
1	0	0	0	0	0	1	1	1
1	0	0	1	0	0	0	0	1
1	0	1	0	0	0	0	0	1
1	0	1	1	1	1	1	1	0
1	1	0	0	0	0	1	1	1
1	1	0	1	1	1	0	0	0
1	1	1	0	0	1	1	0	0
1	1	1	1	1	0	1	0	0

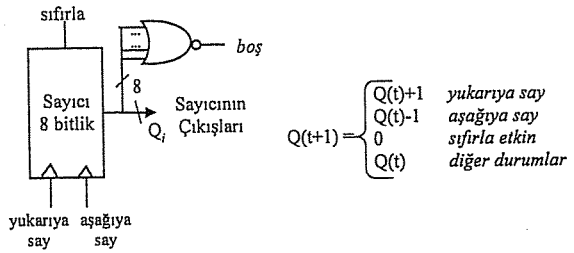
Bu devrenin gerçekleştirilmesi için yukarıdaki içerik bellek içerisine bir şekilde kazınır; yani kalıcı olması sağlanır. RAM dışında bir bellek türü kullanılmalıdır; eğer RAM kullanılırsa, devreye her enerji verilişinde, önce bu değerler belleğe yazılmalıdır.

Örnek 12.3.

Bir alışveriş merkezinde bir giriş kapısı ve bir çıkış kapısı vardır. Herhangi bir anda alışveriş merkezinde kaç kişi olduğu veya alışveriş merkezinin boşalmış olup olmadığı tasarlanacak bir lojik devreyle gözlenmek istenmektedir. Bu amaçla giriş ve çıkış kapılarına birer algılayıcı yerleştirilmiş olup algılayıcılar her geçişte bir darbe üretmektedirler. Alışveriş merkezinin en fazla 255 kişiyi alabildiği varsayımıyla gerekli lojik devreyi modüler yaklaşım dayanarak tasarlayınız; alışveriş merkezinde kimse yoksa *boş* çıkışının lojik 1 değerini alması istenmektedir.



İstenen bu lojik devre sayıcı yaklaşımıyla kolayca tasarlanabilir; çünkü, bir sayma işlemi var. Alışveriş merkezinin kapasitesi 255 kişi olduğuna göre 8 bitlik bir sayıcı bu devre için temel birimdir; eğer, 8 bitlik yukarı/aşağı sayıcı kullanılırsa birkaç kapıyla bu devre gerçekleştirilebilir. Giriş kapısı algılayıcısından alınan darbeler yukarı sayma, çıkış algılayıcısından alınan darbelerde aşağıya sayma işleminde kullanılırsa sayıcı herhangi bir anda içeride kaç kişi olduğunu gösterir. İçeride kimse olmadığını gösteren *boş* adlı çıkışta sayıcının çıkışlarına bağlanmış 8 girişli bir TVEYA kapısıyla anlaşılabilir; tüm çıkışlar lojik 0 değerinde ise TVEYA çıkışı lojik 1 olur; aksi halde boş adlı çıkış 0 değerindedir. Devrenin şematik çizimi aşağıda şekilde (Şekil-12.7) görüldüğü gibi olur:

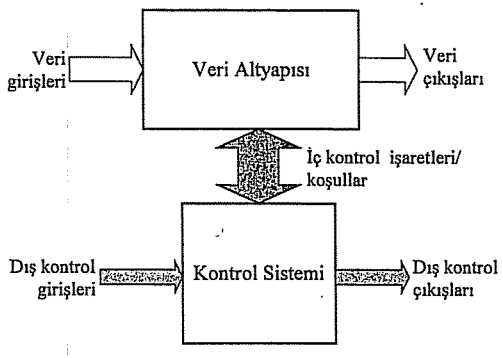


Şekil-12.7. Örnek-12.3'ün çözümü.

12.3. Algoritmik Yaklaşım/Tasarım Yöntemi

Algoritmik yaklaşımda ardışıl devre Şekil-12.8 de görüldüğü gibi iki kısımdan oluşuyormuş gibi düşünülür; biri veri altyapısı, diğeri kontrol sistemi olarak tüm sistem iki parçaya ayrılır. Veri altyapısı durum ve giriş bilgilerinin tutulduğu saklayıcı, sayıcı, bellek gibi modüllerden oluşur; kontrol sistemi de sistemden beklenen fonksiyonel davranışın yapabilmesi için veri altyapısında bulunan birimlerin çalışmasını

sağlayacak kontrol işaretlerini üretir. Örneğin saklayıcı için yükle, sayıcı için bir adım say, bellek için yaz gibi işaretler kontrol sistemi tarafından üretilir.



Şekil-12.8. Algoritmik yaklaşım mimarisi.

- **Veri Altyapısı:**

Sistemde veri saklayabilecek tüm saklayıcı, sayıcı, flip-flop, bellek ve diğer lojik elemanları içeren kısımdır. Bu birimlerin birbirlerine bağlanış biçimi, bir yerden çıkan verinin nereye gideceği; yani, veri akışı bu altyapıda açıkça belirtilir. Modüler tasarımda kullanılan tüm birimler burada doğrudan kullanılır. Birimlere ait kontrol giriş ve çıkışları veri altyapısında boşta bırakılır. Çünkü gerekli kontrol işaretleri kontrol alt birimi tarafından üretilmektedir. Dolayısıyla hangi kontrol girişine ne uygulanacağı ve hangi kontrol girişinin nereye bağlanacağı kontrol sistemi tasarımı sonrası ortaya çıkar. Veri altyapısında saklama özelliği olan birimler arasındaki veri transferine Saklayıcı Aktarım Lojiği, kısaca RTL (Register Transfer Logic) denir.

- **Kontrol Sistemi:**

Veri altyapısında bulunan birimlerin birbirleriyle etkileşimi ve verinin akışı kontrol sistemi tarafından üretilen işaretlerle sağlanır. Kontrol sistemi giriş bilgileri ya dışarıdan gelir ya da koşul altında veri altyapısında alınır; çıkışlarıysa veri akışı için gerekli kontrol işaretleri ve dışarıya verilen çıkış işaretlerdir.

Kontrol sistemi, sıkıbağlı[4], ROM'a dayalı ve mikroprogramlı olarak gerçekleştirilebilir; modüler yaklaşım şekil de kullanılabilir. Dolayısıyla Tablo-12.2 de verilen yöntemler algoritmik yaklaşımda, kontrol sisteminin tasarımında kullanılabilir.

[4] Sıkıbağlı, yazarlarımızın "hardwired" için önerdikleri sözcüktür; yerine göre donanım tabanlı, donanım bağlı denilmektedir.

Hangisinin seçileceği;

- hıza
- tasarım kolaylığına
- modülerliğine ve
- esnekliğine

göre değişir. Genel olarak istenen, en iyi başarım, en düşük tasarım ve gerçekleştirme maliyeti ve günün teknolojisine göre tasarım kolaylılığıdır. Sıkıbağlı (*hardwired*) tasarımda çıkışlar ve durum geçişleri kombinezonsal devrelerle gerçekleştirilir; bir kez yapıldıktan sonra değişiklik yapılması zordur. Bu yöntemle en hızlı çözüm yapılmış olunur. Fakat istenen küçük bir değişiklik kontrol sistemin yeniden tasarlanmasını gerektirebilir. Tasarım sürecini kolaylaştırmak için sayıcı, MUX, PLA ve ROM gibi birimler kullanılabilir.

Örnek 12.4.

Standart lojik birimler/elemanlar kullanılarak aşağıda özellikleri verilen aritmetik tabanlı basit bir işlemci tasarlanması istenmektedir. Bu işlemciyi algoritmik yaklaşıma dayanarak tasarlayınız ve ilgili adımları açıkça gösteriniz.

İşlemcinin Özellikleri:

- **Girişler Uçları:**

  4 bitlik *Veri Girişi*

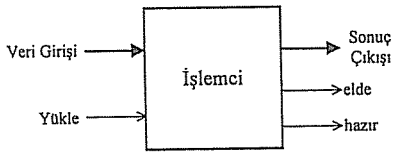
  1 bit *Yükle* girişi

- **Çıkış Uçları**

  4 bitlik *Sonuç Çıkışı*

  1 bitlik *elde*

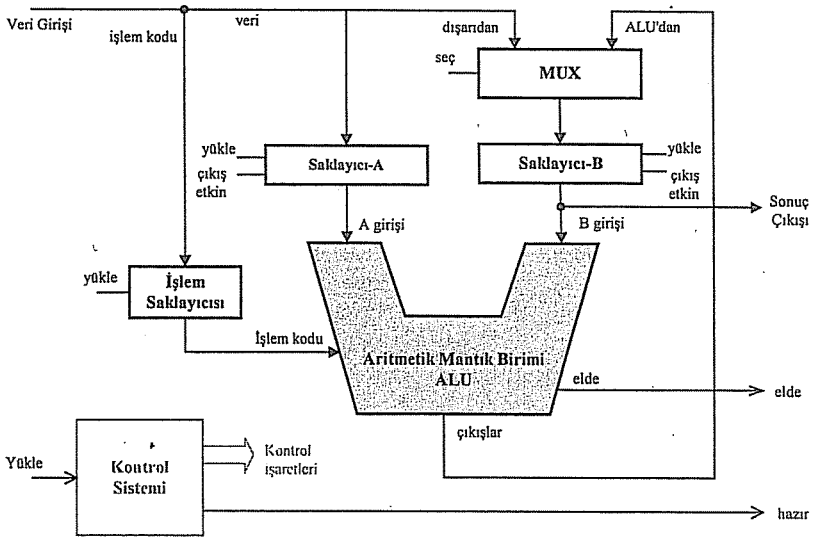
  1 bitlik *hazır*



İşlemcinin Davranışı:

Veri Giriş ucuna önce 16 tane olan işlem kodundan biri uygulanıp *Yükle* girişi etkin yapılmakta; böylece dışarıdaki işlem kodu işlemci içerisindeki bir saklayıcıya alınmaktadır. Ardından 4 bitlik iki tane verinin (parametrelerin) işlemci içerisine alınması gerekmektedir. Bunun için ilk veri *Veri Giriş* ucuna koyulup *Yükle* ucu etkin yapılmakta ve ardından aynı işlem ikinci veri için yapılmaktadır. İşlemci, belirtilen işlemi yaptıktan sonra sonucu *Sonuç Çıkışına* çıkartmakta ve aynı zamanda, sonucun hazır olduğunu belirtmek için, *hazır* ucunu lojik 1 yapmaktadır. Eğer işlem sonrası elde vs gibi bir bitlik çıkış varsa, bunu da *elde* çıkışıyla dışarıya vermektedir.

İşlemci tasarımının algoritmik yaklaşıma göre tasarımı yapılması istenmektedir; bunun için herşeyden önce veri altyapısı açıkça belirlenmeli ve ardından kontrol sistemi tasarımına geçilmelidir. Veri altyapısının tamamen standart birimlerle gerçekleştirilmesi istenmiştir; kaç tane eleman kullanılacağı konusunda bir sınırlama yapılmamıştır. Dolayısıyla ister kombinezonsal ister ardışıl olsun ihtiyaç duyulan kadar birim kullanılabilir; ancak, veri alt yapısındaki eleman sayısı arttıkça, üretilmesi gerek kontrol işaretleri sayısı artacağından kontrol sistemi karmaşıklığı artar. Dolayısıyla optimum tasarım yapılmaya çalışılmalıdır.

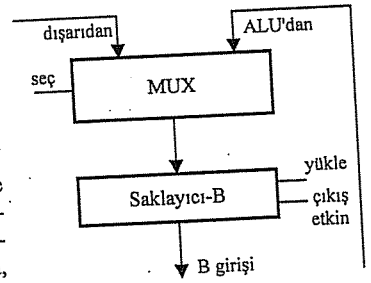


Şekil-12.9. Örnek İşlemci için veri altyapısı (Örnek 12.8).

İstenen özellikte işlemci için gerekli veri altyapısı Şekil-12.9 da görüldüğü gibi olabilir. Bu mutlak bir çözüm değildir; bir başkası daha farklı veri altyapısı ortaya koyabilir. Önemli olan optimum sayıda standart lojik birimler kullanmak ve ona ait kontrol sistemini tasarlamaktır. Önerilen veri altyapısında,

- 1 tane ALU
- 3 tane saklayıcı ve
- 1 tane MUX

kullanılmıştır; ALU ve MUX kombinezonsal ve saklayıcılar da ardışıl birimlerdir; hepsi de standart birimlerdir.



İşlemcinin fonksiyonunu yerine getirebilmesi için herşeyden önce işlem kodu ve verileri kendi içerisindeki saklayıcılarına almalıdır. Bunun için 1 tane işlem saklayıcısı, 2 tane de (A ve B olarak adlandırılmıştır) veri saklayıcısı vardır. ALU (Arithmetic Logic Unit), A ve B girişlerine uygulanan veriler üzerinde İşlem kodu girişinde belirtilen işlemi yapar ve sonucu çıkış ucuna yansıtır; sonuç ta elde vs varsa, onu da elde çıkışıyla bildirir. MUX ise, B saklayıcısına yüklenecek verinin nereden yapılacağını yönlendirir; eğer veri dışarıdan gelirse, ona göre seçme yapar ve dışarıdan gelen veri B saklayıcısına anahtarlanır; eğer işlem sonucu saklanacaksa MUX ona göre seçme yapmalıdır.

Veri altyapısında bulunan saklayıcı, MUX, ALU için birçok kontrol işareti gereklidir; bu işaretlerin üretilmesi Kontrol Sistemi tarafından yapılır. Kontrol Sistemi-nin tasarımında da işe yarayacak olan işlemcinin iç davranışı, yani işlemci içerisindeki veri akışı şöyledir:

❶ İşlem kodu ⇨ *Veri Girişi*;
Yükle → etkin

❷ İlk veri ⇨ *Veri Girişi*;
Yükle → etkin

❸ İkinci veri ⇨ *Veri Girişi*;
Yükle → etkin

Bu aşamada işlem kodu ve veri işlemci içerisindeki saklayıcılar üzerine alınmış oldu.

❹ İşlem Yapılır:
Sonuç ⇨ *Sonuç Çıkışı*; elde sonucu → *elde*
1 → *hazır*

Örneğin, işlem kodunun 0110 (toplama), verilerin de 0111 ile 1100 olduğu varsayılırsa, işlemci içerisindeki saklayıcıların içerikleri şöyle olur.

❶ İşlem Saklayıcısı ⇦ 0110; A ve B saklayıcıları belirli değil.

❷ İşlem Saklayıcısı ⇦ 0110; A ⇦ 0111; B saklayıcısı belirli değil.

❸ İşlem Saklayıcısı ⇦ 0110; A ⇦ 0111; B ⇦ 1100

Bu aşamada işlem kodu ve veri işlemci içerisindeki saklayıcılar üzerine alınmış oldu.

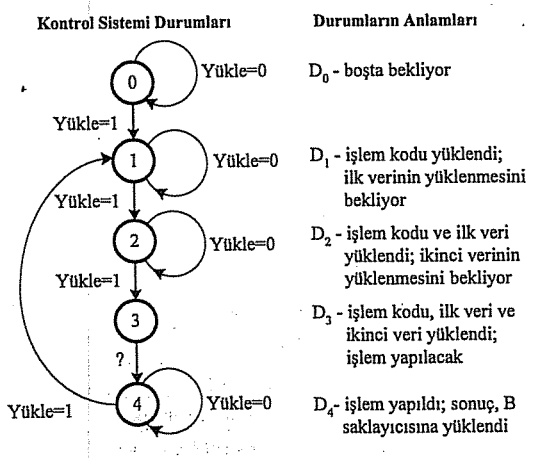
❹ İşlem Yapılır:
İşlem Saklayıcısı ⇦ 0110; A ⇦ 0111; B ⇦ 0011 (toplama sonucu elde oluştu)
0011 ⇨ *Sonuç Çıkışı*; 1 → *elde*; 1 → *hazır*

Kontrol sisteminin tasarlanmasında algoritmik bir yaklaşım kullanılır; yani, sistem davranışı programlama dillerindeki benzer bir notasyonla veya durum diyagramına benzer bir grafla ifade edilir. Buradan, herbir durum için veri altyapısında bulunan birimlere ait kontrol uçlarının pasif veya aktif olacağı belirlenir.

Üzerinde işlem yapılacak veriler işlemci üzerindeki saklayıcılarda tutulur; işlemcinin davranışını ise kontrol sistemi yönlendirir. Dolayısıyla, kontrol sistemi işlemciden beklenen davranışı sağlayacak biçimde kontrol işaretleri üretmelidir; yani, saklayıcıya yükle, MUX'a seç gibi işaretleri uygun sırayla üretmelidir.

Kontrol Sistemi Tasarımı:

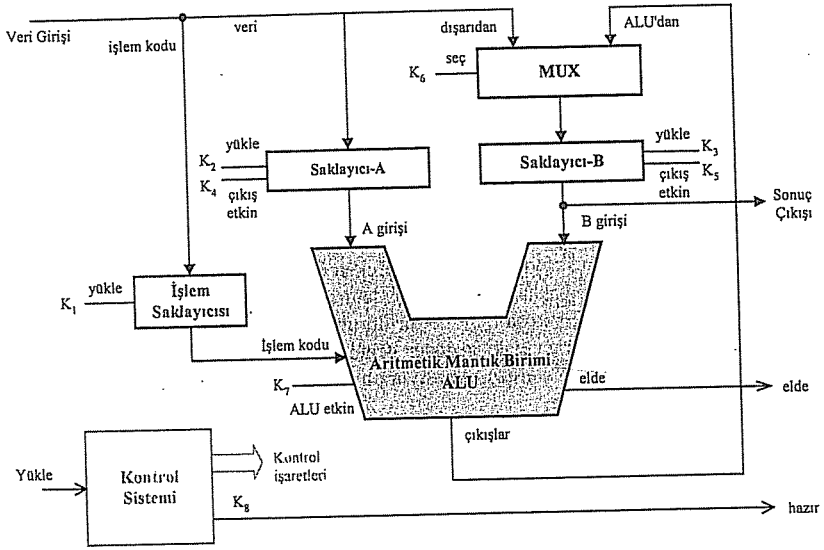
Bir önceki sayfada veri akışı için 4 adım verilmişti; bu işlemcinin kendisinden beklen işi yapması için sırayla bulunması gereken durumlara karşılık düşürülebilir. Bu adımlara bir tane de başlangıç durumu eklenirse toplam 5 tane durum[5] olur. Bu durumlar arasındaki geçiş koşulu ve herbir durumda yapılanlar Şekil-12.10 da görülmektedir; durumlar arasındaki geçişler genel olarak *Yükle* ucunun etkin (0'dan 1'e geçiş etkin olarak varsayılmıştır) olmasıyla olmaktadır. 3üncü durumdan 4üncüye geçişte ise bir koşul yoktur.



Şekil-12.10. Kontrol sistemi durum grafı ve durumların anlamı.

[5] Aslında kontrol sistemi 4 durumlu olacak biçimde de tasarlanabilir; anlatımın açıkça görülebilmesi için 5 durumlu ele alınmıştır. Ancak bu seferde 2 tane flip-flop yerine 3 tane kullanılması gerekir.

Bir kontrol sistemine ait davranış Şekil-12.10 dakine benzer bir durum diyagramıyla ortaya koyulmuşsa, bir sonraki aşama, herbir durumda veri altyapısındaki saklayıcı, sayıcı gibi birimlerin hangi kontrol girişlerinin aktif olacağı ve herbir geçiş için gerekli koşullar belirlenir. Bu aşamaya geçmeden önce Şekil-12.9 da verilen veri altyapısında kontrol girişlerinin aşağıdaki şekildeki gibi isimlendirildiği varsayılsın:

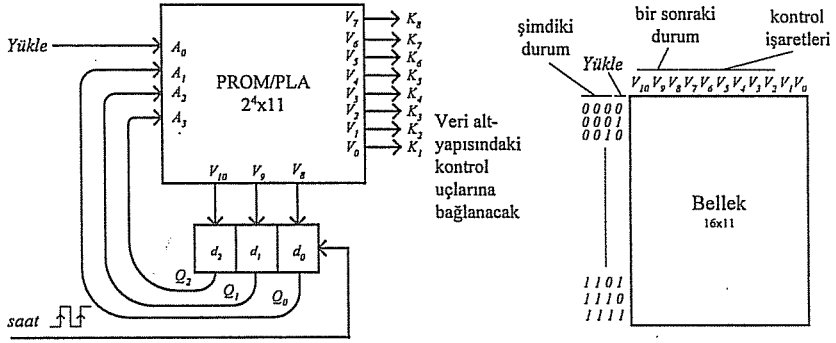


Yukarıdaki şekilde veri altyapısında bulunan saklayıcı, sayıcı gibi birimlere ait kontrol girişlerine K1, K2...K7 gibi isimler verilmiştir. Bu isimlendirmeye göre herbir adımda etkin olması gereken kontrol işaretleri Tablo-12.3 de görüldüğü gibi elde edilir.

Tablo-12.3. Kontrol uçları geçiş değerleri (Örnek-12.4).

Durum No	Etkin Kontrol Uçları	Açıklama
D_0	{ ϕ }	Beklemede.
D_1	{ K1 ← 1 }	İşlem kodu yüklendi.
D_2	{ K2 ← 1 }	İlk parametre yüklendi.
D_3	{ K6 ← 0; K3 ← 1}	İkinci parametre yüklendi.
D_4	{ K4 ← 1; K5 ← 1 } { K7 ← etkin, K8 ← 1 } { K6 ← 1; K3 ← 1 }	İşlem yapıldı. Sonuç *B*'ye yüklendi ve *hazır* çıkışı işlem sonucunun hazır olduğunu göstermesi için 1'lendi.

Kontrol sistemim modüler yaklaşımda sıralanan yöntemlerin birçoğuyla gerçekleştirilebilir. Burada, saklayıcı (veya FF)-Bellek yöntemi kullanılmak istensin. Bu durumda kontrol sistemi mimarisi aşağıdaki (Şekil-12.11) gibi olacaktır. durum sayısı 5 tane olduğu için 3 tane ($log_2 5$'den) durum değişkeni vardır. Bunlar için 3 bitlik saklayıcı veya 3 tane flip-flop kullanılır. Bellekte, bir sonraki duruma ait geçiş bilgileri ve kontrol çıkışları (K_8, K_7... K_1) tutulur. Belleğin adres kısmı ise, o andaki durumlar ve girişler için kullanılır. Bu örnek için gerekli bellek boyu (bkz. Şekil-12.113.), $2^{1+3}x(3+8)= 2^4x(11)= 16x11$ çıkar. Yani, herbir gözü 11 bit olan 16 gözlü bellek gerekir. Bellek gözlerine ait bitlerin ne amaçla kullanıldığı Tablo-12.4 de, adres uçlarının ne amaçla kullanıldığı ise Şekil-12.11 de görülmektedir.



Şekil-12.11. Kontrol sistemi mimarisi ve belleğin durumu.

Bu aşamada Tablo-12.3 e dayanılarak bellek içerisine yazılacak bilgilerin ne olacağı belirlenmelidir. Yukarıdaki şekilden görüleceği gibi durum saklama birimlerine dışarıdan bağımsız bir saat işareti uygulanmıştır; saat işaretinin yükselen kenarında tetikleme yapıldığı varsayılmıştır. Yükle girişi etkin olduğunda devre bir sonraki duruma geçer; pasif olduğunda olduğu durumda çevrime girer (bkz. Şekil-12.10). Etkin durum lojik 1, pasif durum da lojik 0 olarak değerlendirilmiştir.

Tablo-12.4 de bellek içerisine yazılması gereken bilgilerin ne olduğu bit bit gösterilmiştir. Şimdiki durum ve Yükle girişi belleğin adreslerine, bir sonraki durum ve kontrol çıkışları da belleğin veri çıkışlarına bağlanırlar. Devrenin durum tablosunda 5 farklı durum vardır; $\log_2 5$ bağıntısından 3 tane durum değişkeni kullanılması gerektiği ortaya çıkar; ancak 3 tane durum değişkeniyle 8 durumlu bir devre tasarlanabilir. Bu örnekte 3 tane durum kullanılmamaktadır. Tabloda gri olan yerler kullanılmayan bu durumlara aittir.

Tablo-12.4. Bellek içerisine yazılması gereken bilgiler (Örnek-12.4).

şimdiki durum			*Yükle*	bir sonraki durum			kontrol işaretleri							
Q_2	Q_1	Q_0		Q_2	Q_1	Q_0	K_8	K_7	K_6	K_5	K_4	K_3	K_2	K_1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0	0	0	0	0	0	0	1
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	0	0	0	0	0	0	1	0
0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0	0	0	1	0	0
0	1	1	0	1	0	0	0	1	1	1	1	1	0	0
0	1	1	1	1	0	0	0	1	1	1	1	1	0	0
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0	0	0	0	0	0	0	1
1	0	1	0	kullanılmayan durumlara ait										
1	0	1	1											
1	1	0	0											
1	1	0	1											
1	1	1	0											
1	1	1	1											

Yukarıdaki tablo saklayıcı bellek kullanımı yaklaşımıyla çözülen kontrol sisteminde kullanılacak bellek birimini içerisine yazılacak bilgileri göstermektedir. Belleği adres uçlarına şimdiki durum ve *Yükle* sütunları, veri uçlarına da bir sonraki durum ve kontrol işaretleri bağlanır. Örneğin, devre, başlangıç durumundayken (D_0) durum değişkenleri 000 değerindedir; *Yükle=1* ise, tetikleme anında bir sonraki durum olan D_1'e geçer ve K_1 kontrol çıkışı 0 iken 1 olur. MUX dışındaki birimlere ait kontrol girişlerinin 0'dan 1'e geçişi aktif; 1'den 0'a geçmesi veya 0'da kalması pasif kabul edilmiştir.

Kontrol sistemi için yapılan bu çözüm, saklayıcı ve bellek kullanımı açık olarak gösterilmek için yapılmıştır; aynı işlemi yapacak kontrol sistemi daha küçük kapasiteli bellek kullanılarak da gerçekleştirilebilir. Örneğin devrenin durum sayısı 4'e indirilirse 2 durum değişkeni yeterlidir; bir de, *Yükle* giriş ucu doğrudan saat işareti yerine kullanılabilir. Bu durumda gerekli bellek boyu 2^2x10=4x10 çıkar. Görüleceği gibi 16x11 boyunda bellek yerine 4x10 boyunda bellek kullanılarak aynı işi yapabilecek kontrol sistemi tasarlanabilir. Bölüm 6'nın sorusunda bunun nasıl yapılacağı sorulmuştur!

12.4. Özet

Ardışıl lojik devre tasarımı için 3 temel yaklaşım şekli vardır denilebilir; ilki, kanonik yaklaşım/tasarım olarak adlandırılır ve bu yaklaşım şekli ardışıl devrelerin mimarisini ve davranışını göstermek için de kullanılır. Diğer bir yaklaşım şekli modüler olanıdır; bu yaklaşım şeklinde tasarlanmak istenen devrenin davranışı var olan standart ardışıl devrelerin davranışlarıyla karşılaştırılır ve en uygun olanına dayanarak standart ardışıl lojik devrelerle tasarım yapılır. Üçüncü yaklaşım şekliyse, algoritmik olanıdır. Büyükçe ardışıl lojik devrelerin tasarlanmasında algoritmik yaklaşım şekli kullanılmalıdır denilebilir. Kanonik yaklaşımla, daha çok, küçük ölçekli; modüler yaklaşım şekliyle küçük ve orta ölçekli; algoritmik yaklaşımla ise orta ve büyük ölçekli tasarımlar yapılır. Uygulamada, çoğu zaman, kanonik ve modüler yaklaşımlar, algoritmik yaklaşımda gerekli altbirimlerin tasarlanmasında kullanılır.

12.5. Sorular

1. Sayısal bir sistemin denetim birimine ait durum diyagramı aşağıdaki gibi verilmiştir. Şekilden görüldüğü gibi diyagramda 4 adet durum ve 2 adet bağımsız giriş vardır; durumlar d_0, d_1 ve bağımsız girişlerde x, y ile gösterilmektedir. Bu denetim birimini istediğiniz türden flip-flop kullanarak tasarlayınız.

2. Herbiri 4 bit olan iki işaretsiz sayının farkının mutlak değerini bulan bir ardışıl devre algoritmik yaklaşıma göre tasarlanmak istenmektedir. Devre başlama işareti alana kadar hiçbir işlem yapmamaktadır; başlama işaretiyle birlikte sayılar veri altyapısı içerisinde bulunan A ve B sayıcılarına yüklenmektedir. Daha sonra, her saat işaretinde (senkronlama anında), A ve B sayıcılarının içerikleri bir eksiltilmektedir; bu işlem, sayıcılardan birinin içeriği sıfır olana kadar devam etmektedir. Sayıcılardan birinin içeriği sıfır olursa diğer sayıcının içeriği farkın mutlak değeri olarak elde edilmiş olunacaktır. Bu aşamada sonuç C saklayıcısına yüklenecek ve başlangıç durumuna dönülecektir.

   a) Devrenin Veri Altyapısını belirleyiniz ve çiziniz; A, B sayıcıları ve C saklayıcısı dışında gereksinim duyulan kapı, seçici vs. gibi birimlerden istenilen kadar kullanılabilir.

   b) Yukarıda tanımlanan adımları a şıkkında belirlenen veri altyapısı üzerinde kotaracak kontrol sistemini tasarlayınız ve şemasını çiziniz. Kontrol sistemi tasarımında saklayıcı+bellek yaklaşımı veya istenilen herhangi bir yaklaşım kullanılabilir.

3. JK flip-flop'undan oluşturulmuş olan 4 bitlik bir saklayıcı için aşağıdaki soruları yanıtlayınız.

a) Girişlerindeki değeri yüklemesi için herbir flip-flop'un J ve K girişlerine hangi işaretler uygulanmalıdır?

b) İçeriği değişmeden saklı tutmak için herbir flip-flop'un J ve K girişlerine hangi işaretler uygulanmalıdır?

   c) İçeriğin bir artması için herbir flip-flop'un J ve K girişlerine hangi işaretler uygulanmalıdır?

d) İçeriğin bire tümlenmesi için herbir flip-flop'un J ve K girişlerine hangi işaretler uygulanmalıdır?

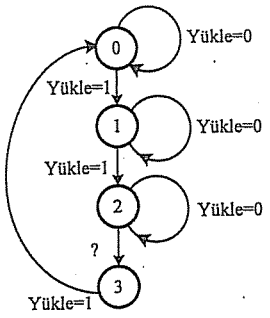
e) Yukarıdaki saklayıcıyı ve gerektiği kadar kapı, seçici kullanarak aşağıda tanımlanan işlemleri yapan ardışıl devreyi tasarlayınız ve çiziniz. Devrenin iki tane kontrol girişi (S_0, S_1) vardır ve bu girişlerin aldığı değere göre aşağıdaki işlemlerden biri yapılmaktadır:

S_1	S_1	İşlev
0	0	yükle
0	1	saklı tut
1	0	1 arttır
1	0	1'e tümle

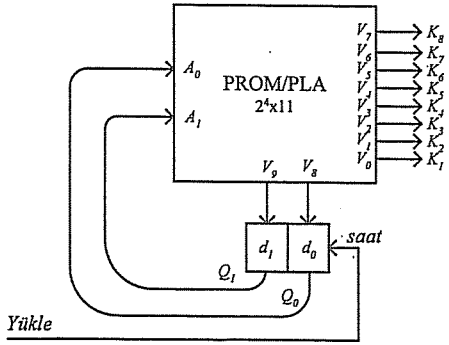
4. Otogardan şehir merkezine 10 dakikada bir seferler yapılmaktadır. Hizmet aracı 15 kişilik olup ya süresi dolmadığı halde dolarsa ya da süresi dolduğunda en az 8 yolcusu varsa kalkmaktadır; aksi durumda kalkmamaktadır. Aracın içerisinde 4 bitlik sayıcı olup her binen yolcuda içeriğini bir arttırmaktadır; bir de, 10 dakikada bir sürenin dolduğunu işaret eden bir saat/zamanlayıcı birim vardır. Saat biriminin çıkışı her 10 dakikada bir darbe üretmektedir. Bu koşullar altında kalkışa hazır işareti üretecek lojik devreyi tasarlayınız ve şemasını çiziniz.

5. Ard arda bit düzeyinde gelen 7 bitlik veri paketi içerisinde 1'lerin tek sayıda olup olmadığını belirleyen bir ardışıl devre tasarlanmak istenmektedir. Devre 1 girişli 1 çıkışlı olup 7 bit içerisinde 1'ler tek sayıda çıkışı 1, aksi durumda 0 yapması beklenmektedir. Devreyi kanonik yaklaşım ve modüler yaklaşım yöntemleriyle tasarlayınız.

6. Örnek-12.4 de veri altyapısı verilen ve durum diyagramında 5 durum olan işlemcinin durum diyagramı aşağıdaki gibi 4 duruma indirgenirse, kontrol sisteminin lojik şemasının çiziniz ve bellek içerisine yazılacak bilgileri belirleyiniz. Yükle ucu yine doğrudan belleğe uygulanmaktadır.



7. Bir yukarıdaki soruyu, 6ncı soruyu, *Yükle* girişinin aşağıdaki şekildeki gibi saat işareti olarak kullanılması durumunda bellek içerisine yazılacak bilgileri belirleyiniz. İstenen işin yapılabilmesi için aşağıdaki şemaya kapı vs. gibi eleman eklenmesi gerekir mi?



8. Herbir gözünde bir harf olan bir ROM 128x8 boyundadır; dolayısıyla içerisinde 128 tane karakterden oluşan bir mesaj vardır. Öyle bir ardışıl devre tasarlayınız ki, ROM içerisindeki karakterleri ard arda ROM'un çıkışlarında üretsin ve mesaj tamamlanınca *tur* adlı kontrol çıkışını 1 yapsın ve yaptığı işlemi tekrarlamak için başa dönsün.

9. Bir önceki soruda mesajın 128 karakterden oluştuğu varsayılmıştı; aynı soruyu mesajın 128 karakterden oluştuğu veya mesaj 128 karakterden daha az ise 11111111 karakteriyle sonlandığı varsayılarak devreyi yeniden tasarlayınız. Dolayısıyla 11111111 karakteri mesaj içerisinde bulunmayacaktır.

10. Temeli RAM kullanılmasına dayanan bir mesaj gösterme devresi tasarlanmak istenmektedir. Devre temel olarak bir klavye, mesaj saklanması için RAM, mesajın gösterilmesi için alfasayısal gösterge ve veri alt yapısında saklayıcı, sayıcı, flip-flop gibi birimlere sahiptir. Mesaj sistemini, mesajı hem saklayacak (yükleme modu) hem de gösterecektir (gösterme modu). Klavyenin ve göstergenin giriş çıkış uçlarını belirleyerek devreyi algoritmik yaklaşıma dayanarak tasarlayınız; veri altyapısını belirleyiniz ve kontrol sistemini tasarlayınız. *Bu sorunun uygulanabilir gerçekçi çözümü için [TOROS ve ÇÖLKESEN-1991] kaynağı incelenebilir.*

13.

Benzetim Ortamında Lojik Devre Tasarımı

Benzetim ortamları, devreleri fiziksel olarak gerçekleştirmeden bilgisayar ortamında tasarlamak ve fonksiyonel davranışını sınamak amacıyla kullanılır. Tasarımcılar, geliştirdikleri sistem veya algoritmaların davranışlarını görmek ve farklı giriş parametrelerine göre sistemin nasıl davranacağını izlemek amacıyla benzetim ortamları kullanırlar. Bu amaçla, hemen hemen tüm disiplinler için birçok yardımcı araç geliştirilmiştir. Örneğin, *CADENCE Design System*, *multiSIM*, *MATLAB / Simulink* hemen akla gelen benzetim araçlarıdır. Tasarımcı bu yardımcı araçların sunduğu ortam aracılığıyla tasarımını fiziksel olarak gerçekleştirmeden çalışıp çalışmadığını sınayabilir. Bu bölümde orta ölçekli lojik devre tasarımında benzetim ortamı olarak kullanılan *multiSIM*[1] tanıtılmış ve kullanımı aşağıdaki başlıklar altında ele alınmıştır:

13.1. multiSIM Programının Özellikleri
- multiSIM Kullanıcı Arayüzü
  - *Tasarım Araç Çubuğu*
  - *Kullanıcı Arayüzünü Özelleştirme*
  - *Kullanıcı Tercihleri*
- Benzetim Ortamı Arayüzü

13.2. Lojik Devrelerin Benzetimi
13.3. Örnek Benzetim Çalışmaları
13.4. Şemadan Baskılı Devreye Geçiş
13.5. Özet
13.6. Sorular

Karmaşık bir lojik devrenin daha gerçekleştirilmeden önce *CADENCE Design System* gibi bir yardımcı araçla; tasarım öncesi deneysel gerçekleştirilmesi zor bir kontrol sisteminin *Matlab/Simulink* gibi bir yardımcı araçla benzetimi yapılarak sınanması, çoğu zaman tutulan bir yoldur. *CADENCE Design System* lojik devrelerin benzetimi yapılması için güçlü bir yardımcı araçtır ve bu alanda yoğun olarak kullanılır; benzer şekilde "Electronics Workbench" şirketi tarafından geliştirilen

[1] "Electronics Workbench" firmasının geliştirmiş olduğu *multiSIM* ve *Ultiboard* yazılımlarının Türkiye Distribütörü "ERA Bilgi Sistemleri ve Yayıncılık Ltd.Şti."dir ve bu kitabın okumakta olduğunuz onüçüncü bölümü ERA'nın katkılarıyla *multiSIM* benzetim programına ayrılmıştır.

multiSIM benzetim programı da çok geniş Eleman veritabanı, şematik çizim arabirimi, *SPICE/VHDL* benzetim arabirimleri, *FPGA/CPLD* sentezi ile RF yetenekleri olan ve *Ultiboard* gibi PCB tasarım programlarına doğrudan transfer imkanı yaratabilen çok gelişmiş bir devre benzetim programıdır ve yoğun tasarım ve sentez gerektiren birçok disiplinlerde oldukça yoğun olarak kullanılmaktadır.

Benzetim ortamları, genelde, sanal dünyada yapılıyormuş gibi düşünülebilir; devre aşağıda görüldüğü gibi grafiksel olarak oluşturulur (veya HDL ile tanımlanır) ve olası giriş değerlerine karşı çıkış değerleri üretilerek davranışı gözlenir. Çünkü, gerçekte sistemin veya devrenin kendisi yoktur; giriş parametreleri, modellenmiş sanal bir sisteme veya devreye uygulanır ve davranışı izlenir. Bu tasarımı tamamlanmış bir devrenin çalışmasının kontrol edildiği ilk aşamadır. Eğer devrenin çalışmasında aksilikler ya da tasarımdan dolayı eksiklikler varsa benzetim ortamları bunların görülebilmesi açısından tasarımcı için çok büyük önem taşımaktadır. Özellikle mikroişlemci veya mikrokontrolör tabanlı tasarımların bu tür ortamlarda sınanabilmesi ve baskılı devrelerinin oluşturulabilmesi için gerekli PCB (*Printed Circuit Board*) transferinin yapılabilmesi tasarımcı açısından çok büyük bir şans oluşturmaktadır. Ayrıca, benzetim sonuçları *MathCAD* ve Excel formatına dönüştürülebilmektedir. Bu da verilerin farklı ortamlarda yorumlanabilmesine imkan sağlamaktadır.

13.1 multiSIM Programının Özellikleri

multiSIM, aşağıdaki yeteneklere sahip, kullanımı kolay, grafik arabirimi olan tam anlamıyla bir sistem tasarım ve benzetim programıdır.

- Geniş Eleman veritabanı
- Şematik çizim ortamı
- Analog/dijital SPICE benzetimi
- VHDL/Verilog tasarım ve benzetimi
- FPGA/CPLD sentezi
- RF devreleri üzerine işlemler
- Şematik çizimi PCB programlarına transfer etme.

Özellikle lojik devre tasarımı açısından değerlendirildiği zaman, multiSIM'in bu iş için biçilmiş kaftan olduğunu söylemek yanlış olmaz. TTL ve CMOS ailelerini desteklemesi ve her iki aileye ait çok büyük birer veritabanını barındırmasının yanı sıra kullanıcıya yeni bir tümdevrenin özelliklerini tanıtmasına da imkan vermektedir. Bütün bunların yanında kullanıcının benzetimleri ideal devreler için ya da gerçek hayattaki pratik devreler için gerçekleştirebilmesine de imkan sağlamaktadır.

multiSIM Kullanıcı Arayüzü

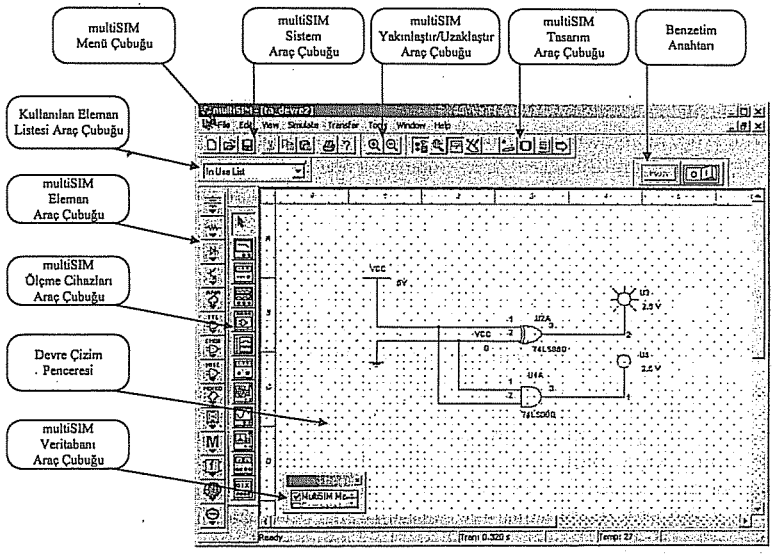
multiSIM programı, kullanıcılarına kullanımı oldukça kolay bir arayüz sunmaktadır. Genel yapısı Şekil-13.1 de verilen bu arayüzde, başlık çubuğu, menüler, durum çubuğu gibi geleneksel Windows programı özelliklerinin yanı sıra kullanıcıya devre çizimi ve benzetimi için gerekli olan bütün araç çubukları da yer almaktadır. multiSIM kullanıcı arayüzünde bulunan menü ve araç çubukları;

- **multiSIM Menü Çubuğu** ***(Menu Bar)***

multiSIM benzetim programı Windows'un klasik menü yapısına benzer bir menü çuğuna sahiptir. Bu menü'de sırasıyla, File (Dosya), Edit (Düzen), View (Görünüm), Simulate (Benzetim), Transfer, Tools (Araçlar), Window (Pencere), Help (Yardım) menüleri yer almaktadır.

- **multiSIM Sistem Araç Çubuğu** ***(System Toolbar)***

Yeni Dosya oluşturma, Mevcut dosyayı açma,saklama, kes, kopyala, yapıştır, yazdır, yardım, yakınlaştır ve uzaklaştır işlemlerini yerine getirmeye yarayan kısa yol düğmelerini barındıran araç çubuğudur. Sistem Araç Çubuğu, *View* menüsünde *Toolbars/System* seçeneği ile aktif hale getirilir.



Şekil-13.1. multiSIM kullanıcı arayüzü.

- **multiSIM Yakınlaştır/Uzaklaştır Araç Çubuğu *(Zoom Toolbar)***

multiSIM devre çizim penceresinde oluşturulan devre şemasını daha yakından ya da kuşbakışı inceleyebilmek için kullanılan araç çubuğu. Yakınlaştır/Uzaklaştır Araç Çubuğu, *View* menüsünde *Toolbars/Zoom* seçeneği ile aktif hale getirilir.

- **multiSIM Tasarım Araç Çubuğu *(Design Toolbar)***

multiSIM'in karmaşık fonksiyonlarına erişim imkanı sağlayan araç çubuğudur. Bu araç çubuğunda yer alan bütün kısayol düğmelerinin işlevleri ileride ayrıntılı olarak açıklanacaktır. Tasarım Araç Çubuğu, *View* menüsünde *Toolbars/Design* seçeneği ile aktif hale getirilir.

- **Kullanılan Eleman Listesi Araç Çubuğu *(In Use List Toolbar)***

Devre çizim alanında mevcut olan Elemanların listesini veren araç çubuğudur. Kullanılan Eleman Listesi Araç Çubuğu, *View* menüsünde *Toolbars/In Use List* seçeneği ile aktif hale getirilir.

- **multiSIM Eleman Araç Çubuğu *(Components Toolbar)***

Her bir kısayol düğmesinin bir Eleman ailesine erişime imkan sağlayan araç çubuğudur. Benzetim öncesi devre kuruluş aşamasında en çok kullanılacak araç çubuğudur. Eleman Araç Çubuğu, *View* menüsünde *Toolbars/Components* seçeneği ile aktif hale getirilir.

- **multiSIM Ölçme Cihazları Araç Çubuğu *(Instruments Toolbar)***

Distorsiyon ve lojik analiz cihazları ile fonksiyon üreteci, sınama sözcüğü üreteci, osiloskop, multimetre, wattmetre gibi her türlü ölçüm cihazını barındıran Araç Çubuğudur. Ölçme Cihazları Araç Çubuğu, *View* menüsünde *Toolbars/Instruments* seçeneği ile aktif hale getirilir. Ölçme Cihazlarına *Simulate* menüsünde *Instruments* seçeneği ile erişmek mümkündür.

- **multiSIM Veritabanı Araç Çubuğu *(Database Toolbar)***

multiSIM'de mevcut veya kullanıcı tarafından oluşturan elemanları içeren veritabanlarına erişim için kullanılan araç çubuğudur. Veritabanı Araç Çubuğu, *View* menüsünde *Toolbars/Database* seçeneği ile aktif hale getirilir.

- **Benzetim Anahtarı *(Simulate Switch)***

Devre çizim penceresinde tasarım işlemi tamamlandıktan sonra benzetimin başlatılmasını veya başlatılmış olan benzetimin sonlandırılmasını sağlayan açma/kapama anahtarıdır. Bu anahtarın yanında yer alan bir düğme ile çalışmakta olan benzetimi duraklatma ve duraklatılmış benzetimi sürdürme işlemleri gerçekleştirilebilir. Benzetim Anahtarı, *View* menüsünde *Show Simulate Switch* ile aktif hale getirilir.

multiSIM Tasarım Araç Çubuğu

Tasarım Araç Çubuğu *(Design Toolbar)*'ın multiSIM'in karmaşık fonksiyonlarına erişim imkanı sağlayan bir araç çubuğu olduğu daha önce açıklanmıştı. Karmaşık fonksiyonlara erişim, tasarım araç çubuğunda bulunan kısa yol düğmeleriyle sağlanmaktadır. Bu kısa yol düğmelerinin detaylı açıklamaları aşağıda verilmiştir.

Eleman *(Components)* Düğmesi

Eleman Araç Çubuğunun *(Components Toolbar)* multiSIM penceresinde gösterilmesini ya da gösterilmemesini sağlar. multiSIM eleman *(Components)* araç çubuğunu görebilmek için multiSIM Veritabanı *(Database)* araç çubuğunda MultiSIM Master veritabanının seçilmiş olması gerekmektedir.

Eleman Editörü *(Component Editing)* Düğmesi

Yeni bir Eleman oluşturmak, mevcut Elemanlar üzerinde değişiklik yapmak, bir komponenti kopyalamak ya da silmek, ayrıca mevcut veritabanlarından herhangi birini seçip mevcut elemanları incelemek gibi işlemleri yapmaya yarayan erişim düğmesidir.

Ölçme Cihazları *(Instruments)* Düğmesi

Distorsiyon ve lojik analiz cihazları ile fonksiyon üreteci, sınama sözcüğü üreteci, osiloskop, multimetre, wattmetre gibi her türlü ölçüm cihazını barındıran Ölçme Cihazları Araç Çubuğunu aktif ya da pasif hale getiren düğmedir.

Benzetim *(Simulation)* Düğmesi

Benzetim düğmesi, devre şeması çizilerek hazır hale getirilmiş benzetimi başlatma, sonlandırma, duraklatma ve sürdürme işlerini yerine getiren Benzetim Anahtarı ile aynıdır. Kısaca, devre şeması tamamlandıktan sonra kullanıcı benzetimi buradan da başlatabilir, çalışmakta olan benzetimi sonlandırabilir ya da duraklatabilir. Ayrıca duraklatılmış benzetimi sürdürebilir.

Analiz *(Analyses)* Düğmesi

Kurulu devre düzeneği üzerinde, DC çalışma noktası, AC, geçiş *(transient)*, Fourier, Gürültü, Distorsiyon, Duyarlık ve benzeri analiz işlemlerinin yapılabilmesine imkan sağlayan düğmedir. Bununla birlikte, kullanıcının özel analizlerini yapabilmesi için Kullanıcı Tanımlı Analiz bölümü de yer almaktadır. Analiz yöntemlerine *Simulate* menüsünde *Analyses* seçeneği ile erişmek mümkündür.

İleri işlemciler *(Post Processor)* Düğmesi

Benzetim sonuçları üzerinde çeşitli ileri işlemlerin yapılmasına imkan sağlayan bölümdür. Benzetim sonucu eğrisi üzerinde adım adım izleme yaparak sonucu yo-

rumlamak, hatta çeşitli analiz sonuçlarını da ekleyerek tam kapsamlı yorumlarda bulunmak mümkündür. İleri İşlemcilere *Simulate* menüsünde *Postprocess* seçeneği ile erişmek mümkündür.

VHDL/Verilog HDL Düğmesi

VHDL modelleme ile çalışmaya imkan sağlayan erişim düğmesidir. VHDL tasarım bütün multiSIM sürümlerinde mevcut olan bir özellik değildir. Alacağınız sürümde mevcut olup olmadığını mutlaka sorunuz. Sonuçta, multiSIM programıyla VHDL benzetimi veya sentezi ile Verilog HDL benzetimi bu düğme yardımıyla gerçekleştirilebilir. VHDL ve Verilog benzetimlerine *Simulate* menüsünde *VHDL Simulations* ve *Verilog HDL Simulations* seçeneği ile VHDL sentezine ise *Transfer* menüsünde *VHDL Synthesis* seçeneği ile erişmek mümkündür.

Raporlar *(Reports)* Düğmesi

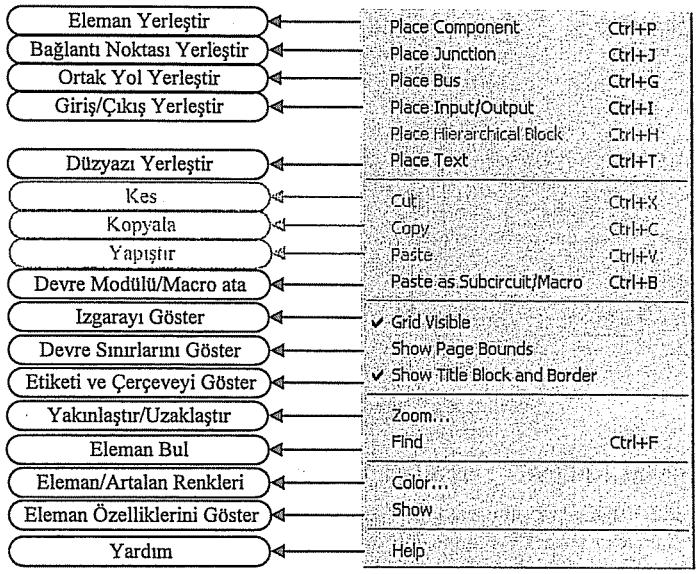
Raporlar düğmesi ile Maliyet Analizi yapılabileceği gibi Eleman Araç Çubuğundan bir eleman seçilirken açılan pencereden Veritabanı Eleman Ailesi Listesi, Eleman Ayrıntılı Raporuna ilişkin bilgiler alınabilir. Ayrıca devrede kullanılan Ölçme Cihazlarının yapısı yazıcıdan çıkış olarak alınabilir.

Transfer *(Transfer)* Düğmesi

Bu düğme vasıtasıyla, oluşturulan devre şemasından, Electronics Workbench şirketinin ürettiği Ultiboard veya diğer şirketlere ait baskılı devre (PCB) hazırlama programlarına doğrudan transfer yapılabilir. Ayrıca benzetim sonuçlarının yorumlanabilmesi ve diğer programlar tarafından kullanılabilmesi için MathCad ve Excel programları formatında doğrudan veri transferi de mümkündür. Transfer işlemlerine *Transfer* menüsünden ulaşmak mümkündür.

multiSIM Kullanıcı Arayüzünü Özelleştirme

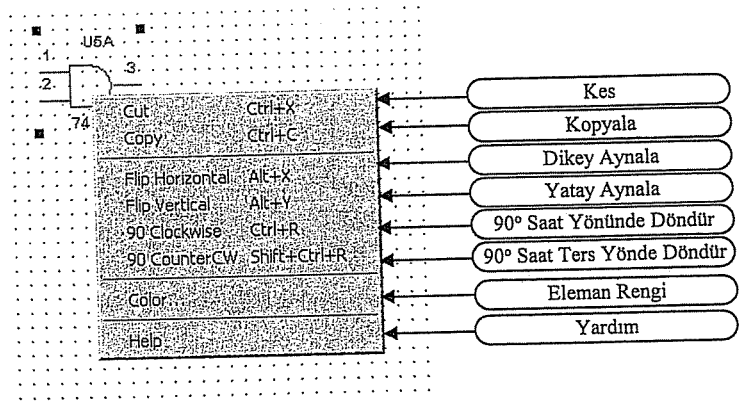
multiSIM, kullanıcılarına araç çubuklarını, devre renklerini, sayfa yapısını, yakınlaştır/uzaklaştır ölçeklerini, otomatik saklama ayarlarını, kullanılan devre standartlarını (ANSI veya DIN) ve yazıcı ayarlarını değiştirmelerine imkan sağlaması açısından kullanıcı dostu bir benzetim programıdır. Ayrıca bütün bu işlemleri her bir devre için ayrı ayrı yapmak da mümkündür. Bunun yanı sıra eleman, bağlantı noktası, ortak yol, giriş/çıkış, yazı ekleme işlemleri gibi bir çok işlemi devre çizim penceresi içinde mouse sağ tuşuna basılarak açılacak olan menüden gerçekleştirilebilir. Bu işlem yapıldığında açılan pencerenin genel yapısı ve açıklamaları Şekil-13.2 de verilmiştir.



Şekil-13.2. multiSIM devre çizim ayarları menü penceresi.

Devre çizim penceresi içinde yerleştirilmiş olan bir elemanı taşımak, kopyalamak, sağa sola döndürme ve aynalama işlemleri ile eleman rengini değiştirmek mümkündür. Bu işlemler, elemanın üzerindeyken farenin sağ tuşuna basılarak açılacak olan menüden gerçekleştirilebilir. Açılan bu menü penceresinin genel yapısı ve açıklamaları Şekil-13.3 de verilmiştir.

MultiSIM Windows tabanlı bir uygulama programı olduğu için bloklama, kopyalama, taşıma ve silme işlemleri genel windows kuralları çerçevesinde gerçekleştirilebilmektedir. Özellikle Windows arayüzüne alışkın kullanıcılar için bu benzerlik çok büyük kolaylıklar sağlamaktadır.

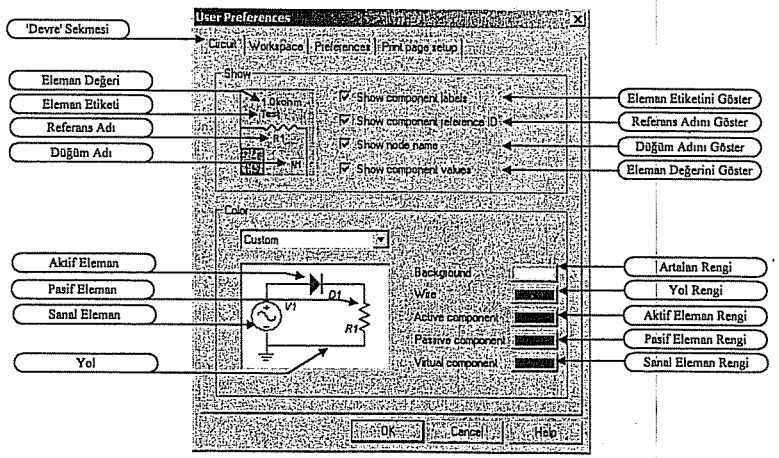


Şekil-13.3. multiSIM eleman menü penceresi.

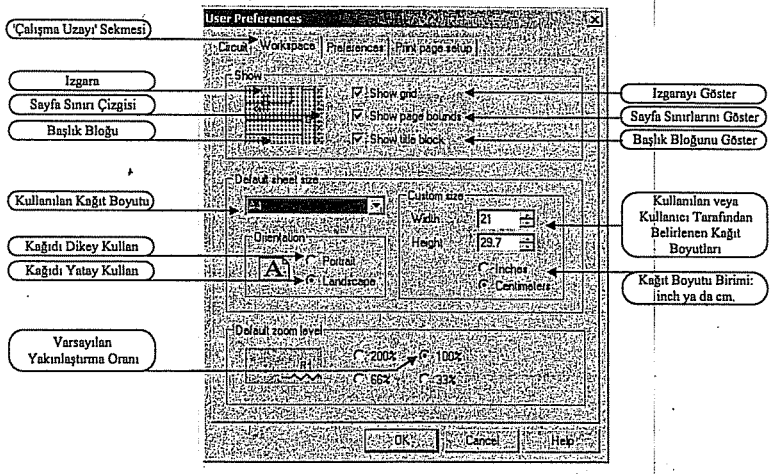
multiSIM Kullanıcı Tercihlerinin Tanımlanması

Benzetim çalışmaları yapılacak devreye ilişkin devre şemasını oluşturma işlemlerine başlamadan önce, devrenin sahip olacağı renk, etiket adı, eleman değeri, referans adı gibi özelliklerle, çalışma alanında ızgaranın gösterilmesi, kağıt boyutunun belirlenmesi ve yazdırma özelliklerini belirlenmesi gibi işlemlerin gerçekleştirilmesi gerekmektedir. Kullanıcı eğer mevcut yapıdan memnun ise bu özellikler üzerinde bir değişiklik yapma ihtiyacı duymaz ve *File* menüsünden *New* seçeneği ile yeni tasarıma başlar. Eğer kullanıcı yukarıda sayılan özellikleri kendi tercihleri doğrultusunda yeniden düzenlemek isterse bu işlemi *Edit* Menüsü *User Preferences* seçeneği ile belirlemeli ve bu değişiklikleri yürürlüğe koymak için *File* Menüsü *New* seçeneği ile yeni bir multiSIM sayfası açmalıdır, yeni sayfa açılmadan yapılan değişiklikler yürürlüğe girmez. Kullanıcı referansları yapı itibarıyla; *Circuit* (Devre), *Workspace* (Çalışma Uzayı), *Preferences* (Tercihler), *Print page setup* (Sayfa Yazdırma Ayarı) şeklinde dört bölüme ayrılmıştır. *Circuit* (Devre) adı verilen ilk bölümde, devre elemanlarının gösterileceği renklerle, etiket adı, eleman değeri, referans adı gibi özellikler değiştirilebilmektedir.

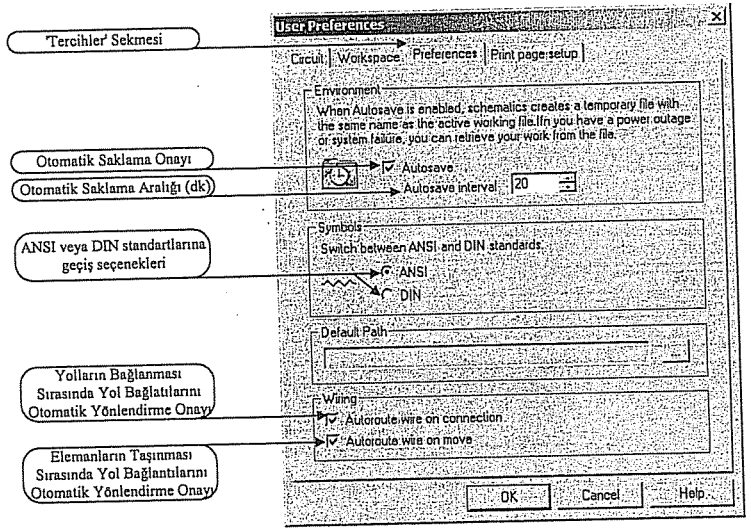
Edit menüsünden *User Preferences* seçeneği ile kullanıcı referansları tanımlanmak için açıldığında *Circuit* (Devre) sekmesini ilk sekme olduğu için otomatik olarak açılır. Adından da anlaşılacağı gibi devre özellikleri üzerinde değişikliklerin yapılabildiği bu bölüme ilişkin pencere yapısı Şekil-13.4 de gösterilmiştir.



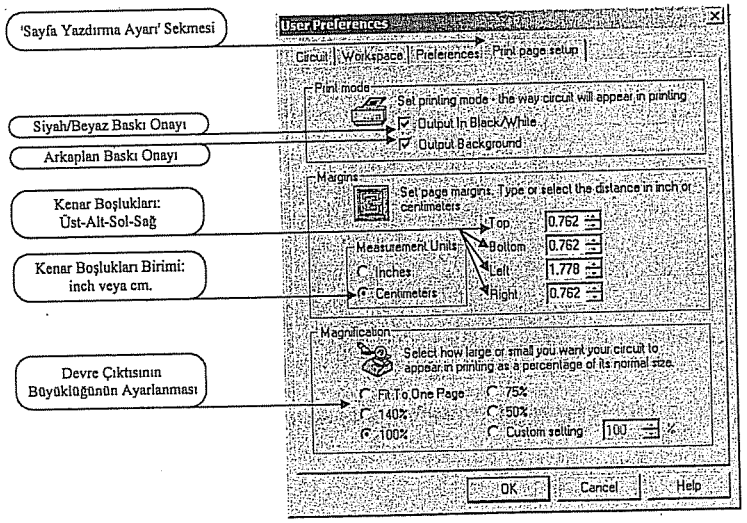
Şekil-13.4. User Preferences/Circuit (devre) sekmesi.



Şekil-13.5. User Preferences/Workspace (Çalışma Uzayı) sekmesi.



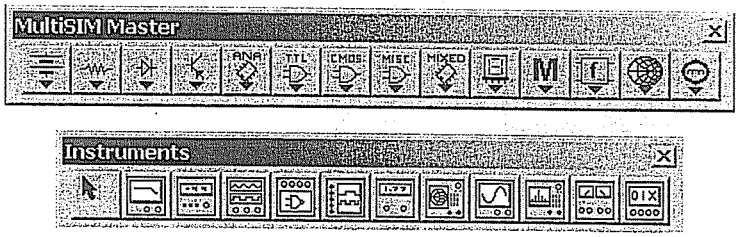
Şekil-13.6. User Preferences/Preferences (Tercihler) sekmesi.



Şekil-13.7. User Preferences/Print page setup (Sayfa Yazdırma Ayarı) sekmesi.

multiSIM Benzetim Ortamı Arayüzü

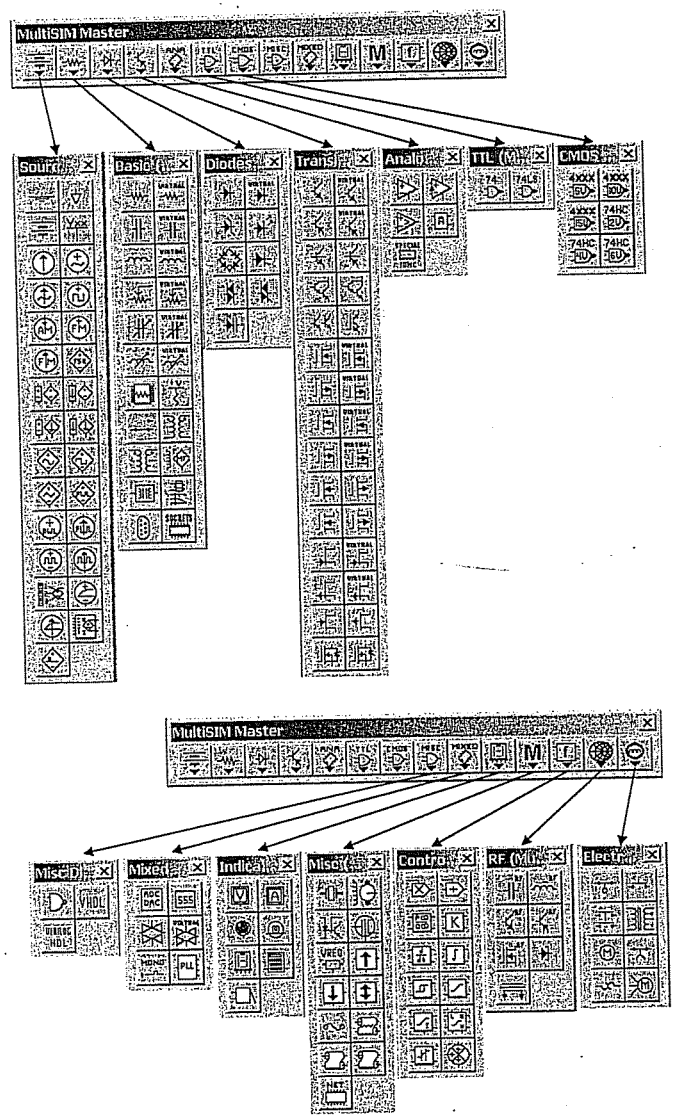
Bir devreye ilişkin benzetim sonuçlarını görebilmek için öncelikle o devreye ilişkin elemanların doğru olarak seçilmesi ve gerekli besleme devreleri ile gerekli görüntüleme/ölçme cihazlarının doğru olarak yerleştirilmesi gerekmektedir. Bu amaçla **multiSIM Eleman** *(Components)* **Araç Çubuğu** ile **multiSIM Ölçme Cihazları** *(Instruments)* **Araç Çubuğu** araç çubuklarından yararlanılarak devre şemasını benzetim ortamı arayüzünde oluşturmak gerekir. multiSIM eleman *(Components)* araç çubuğunu görebilmek için multiSIM Veritabanı *(Database)* araç çubuğunda MultiSIM *Master* seçeneğinin işaretlenmiş olması gerekmektedir. Aksi taktirde *View* menüsünden *components* araç çubuğu seçilirse bu araç çubuğu aktif olmayacaktır. multiSIM Eleman *(MultSIM Master Components)* ve Ölçme Cihazları *(Instruments)* araç çubuklarının genel yapıları Şekil-13.8 de verilmiştir.



Şekil-13.8. multiSIM Eleman (Master Components) ve Ölçme Cihazları (Instruments) araç çubuklarının genel yapısı.

multiSIM ortamında bir devre şemasını çizmek için MultiSIM *Master Components* araç çubuğundan yararlanılır. Bu araç çubuğu tek başına bir benzetim için gerekli bütün bileşenleri içinde barındırır. Şekil-13.9 da da gösterildiği gibi bu bileşenler sırasıyla aşağıdaki gibidir:

Kaynaklar (Sources)	Çeşitli Sayısal (Misc Digital)
Temel Elemanlar (Basic)	Çeşitli (Mixed)
Diyotlar (Diodes)	Göstergeler (Indicators)
Transistörler (Transistors)	Çeşitli (Misc)
Analog (Analog)	Kontrol (Controls)
TTL (TTL)	Radyo Frekansı (RF)
CMOS (CMOS)	Elektro-mekanik (Electro-mechanical)



Şekil-13.9. multiSIM Eleman (Master Components) araç çubuklarının alt bileşenleri.

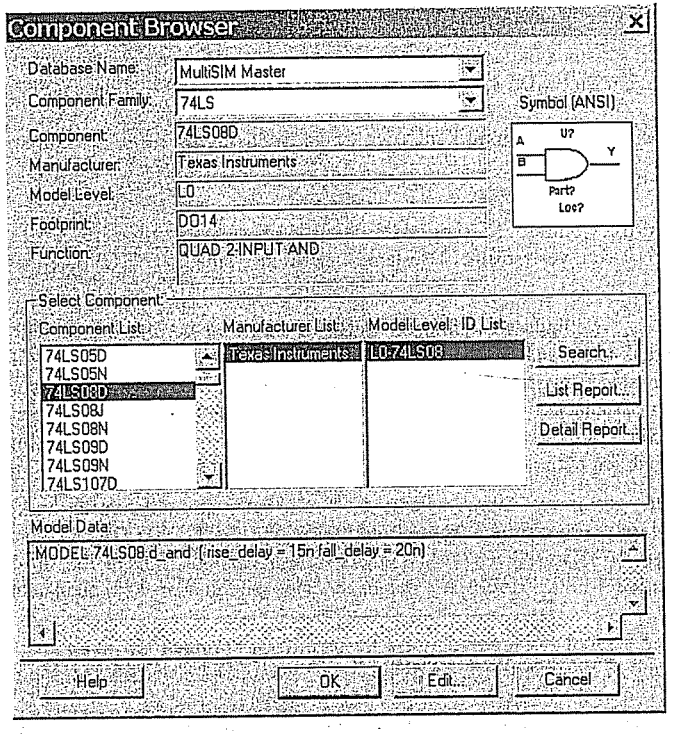
13.2. Lojik Devrelerin Benzetimi

Burada Ayrıt 7.4.1 de Şekil-7.6 da ayrıntılı olarak incelenen yarı toplayıcı devrenin benzetimi yapılmaya çalışılmıştır. Bunun için bir tane DVEYA bir tane de VE kapısına ihtiyaç vardır. Ayrıca besleme, toprak gerilim değerlerine ve ayrıca çıkışların lojik seviyelerini göstermek için 2 adet lojik *probe*'a ihtiyaç vardır. Benzetim programlarında devreyi gerçeklemeden önce malzeme listesini hazırlamak devrenin hızlı kurulmasında çok büyük bir avantaj oluşturur.

Kuruluş sırasında ilk tercih edilecek başlangıç adımı besleme kaynağının seçimidir. Bunun için MultiSIM *Master Components* araç çubuğunun ilk bileşeni olan *Source* altındaki toprak gerilimini ifade eden simge fare sol tuşu ile seçilir ve toprak bağlantısını gösteren simgesi istenen yere fare sol tuşuna bir kez basılarak yerleştirilir. Sonra simgesiyle gösterilen besleme gerilimi benzer şekilde istenen yere yerleştirilir. Besleme gerilimi simgesi TTL eleman için VCC 5V şeklinde gösterilmiştir. Herhangi farklı bir gerilim değeri devreye uygulanmak istenirse bu sembol üzerinde fare sol tuşuna iki kez basılarak açılan pencerede istenen gerilim değeri yazılır ve OK seçeneği ile uygulamaya sokulur. Burada tasarlanacak devreye göre lojik kapı elemanları, analog elemanlar ve MultiSIM kütüphanelerinde bulunan bütün elemanlar ve kullanıcı tarafından tanımlanabilen bütün elemanları kullanmak mümkündür.

Besleme gerilimi ve toprak uçlarını oluşturduktan sonra kullanılacak lojik kapı elemanlarını yerleştirmek uygun bir adım olacaktır. Bu amaçla DVEYA kapısı için 74LS86, VE kapısı için 74LS08 tümdevrelerinden birer kapıya ihtiyaç duyulur. Bunun için MultiSIM *Master Components* araç çubuğundan TTL elemanlarını ifade eden sembol fare sol tuşu ile seçilir. Bu durumda bir ikinci araç çubuğu daha açılarak standart TTL ya da *Low Power Shottky* TTL serilerinden birinin seçilmesi istenir. LS serisi TTL tercih edilirse simgesi fare sol tuşu ile seçilir. Bu durumda Şekil-13.10 da gösterilen Eleman Tarayıcı *(Component Browser)* adı verilen ve LS serisi TTL tümdevre elemanlarının bir listesini, sembolik gösterilimini, giriş/çıkış gecikme sürelerini, tümdevrenin kaç kapı içerdiği ve kapıların kaç giriş içerdiği gibi gerekli bilgileri barındıran bir pencere açılır. Kullanıcı buradan istediği elemana kolaylıkla ulaşabilir. Bu alanda ilk olarak 74LS08 seçilir ve devre çizim penceresine yerleştirilir ve hemen sonra da 74LS86 yerleştirilir. Bundan sonra lojik probe yerleştirmek için MultiSIM *Master Components* araç çubuğundan Gösterge *(Indicators)* elemanlarını ifade eden sembol fare sol tuşu ile seçilir. Bu du-

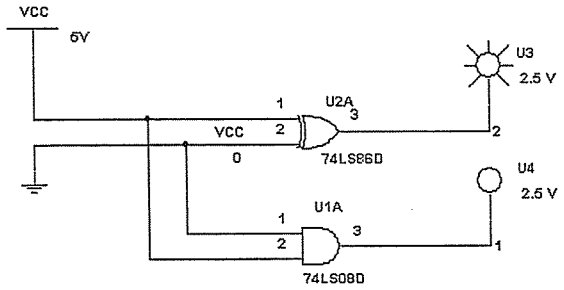
rumda açılan ikinci araç çubuğundan lojik probe'u ifade eden simge fare sol tuşu ile seçilir ve lojik probe bağlantısını gösteren 2.5 V simgesi istenen yere fare sol tuşuna bir kez basılarak yerleştirilir.



Şekil-13.10. multiSIM eleman tarayıcı (component browser) penceresi.

Bütün bu işlemlerden sonra devre uç bağlantılarının yapılması gerekmektedir. Her elemanın elektriksel bağlantı yapılabilen uçlarından birine fare imleci ile temas edildiğinde fare imleci ✦ şekline dönüşür. Bu durumda fare sol tuşuna basılır ve bu ucun bağlanacağı başka bir ucun ya da herhangi bir yolun üzerinde tekrar fare sol tuşuna basılarak burada bir düğüm noktası oluşturulur ve bağlantı sağlanır. Devre şemasına bakıldığında düğüm noktaları ┼ şeklinde gösterilecektir. Herhangi bir şekilde bu düğüm noktasından başka bir elemanın ucuna, herhangi bir yola ya da herhangi bir

ka bir düğüm noktasına bağlantı kurulabilir. Bütün elemanlara ilişkin tüm bağlantılar tamamlandıktan sonra devre şeması Şekil-13.11 de olduğu gibi elde edilir.



Şekil-13.11. Benzetimi yapılacak yarı toplayıcı devre şeması.

Bu aşamadan sonra benzetim çalıştırmak gerekmektedir. Bu işlem;

- Tasarım (*Design*) Araç Çubuğundan Benzetim (*Simulation*) düğmesinden,
- Benzetim Anahtarı (*Simulate Switch*) aç/kapa (*on/off*)düğmesinden,
- Simulate menüsü, Run/Stop seçeneğinden.

gerçekleştirilebilir. Benzetimi durdurma işlemi başlatma işleminin yapıldığı yerlerden gerçekleştirilebilmektedir.

13.3. Örnek Çalışmalar

Bu ayrıtta bazı devrelerin devre şemaları çizilmiş ve benzetim çalışmaları gerçekleştirilmiştir. Devre şemasının çizilmesi ve devre uçları arasında bağlantıların yapılması ve benzetim çalışmasının gerçeklenmesi Ayrıt 13.4 de açıklandığı gibidir. Burada sırasıyla,

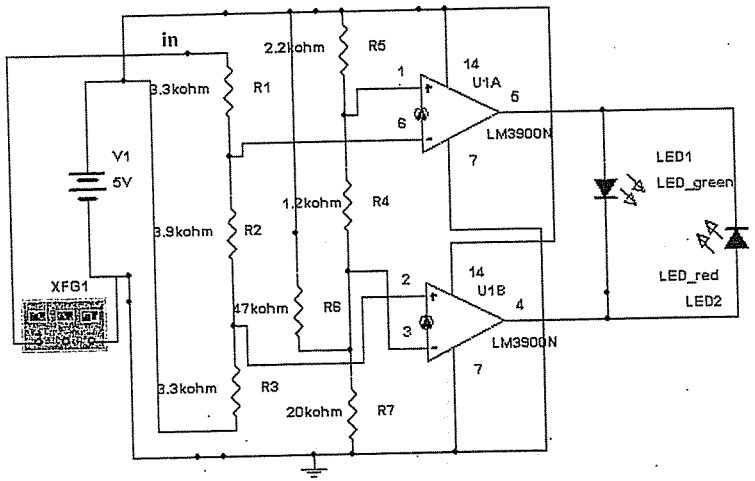
- Lojik Seviye Algılayıcısı *(Logic Probe)*
- Ondalık Sayıcı *(Decade Counter)*
- Ondalık Sayıcı +7-parçalı gösterge *(Decade Counter+7-segment display)*
- 4-bitlik İkili Yukarı Sayıcı *(4-bit Binary Up Counter)*

benzetimleri gerçekleştirilmiştir[2].

[2] Bu kısımlarda desteği için meslektaşımız Sayın Murat Başkan'a teşekkür ederiz.

Lojik Düzey Algılayıcısı (*Logic Monitor*)

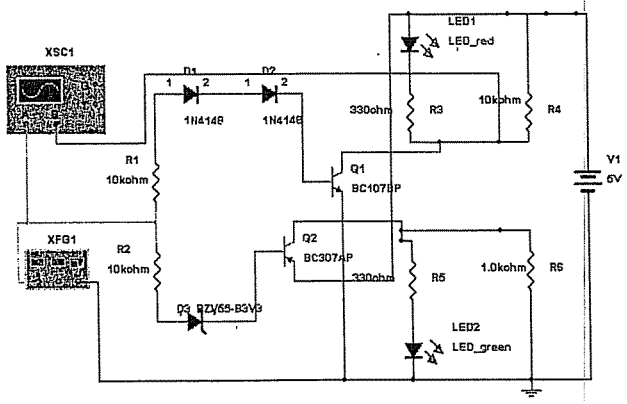
Lojik düzey algılayıcısında, iki adet *norton opamp* kullanılarak girişteki lojik düzeyin değeri belirlenmektedir. Devredeki işlemsel kuvvetlendiriciler (*OpAmp-Operational Amplifier*) birer gerilim karşılaştırıcısı olarak çalışmakta ve devrenin giriş işaretine göre birinin çıkış gerilimi artı olurken diğeri eksi (toprak potansiyeli) olmakta ve çıkıştaki ışık saçan diyotlar (led'ler) parlamaktadırlar. Böylece girişteki lojik düzeyin "0" ya da "1" olduğu belirlenmektedir. Devredeki yeşil led lojik 1, kırmızı olanı ise lojik 0'ı göstermektedir. Devrede ***in*** girişi lojik 1 ve lojik 0'a çekilerek denemeler yapılabilir. Devreye ilişkin lojik şema Şekil-13.12 de verilmiştir.



Şekil-13.12. Lojik Seviye Algılayıcıya ilişkin devre şeması.

Lojik Probe

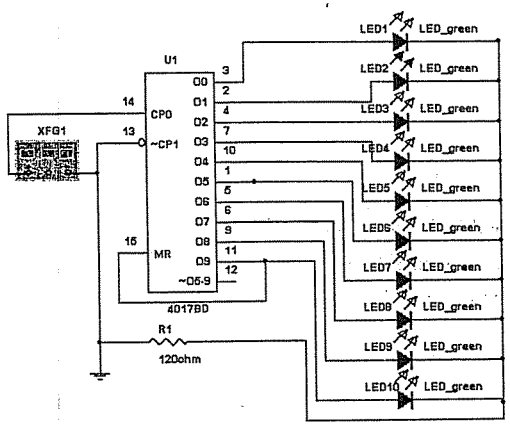
Bu devrede lojik monitör devresine benzemektedir. Burada TTL lojik seviyelerin belirlenmesinde transistör kullanılmıştır. Devrenin girişine 0-2V arasında bir gerilim uygulandığında PNP transistör doyuma giderek yeşil led'i yakar (lojik 0). 2 Voltun üzerinde bir gerilim uygulandığında ise NPN transistör doyuma giderek kırmızı led'i yakar (lojik 1). Devreye ilişkin lojik şema Şekil-13.13 de verilmiştir.



Şekil-13.13. Lojik Probe'a ilişkin devre şeması.

Ondalık Sayıcı-Yürüyen Işık (*Decade Counter*)

4017 Entegresi onlu sayıcı ve kod çözücü işlemlerini gerçekleştirir. Entegre saat darbesine bağımlı olarak birden ona kadar olan çıkışlarını lojik 1 yapar, bu anda diğer çıkışlar lojik 0 dır. Bu devre çıkışlarına led'ler uygun şekilde bağlanarak basit bir yürüyen ışık devresi oluşturulabilir. Bir led aydınlatılıp söndürülürken hemen yanındaki yakılarak sanki ışık ilerliyormuş etkisi yaratılır. Devreye ilişkin lojik şema Şekil-13.14 de verilmiştir.

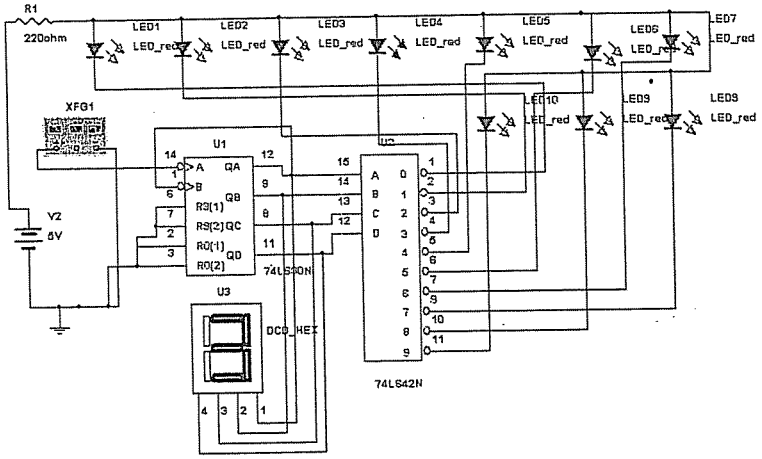


Şekil-13.14. Yürüyen Işık devresine ilişkin benzetim şeması.

4-bit BCD Sayıcı ve 7-parçalı Gösterge (*4-bit BCD Counter*)

Bu devrede 74LS90 entegresi 4 bit BCD sayıcı olarak çalışmakta ve 7441 entegresi ise 4 bit BCD girişi olan ve girişleri çözerek onluk sisteme çevirmektedir. Bu entegreler özel olarak nümerik gösterge tüplerini sürmek üzere yapılmışlardır.

Devre çalıştırıldığında led'ler üzerinden ve 7-parçalı göstergeden çalışması izlenebilir. Devreye ilişkin lojik şema Şekil-13.15 de verilmiştir.



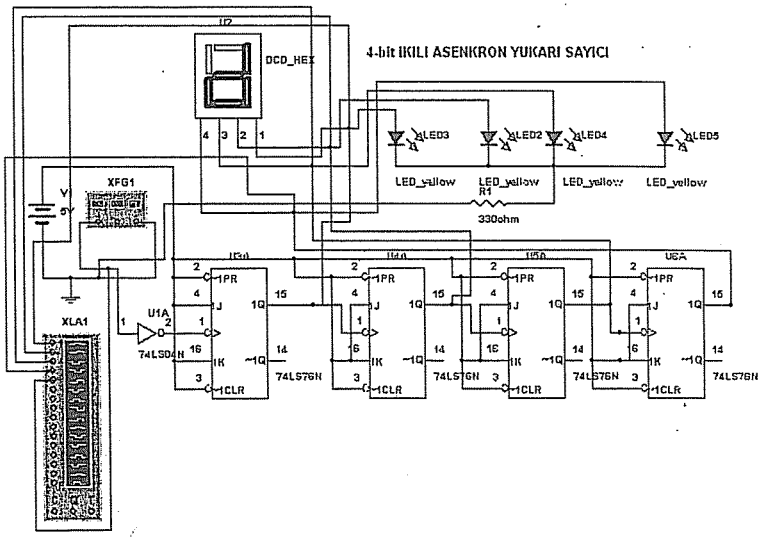
Şekil-13.15. 4-bit BCD Sayıcı ve 7-parçalı gösterge.

4-bitlik İkili Yukarı Sayıcı (*4-bit Up Counter*)

Devre J-K Flip-Floplarıyla gerçekleştirilmiş 4-bitlik ikili asenkron yukarı sayıcı devresidir. Sayma işlemi devredeki 7-parçalı göstergeden (*seven segment display*) izlenebildiği gibi aynı anda çıkıştaki ikili değerler ışık saçan diyot olarak adlandırılan led'ler aracılığıyla gözlenebilir.

Devredeki lojik analizör ile çıkıştaki dalga formu (sayıcılar aynı zamanda bir frekans bölme işlemi yaptığı için) izlenebilir. Devreye ilişkin lojik şema Şekil-13.15 de verilmiştir.

Örnek olarak verilen bütün devreler *Electronics Work Bench* programının anlaşılmasına yönelik olup program ile neler yapabileceğiniz sizin lojik dünyadaki ufkunuza bağlıdır.



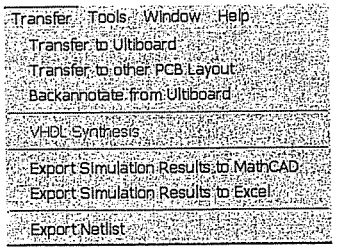
Şekil-13.16. 4-bitlik ikili asenkron yukarı sayıcı devreye ilişkin devre şeması.

13.4. Lojik Devre Şemasından Baskılı Devreye Geçiş

Bir lojik devre tasarımcısının benzetim çalışmalarını da sonuçlandırdıktan sonra yapması gereken son işlem tasarladığı devreyi, baskılı devre olarak üretmektir. Tasarım, maliyet hesabı, karmaşıklık hesabı ve devrenin çalışmasının sınandığı benzetim çalışmaları, tasarımcının takip ettiği uzun bir yol olmakla birlikte son adım atılmadan tasarım tamamlanmış sayılmaz. İşte bu ayrıtta tasarımcının son adımı; baskılı devre tasarımına yer verilmiştir.

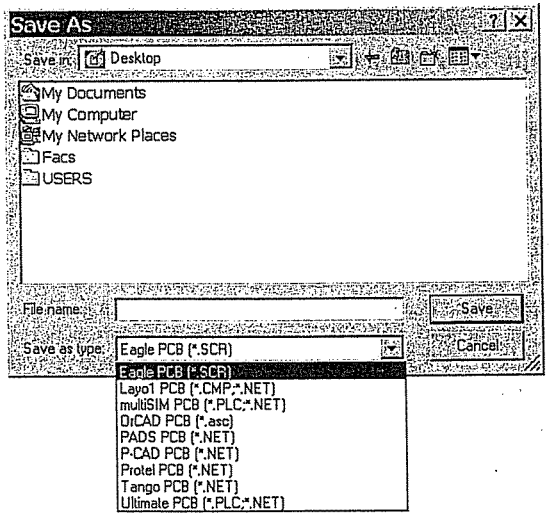
Tasarımcı, benzetim programı olarak *multiSIM* kullanıyor ise baskılı devreye geçiş aşamasında programın özelliğinden dolayı çok büyük avantajlarla karşılaşırlar. Bu avantajların en önemlisi *multiSIM*'in başta kendi baskılı devre programı olan *Ultiboard* olmak üzere bir çok baskılı devre tasarım programına uygun transfer çıkışları vermesidir.

Benzetim çalışması durdurulduktan sonra *Transfer* menüsünde yer alan *Transfer to Ultiboard* ve *Transfer to other PCB Layout* seçenekleri baskılı devreye geçişin ilk adımıdır. Bu yapı Şekil-13.17 de verilmiştir.



Şekil-13.17. Transfer menüsü ve alt menüler.

Transfer menüsündeki *Transfer to Ultiboard* aslında *Transfer to other PCB Layout* seçeneğinin bir alt kümesidir. Bu yüzden burada *Transfer to other PCB Layout* seçeneğinin fare sol tuşu ile seçilmesi durumu tartışılmıştır. Bu durumda otomatik olarak bir Farklı Kaydet (*Save As*) penceresi açılır ve Farklı Kaydet Türleri içinde Şekil-13.18 de de belirtildiği gibi özellikle *Eagle*, *Layo1*, *multiSIM* (*UltiBoard*), *OrCAD*, *PADS*, *P-CAD*, *Protel*, *Tango* ve *Ultimate* programlarına doğrudan çıkış üretmek mümkün olmaktadır. Bir dosya adı verip, uygun baskılı devre tasarım programına ilişkin çıkış formatını seçtikten sonra Kaydet seçeneği ile uygun dosya üretilir.



Şekil-13.18. Transfer menüsünden *Transfer to other PCB Layout* seçeneği.

13.5. Özet

Günlük hayatta kullanılmakta ve çok basit bir işlevi yerine getirmekte olan herhangi bir cihaz, aslında çeşitli tasarım aşamalarından geçtikten sonra yaşantımız içindeki yerini alabilmiştir. Tasarımcılar, önce hayatımızı kolaylaştıracak bir cihazı düşünmek hatta hayal etmek durumundadırlar. Bir süre sonra, tabi teknolojinin de yardımıyla o hayalin gerçekleşmesi için uygun ortam oluşmaya başlar. Bu noktada tasarımcı hayalini gerçekleştirebilmek için uygun teknolojik gelişmeleri kullanarak tasarım yapar. Bu sırada benzetim ortamları ile tasarımlarını sınayarak doğru yolda olup olmadıklarını öğrenmeye çalışırlar. Bu yardımcı araçlar vasıtasıyla tasarımcı hayalindeki icadın ilk deneme çalışmalarını yapar. Bunu takip eden adım, devrenin hayata geçirilmesidir. Bu konuda yine benzetim programları kullanılarak baskılı devre tasarımı için gerekli transfer işleri gerçekleştirilir ve bir prototip oluşturmak için ilk adım böylece atılmış olunur. İşte bu bölümde tanıtılan *multiSIM* benzetim programı, bütün bu işlemlerin yapılabilmesi için kullanılabilecek en temel, kolayca öğrenilebilen bir yardımcı programdır. Prototip aşamasından sonrası ise karmaşıklığı/maliyeti en aza indirerek seri üretime geçmektir. Aslında lojik devre tasarımcısı, en genel adıyla mikroişlemcili sistem tasarımcısı bugün teknolojik alanda insanlığın gelmiş olduğu noktada en önemli sorumluluğu; teknoloji geliştirme görevini yerine getiren kişi konumundadır.

13.6. Sorular

Diğer bölümlerdeki tüm soruların çözümlerini burada deneyebilir ve tasarımlarınızı benzetim ortamında sınayabilirsiniz.

Ek A

Lojik Devre Katalog Bilgileri

Tümdevreler, direnç, kapasitör, diyot, transistör ve diğer elektronik elemanlarla, bunların bağlantılarından oluşan küçük silikon yarıiletken parçacıklar üzerinde gerçeklenmiş devreler topluluğudur. Analog ve Sayısal Tümdevreler olmak üzere iki çeşit tümdevre yapısı mevcuttur.

Analog Tümdevreler, analog işaretleri kullanan devrelerdir. Bu yapıdaki elemanlara işlemsel kuvvetlendiriciler, gerilim regülatörleri, frekans çoğullayıcılar, modülatörler ve osilatörler örnek olarak verilebilirler.

Sayısal Tümdevreler, sayısal işaretleri kullanan devrelerdir. Bu yapıdaki elemanlara lojik kapılar, saklayıcılar, kodlayıcılar, kod çözücüler, sayıcılar, bellek elemanları, karşılaştırıcılar, veri çoğullayıcılar ve veri toplayıcılar örnek olarak verilebilir. Sayısal Tümdevreler, içerdikleri kapı yoğunluklarına göre;

- Küçük Ölçekli Tümdevreler (SSI-Small Scale Integration)
- Orta Ölçekli Tümdevreler (MSI-Medium Scale Integration)
- Büyük Ölçekli Tümdevreler (LSI-Large Scale Integration)
- Çok Büyük Ölçekli Tümdevreler (VLSI-Very Large Scale Integration)

şeklinde dörde ayrılır. Bir tümdevre, kapı sayısı 10'dan az ise *küçük ölçekli tümdevreler* olarak isimlendirilir. Benzer şekilde, kod çeviriciler, kodlayıcılar, çoğullayıcılar ve veri toplayıcılarda olduğu gibi kapı sayısı 10-100 arasındaysa *orta ölçekli tümdevreler*, hesaplayıcılar, sayısal saatler ve bellek birimleri olduğu gibi kapı sayısı 100 ile birkaç bin arasındaysa *büyük ölçekli tümdevreler*, mikrobilgisayarlarda, yeni tip mikroişlemcilerde ve mikrokonrolörlerde olduğu gibi kapı sayısı birkaç binden büyükse devre çok büyük ölçekli tümdevreler olarak adlandırılır.

Sayısal tümdevreler, üretim teknolojilerine göre de sınıflanmaktadırlar:

Direnç-Transistör Lojiği	RTL	Resistor-Transistor Logic
Diyod-Transistör Lojiği	DTL	Diode-Transistor Logic
Yüksek-Eşik Lojiği	HTL	High-Treshold Logic
Transistör-Transistör Lojiği	TTL	Transistor-Transistor Logic
Emitör-Kuplajlı Lojik	ECL	Emitter-Coupled Logic
Metal-Oksitli Yarıiletken	MOS	Metal-Oxide Semiconductor
Tümler-Metal-Oksitli Yarıiletken	CMOS	Complementary MOS
Enjeksiyonlu-Tümleşik Lojik	IIL	Injected-Integrated Logic

Lojik Devre tasarımcısı hangi teknoloji ile üretilmiş elemanı kullanacağına ihtiyacı doğrultusunda karar verir. Lojik devre tasarımında TTL teknolojisi standart olarak kabul edilir. Tasarımcı, eğer işaret işleme gibi yüksek hızlara ihtiyaç duyuluyorsa bu durumda ECL teknolojisi ile üretilmiş bir elemanı tercih edecektir. Eğer elemanın düşük güç tüketmesi tasarımcı için önemli bir kriter ise, bu durumda tasarımcı, CMOS teknolojisi ile üretilmiş elemanı tercih eder.

TTL ve CMOS ailesinden tümdevreler, tasarım özelliklerine göre belirli bir numaralandırma ve harf ekleriyle ifade edilirler. Üretim teknolojilerine göre tümdevrelerin lojik kodları Tablo/Ek-A.1 de verildiği gibidir:

Tablo/Ek-A.1. Üretim teknolojilerine göre tümdevrelerin lojik kodları.

Üretim Teknolojisi	Kodu
Standart TTL	74
Yüksek Hızlı TTL	74H
Düşük Güç Tüketen TTL	74L
Schottky TTL	74S
Düşük Güç Tüketen Shottky TTL	74LS
İleri Shottky TTL	74AS
Düşük Güç Tüketen İleri Shottky TTL	74ALS
CMOS	40
TTL ile bağlantı uyumlu CMOS	74C
Yüksek hızlı ve TTL ile bağlantı uyumlu CMOS	74HC
Yüksek hızlı ve TTL ile elektriksel uyumlu CMOS	74HCT

Kombinezonsal Devreler

Lojik Devre Tasarımı isimli bu kitapta, örneklerde adı geçen ve lojik devre tasarımcısının ihtiyaç duyabileceği bazı tümdevrelerin, özellikle temel ve türetilmiş lojik kapı elemanlarının listesi verilmiştir; tam liste için ilgili üretici kataloglarına bakılmalıdır. Tablo/Ek-A.2 de küçük ölçekli ve düşük güç tüketen *Schottky* TTL tümdevre ailesinden tümdevreler verilmiştir.

Tablo/Ek-A.2. Kombinezonsal lojik devre elemanları, özellikleri ve isimleri.

Tümdevre No	Tümdevrenin Adı	Orijinal Adı	Uç Sayısı
74LS00	Dörtlü 2-girişli TVE	Quad 2-input NAND	14
74LS02	Dörtlü 2-girişli TVEYA	Quad 2-input NOR	14
74LS04	Altılı TÜMLEME	Hex INVERTER	14
74LS08	Dörtlü 2-girişli VE	Quad 2-input AND	14
74LS10	Üçlü 3-girişli TVE	Triple 3-input NAND	14
74LS11	Üçlü 3-girişli VE	Quad 3-input AND	14
74LS20	İkili 4-girişli TVE	Dual 4-input NAND	14
74LS21	İkili 4-girişli VE	Dual 4-input AND	14
74LS27	Üçlü 3-girişli TVEYA	Triple 3-input NOR	14
74LS30	8-girişli TVE	8-input NAND	14
74LS32	Dörtlü 2-girişli VEYA	Quad 2-input OR	14
74LS54	3-2-2-3 girişli VE ve TVEYA	3-2-2-3-input AND-OR-INVERT Gate	14
74LS55	İkili 4-girişli VE ve TVEYA	2-wide 4-input AND-OR-INVERT Gate	14
74LS86	Dörtlü 2-girişli DVEYA	Quad 2-input XOR	14
74LS133	13-girişli TVE	13-input NAND	14

Tümleşik Kombinezonsal Devreler

Tümleşik kombinezonsal devrelerde belirli bir işi yerine getiren kapılarla tasarlanmış devrelerdir. Bu devreler içinde yarı ve tam toplayıcılar, kod çözücüler, kodlayıcılar, seçiciler ve çoğullayıcılar gibi elemanlar sayılabilir. Orta ölçekli tümleşik devrelere ek olarak bu ve benzeri tümleşik lojik devreleri Programlanabilir Lojik Düzenler adı verilen özel türdeki düzenlerle gerçekleştirmek de olasıdır. Programlanabilir Lojik Düzenler içinde PAL, PLA ve ROM elemanları yer almakta olup, bu elemanlar özellikle orta ölçekte tümleşik devrelerin gerçeklenmesinde iyi bir seçenek olarak kullanılmaktadırlar. Bu açıdan değerlendirildiğinde gerek orta

ölçekli tümleşik kombinezonsal devre elemanlarının ve gerekse programlanabilir lojik düzenlerin özellikle belleksiz devrelerin tasarımında oldukça büyük bir yer kapladığı söylenebilir. Bu ayrıtta son olarak orta ölçekli tümleşik devre elemanları Tablo/Ek-A.3 de, PLD elemanları ise Tablo/Ek-A.4 te tanıtılmış ve özellikleri verilmiştir.

Tablo/Ek-A.3. Tümleşik kombinezonsal lojik devre elemanları, özellikleri ve isimleri.

Tümdevre No	Tümdevrenin Adı	Orijinal Adı	Uç Sayısı
74LS138	3×8 Kod Çözücü	1-of-8 Decoder	16
74LS139	İkili 2×4 Kod Çözücü	Dual 1-of-4 Decoder	16
74LS148	Öncelik Kodlayıcı	8-input to 3-line Priority Encoder	16
74LS151	8×1 Seçici	8-input Multiplexer	16
74LS153	İkili 4×1 Seçici	Dual 4-input Multiplexer	16
74LS154	4×16 Kodlayıcı	4-to-16 decoder	24
74LS155	İkili 2×4 Kod Çözücü/Seçici	Dual 1-of-4 Decoder/Demultiplexer	16
74LS157	Dörtlü 2-girişli Seçici	Quad 2-input Mux	16
74LS283	4-bitlik Tam Toplayıcı	4-Bit Full Adder with Fast Carry	16
74LS351	16×1 Seçici	16-input Multiplexer	24

Tablo/Ek-A.4. Programlanabilir Lojik Düzenler ve ilgili devre elemanları ve özellikleri.

Tümdevre No	Tümdevrenin Adı	Orijinal Adı	Uç Sayısı
PAL 16L8A	10 giriş, 2 çıkış, 6 seçilebilir giriş-çıkışlı Programlanabilir Diziler (PAL)	Standard High-Speed PAL Circuits High Speed (35 MHz)	20
PAL 16L8A-2	10 giriş, 2 çıkış, 6 seçilebilir giriş - çıkışlı Programlanabilir Dizi elemanı (PAL)	Standard High-Speed PAL Circuits Half Power (18 MHz)	20
PAL20L8A	14 giriş, 2 çıkış, 6 seçilebilir giriş - çıkışlı Programlanabilir Dizi elamanı (PAL)	Standard High-Speed PAL Circuits Half Power (18 MHz)	24
PLA839M	14 giriş 6 çıkışlı Programlanabilir Lojik Dizi elamanı (PLA)	14 × 32 × 6 Field-Programmable Logic Arrays	24

Ardışıl Devreler

Lojik devreler sadece kombinezonsal devrelerden ibaret olmayıp kendi içinde bellek elemanı barındıran lojik devreler de bulunmaktadır. Bu devrelere ardışıl devreler adı verilir. Ardışıl lojik devreler,en temelde, Flip-Flop adı verilen elemanlar kullanılarak tasarlanırlar. Bilinen en temel ardışıl devreler sayıcılardır; saklayıcı, mikroişlemci, ALU ardışıl devre elemanları olarak bilinirler. Tablo/Ek-A.5 de bazı ardışıl devre elemanları listelenmiş ve özellikleri verilmiştir.

Çok farklı özelliklere sahip ardışıl devreler vardır. Bunlarla ilgili ayrıntılı bilgilere ilgili üretici firma kataloglarına bakılmalıdır.

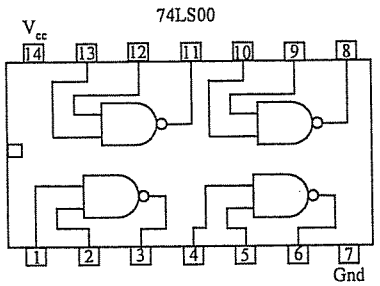
Tablo/Ek-A.5. Ardışıl lojik devre elemanları, özellikleri ve isimleri

Tümdevre No	Tümdevrenin Adı	Orijinal Adı	Uç Sayısı
74LS74A	İkili D Flip-Flopu (Yükselen Kenar Tetiklemeli)	Dual D type Positive Edge Triggered Flip-Flop	14
74LS90	Ondalık sayıcı	Decade Counter	14
74LS91	8-bitlik Ötelemeli Saklayıcı	8-bit shift register serial-in serial-out	14
74LS93	4-bitlik ikili sayıcı	4-bit Binary Counter	14
74LS95B	4-bitlik Ötelemeli Saklayıcı	4-bit Shift Register	14
74LS109A	İkili JK Flip-Flop (Yükselen Kenar Tetiklemeli)	Dual JK Positive Edge Triggered Flip-Flop	16
74LS112A	İkili JK Flip-Flop (Düşen Kenar Tetiklemeli)	Dual JK Negative Edge Triggered Flip-Flop	16
74LS113A	İkili JK Flip-Flop (Düşen Kenar Tetiklemeli) (Sıfırlama girişi yok)	Dual JK Negative Edge Triggered Flip-Flop	14
74LS114A	İkili JK Flip-Flop (Düşen Kenar Tetiklemeli) (Ana-uydu tetiklemeli)	Dual JK Negative Edge Triggered Flip-Flop	14
74LS181	4-bit Aritmetik Lojik Birim	4-bit ALU	24
74LS193	4-bitlik ikili öndeğer verilebilir yukarı/aşağı sayıcı	Presettable 4-bit Binary Up/Down Counter	16
74LS195A	4-bitlik Universal Ötelemeli Saklayıcı	Universal 4-bit Shift Register	16

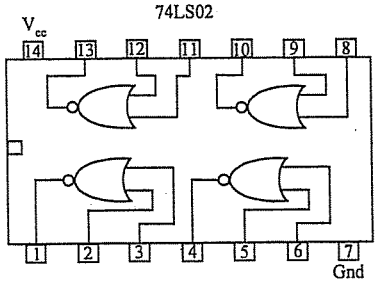
Bazı Tümdevrelerin Şematik Gösterimleri

Burada düşük güç tüketen *Schottky* TTL tümdevre ailesine ilişkin bazı temel kapılara ilişkin tümdevrelerin lojik şemaları verilmiştir. Bu devreler sırasıyla şöyledir:

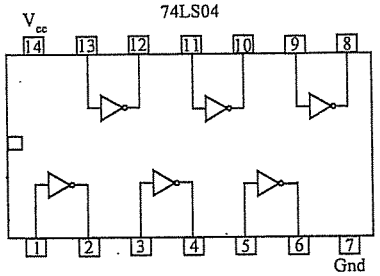
74LS00 - Dörtlü 2-girişli TVE
74LS02 - Dörtlü 2-girişli TVEYA
74LS04 - Altılı TÜMLEME
74LS08 - Dörtlü 2-girişli VE
74LS32 - Dörtlü 2-girişli VEYA
74LS86 - Dörtlü 2-girişli YA DA



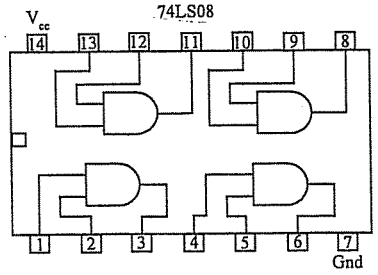
74LS00 Dörtlü iki-girişli TVE



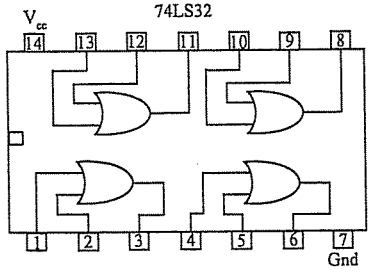
74LS02 Dörtlü iki-girişli TVEYA



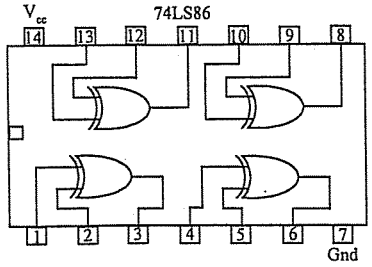
74LS04 Altılı TÜMLEME



74LS08 Dörtlü iki-girişli VE



74LS32 Dörtlü iki-girişli VEYA

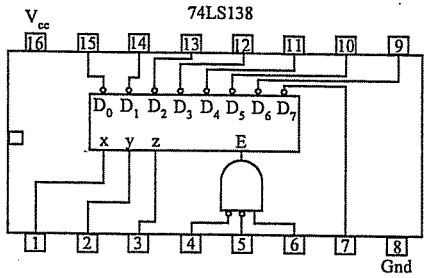


74LS86 Dörtlü iki-girişli YA DA

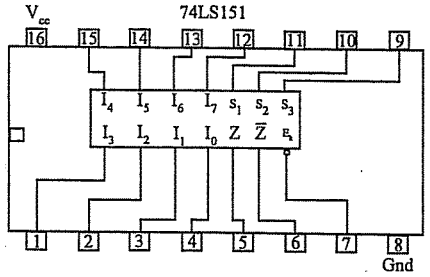
Burada, sırasıyla düşük güç tüketen *Schottky* TTL tümdevre ailesine ilişkin;

74LS138 - 3×8 Kod Çözücü (Decoder)
74LS151 - 8×1 Seçici (MUX)
74LS283 - 4-bitlik Tam Toplayıcı

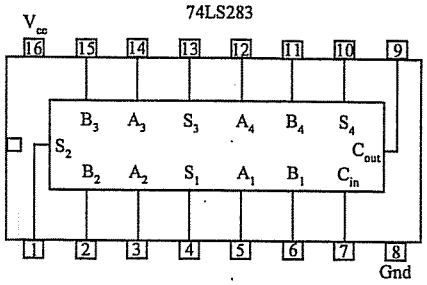
tümleşik kombinezonsal tümdevrelerin lojik şemaları verilmiştir.



74LS138 3e8 Kod Çözücü (Decoder)



74LS151 8×1 Seçici (Multiplexer)

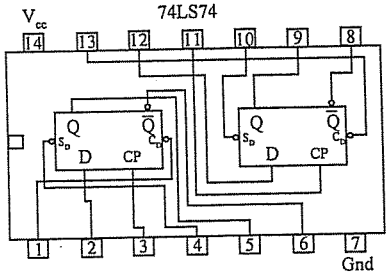


74LS283 4-bitlik Tam Toplayıcı

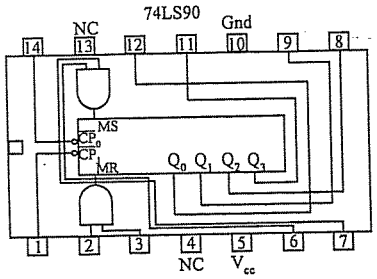
Burada düşük güç tüketen *Schottky* TTL tümdevre ailesine ilişkin,

74LS74 - İkili, kenar tetiklemeli D-Flip-Flop'u
74LS90 - Yukarı ondalık sayıcı
74LS194A - 4-bitlik üniversal ötelemeli saklayıcı

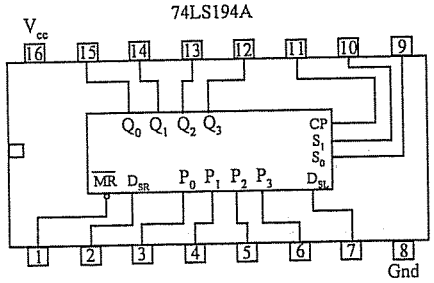
ardışıl lojik tümdevrelerin lojik şemaları verilmiştir.



74LS74 İkili Kenar tetiklemeli D-Flip-Flop'u



74LS90 Yukarı Ondalık Sayıcı



74LS194A 4-bitlik Universal Ötelemeli Saklayıcı

Ek B

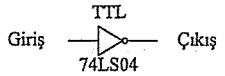
TTL ve CMOS Tümdevre Özellikleri

Bütün lojik kapı elemanları, Flip-Flop'lar, kodlayıcılar, kod çözücüler, seçiciler, dağıtıcılar gibi lojik elemanlar TTL ve CMOS teknolojisi ile üretilmektedir. Dolayısıyla lojik devre tasarımcısı bu iki teknolojiye ait tümdevreleri oldukça yoğun bir şekilde kullanmak durumunda kalır. Durum böyle olunca da bu iki aileye ait karakteristik değerleri öğrenmek veya bu bilgileri bir şekilde bilmek yararlı olur. Bu bölümde lojik devre tasarımcısının her dönemde ihtiyaç duyacağı bilgiler özetlenmeye çalışılmıştır. Bölüm içinde sırasıyla: TTL ve CMOS tümdevre karakteristikleri ele alınan birer tümleme kapısı üzerinde açıklanmış, bunların yanı sıra propogasyon gecikmesi gibi bazı önemli kavramlar da irdelenmiştir.

TTL Tümdevre Özellikleri

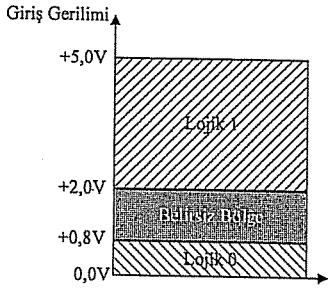
TTL özellikleri, lojik tasarım yapan mühendisler ve teknisyenler tarafından bilinmek zorundadırlar. Bir tasarımın ve üretimin temel başlangıç noktası da bu bilgilerdir.

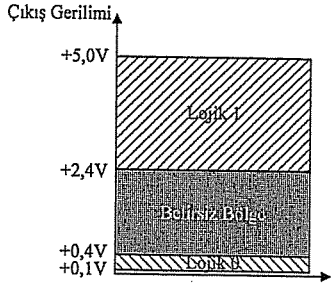
Bilindiği üzere lojik devreler 0 ve 1 ile ifade edilen ikili işaretlerle çalışırlar. Ancak her lojik ailenin lojik 0 ve lojik 1 seviyeleri birbirinden farklıdır.



TTL tümdevrelerin besleme gerilimi +5 V'dur. Bu aileye ait bütün elemanların lojik giriş seviyeleriyle lojik çıkış seviyeleri aynıdır. Bir TTL tümdevrede, örneğin 74LS04'te, giriş değeri, giriş gerilimi 0 V-0,8 V arasındaysa lojik 0; 2,0 V-5,0 V arasındaysa lojik 1 seviyesindedir. 0,8 V-2,0 V arası ise belirsiz bölgedir. 0,8 V-2,0 V arasında bir gerilim uygulanması durumunda TTL devre girişindeki işaret belirlenemez ve bu değer bir lojik seviye olarak yorumlanamaz. Kısacası giriş ne lojik 0'dır ne de lojik 1'dir; bir başka deyişe belirsizdir. Yine, 74LS04 tümdevresinde,

çıkış değeri, çıkış gerilimi 0,1 V-0,4 V arasındaysa lojik 0; 2,4 V-5,0 V arasındaysa lojik 1 seviyesindedir. 0,4 V-2,4 V arası ise yine belirsiz bölge olarak isimlendirilmiştir. Bu arada bir gerilim uygulanması durumunda TTL devre çıkışında bir lojik seviye belirten bir işaret söz konusu değildir. Giriş ve Çıkış gerilimlerine ilişkin özellikler aşağıdaki şekillerden görülmektedir:





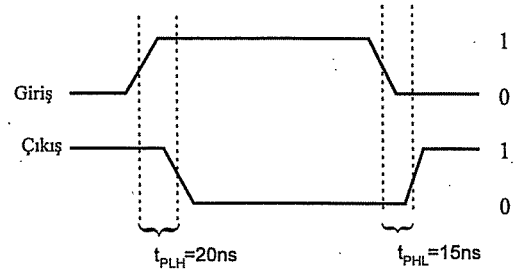
Lojik 1 seviyesi için giriş gerilimi ile çıkış gerilimi arasında bir gerilim farklılığı bulunmaktadır. Bunun sebebi sayısal sisteme her an etkiyebilecek olan istenmeyen gürültülerden lojik seviyelerin etkilenmesini engellemektir. Bu farklılık lojik 0 seviye için de geçerlidir. Girişte 0 V-0,8 V'luk bir aralık bulunmasına rağmen, çıkışta bu aralık 0,1 V-0,4 V'a kadar düşüş göstermektedir.

Propagasyon Gecikmesi

Bir elemanının girişinde oluşan bir işaret değişiminin çıkışta görülmesi için geçen süreye propagasyon gecikmesi adı verilir. Bu gecikmeyi göstermek için 7404 TÜMLEME kapısı örnek olarak seçilmiştir. Kapının girişine uygulanan 0-1-0 geçişli bir işarete karşılık oluşan 1-0-1 şeklindeki işaret değişimi arasındaki gecikmeler değerlendirildiğinde şu sonuçla karşılaşılır: lojik kapının 1-0 geçişine oranla 0-1 geçişinde daha büyük bir gecikme söz konusudur. 0-1 geçişinde yaklaşık 20 ns'lik bir gecikme oluşurken 1-0 geçişinde sadece 15 ns'lik bir propagasyon gecikmesi söz konusudur (bkz. yan sayfadaki şekil). Bu değerler farklı türdeki TTL'ler için farklıdır! Özellikle ileri *Shottky* teknolojisi ile üretilen tümdevrelerde, örneğin 74ALS04, propagasyon gecikmesi 1,5 ns mertebesindedir. Hızlı elemanlarda propagasyon gecikmesi daha küçüktür.

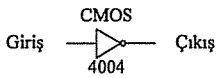
Standard TTL elemanların güç harcama miktarı 10 mW (mili Watt) civarındadır; genel olarak 1-10 mW mertebesinde olduğu söylenebilir; örneğin 74LS00 kapısı için bu değer tipik olarak 1 mW'dır; tabi ki, tümdevre içerisindeki kapıların kullanılma durumları da güç tüketimini etkiler. Tüm kapılar en çok akım çekecek durumda sürülürlerse daha çok güç harcanır.

Bütün bunların yanısıra, bir elemanın çıkışında sürebileceği eleman sayısı da önem taşımaktadır. Standart TTL ve ileri *Shottky* teknolojisi ile üretilen TTL elemanlar çıkışında aynı cins 10 taneye kadar kapı sürebilir, ancak bir ileri *Shottky* TTL eleman çıkışında sadece bir tane standart TTL sürebilir.



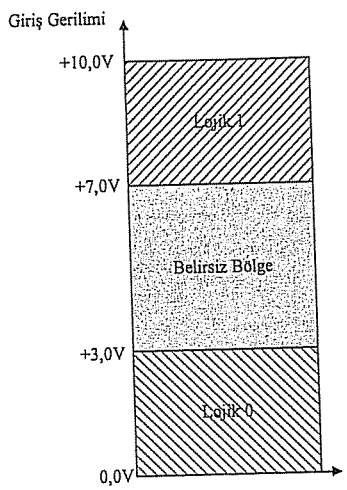
CMOS Tümdevreler

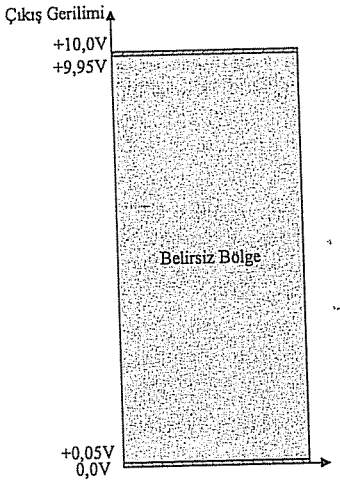
CMOS'larda besleme gerilimi 10 V'dur. Ancak, HCT kodlu CMOS tümdevrelerde besleme gerilimi seviyeleri TTL tümdevrelerle aynıdır. Çünkü HCT serisi CMOS tümdevreler elektriksel olarak TTL uyumlu tümdevrelerdir. Aşağıda CMOS, 4004 tümdevresinin içerisinde bulunan bir TÜMLEME kapısı görülmektedir.



Bir CMOS tümdevrede, örneğin 4004'te, giriş değeri, giriş gerilimi 0,0 V-3,0 V arasındaysa lojik 0; 7,0 V-10,0 V arasındaysa lojik 1 seviyesindedir. 3,0 V-7,0 V arası ise belirsiz bölge olarak isimlendirilmiştir. Bu aralıkta bir gerilim uygulanması durumunda CMOS devre girişindeki işaret belirlenemez ve bu değer bir lojik seviye olarak yorumlanamaz. Kısacası, giriş ne lojik 0'dır ne de lojik 1'dir; belirsizdir. Yine 4004 tümdevresinde, çıkış değeri, çıkış gerilimi 0,0 V-0,05 V arasındaysa lojik 0; 9,95 V-10,0 V arasındaysa lojik 1 seviyesindedir. 0,05 V-9,95 V arası ise yine belirsiz bölge olarak isimlendirilmiştir. Bu arada bir gerilim uygulanması durumunda CMOS devre çıkışında bir lojik seviye belirten bir işaret söz konusu değildir. Giriş ve Çıkış gerilimlerine ilişkin özellikler aşağıdaki şekillerde verilmiştir.

TTL teknolojisinde olduğu gibi CMOS teknolojisinde de lojik 1 ve lojik 0 seviyeleri için giriş gerilimi ile çıkış gerilimi arasında bir gerilim farklılığı bulunmaktadır. Bunun sebebinin sayısal sisteme her an etkiyebilecek olan istenmeyen gürültülerden lojik seviyelerin etkilenmesini engellemek olduğu daha önce belirtilmişti. Özellikle çıkış gerilimi için CMOS devrelerin bu gürültü filtreleme konusunda TTL teknolojisinden daha iyi olduğu söylenebilir.





Propagasyon Gecikmesi

7404 TTL TÜMLEME kapısında oluşan propagasyon gecikmesi 74HC04 CMOS TÜMLEME kapısı için de geçerlidir. Standart CMOS tümdevrelerde propagasyon gecikmesi 25-100 ns arasında değişmesine rağmen, özellikle yeni nesil yüksek hızlı CMOS devrelerde (HC serisi) bu gecikme 8 ns mertebesine düşmektedir. Dolayısıyla tasarımın yüksek hızda çalışması isteniyor ya da planlanıyorsa mutlaka HC serisi CMOS teknolojisi ile üretilmiş lojik elemanların kullanılması önerilir. CMOS elemanların güç harcama miktarı 0,01-1 mWatt mertebesindedir.

TTL ve CMOS Karşılaştırılması

CMOS teknolojisi, TTL teknolojisi ile üretilen tümdevrelere göre, gürültüye daha az duyarlıdırlar; dolayısıyla daha iyidir. Standart olarak bir TTL'de propagasyon gecikmesi 10-15 ns, CMOS'da 25 ns düzeyindedir. Ancak heriki ailenin de çeşitli türleri vardır; bunların propagasyon gecikmeleri ve harcadıkları güç miktarı değişmektedir. Eski neslin aksine, yeni nesil CMOS tümdevrelerde propagasyon gecikmesi TTL elemanlara oranla daha küçüktür denilebilir; böylece daha hızlı çalışan lojik devreler gerçekleştirmek olasıdır. Bununla birlikte, güç tüketimleri açısından bir karşılaştırma yapıldığında CMOS teknolojisinin TTL teknolojisine göre daha üstün olduğu görülür. Standart TTL'de 10 mW, düşük güç harcayan LS serisi TTL devrelerde 1 mW civarına güç tüketilirken, CMOS teknolojisinde güç harcama 0,01 mW-1 mW arasındadır. Bu yüzden özellikle pille çalışması istenen tasarımlarında ya CMOS ya da çok az enerji tüketen diğer TTL teknolojisi tercih edilir.

Ek C

Lojik Devreler Lab. Deney Önerileri

Lojik devre tasarımını anlamanın ve edinilen bilginin pekiştirilmesinin en kolay yolu konularla ilgili uygulama yapmaktan geçer. Böylece teorik olarak görülen cebirsel ifadelerin indirgenmesi, kombinezonsal devreler, ardışıl devreler gibi konuların pratikte uygulaması ve gerçek dünyada nasıl kullanılacağı görülmüş olur. Burada, bir lojik laboratuarında yapılması gereken on tane deney önerisi verilmiştir; bir lojik devre laboratuarı uygulamasında en az sekiz tanesinin yapılması gerektiği ön görülmektedir;

Deney 1. Lojik Kapı Elemanları
Deney 2. TTL ve CMOS Özelliklerinin Çıkarılması
Deney 3. Lojik Fonksiyonların İndirgenmesi
Deney 4. Aritmetik İşlem Devreleri
Deney 5. PLD Uygulaması
Deney 6. İşaret Üreteçleri ve Frekans Bölme
Deney 7. Üç-Durumlu Çıkışlar ve Ortak Yol Kullanımı
Deney 8. Saklayıcı ve Sayıcı Uygulamaları
Deney 9. Ardışıl Devre Uygulaması
Deney 10. ADC/DAC Uygulamaları

Yukarıda listelenen deneyler iki şekilde yapılabilir; ya, doğrudan elemanlara ait tümdevrelerin deney düzenekleri üzerinde fiziksel kurulumlar, ya da lojik devre benzetim ortamları kullanılır. Deney düzenekleri lojik kapı vs. gibi tümdevreler, proto-board, güç kaynağı, osiloskop, lojik probe gibi ekipmanlardan oluşabileceği gibi, bunların birçoğunu üzerinde barındıran ve genel olarak Deney Seti (Örneğin, C.A.D.E.T., Wish Maker vb.) olarak adlandırılan hazır düzenekler de olabilir.

Yapılan her deney sonrası bir deney raporu hazırlanması gerekir; rapor, deneyi yapan kişinin deney sırasında edindiği bilgi birikimini açıkça yansıtmalı ve bir teknik rapor ciddiyetinde olmalıdır. Bu yapılırken de standart mühendislik formatlarının kullanılması önerilir.

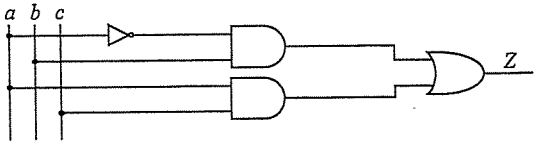
Deney 1.

Lojik Kapılar ve Fonksiyon İndirgeme

Bu deneyin amacı, temel lojik kapıları ve bu elemanlardan türetilen kapı elemanlarını pratik kullanım açısından incelemektir. Deneyde, sırasıyla VE, VEYA ile TÜMLEME temel kapılarına ilişkin lojik ifadelerin doğruluğu örnek girişler verilerek sınanacaktır. Buna ilave olarak lojik devre tasarımcısının temel kriterlerinden biri olan maliyet faktörünü en aza indirgemek amacıyla tek tip kapı elemanlarıyla (TVE ya da TVEYA kapıları kullanılarak) nasıl tasarım yapıldığı yine bu deneyde tartışılacaktır. Sonraki aşamalarda, verilen bir lojik fonksiyon indirgenecek ve hem ilk fonksiyona hem de indirgenmiş fonksiyona ait doğruluk tabloları oluşturularak çıkış fonksiyonlarının aynı lojik değerleri aldığı gösterilecektir.

Deneyden Önce Yapılacaklar:

- Şekil-D1.1.'deki devreye ait **Z** çıkış ifadesini bulunuz.



Şekil-D1.1. Örnek devre.

- Devreye ilişkin doğruluk tablosunu oluşturunuz.
- Z çıkış ifadesini *Karnaugh* diyagramıyla indirgeyiniz.
- İndirgenmiş fonksiyonu *sadece TVE kapıları* ile gerçekleyiniz.
- İndirgenmiş fonksiyonu *sadece TVEYA kapıları* ile gerçekleyiniz.
- Hangi tasarımın daha az kapı gerektirdiğini belirleyiniz.

Gerekli Önbilgiler:

Bu deneye başlamadan önce lojik fonksiyonların gerçeklenmesinde kullanılan VE, VEYA, TÜMLEME kapıları ile TVE, TVEYA, YA DA ve TYA DA gibi türetilen kapılara ilişkin doğruluk tabloları incelenmelidir; ayrıntılı bilgi için bkz. Ayrıt 5.1.

Ayrıca, TVE ile TVEYA türetilmiş kapıları kullanılarak temel kapıların oluşturulması için gerekli yöntemlerin anlatıldığı Ayrıt 6.5 i inceleyiniz.

Deneyde Kullanılan Malzemeler:

- 1 x 7400 tümdevresi;4'lü 2-girişli NAND (TVE)
- 1 x 7402 tümdevresi;4'lü 2-girişli NOR (TVEYA)
- 1 x 7404 tümdevresi;6'lı INVERTER (TÜMLEME)

- 1 x 7408 tümdevresi;4'lü 2-girişli AND (VE)
- 1 x 7432 tümdevresi;4'lü 2-girişli OR (VEYA)
- Besleme kaynağı, bağlantı kabloları.

Deneyin Yapılışı:

a) TÜMLEME, VE ile VEYA kapılarına ilişkin tümdevreleri (7404, 7408, 7432) teker teker deney düzeneğine kurunuz ve kapıların her girişine ilişkin mümkün lojik değerleri vererek her kapı elemanına ilişkin doğruluk tablolarını elde ediniz.

b) TVE (7400) kapıları kullanarak, VE, VEYA ile TÜMLEME kapıları gibi çalışacak devreye ilişkin eşdeğer devreleri kurunuz ve doğruluk tablolarını elde ediniz.

c) TVEYA (7402) kapıları kullanarak, VE, VEYA ile TÜMLEME kapıları gibi çalışacak devreye ilişkin eşdeğer devreleri kurunuz ve doğruluk tablolarını elde ediniz.

d) Deneyden önce yapılacaklar kısmında verilen devreye eşdeğer devreyi en az sayıda kullanılacağı tespit ettiğiniz aynı tür kapılarla (TVE ya da TVEYA) kurunuz. Giriş değişkenlerine mümkün lojik değerleri vererek devreye ilişkin doğruluk tablosunu oluşturunuz. Sonucu deneyden önce yapılacaklar kısmında bulduğunuz doğruluk tablosu ile karşılaştırınız.

Deney Raporunda Yapılması İstenenler:

1. Herbir deneyde elde ettiğiniz, gözlemlediğiniz sonuçları şekillerle destekleyerek açıklayınız.
2. Aynı tür kapı elemanları kullanmanın olumlu yönlerini açıklayınız. Hangi durumda TVE, hangi durumda TVEYA kapısının tercih edilmesi gerektiği konusunda bir yöntem önermek mümkün müdür?
3. TTL ailesine ait 7400 ve 74LS00 tümdevrelerini, propagasyon gecikmesi ve çıkışta sürebildikleri kapı sayısı açısından karşılaştırınız.

Deney 2.

TTL ve CMOS Özelliklerinin Çıkarılması

TTL ve CMOS tümdevreler lojik devre tasarımında standart birimlerdir denilebilir; kodlayıcı, kod çözücü, saklayıcı, sayıcı gibi birçok birim TTL ve CMOS teknolojisiyle birer tümdevre olarak üretilmişlerdir. Bu deneyle yapılmak istenen TTL ve CMOS teknolojileriyle üretilen kapı ve diğer temel birimlerin gecikme, G/Ç akım-gerilim eğrileri ve harcanan güç miktarının belirlenmesidir.

Deneyden Önce Yapılacaklar:

- Tümdevre kataloglarından araştırılarak TTL ve CMOS teknolojileriyle üretilmiş kapı tümdevrelerinin özellikleri incelenmelidir.

Gerekli Önbilgiler:

Bir kapının girişine uygulanan değişikliğin çıkışına yansıtılmasına kadar geçen süre gecikme olarak adlandırılır; ve, bu gecikme değeri, 0'dan 1 geçişle (τ_y), 1'den 0'a geçiş (τ_d) arasında farklıdır. Kapıların girişine uygulanan gerilim 0 Volt ile belirli bir Volt arasında ise lojik 0 olur; arada bir belirsiz bölge vardır; daha sonra belirli bir Volt ile 5 Volt arasıda lojik 1 olur. Bu belirli bölgeler TTL ve CMOS için farklıdır.

- ***Boşta çalışma özelliğinin bulunması***

Çıkışa herhangi bir bağlantı yapılmamışken (Şekil-D2.1.'de yük olarak kullanılan 3 TVE kapısı bağlı değilken), yani yüksüzken, girişindeki gerilim (V_g) değeri 0 V'dan 5 V'a kadar yavaş yavaş arttırılır ve çıkıştaki gerilim ($V_ç$) ölçülür. Bu değerler, V_g yatay eksen, $V_ç$ dikey eksen olmak üzere $V_ç=f_b(V_g)$ bağıntısı uyarınca G/Ç eğrisi elde edilir.

- ***Yükte çalışma özelliğinin bulunması***

Bir kapının çıkışına birkaç kapı bağlıyken (bkz. Şekil-D2.1), yani çıkış yüklü iken G/Ç eğrisi elde edilir; $V_ç=f_y(V_g)$ eğrisinin bulunması.

- ***Giriş gerilimi giriş akımı arasındaki ilişkinin çıkarılması***

Çıkışa herhangi bir yük bağlanmamışken $V_ç=f_{ggça}(I_g)$ bağıntısına ait eğrinin bulunması; bu amaçla kapı girişindeki gerilim arttırılırken akımın değişimi bir yere yazılır ve $V_ç/I_g$ eğrisi çizilir.

- ***Bir kapı için güç harcama miktarının bulunması***

Güç, besleme gerilimiyle (V_{cc}) çekilen akımdan (I_{cc}) hesaplanır. İlk önce kapının çıkış ve girişleri boştayken harcana güç bulunur; daha sonra kapının girişlerine 1, 2, 3, 4 ve 5 MHz'lik işaretler uygulanarak ayrı ayrı harcanan güç miktarı bulunur. Besleme gerilimi ve çekilen akımı ölçmek için Voltmetre besleme gerilimine paralel,

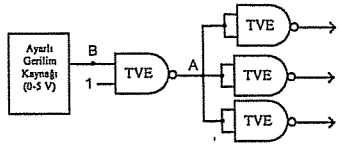
Ampermetre seri olarak bağlanır; DC gerilim ve akımla çalışıldığı için Voltmetre ve Ampermetre doğru kutuplanmalıdır (aksi durumda ibre ters yönde sapar veya eksi değer gösterilir).

- ***Yükselen ve düşen kenarlarda gecikme değerlerinin bulunması***

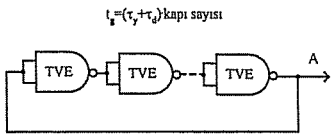
Bir kapının girişleri birleştirilip girişine bir kare dalga işaret uygulanır; giriş ve çıkış işaretleri bir osiloskop ekranında üst üste bindirilerek τ_y ve τ_d değerleri bulunur.

Deneyde Kullanılan Malzemeler:

- TTL TVE (NAND) kapıları; *örneğin 7400 TTL tümdevreleri (2 adet)*
- CMOS TVE (NAND) kapıları; *örneğin 4000 CMOS tümdevreleri (2 adet)*
- 1 adet Ayarlı gerilim kaynağı, 2 adet AVO metre, Osilaskop cihazı.



Şekil-D2.1. TVE kapısının yüklenmesi



Şekil-D2.2. TVE kapısıyla İşaret üretici

Deneyin Yapılışı:

a) TTL kapının özelliklerinin bulunması
   - i) Boşta çalışma özelliği
   - ii) Yükte çalışma özelliği
   - iii) Yükselen ve düşen kenarlarda gecikme değerleri (τ_y ve τ_d)
   - iv) Bir kapı için güç harcama miktarı
   - v) Giriş gerilimi giriş akımı arasındaki ilişkinin çıkarılması

b) CMOS kapının özelliklerinin bulunması
   - i) Boşta çalışma özelliği
   - ii) Yükte çalışma özelliği
   - iii) Yükselen ve düşen kenarlarda gecikme değerleri (τ_y ve τ_d)
   - iv) Bir kapı için güç harcama miktarı
   - v) Giriş gerilimi giriş akımı arasındaki ilişkinin çıkarılması

c) Kapılarla işaret üreteci elde edilmesi ve gecikme
   - i) 7 tane iki girişli TTL TVE kapısını Şekil-D2.2 deki gibi bağlayınız ve A noktasında görülen işareti osiloskopta gözleyiniz. İşaretin periyodu ve frekansı hangi büyüklüktedir; deney a-iii'de elde edilen gecikme değerleriyle nasıl bir ilişki vardır?

ii) Bir önceki deneyi (Deney i) CMOS TVE kapısı için deneyiniz. Elde edilen işaretin periyodu ve frekansı hangi büyüklüktedir; deney b-iii'de elde edilen gecikme değerleriyle nasıl bir ilişki vardır? τ_y, yükselen; τ_d, düşen kenarda gecikmelerdir.

Deney Raporunda Yapılması İstenenler:

1. Herbir deneyde elde ettiğiniz, gözlemlediğiniz sonuçları şekillerle destekleyerek yazınız.
2. TTL kapı ile CMOS kapı arasında en belirgin fark nelerdir? Sizce hangi durumlarda hangisi seçilmelidir?
3. Deney c'de neden kapı sayısı tek sayı olan 7 seçilmiştir; çift sayıda kapı kullanılmış olsaydı nasıl bir sorun çıkabilirdi?

Deney 3.

Kombinezonsal Devre Tasarımı

Bu deneyin amacı, sözle tanımı yapılan işlevleri yerine getiren kombinezonsal devrelerin tasarımı gerçekleştirmektir. Devre tasarımında aynı tür kapı elemanlarının kullanılması şart koşularak devre karmaşıklığının/maliyet faktörünün ve devre gecikmelerinin en aza indirilmesine çalışılmıştır.

Deneyden Önce Yapılacaklar:

- Deneyin yapılmasına geçmeden önce aşağıda sözle tanımlanan problemlerin tasarımının tamamlanmış olması gerekmektedir. Problemler şu şekilde tanımlanmışlardır;

1. x, a ve b şeklinde üç girişi bulunan bir devre gözönünde bulundurulsun. Devre aşağıdaki çıkışları vermektedir.

x=0 ve a=0 için devre çıkışı lojik 0,

x=1 ve b=1 için devre çıkışı lojik 1,

a=1 ve b=0 için devre çıkışı lojik 1,

diğer durumlar dikkate alınmamıştır.

Bu kombinezonsal devreyi tasarlayınız. Devreyi sadece TVE (NAND) kapıları ile gerçekleştiriniz.

2. x, a ve b şeklinde üç girişi bulunan bir devre x girişine gelen lojik seviyelere göre çalışmaktadır. Buna göre, x=0 ise TVEYA, x=1 ise TVE gibi çalışan devre için doğruluk tablosunu oluşturunuz, çıkış işaretini indirgeyiniz ve devreyi sadece TVEYA kapıları kullanarak tasarlayınız.

Gerekli Önbilgiler:

Kombinezonsal devrelerin ve kombinezonsal devre elemanlarının tanıtıldığı Ayrıt 7.1 ve Ayrıt 7.2 ile Kombinezonsal devre tasarımının ayrıntılı olarak açıklandığı Ayrıt 7.3 ü inceleyiniz.

Ayrıca, indirgenmiş ifadelerin aynı tür kapılarla gerçeklenme şartlarının anlatıldığı Ayrıt 6.6 yı gözden geçiriniz.

Deneyde Kullanılan Malzemeler:

- 1 x 74HC00 tümdevresi; 4'lü 2-girişli NAND (TVE)
- 1 x 74HC02 tümdevresi; 4'lü 2-girişli NOR (TVEYA)
- Besleme kaynağı ve bağlantı kabloları.

Deneyin Yapılışı:

a) Deneyden önce yapılacaklar kısmında tanımlanan ilk probleme ilişkin tasarladığınız devreyi kurunuz. Devreye ilişkin doğruluk tablosunu elde ediniz.

b) Deneyden önce yapılacaklar kısmında tanımlanan ikinci probleme ilişkin tasarladığınız devreyi kurunuz. Devreye ilişkin doğruluk Tablosunu elde ediniz.

Deney Raporunda Yapılması İstenenler:

1. Herbir deneyde elde ettiğiniz, gözlemlediğiniz sonuçları şekillerle destekleyerek açıklayınız.
2. Bir bitlik bir toplama ve çıkarma devresi tasarlayınız. Bu devrede x, a ve b şeklinde giriş değişkenleri, f_1 ve f_2 şeklinde çıkış değişkenleri tanımlanmıştır. f_1 çıkışı Borç ya da Elde, f_2 çıkışı Toplam ya da Fark çıkışı olarak tanımlanmıştır. x=0 ise devre çıkarma (a-b), x=1 ise devre toplama (a+b) işlemi yapmaktadır. Devreyi sadece TVEYA (NOR) kapıları ile tasarlayınız.
3. Beş değişkenli bir fonksiyonun *Karnaugh* diyagramıyla indirgenmesi mümkün müdür? Bu konuyu araştırarak, eğer mümkün ise, bir örnek üzerinde açıklayınız.

Deney 4.

Aritmetik İşlem Devreleri

Bu deneyde, toplama ve çıkarma işlemlerinin gerçeklendiği aritmetik devreler tanıtılacaktır. Tasarım yöntemi ve çalışma prensibi açısından bu devreler birer kombinezonsal devredir. Deney aşamasında, *yarı toplayıcı*, *tam toplayıcı* devrelerle, 4-bitlik toplama ve çıkarma devresi gerçekleştirilerek, çalışma biçimleri incelenecektir.

Deneyden Önce Yapılacaklar:

- Yarı toplayıcı devrenin gerçekleştirilmesi için gerekli kapı elemanlarını ve bunlara ilişkin tümdevreleri belirleyiniz.
- Tam toplayıcı devrenin iki adet yarı toplayıcı devreyle gerçekleştirilmesi için gerekli kapı elemanlarını ve bunlara ilişkin tümdevreleri belirleyiniz.
- 4-bitlik toplama ve çıkarma devresini, 74HC283 tümdevresi ile gerçekleştirmek için gerekli bağlantı şemasını oluşturunuz.
- 4-bitlik toplama ve çıkarma işlemlerini yapabilmek amacıyla bir Aritmetik Lojik Birim (ALU) kullanılmak istenmektedir. Bu işlemleri, 74181 ALU-tümdevresi ile gerçekleştirmek için gerekli bağlantı şemasını oluşturunuz.

Gerekli Önbilgiler:

Yarı toplayıcı devrenin tasarımına ilişkin ifadeleri gözden geçiriniz. (*bkz. Ayrıt 7.4.1*)

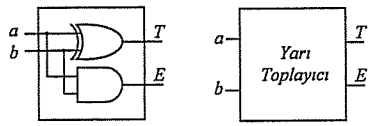
Tam toplayıcı devrenin iki adet yarı toplayıcı devre ile tasarımına ilişkin ifadeleri gözden geçiriniz. (*bkz. Ayrıt 7.4.1*)

Deneyde Kullanılan Malzemeler:

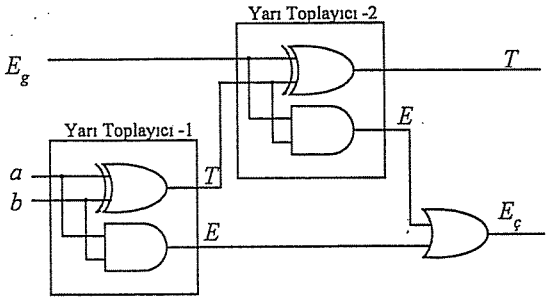
- 1 x 74HC86 tümdevresi; 4'lü 2-girişli XOR (YA DA)
- 1 x 74HC08 tümdevresi; 4'lü 2-girişli AND (VE)
- 1 x 74HC32 tümdevresi; 4'lü 2-girişli OR (VEYA)
- 1 x 74HC283 tümdevresi; 4-bit ikili Tam Toplayıcı
- Besleme kaynağı ve bağlantı kabloları

Deneyin Yapılışı:

a) Yarı toplayıcı devreye ilişkin devre aşağıda verilmiştir. Bu devreyi kurunuz. Yarı toplayıcıya ilişkin doğruluk tablosunu elde ediniz.



b) Tam toplayıcı devreye ilişkin devre şeması aşağıda verilmiştir.



İlk deneyde oluşturmuş olduğunuz devreyi bozmadan, yukarıdaki devreyi kurunuz. Tam toplayıcıya ilişkin doğruluk tablosunu elde ediniz.

c) Deneyden önce yapılacaklar kısmında sorulan 4-bitlik toplayıcı tümdevresi {74HC283} ile tasarladığınız 4-bitlik toplama ve çıkarma devresini kurunuz. Devreyi kullanarak aşağıdaki işlemleri gerçekleştiriniz ve sonuçları yorumlayınız.

1001	1001	0111	0111
+ 0110	- 0110	+ 1011	- 1011

d) Deneyden önce yapılacaklar kısmında sorulan 4-bitlik ALU tümdevresini {74181} Devreyi kullanarak yukarıdaki ilk iki işlemi gerçekleştiriniz ve sonuçları yorumlayınız.

Deney Raporunda Yapılması İstenenler:

1. Herbir deneyde elde ettiğiniz, gözlemlediğiniz sonuçları şekillerle destekleyerek açıklayınız.
2. 4-bitlik toplama ve çıkarma devresinin 2.3.7 de verilen tasarımı ve 74HC283 tümdevresinin kullanılmasıyla gerçeklenen tasarımı kapı yoğunluğu ve maliyet açısından karşılaştırınız.
3. Tasarlamış olduğunuz 4-bitlik toplama-çıkarma devresi ile 8-bitlik toplama-çıkarma devresinin tasarımını gerçekleştiriniz. *Elde* ya da *borç* oluştuğu durumu açıklayınız.

Deney 5.

PLD Uygulamaları

Bu deneyde, Programlanabilir Lojik Düzenekler *(PLD-Programmable Logic Devices)* genel olarak incelenecek ve en çok bilinen/kullanılan elemanlarından biri olan ROM üzerinde deneyler yapılacaktır.

Programlanabilir Lojik Devrelerde amaç, karmaşık lojik işlevleri tek bir devre elemanı kullanarak gerçeklemektir ve buradaki programlanabilirlik özelliği lojik kapılar bağlamında istenilen işlevin sağlanması anlamındadır. Bir Programlanabilir Lojik Düzen, kendi bünyesinde programlanabilir özellikte VE ve VEYA kapılarını barındırmaktadır. Bu kapıları programlama işlemi, değişkenleri kapı girişlerine bağlantısını sağlayan sigortalarla gerçeklenir. Dolayısıyla, bir değişkenin kapı girişinde etkimesi isteniyorsa sigorta olduğu gibi bırakılır, eğer değişken bağlantısı istenmiyorsa o değişkene ilişkin sigorta attırılır. PLD elemanları programlanabilirlik özelliklerine göre, bir kez programlanabilenler ve çok kez programlanabilenler olmak üzere iki sınıfa ayrılırlar. Bir kez programlanabilen PLD elemanları içinde, PROM, PAL ve PLA, çok kez programlanabilen PLD elemanları içinde ise EPROM, E^2PROM, LCA/LGA sayılabilir.

Deneyden Önce Yapılacaklar:

- Kataloglardan 2864 E2PROM'a ilişkin giriş-çıkış ve programlamak için gerekli bağlantıları belirleyiniz.
- Tam Toplayıcı devrenin iki adet Yarı Toplayıcı devreyle gerçekleştirilmesi için gerekli kapı elemanlarını ve bunlara ilişkin tümdevreleri belirleyiniz.
- 4-bitlik toplama ve çıkarma devresini, 74HC283 tümdevresi ile gerçekleştirmek için gerekli bağlantı şemasını oluşturunuz.

Gerekli Önbilgiler:

PLD elemanlarının genel yapılarını, özellikle PROM programlanmasına ilişkin bilgileri inceleyiniz (*bkz. Ayrıt 7.4.1 ve Ayrıt 11.3*)

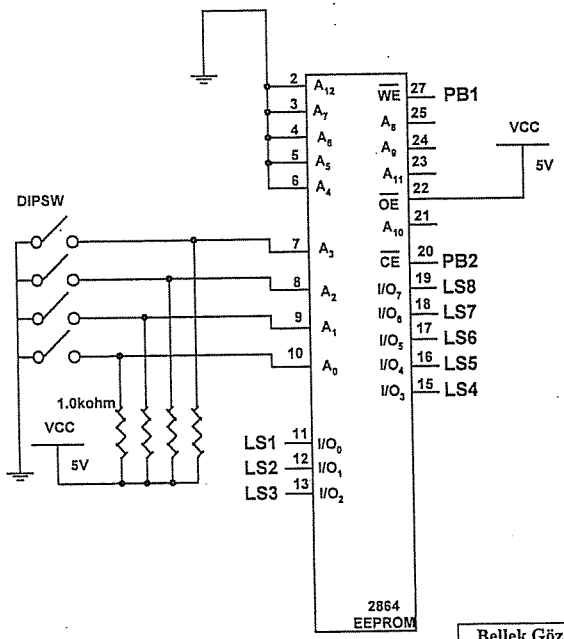
Deneyde Kullanılan Malzemeler:

- 1 x 2864 E2PROM
- 1 x Direnç 1kΩ
- 4 x Anahtar DIP switch
- 1 x Hazır Deney Seti*
- Bağlantı kabloları

* Bu deneyde, C.A.D.E.T veya WishMaker gibi bir deney setinin kullanılması özellikle önerilir.

Deneyin Yapılışı:

a) 2864 Elektriksel Silinebilir ve Programlanabilir E²PROM'u programlayabilmek için gerekli bağlantı şeması aşağıda verilmiştir. Bu bağlantıları Hazır deney seti üzerinde gerçekleştiriniz. Tümdevrede, çizimde gösterilmeyen 14 numaralı uç toprak, 28 numaralı uç ise besleme gerilimi (V_{cc}) içindir. Bu bağlantıları yapınız.



b) E²PROM'un 11-13 ve 15-19 numaralı uçları veri giriş uçlarıdır (LS1-LS8). DIP switch'ler ile E²PROM'a ilişkin düşük anlamlı bellek gözeleri seçilir ve bu adresten itibaren 20 numaralı uç ile $\overline{CE}$ Lojik 0 yapılarak tümdevre aktif hale getirilirken, 27 numaralı uç $\overline{WE}$ yine Lojik 0 yapılarak veri giriş uçlarından girilen bilginin E²PROM'da saklanması sağlanır.

Bellek Gözesi Adresi	Veri
000	30
001	31
002	32
003	33
004	34
005	35
006	36
007	37
008	38
009	39
00A	40
00B	41
00C	42
00D	43
00E	44
00F	45

Bu açıklamalardan yararlanarak ve yanda verilen tabloyu kullanarak verilen adres gözlerinde istenen verilerin saklanmasını sağlayınız, bir başka deyişle E²PROM'u programlayınız.

c) E²PROM'un beslemesini kesiniz. Veri giriş uçlarının bağlantılarını ayırınız. 20 numaralı uç $\overline{CE}$ ve 27 numaralı uç $\overline{WE}$ Lojik 1' e çekilerek 16 tabanında girilen verilerin ASCII olarak okunması sağlatılır.

Bu bağlantıları yaparak çıkış değerlerini kaydediniz.

Deney Raporunda Yapılması İstenenler:

1. Herbir deneyde elde ettiğiniz, gözlemlediğiniz sonuçları şekillerle destekleyerek açıklayınız.
2. 2864 E²PROM'unun katalog bilgilerini inceleyerek bellek kapasitesi, propagasyon gecikmesi, güç harcaması gibi karakteristik değerleri belirleyiniz. Bu değerleri temel lojik kapı elemanları ile karşılaştırınız.
3. Bir tablo ile verilen verileri saklamak amacıyla deneyde kullanılan E²PROM yerine başka hangi programlanabilir devre elemanı kullanılabilir? Bu devrenin özelliklerini ve nasıl programlandıklarını açıklayınız.

Deney 6.

İşaret Üreteçleri Ve Frekans Bölme

İşaret üreteçleri, lojik devre açısından periyodik özellikte kare dalga veya darbe şeklinde ikili işaret üreten birimlerdir; kombinezonsal devrelere giriş işareti, ardışıl devrelere saat işareti olarak uygulanırlar. Frekans bölme ise, var olan periyodik ikili işaretin daha küçük frekans değerine sahip hale getirilmesidir; yani, periyot süresi olan T'nin arttırılmasıdır (frekans değeri=1/periyot süresi).

Deneyden Önce Yapılacaklar:

- Ayrıt-11.5 deki kenar ve düzey tetikleme özellikleri ve Ayrıt 10.1.6 daki Saat işareti ve senkron hücrelerin zaman davranışı başlıklı ayrıtlar okunmalıdır.
- CMOS ve TTL teknolojisiyle üretilmiş TVE kapıları içeren tümdevrelerin uçları ve kapıların tipik gecikme değerlerini araştırınız ve bir yere yazınız. bkz. Ek-A ve Ek-B.
- *Schmitt Trigger* olarak adlandırılan devrenin çalışmasını araştırınız; 7414 içerisinde *Schmitt Trigger* olan bir tümdevredir. 7414'ü inceleyiniz.
- Tek kararlı ikili devre olarak adlandırılan 74121 tümdevresini inceleyiniz.

Gerekli Önbilgiler:

Belirli aralıkla kendisini tekrarlayan işaretlere periyodik işaret denir. Periyodik bir ikili işarette biri darbe (lojik 1) diğeri boşluk (lojik 0) olarak adlandırılan iki kısım vardır. Darbe boşluk oranı yaklaşık olarak eşitse, kare; darbe oranı boşluğa göre çok çok küçük ise darbesel işaret olarak adlandırılır.

Deneyde Kullanılan Malzemeler:

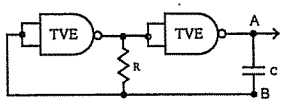
- CMOS ve TTL TVE kapıları: örneğin 7400 ve 4011 tümdevreleri
- Schmitt Trigger devresi: örneğin 7414 tümdevresi
- JK flip-flop'u: örneğin 74109 veya 74112 tümdevreleri
- Tek kararlı ikili devre: örneğin 74121 tümdevresi
- Çeşitli değerde dirençler (R) ve kondansatörler (C).

Deneyin Yapılışı:

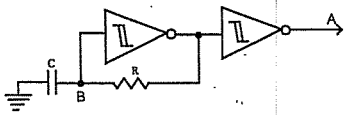
a) CMOS TVE kapılarını kullanarak Şekil-D6.1 deki gibi devre kurunuz; R=10 kΩ, C=10 nF olarak alınız. C'nin değerini sırasıyla, yaklaşık olarak 1, 10, 20, 50, 100 nF yaparak A ve B noktalarındaki işaretlerin değişim şekilleri gözleyiniz ve bir yere çiziniz. C'nin değeri sabit bırakıldığında, örneğin 20 nF, R'nin değeri hangi değere kadar küçültülürse titreşim sönmektedir/işaret üretimi durmaktadır? Belirleyiniz. R'nin başlangıçtaki değeri 10 kΩ

b) Bir önceki deneyi TTL TVE kapısı için tekrarlayınız.

c-) Şekil-D6.2 deki gibi bir devreyi Schmitt Trigger devresiyle (7414) kurunuz; R=330Ω, C=10 nF olarak alınız. C'nin 10, 100, 1000 nF değerleri için A ve B noktalarındaki işaretlerin değişim şeklini gözlemleyiniz ve bir yere çiziniz. C=100 nF için R'nin değerini arttırarak işaret üretiminin sustuğu R değerini belirleyiniz.



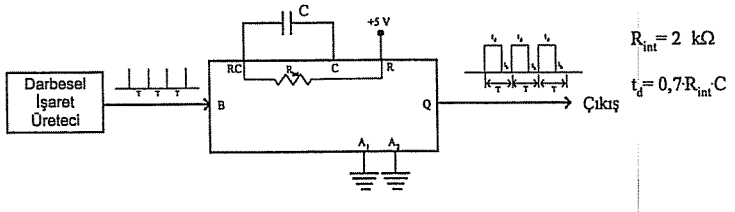
Şekil-D6.1. TVE kapısıyla işaret üreteci.



Şekil-D6.2. *Schmitt Trigger* ile işaret üreteci.

d) Deney a veya b'de elde edilen ikili işareti JK girişleri kısa devre edilmiş ve lojik 1'e çekilmiş bir flip-flop'un saat işareti girişine bağlayınız ve flip-flop'un saat işaretiyle Q çıkışı arasındaki bağıntıyı belirleyiniz. Bir bölme işlemi gerçekleşti mi? Eğer evet ise, işaretin frekansı kaça bölündü? Neden?

e) Şekil-D6.3 deki devreyi kurup A ve B noktalarındaki işaretin değişim şeklini gözleyiniz. Girişe darbe oranı çok kısa olan bir işaret uygulanırsa B noktasındaki işaret nasıl olur belirleyiniz ve yorumlayınız.



Şekil-D6.3. 74121 ile işaret şekillendirici devre.

Deney Raporunda Yapılması İstenenler:

1. Herbir deneyde elde ettiğiniz, gözlemlediğiniz sonuçları şekillerle destekleyerek yazınız.
2. Deney a ve deney b'de elde ettiğiniz işaretlerin frekansları farklı mı? Nedenini açıklayınız.
3. Deney 4'de en basitinden bir ikili işaret frekans bölücü yapılmıştır. Bu şekilde 1 tane flip-flop'la 2'ye bölme yapılabilir; 2 taneyle 4'e, 3 taneyle 8'e ve 4 taneyle 16'ya bölme yapılabilir. Veya, 4 bitlik doğal kodda bir ikili sayıcının saat işareti girişine uygulanırsa sayıcının en anlamlı çıkışı olan Q_4'de frekansı 16'ya bölünmüş işaret elde edilir. Eğer işaretin frekansı 10 bölünmek istenirse kaç tane flip-flop gerekir? Böyle bir frekans bölücü devresini tasarlayınız ve çiziniz.
4. Sabit frekanslı analog işaret üretmek için kristal olarak adlandırılan elemanlar kullanılır. Kristal'in kullanıldığı analog işaret üreten bir devreyi araştırınız ve çiziniz. Çalışma şeklini de açıklayınız.

Deney 7.

Üç-Durumlu Çıkışlar VE Ortak Yol Kullanımı

Karmaşık bir lojik devre birçok eleman içerir ve bu elemanların birbiriyle olan bağlantısını sağlamak kolay olmayabilir. Çünkü birinin çıkışı diğerinin veya diğerlerinin girişlerine bağlanacaktır. Bu şekilde birçok bağlantı hattı olacaktır. Bağlantı hattı sayısını azaltmak ve tasarım esnekliği sağlamak için lojik birimlerin çıkışı ortak yola bağlanacak şekilde tasarlanabilir. Böylece, birimler arasında ortak bir yol *(common bus)* olur ve birimler arasındaki veri aktarımı ortak yolun paylaşılmasıyla sağlanır. Herhangi bir anda yolu bir birim kullanırken diğerleri ya giriş durumunda ya da bekleme durumunda kalırlar. Bu deneyde 3-durumlu *(3-states)* çıkışa sahip birimlerle ortak yol kullanımına ilişkin uygulama yapılacaktır.

Deneyden Önce Yapılacaklar:

- Ortak yol kullanımı sağlayan 3-durumlu çıkışları ve açık kolektörlü çıkışları inceleyiniz; bu amaçla bakınız Ayrıt 7.5.
- 3-durumlu çıkışlara sahip saklayıcı tümdevrelerini inceleyeniz.

Gerekli Önbilgiler:

Ortak yolu süren çıkışlarından yalnızca biri yolu sürmelidir; diğerleri yüksek empedans olan 3. durumda bulunmalıdırlar. Eğer aynı anda iki birim yolu sürerse kısa devre oluşur.

Deneyde Kullanılan Malzemeler:

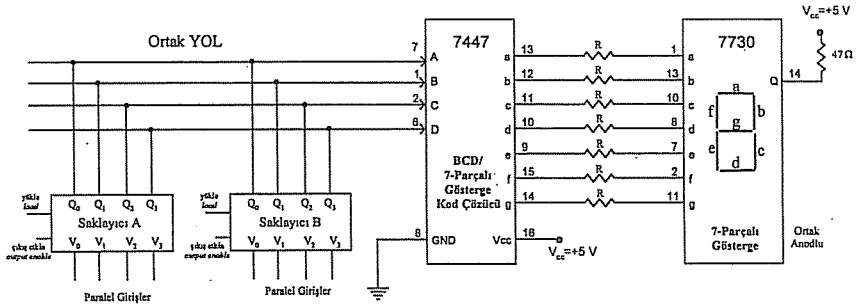
- 4 bitlik,çıkışları 3-durumlu saklayıcı: örneğin 74173 (D flip-flop'u) tümdevresi (2 tane)
- BCD/7-Parçalı Gösterge Kod çözücü: örneğin 7447 tümdevresi
- 7-Parçalı Gösterge (ortak anotlu)

Deneyin Yapılışı:

a) Şekil-D7.1 de görülen devrenin sağ tarafında bulunan kısmını kurunuz. Bu kısım BCD girişleri 7-parçalı göstergede oluşturmalıdır. Örneğin BCD girişler 0001 ise göstergede 1_{10}, 0101 ise 5_{10} yazmalıdır.

b) Saklayıcıları, çıkışları ortak yola bağlı olacak şekilde kurunuz. A saklayıcısına 1, B saklayıcısına 5 değerini yükleyiniz. Ortak yolu, herhangi bir anda yalnızca birinin kullanacağı şekilde sürünüz. A saklayıcısıyla sürüldüğünde göstergede 1_{10}, B ile sürüldüğünde 5_{10} görülmelidir.

c) Heriki saklayıcı da yolu sürmediği zaman göstergede hangi sayı göründüğünü belirleyiniz.



Şekil-D7.1. İki saklayıcının ortak yola erişmesi ve 3-durumlu çıkışlar.

Deney Raporunda Yapılması İstenenler:

1. Deneyde elde ettiğiniz sonuçları ve yorumları şekillerle destekleyerek açıklayınız.
2. Çıkışları 3-durumlu olmayan birimler ortak yol kullanılacak biçimde nasıl bağlanırlar? Hangi yöntemler vardır.
3. Bir ortak yola en fazla kaç tane 3-durumlu çıkış bağlanabileceğini araştırınız; sayı arttıkça devrenin hızında azalma olur mu?
4. Ortak yol kullanmanın olumlu ve olumsuz yanlarını araştırınız ve yazınız.
5. Açık kolektörlü çıkışlarla ortak yol kullanımını bir örnek üzerinde şekil çizerek açıklayınız.

Deney 8.

Saklayıcı ve Sayıcı Uygulamaları

Saklayıcı ve sayıcılar ardışıl devre uygulamasında bolca kullanılırlar; saklayıcı ile sayıcı arasındaki fark, sayıcının içeriğini belirli bir sıra düzeninde değiştirebilmesidir. Bu deneyde, ilk önce saklayıcı uygulaması yapılacaktır; bir SIPO saklayıcıya verinin seri girmesi ve paralel çıkması ve daha sonra bir sayıcının 0'dan başlayarak saydırılması işlemleri gerçekleştirilecektir. Heriki deneyde de çıkışlara bir BCD/7-parçalı gösterge kod çözücü ve 7-parçalı gösterge bağlanacaktır. Böylece çıkışların değeri 7-parçalı gösterge üzerinde görülmesi sağlanacaktır.

Deneyden Önce Yapılacaklar:

- Kitabınızdaki saklayıcı ve sayıcılar kısmına göz atınız; türleri ve çalışma şekilleri deneye başlamadan önce öğrenilmelidir. (bkz. Ayrıt-11.1 ve Ayrıt-11.2)
- BCD kod dönüştürücü ve 7-parçalı kod göstergelere göz atılmalıdır. (bkz. Ayrıt-7.1.4)
- Deneyde kullanılacak tümdevrelerin bacakları ve anlamları katalogdan incelenmelidir.

Gerekli Önbilgiler:

Deneyde kullanılacak saklayıcı, sayıcı, BCD/7-parçalı gösterge kod çözücü devreler hazır tümdevre olarak üretilmişlerdir; üstelik değişik özelliklere sahip birçok türleri vardır. Üniversal bir saklayıcı her tür ötelemeli saklayıcı olarak kullanılabilir: SISO, SIPO, PIPO, PISO gibi. Sayıcılar, önyüklemeli ise saymadan önce sayıcı içerisine saklayıcıda olduğu gibi bir sayı yüklenir ve sayma işlemi oradan başlar.

Deneyde Kullanılan Malzemeler:

- 4 bitlik Ötelemeli Saklayıcı (SIPO-seri giriş paralel çıkışlı): örneğin 74194 tümdevresi
- 4 bitlik Sayıcı: örneğin 74193, 74192, 74191 tümdevreleri
- BCD/7-Parçalı Gösterge Kod Çözücü: örneğin 7447 tümdevresi
- 7-Parçalı Gösterge (ortak anotlu).

Deneyin Yapılışı:

a) Şekil-D8.1 de görülen devrenin sağ tarafında bulunan kısmını kurunuz. Bu kısım BCD girişleri 7-parçalı göstergede oluşturmalıdır. Örneğin BCD girişler 0011 ise göstergede 3, 0111 ise 7 yazmalıdır.

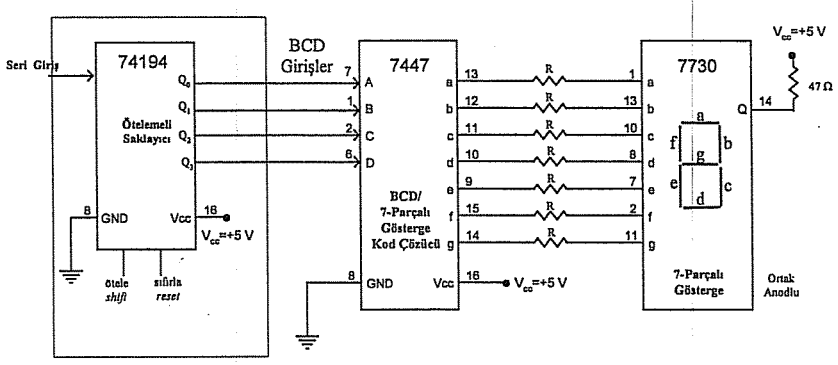
b) Saklayıcı deneyi olarak, Şekil-D8.1 de sol tarafta görülen saklayıcı devresini kurunuz. Saklayıcının sıfırla girişini etkin yapınız; bu durumda 7-parçalı göstergede 0 sayısı belirecektir. Saklayıcıya 0 ile 9 arasında olacak şekilde 4 bitlik sayıyı seri giriş üzerinden bit bit giriniz.

i) Örneğin 0101 sayısını (sağdaki bit en anlamsız) girdiğinizde; ilk bit 1, seri girişi lojik 1 yapın ve ötele girişini tetikleyiniz. Bu durumda saklayıcı içeriği 0001 (1_{10}) olacaktır ve göstergede 1_{10} görülecektir. Seri giriş lojik 0 yapılıp ötele girişi

tetiklendiğinde saklayıcı içeriği 0010 (2_{10}) olacak ve göstergede 2_{10} sayısı görülecektir. 0101 sayısının diğer bitleri içinde benzer işlemi yapıp sonucu değerlendiriniz.

ii) Kurduğunuz devrenin seri girişine sırasıyla 0111, 1001 ve 1111 sayılarının bitlerini giriniz. Herhangi bir anda saklayıcı içerisindeki veri 0 ile 9 arasında bir sayıya karşılık geliyorsa göstergede ilgili sayı görülür; eğer, sayı 9'dan büyük ise ne görülüyor?

c) Şekil-D7.1 deki şekilde kesikli çizgilerle belirtilen yere 4 bitlik yukarı sayıcı koyarak devreyi kurunuz. Sayıcının saat girişine elle darbeler üretiniz ve sayma işleminin gerçeklendiğini görünüz. Saat girişine yüksek frekanslı bir ikili işaret uygulandığında göstergenin tüm led'lerinin sürekli ışıldayarak 8 oluştuğu görülür. Neden?



Şekil-D8.1. Saklayıcı ve sayıcı deneyi lojik şeması.

Deney Raporunda Yapılması İstenenler:

1. Deneyde elde ettiğiniz sonuçları ve yorumları şekillerle destekleyerek açıklayınız.
2. Deney a'da saklayıcı içeriği 0001 olsa ve saklayıcı çıkışlarından en anlamlı olanı, yani Q_3, seri giriş ucuna bağlansa, her tetikleme anı sonrası 7-parçalı göstergede hangi sayılar görüler. Belirleyiniz.
3. Deney b'de 1 hane için yaptığınız ve sayıcı kullandığınız deneyi 2 hane için yapacak lojik devrenin şemasını çiziniz. 09'dan 10'a geçişin nasıl olduğunu açıklayınız. Sayıcı çıkışları 99 olduğunda bir sonraki sayı ne olur?
4. Bir tane BCD/7-Parçalı Gösterge Kod Çözücü kullanılarak birden çok gösterge süren/onlara çıkış veren bir devre nasıl olabilir? Araştırıp lojik şemasını çiziniz.

Deney 9.

Ardışıl Devre Uygulaması

Yalnızca 1 bitlik bilginin tutulduğu devre de, bir mikroişlemcinin iç mimarisi de ardışıl devre özelliğindedir. Ardışıl devrelerde çıkış veya çıkışlar o andaki girişlere ek olarak geçmiş girişlerin belirlediği durumlara da bağlıdır. Çoğu lojik uygulamanın doğası ardışıl devreye dayanmaktadır denilebilir. Bu deneyde bir girişli bir çıkışlı ardışıl devre tasarlanacak ve gerçekleştirilecektir; devrenin işlevi şifre çözme veya yakalama olarak adlandırılabilir.

Deneyden Önce Yapılacaklar:

- Şekil-D9.2 deki bir girişli bir çıkışlı devre kurulacaktır. Devrenin fonksiyonu şöyledir: girişine sırasıyla 0, 1, 1, 0 bit dizisi geldiğinde (0110 olarak gösterilir) çıkışa bağlı led ışık saçacak; aksi durumda led sönük olacaktır. Bu işi yapacak ardışıl devreyi JK flip-flop'ları ve VE, TVE, VEYA kapıları kullanarak tasarlayınız (tasarım için bkz. Ayrıt-10.5). Aynı işi, ötelemeli saklayıcıyla gerçekleştirecek şekilde tasarlayınız ve deneye başlamadan önce belirleyiniz. (tasarım için bkz. Ayrıt-12.2)

Gerekli Önbilgiler:

Ardışıl devre tasarımında, ilk önce durum diyagramı ve durum tablosu elde edilmelidir. Devrenin durum sayısı, kaç tane durum değişkeni olacağını belirler; herbir durum değişkeni için bir tane flip-flop kullanılır. Örneğin 4 tane durum varsa 2 tane flip-flop, 5 ile 8 arasında durum varsa 3 tane flip-flop gerekir.

Led'lerin ışık saçması doğru yönde kutuplanması gerekir; yani, üzerinden akım geçirmesi gerekir. Uçları katot ve anot olarak adlandırılır; katot ucu daha düşük gerilim seviyesine bağlanmalıdır.

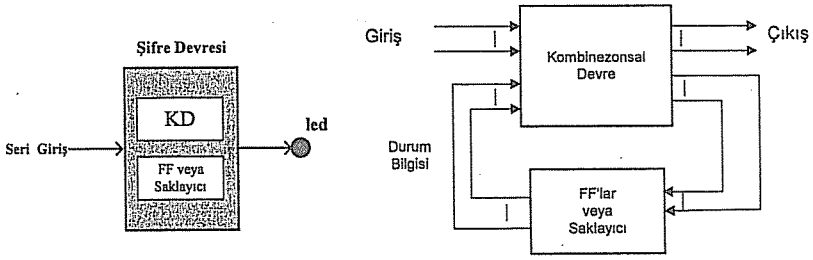
Deneyde Kullanılan Malzemeler:

- JK flip-flop'ları: Örneğin 7474 tümdevresi
- Kapılar; VE, TVE, VEYA: örneğin 7408, 7400 ve 7432 tümdevreleri
- 4 bitlik ötelemeli saklayıcı: örneğin 74198 tümdevresi.
- 1 tane led: örneğin kırmızı veya yeşil bir led.

Deneyin Yapılışı:

a) JK flip-flop'larıyla tasarladığınız devreyi kurunuz. Çeşitli kombinezonlar vererek doğru çalışıp çalışmadığını sınayınız. Doğru şifre, yani 0110, arda arda iki kez girilirse devrenin davranış nasıl olmaktadır?

b) Ötelemeli saklayıcıyla tasarladığınız devreyi kurunuz. Çeşitli kombinezonlar vererek doğru çalışıp çalışmadığını sınayınız. JK flip-flop'uyla tasarlanan devreyle aynı davranışı gösterip göstermediğini sınayacak örnek şifreler giriniz.



İstenen şifre devresinin genel gösterilimi — Ardışıl devrelerin genel mimarisi

Şekil-D9.1. Şifre devresi giriş ve çıkışı.

Deney Raporunda Yapılması İstenenler:

1. Deneyde elde ettiğiniz sonuçları ve yorumları şekillerle destekleyerek açıklayınız.
2. Ardışıl devre olarak tasarlanması istenen bir şifre devresinin girişine ard arda 1100111 gelirse çıkışını lojik 1 yapması istenmektedir; aksi durumda çıkışı lojik 0 olmalıdır. Bu devreyi ötelemeli saklayıcı kullanarak ve D flip-flop'u ile belle kullanarak ayrı ayrı tasarlayınız ve lojik şemasını çiziniz.
3. Yeteri kadar büyüklükte ikili sayıcı kullanılarak 2 nokta arasındaki mesafenin ne kadar sürede alındığını/gidilebildiğini ölçen bir devre nasıl yapılabilir; düşüncenizi kavramsal olarak şekillerle destekleyerek açıklayınız.

Deney 10.

ADC Uygulamaları

Analog işaretlerin sayısal sistemlerinde değer olarak kullanılabilmesi için bir dönüşüm yapılmalıdır; yani, analog değerden örneklemeler alınıp onlara karşılık gelen sayısal kodlar üretilmelidir. Bu amaçla ADC *(Analog-Digital-Converter)* olarak adlandırılan birimler kullanılır; tersi olarak da sayısal koddan analog işaret elde etmek için de DAC *(Digital-Analog-Converter)* olarak adlandırılan birimler kullanılır. Bu deneyde, analog işaretin sayısala dönüştürülmesi uygulaması yapılacaktır.

Deneyden Önce Yapılacaklar:

- Ayrıt 2.2 de anlatılan Analog/Sayısal işaret dönüşüm konularına göz atınız. Dönüştürmeyle ilgili kavramları öğreniniz: dönüştürme hızı, duyarlılık vs gibi.

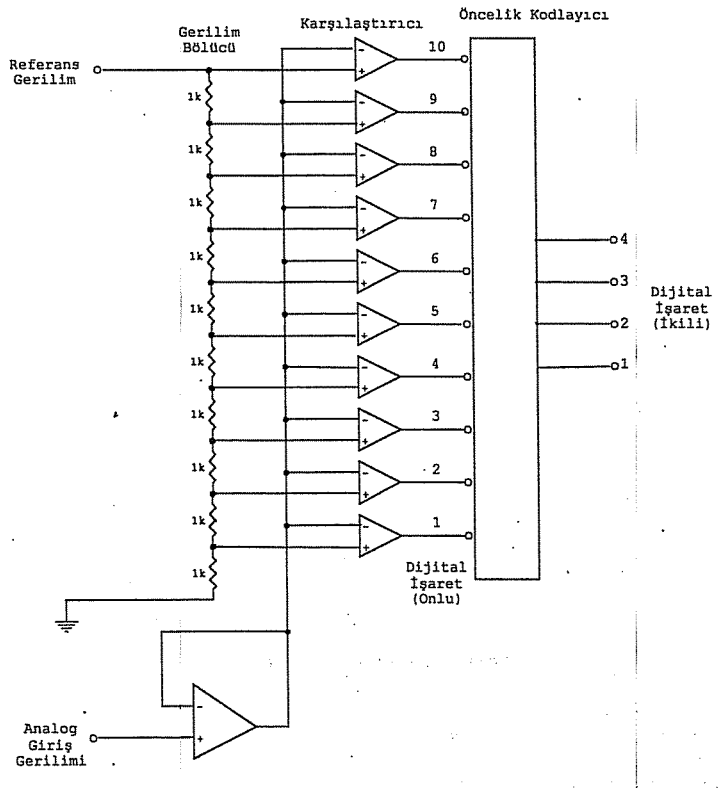
Gerekli Önbilgiler:

Genel olarak, paralel ve seri olmak üzere iki çeşit ADC yapısı bulunmaktadır. Şipşak *(Flash)* ADC'ler paralel, ardışıl ADC'ler seriye örnek olarak verilebilir; paralel ADC'Ler karmaşık yapılarına karşın daha hızlı dönüştürme yaparlar; Şekil-D10.1 de şipşak ADC'nin iç yapısı verilmiştir: Herbir karşılaştırıcı girişinde gerilim bölücülerle bölünmüş bir gerilim ve dönüştürülecek analog işaret bulunur. Karşılaştırıcının çıkışında bir ondalık kodlayıcı kullanılarak kodlanmış dijital işaret elde edilir.

Ardışıl Yaklaşımlı ADC, çok bilinen ve kullanılan ADC türüdür. Ardışıl Yaklaşımlı ADC'nin dönüştürme hızı yaklaşım yöntemine göre sabit olabildiği gibi giristeki analog işaretin gerilim düzeyine de bağlı olabilir. Eğer doğrusal yaklaşım yöntemi kullanırsa dönüştürme hızı giristeki analog işaretin gerilim düzeyine bağlı olur. Çünkü 0'dan başlanarak ardışıl yaklaşıldığı için gerilim düzeyi düşükse çabuk yaklaşılır/bulunur; gerilim düzeyi yüksekse daha çok sayma adımı gerekeceğinden dönüşme daha çok saat işareti gerektirir. Bölmeli ardışıl yaklaşım kullanılırsa daha hızlı dönüştürme yapılır. Burada, bilinmeyen bir sayının verilen cevaplara büyük-küçük denilerek sıkıştırılıp bulunması gibi bir yöntem kullanılır. Örneğin 1 ile 64 arasındaki bir sayı, verilen cevaplara büyük veya küçük olduğu söylenmesi durumunda en kötü koşulda 6 söylemede bulunabilir. Bu doğrusal yaklaşımla bulunmak istenseydi, en kötü durumda 64 söylemede bulunurdu; 64'e karşılık 6 sabit sayılabilir.

Bölmeli Ardışıl Yaklaşımlı bir ADC'nin iç yapısı Şekil-D10.2 de verildiği gibidir: içerisinde kontrol lojiği, ardışıl yaklaşım saklayıcısı (SAR), karşılaştırıcı, DAC ve çıkış işaretini tutan bir adet Tutucu/saklayıcı bulunur. Doğrusal ardışıl yaklaşımlı ADC'nin mimarisi de aynıdır; tek fark, saklayıcı yerine, 0'dan başlayıp dönüştürme değerine kadar sayan sayıcı kullanılmasıdır.

Bölmeli ardışıl yaklaşımda dönüştürme işlemi şöyle gerçeklenir: SAR saklayıcısının en yüksek anlamlı bitine yüklenen değer DAC tarafından analog işarete çevrilir. Sonra da dönüştürülecek analog gerilim ile karşılaştırılır. Eğer SAR'daki değer büyük ise SAR içeriği temizlenir. Eğer dönüştürülecek gerilim büyük ise en yüksek anlamlı bit olduğu gibi korunurken, bir sonraki en yüksek anlamlı bit'e bir başka değer yüklenir... Bu işlemler en az anlamlı bit için de yapılana kadar tekrar edilir. En az anlamlı bit de işlendikten sonra dönüştürülecek işaretin sayısal değeri SAR içerisindedir; içeriği çıkıştaki tutucuya/saklayıcıya aktarılır. Bu tutucunun / saklayıcının çıkışları, genel olarak, üç durumlu olup ortak yol uygulamaları için uygundurlar.



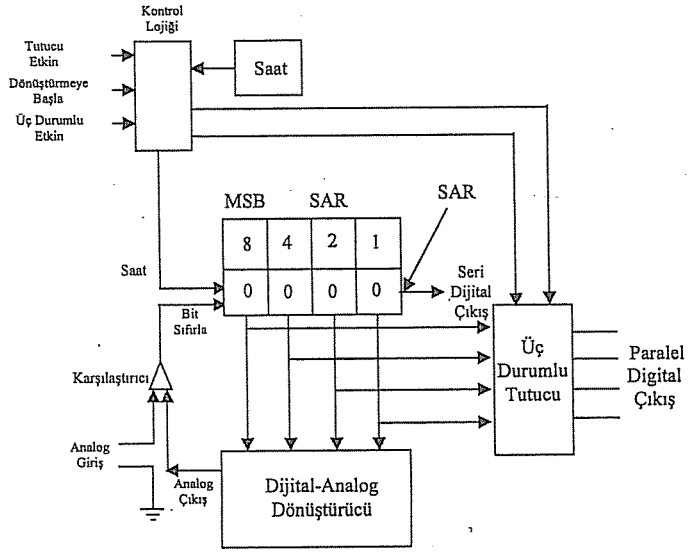
Şekil-D10.1. Şipşak (Flash) ADC İç yapısı.

Deneyde Kullanılan Malzemeler:

- 8 bitlik ADC: örneğin ADC0809 tümdevresi
- Potansiyometre: örneğin 20 kΩ doğrusal özellikte
- Direnç: 7 tane 4.7 kΩ
- Multimetre

Deneyin Yapılışı:

Buradaki deneyler, 8-bitlik analogtan sayısal dönüştürücü olan ADC0809 tümdevresi kullanılarak gerçekleştirilecektir. Bu ADC, ardışıl yaklaşımlı bir dönüştürücü olup, 8-bitlik üç durumlu tutucu çıkışına sahiptir. ADC0809, 0-5 V DC işaret girişlerine izin verme olup, 640 kHz'lik saat işaretinde, çevirme işlemini 100 µs'de tamamlamaktadır.



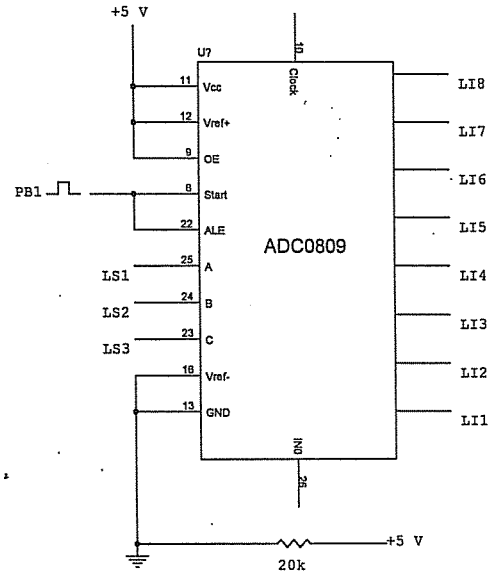
Şekil-D10.2. Ardışıl yaklaşımlı ADC iç yapısı.

a) *ADC-0809* tümdevresini deney setine yerleştiriniz. Tümdevrede, 13 numaralı uç toprak ve 11 numaralı uç besleme gerilimidir (V_{cc} = +5 V DC). Şekil-D10.3 de verilen bağlantıları yapınız.

b) LSI-LS2-LS3 anahtarlarını lojik 0 seviyesine çekiniz. Saat işaretinin frekansını 100 kHz'e ayarlayınız. ADC-0809'un analog gerilim giriş ucu olan 26 numaralı ucuna Voltmetreyi bağlayınız.

c) ADC-0809'un 26 numaralı ucundaki gerilimi 2.50 V DC'ye ayarlayınız. Başla (Start) ve ALE girişlerini lojik 1'e çekerek aktif hale getiriniz. Bu durumda LI8 çıkışının lojik 1 seviyesinde olması beklenir. Yani LI8'de bulunan LED'in yanması gerekir, eğer bu sağlanmıyorsa LI1-LI8 LED'lerinin yaklaşık olarak 128 değerini sağlıyor olması gerekmektedir.

d) *ADC-0809*'un 26 numaralı ucuna çeşitli analog gerilim değerleri uygulayınız ve LI1-LI8 dijital çıkışlarındaki LED konfigürasyonu belirleyiniz. Gerilim değerlerinin değişimiyle dijital değerlerin değişimini göz önünde bulundurarak ADC'nin duyarlılığını belirleyiniz.



Şekil-D10.3. ADC-0809 uç bağlantı şekli.

Deney Raporunda Yapılması İstenenler:

1. Deneyde elde ettiğiniz sonuçları ve yorumları şekillerle destekleyerek açıklayınız.
2. Bir şirkette iki sistem çalışmaktadır; bir tanesi odanın sıcaklığını kontrol etmekte ve sabit değerde tutmaya çalışmakta, diğeri ise çalışanlarını sesleriyle tanıyarak otomatik geçiş sisteminde kullanılmaktadır. Her iki sistemi kontrol eden cihazlarda yer alan ADC'lerin özellikleri için ne söyleyebilirsiniz? Konuyu detaylı araştırarak bu uygulamalar için uygun ADC'leri özellikleriyle birlikte veriniz.
3. Deneylerde kullanılan Analog-Sayısal dönüştürme işleminin duyarlılığını arttırmak için nasıl bir çözüm önerirsiniz? Bunun için bir tümdevrenin bulunup bulunmadığını araştırınız ve özelliklerini yazınız.

Kaynakça

[1] Balasubramian, K., Altun, Z.G., Çölkesen, R., and, Özer, G., ***An Intelligent Data Acquisition and Management Card for the PC***, Laboratory Microcomputer-Incorporating Science Software, v.12, n.3, 1993.

[2] Çölkesen, R., ve Örencik, B., ***On the Design of V.42bis Data Compression and Decompression Chip***, Proceedings of the 6th International Conference od Microelectronics (ICM'94), İstanbul, September 1994.

[3] Çölkesen, R. ve Aliefendioğlu, O., ***Bir Yorumlayıcı Tasarımı ve X-Y Tablası Üzerinde Uygulaması***, Otomasyon ve Bilgisayar Sempozyumu, İstanbul, 1994.

[4] Çölkesen, R., ***An Architecture Simulation: Usuing Object-Oriented Concepts as an HDL***, teknik doküman, İTÜ, 1995.

[5] Çölkesen, R., Balasubramian, K., and Altun, Z.G., ***A Micro-processor Conrolled Versatile 3-D Motion Control System Incorporating Hash Function Command Interpreter***, Laboratory Microcomputer-Incorporating Science Software, .

[6] Danilels, Jerry D., ***Digital Design from Zero to One***, John Wiley & Sons Inc., 1996.

[7] Electronics WORKBENCH, ***multiSIM: Getting Started & Tutorial***, Interactive Image Technologies Ltd., 1999.

[8] Harmancı, E., ***Lojik Devreleri Dersleri***, İTU Elektronik Fakültesi, 2. baskı, 1988.

[9] Horsey, M.P., ***Electronics Projects Using Electronics WORKBENCH***, Newnes, 1998.

[10] Mano, M., ***Digital Design***, Prentice Hall Int. Inc., 2nd Edition, 1991.

[11] SGS-THOMSON, ***Low Power Schottky TTL ICs Databook***, 1st Edition, 1991.

[12] Örencik, B., ***Lojik Tasarım Lab. Deney Kılavuzu***, İTÜ Elektrik-Elektronik Fakültesi, 1987.

[13] Ünalan, E., ***Lojik Devreleri: Cilt I: (Kombinasyonal Devreler)***, *Cilt-II (Ardışıl Devreler)*, İTÜ Elektrik Elektronik Fakültesi, 1988.

[14] Taub, H. and Schilling, D., ***Digital Integrated Electronics***, McGraw-Hill, 1977.

[15] Toros, Z., Çölkesen, R., ***Message Display System: a VLSI Design***, ICTP-INFN Second Course on Basic VLSI Design Techniques (a project design), Trieste, 1991.

[16] Kollektif (Editör: Çölkesen, R.), ***Bilgisayar Mühendisliğine Giriş***, Papatya Yayıncılık Eğitim, İstanbul, 2007.

www.papatya.info.tr

Yazarlarımız

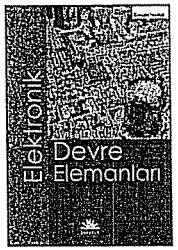
Taner ARSAN (Dr.) - Kadir Has Üniversitesi

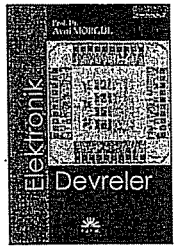
Taner Arsan, 1968 yılında Erzincan'da doğdu. 1990 yılında İTÜ Elektronik ve Haberleşme Mühendisliği Bölümünden mezun oldu. Yüksek Lisans öğrenimini 1994 yılında, doktora öğrenimini ise 1999 yılında İTÜ Kontrol ve Bilgisayar Mühendisliği Bölümü'nde tamamladı. 1997-1998 yılları arasında TÜBİTAK Bilimadamı Yetiştirme Grubu'ndan aldığı burs ile University of Glasgow'da doktora konusu üzerinde çalışmalar yaptı. 1991-2000 yılları arasında İTÜ Kontrol ve Bilgisayar Mühendisliği Bölümünde Araştırma Görevlisi olarak çalıştı. 2000 yılında Kadir Has Üniversitesi Bilgisayar Mühendisliği Bölümü'nde Yardımcı Doçent Doktor olarak göreve başladı. 2006-2007 yılları arasında İsviçre'de Ecole Polytechnique Fédéral de Lausanne (EPFL) - Signal Processing Institute'de doktora sonrası çalışmalar yaptı. Halen Kadir Has Üniversitesi Bilgisayar Mühendisliği Bölümünde Bölüm Başkan Yardımcılığı ve Rektör Danışmanlığı görevlerini sürdürmektedir. Öngörülü Kontrol, Lojik Devreler, Mikroişlemcili Sistem Tasarımı, Bilgisayar Ağları ve Yazılım Mimarisi Tasarımı konuları ilgi alanları içinde yer almaktadır.

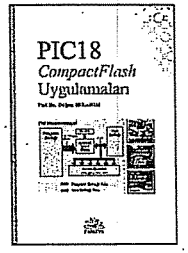
Rifat ÇÖLKESEN (Dr.) - Post-Edu Enstitüsü

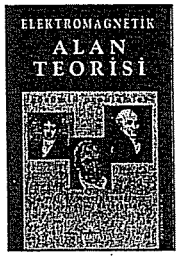
İstanbul'da bir üniversitede öğretim üyesi olarak çalışmalarını sürdürmektedir. Eğitimi: Elektronik ve Haberleşme Mühendisi (KTÜ), Bilgisayar Yüksek Mühendisi (İTÜ), doktora çalışmalarını İstanbul Teknik Üniversitesi, ICTP Microprocessor Lab. (Trieste-İtalya) ve Çukurova Üniversitesinde yaptığı çalışmalarla tamamladı. Profesyonel yaşam: Yüksek Lisans ve doktora öğretimi/çalışmaları süresince İTÜ Vakfı eğitim bölümünde bilgisayar programlama dersleri verdi; ve dört yıl kadar bilgisayar sektöründe "araştırma ve proje müdürlüğü" görevinde bulundu. Bu süre içerisinde "Bilgisayar Ağları ve İnternet Teknolojileri" konusunda katma değeri yüksek birçok proje tasarladı; algoritmalar ve bilgisayar ağları çalıştığı ana konulardır. Akademik kitaplar basan bir yayınevinde yayın danışmanlığı ve hakem kurulu başkanlığı görevini de yürütmektedir. Yayınları: Bilgisayar Mühendisliği ve çalışma/araştırma yaptığı konularda çeşitli kitaplar yazdı: "Bilgisayar Haberleşmesi ve Ağ Teknolojileri (Bülent Örencik ile)", "İşte C programlama Dili", "Veri Yapıları ve Algoritmalar", "Programlama Sanatı ve Algoritmalar" ve "Bilgisayar ve Elektronik Mühendisliği için Lojik Devre Tasarımı (Taner Arsan ile)" adlı kitapları en çok bilinen eserlerdir. Türkçeye aşık olup Türkçenin iyi bir bilim dili olacağına inanmaktadır; kitaplarını da, bilimselliğe ek olarak alanında Türkçenin güzel kullanılmasını yaygınlaştırmak amacıyla yazmaktadır. Toplam 10 civarında kitabı ve 30'un üzerinde ulusal ve uluslararası yayını vardır. Diğer: 7 kitabın editörlüğünü ve Türkiye Bilişim Ansiklopedisi'nin baş editörlüğünü yapmıştır.

www.papatya.info.tr









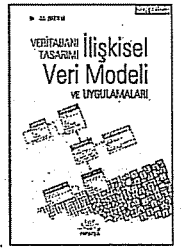
ALGORİTMA
geliştirme
VERİ yapıları

Veri Yapıları ve Algoritmalar

Programlama
Sanatı Algoritmalar

Sistem Analizi
ve Tasarımı

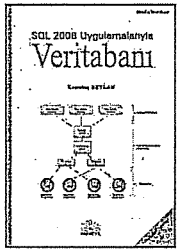
SİSTEM ANALİZİ
TASARIM


VERİTABANI TASARIMI İlişkisel
Veri Modeli
VE UYGULAMALARI

Yazılım Mühendisliği
Yöntemleri - İleri Konular

Yazılım
Mühendisliği
Temelleri

Yazılım Test
Mühendisliği


SQL 2008 Uygulamalarıyla
Veritabanı

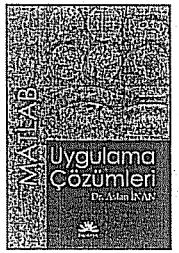
veri
madenciliği

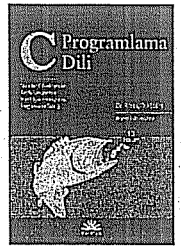
Veri
Madenciliği
Yöntemleri

JAVA

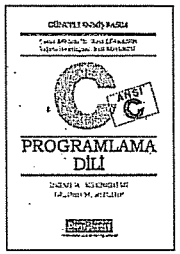
JAVA
Programlama Dili
YAZILIM TASARIMI

MATLAB
Kılavuzu


MATLAB
Uygulama
Çözümleri


C Programlama
Dili

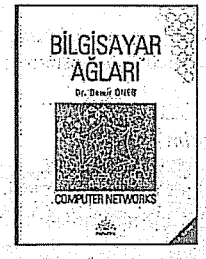
C# Programlama Dili
Yazılım Tasarımı


PROGRAMLAMA
DİLİ

YAZILIM
MÜHENDİSLİĞİ

Bilgisayar
Mühendisliğine
Giriş

BİLGİSAYAR AĞLARI ve
İNTERNET MÜHENDİSLİĞİ


BİLGİSAYAR
AĞLARI
COMPUTER NETWORKS

Bilgisayar
Haberleşmesi
ve Ağ Teknolojileri

post-EDU®
institute

üniversitenin de ötesi...

Öğrenim sonrası eğitim veren özel sektörel eğitim enstitüsüdür. Amacı, Üniversite veya Meslek Yüksek Okulu mezunlarını sektörün aradığı katma değeri yüksek olan bireyler haline getirmektir; üniversite öğretimini sektörel açıdan tamamlamaktadır. Bu amaçla hem diploma programları düzenlemekte hem de bağımsız dersler/kurslar açmaktadır. Bilişim ve İşletme ana eğitim alanlarıdır:

Bilişim & Bilgisayar
İş Dünyası & İşletme
İnsan & İnsan Bilimleri
Temel Bilimler & Teknik

Programlar

- Bilişim Uygulama Uzmanı
- Bilişim Satışçılığı
- Yazılım Tasarımcısı
- Donanım Tasarımcısı
- Network Uygulama Uzmanlığı
- Internet/ISP Uygulama Uzmanlığı
- Bilgi İşlem Merkezi Yöneticiliği
- E-İş Uygulama Uzmanlığı
- MIS Uzmanlığı
- Satış Yöneticiliği
- Satış Uzmanlığı
- İnsan Kaynakları Yönetimi

Programlar

Konusunda doktoralı, uzman veya isim olmuş eğitmenler tarafından uygulamalı ve teorik anlatımlarla kısa zamanda yüksek katma değer... Her kursa/derse ait özel hazırlanmış kitaplar ve yardımcı dökümanlarla standart üstü eğitim. Kişilik testiniz ve meslek seçiminizde yol gösterme; hem eğitim hem motivasyon...

sektörel eğitim enstitüsü

bilgi için
www.post-edu.info
info@post-edu.info
212-249 59 08

Dizin

10 tabanı 36
10'dan *n*'e dönüşüm 45
16 tabanı 36, 39
2'li taban 35
  2'den 16'ya dönüşüm 52
  2'den 8'e dönüşüm 52
3-durumlu eleman 192
  devre deneyi 376
74 serisi tümdevreler 351
7400/7402/7404/7408 354
74138/74151/72283 355
74194 356
7432 354
7474 356
7486 354
7490 356
7-parçalı gösterge 188

Aa

ADC 21, 27
  ADC deneyi 382
  ADC mimarisi 28
Aiken kodu 71
alfanumerik kodlar 82
algoritmik model 16
algoritmik yaklaşım 304, 314
ALU 186
ana uydu flip flop 257
analog işaret 14
analog sayısal dönüşüm 27
AND 88
ANSI standardı 161
ardışıl devre 15
  analiz yöntemi 260
  deneyi 380
  tasarım yöntemi 260
  yapısı 230
aritmetik işlemler 38
  aritmetik çıkarma devresi 167
  aritmetik işlem deneyi 369
  aritmetik mantık birimi 186
  aritmetik toplama devresi 167
ASCII 83
  tablosu 85
asenkron ardışıl devreler 232
ASIC 20
asosiyatif bellek 296
aynı tür kapı kullanılması 148

Bb

baskılı devre 345
BCD 67
  BCD çıkarma 69
  BCD sayı 39
  BCD sayıcı 284
  BCD toplama 68
bellek 18, 287
benzetim ortamları 19, 327
binary 37
birleşim 99
bit 21
bitişik kodlar 73
Boole cebri
  aksiyomları 101
  fonksiyonları 107
  teoremleri 101
boş küme 100
boyna fazlalık sınaması 77
Büyük O gösterilimi 210
Byte 21

Cc - Çç

CAM 287, 296
CMOS 19, 350
  CMOS tümdevreler 359
CRC 79
çarpımların toplamı 148
çevrimli fazlalık sınaması 79
çıkarma işlemi ve tümleme 56

Dd

D flip-flop 254
DAC 21, 27, 32
dağıtıcılar 179
de Morgan kuralı 102
decoder 182
değişkende fazlalık özelliği 102
DEMUX 179
devre elemanları 161
donanım tanımlama dili 21
DTL 350
durum diyagramı 234
durum indirgemesi 241
durum tablosu 235
düzey tetikleme 20

Ee-Ff

E2PROM 287
ECL 350
eksik *Boole* işlevi 144
eleman karmaşıklığı 204, 208
EMO 388
encoder 184
EPROM 287, 296
eşdeğerlilik tablosu 241
eşlik biti 76
etkisizlik 192
evrensel küme 100
excess-3 70
flip-flop'lar 253
D flip-flop 254
JK flip-flop 255
SR flip-flop 255
T flip-flop 256
frekans bölme deneyi 374

Gh-Hh

görüşe dayanarak indirgeme 122
Gray kodu 73, 76
halka sayıcı 284
Hamming kodlaması 81
Hamming uzaklığı 74
hata onarma/sezme 76
HDL 21
hexadecimal 39
HTL 350

Iı-İi-Jj

IEEE-1973/1984 standardı 161
ikili işaret 25
ikili kodlanmış ondalık gösterim 67
ikili sayı sistemi 37
ikili sayıcı 284
ikili taban 36
indirgeme deneyi 362
indirgeme görüşe dayanarak 122
indirgeme *Karnaugh* diyagramı 124
işaret üretici deneyi 374
işaretlerin sınıflanması 23
işaretsiz sayı 40
işaretli sayı 40, 53
işlemci tasarımı (örnek) 320
JK flip-flop 255

Kk

kanonik biçim 109
kanonik model 16
kanonik yaklaşım 304
karmaşıklık 200
hesabı 204
Karnaugh diyagramı 124
kart 20
katalog bilgileri 349
kayan noktalı 40
kenar tetikleme 20
kenar ve düzey tetikleme 248
kesişim 99
kod çözücü 182
kod tablosu 14
kodlama sayısal kodlama 66
kodlama 14, 30, 64
kodlanmış sayısal işaret 25
kodlayıcı 184
kombinezonsal devreler 15, 160
deneyi 367
elemanları 160
tasarımı 162
komşuluk kavramı 129
kontrol sistemi 315
kuantalama 29
küme kavramı 97
kümeler cebri 97
kümeler cebrinde tümleme 99

Ll

latch 246, 258
logic monitor 342
lojik devre benzetimi 339
lojik kapı elemanları 98
lojik kapılarla gerçekleştirim 116
lojik probe 342
LRC 77

Mm-Nn

m'den *n*'ye dönüşüm 50
maksimum terimler 107
maliyet 20, 201
hesabı 202
Mealy makinası 231
minimum terimler 107
modüler yaklaşım 304
Moore makinası 231
Mors alfabesi 64
MOS 350
multiSIM 328
MUX 74
n'den 10'a dönüşüm 42
NAND 93
nibble 21
NOR 94
NOT 91

Oo-Öö

O notasyonu 210
octal 39
on tabanı 36
onaltılı taban 36
ondalık sayıcı 284
OR 90
ortak yol kullanım deneyi 376
örnekleme 28
ötelemeli saklayıcı 279

Pp-Rr

PAL 216
iç yapısı 219
PIPO 280
PISO 280 280
PLA 216
iç yapısı 223
PLD 214
PLD deneyi 371
programlanabilir ardışıl devreler 247, 298
programlanabilir kombinezonsal devre 214
PROM 215, 287, 295
iç yapısı 217
propagasyon gecikmesi 358
Quine-McCluskey yöntemi 139
RAM 247, 287, 289
ripple sayıcı 284
ROM 294
RTL 350

Ss-Şş

saat işareti davranışı 250
sabit noktalı 40
saklayıcı 18, 246, 278
saklayıcı deneyi 378
sayı sistemleri arasında dönüşüm 42
sayıcı 246, 282
sayıcı deneyi 378
sayıların gösterimi 40
sayının genel ifadesi 42
sayı sistemleri 35
sayısal işaret 14, 25
seçiciler 174
sekizli sayı sistemi 39
sekizli taban 36
senkron ardışıl devreler 232
Shannon Teoremi 102
SIPO/SISO 280
sonlu durum makinaları 238
SR flip-flop 255
standart tasarım birimleri 245
şifre çözme 239

Tt

T flip-flop 256
taban-1'e göre tümleme 55
tabana gör tümleme 53
tam çıkarıcı 171
tam toplayıcı 168
toplamların çarpımı 151
transistör 17
TTL 19, 350
TTL özellikleri 357
TTL ve CMOS deneyi 364

tutucu 28, 246, 258
tümleme 53
  işlemi 91
  kapısı 146
  taban-1'e göre tümleme 55
tümleşik devreler 167
tümleyen aritmetiği 38
tümleyen-ve/veya 93, 94
türetilen işlemler 93
türetilmiş kapılar 146
Türkçe karakter tablosu 84
TVEYA 94
TYA DA 96

Uu-Üü-Vv

Unikod 84
üç fazlalık kodu 70
VE işlemi 88
veri altyapısı 315
VEYA işlemi 90
XNOR 96
XOR 95

Yy-Zz

YA DA 95
yarı çıkarıcı 170
yarı toplayıcı 168
yutma özelliği 102
yüksek empedans 192
yürüyen ışık 343
zaman karmaşıklığı 204

www.papatya.info.tr

Türkiye'nin İnternet Kitapçısı
www.tdk.com.tr