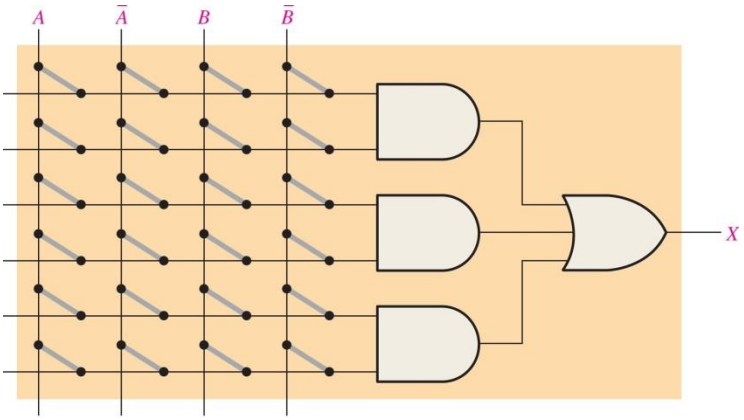
Programlanabilir Mantık

Basit programlanabilir mantık cihazlarının iki türü (SPLD-simple programmable logic devices) PAL ve GAL'dir. PAL, programlanabilir dizi mantığı ve GAL, genel dizi mantığı anlamına gelir. Bir PAL bir kereye mahsus programlanabilir (OTP-one-time programmable) bir cihazdır, GAL ise yeniden programlanabilir bir PAL dir. Hem PAL'lerin hem de GAL'lerin temel yapısı, çarpımların toplamı mimarisine sahip; programlanabilir bir VE dizisi ve sabit bir VEYA dizisidir.

SPLD: PAL

Bir PAL, VEYA kapı dizilerine bağlanmış programlanabilir VE kapısı dizilerinden oluşur. Genel olarak, PAL'ler sigorta işlemi teknolojisi ile uygulanır ve bu nedenle bir defalık programlanabilir (OTP). PAL yapısı, uygulanacak tanımlanmış değişkenlerle, herhangi bir çarpımların toplamı (SOP) mantığı ifadesini sağlar. Herhangi bir kombinasyonel mantık işlevi SOP formunda ifade edilebilir. İki girişli ve bir çıkışlı basit bir PAL yapısı Şekil 10.1'de gösterilmektedir; çoğu PAL'de birçok giriş ve birçok çıkış vardır. Programlanabilir bir dizi, esas olarak her bir çapraz noktada programlanabilir bir bağlantıya sahip satırlar ve sütunlar oluşturan iletkenlerin bir ızgarası veya matrisidir. Her bir programlanabilir bağlantı, PAL'e ait bir sigorta yapısıdır. Her satır bir VE kapısının girişine bağlanır ve her sütun bir giriş değişkenine veya tamamlayıcısına(DEĞİL) bağlıdır. Sigorta bağlantısının varlığını veya yokluğunu programlayarak, istenen herhangi bir çarpımların toplamını oluşturmak için herhangi bir giriş değişkeni veya tamamlayıcı kombinasyonu bir VE kapısına uygulanabilir. VE kapıları VEYA kapısına bağlanarak çarpımların toplamı çıkışını oluşturur.

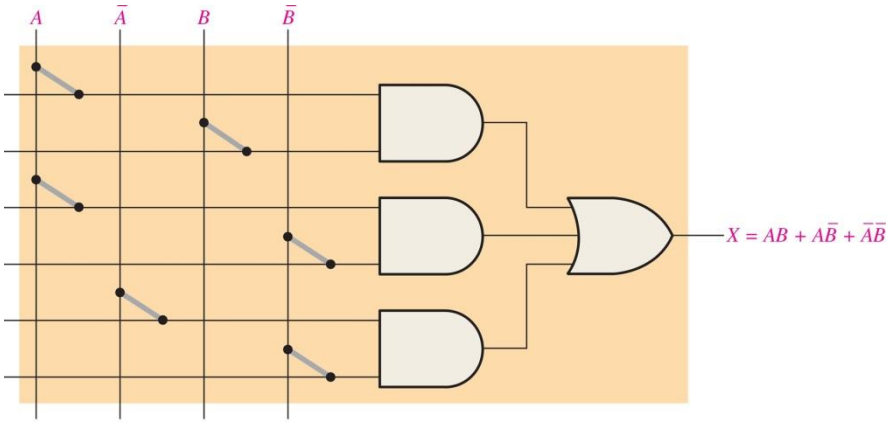


Şekil 10.1

Çarpımların toplamı ifadesini uygulama

Basit bir PAL örneği, Şekil 10-2'de gösterilmiştir. Burada AB çarpım terimi en üstteki VE kapısı tarafından üretilir, $A\bar{B}$ ortadaki VE kapısı tarafından ve $\bar{A}\bar{B}$ alttaki VE kapısı tarafından üretilir.

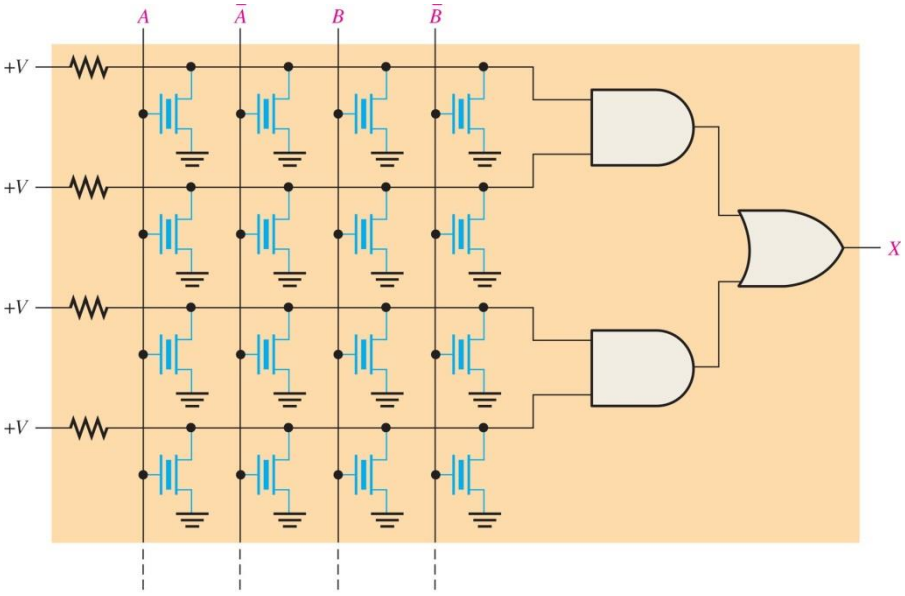
$$X = AB + A\bar{B} + \bar{A}\bar{B}$$



Şekil 10.2

SPLD: GAL

GAL, esasen, yeniden programlanabilen bir PAL'dir. PAL ile aynı VE/VEYA organizasyonuna sahiptir. Temel fark, bir GAL'nin, Şekil 10-3'te gösterildiği gibi, sigortalar yerine EEPROM (E^2CMOS) gibi yeniden programlanabilir bir işlem teknolojisi kullanmasıdır.

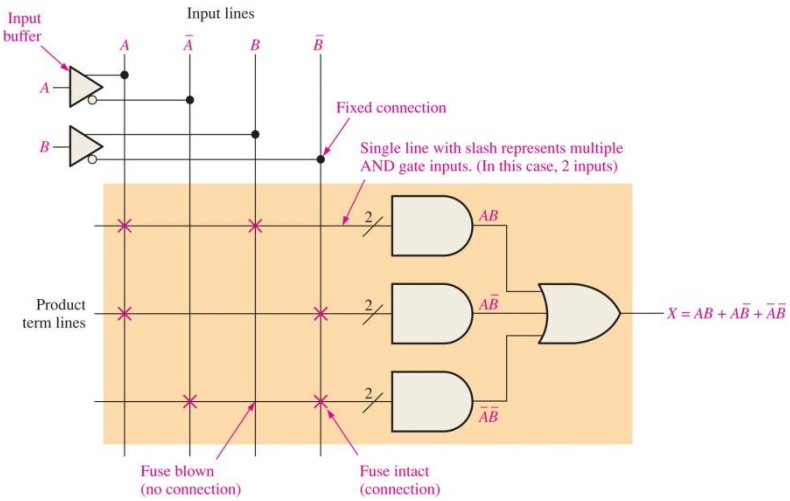


Şekil 10.3 Basit GAL dizisi

PAL / GAL Diyagramları için Basitleştirilmiş Notasyon

Gerçek PAL ve GAL cihazları, diğer elemanlara ek olarak birçok VE ve VEYA kapısına sahiptir ve birçok değişken ve tamamlayıcı kullanabilmektedir.

Bir data sheet'de görebileceğiniz çoğu PAL ve GAL diyagramı, şemayı çok karmaşık yapmaktan korumak için, Şekil 10–4'te gösterildiği gibi basitleştirilmiş gösterimi kullanır.

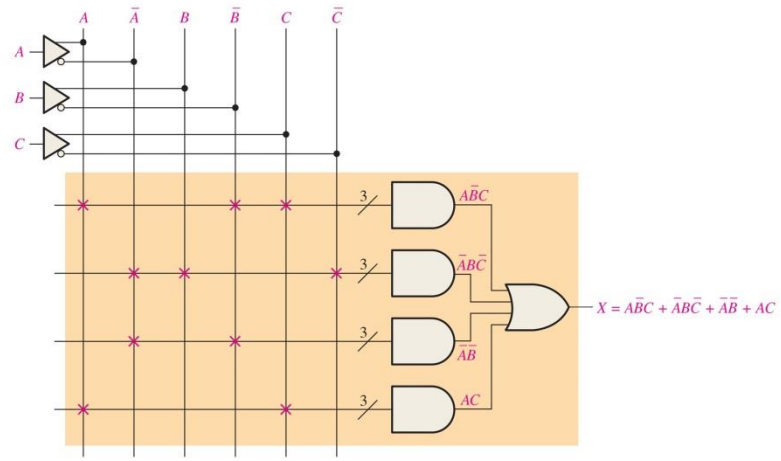


Şekil 10.4 Programlanabilir PAL/GAL in bir bölümü

Bir PAL veya GAL de giriş değişkenleri, genellikle bağlı oldukları çok sayıda AND kapısı girişi tarafından yüklenmesini önlemek için tamponlanır. Şekilde, üçgen sembolü hem değişkeni hem de tamamlayıcısını üreten bir tamponu temsil eder. Giriş değişkenlerinin ve tamponların sabit bağlantıları standart nokta gösterimi kullanılarak gösterilmiştir. PAL'ler ve GAL'ler çok sayıda programlanabilir ara bağlantı hattına sahiptir ve her AND kapısı birden fazla girişe sahiptir. PAL ve GAL'in mantık şekillerinde çoklu girişi ifade eden bir çizgi üzerine çizilmiş yatay bir çizgi ve giriş sayısını gösteren bir rakam bulunur. Şekil 10.4 bunu 2 girişli VE giriş kapıları için göstermektedir. Bir dizideki programlanabilir bağlantılar, sağlam bir sigorta veya başka bir bağlantı tipi için çapraz noktada kırmızı bir X ile ve açık bir sigorta veya başka bir bağlantı tipi için bir X'in bulunmadığı bir şekille gösterilmiştir. Şekil 10.4'te 2 değişkenli mantık fonksiyonu $AB + A\bar{B} + \bar{A}\bar{B}$ olarak programlanmıştır.

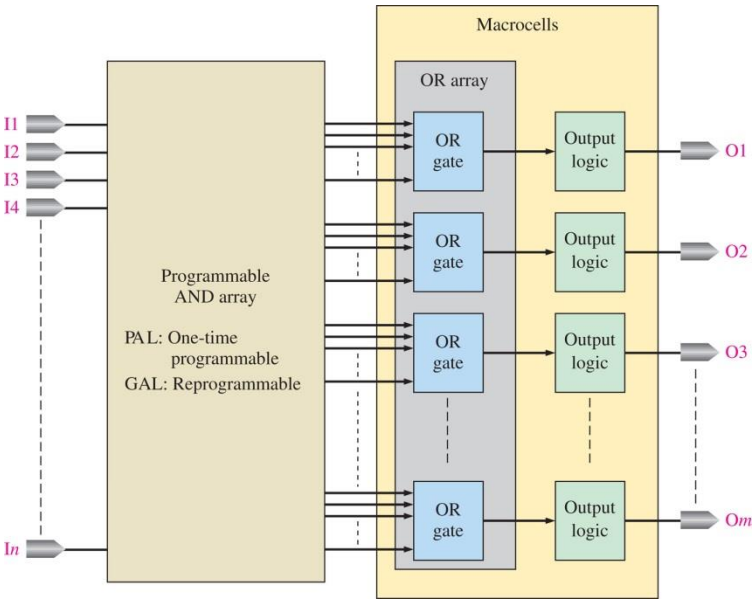
Örnek: Aşağıdaki 3 değişkenli mantık işlevi için bir PAL'nin nasıl programlandığını gösterin:
$X = A\bar{B}C + \bar{A}B\bar{C} + \bar{A}\bar{B} + AC$

Çözüm:



PAL / GAL Genel Blok Şeması

Bir PAL veya GAL'nin blok şeması, Şekil 10-6'da gösterilmektedir. Temel fark bir GAL'nin yeniden programlanabilir bir diziye sahip olmasıdır ve PAL bir kereye mahsus programlanabilir olmasıdır.



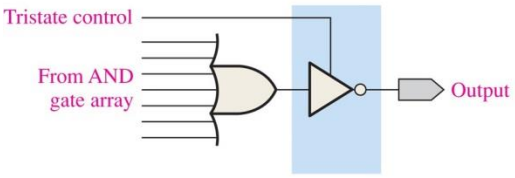
Şekil 10.6

Programlanabilir AND dizisi çıkışları, ek çıkış mantığına bağlı sabit VEYA kapılarına gider. İlişkili çıkış mantığı ile birlikte bir OR kapısı tipik olarak makrosel olarak adlandırılır. Makroselin karmaşıklığı, bulunduğu cihaza bağlıdır ve GAL'lerde çoğu zaman yeniden programlanabilir. Genellikle, SPLD paket yapılandırmaları 20 pin ile 28 pin arasında değişir. Belirli bir PAL veya GAL'nin bir mantık tasarımı için yeterli olup olmadığını belirlemek için kullanabilecek iki faktör; giriş ve çıkışların sayısı ve eşdeğer kapıların veya yoğunluğun sayısıdır. Dikkate alınacak diğer parametreler maksimum çalışma frekansı, gecikme süreleri ve dc besleme voltajıdır. İki çok kullanılan SPLD türü 16V8 ve 22V10'dur. Çeşitli SPLD üreticileri yoğunluğu tanımlamanın farklı yollarına sahip olabilir, bu nedenle belirtilen sayıda eşdeğer kapı kullanmak zorundasınız.

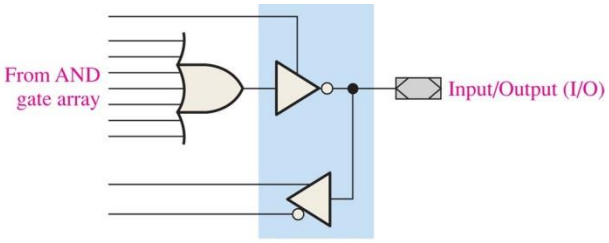
Makroseller

Bir makrosel genellikle bir VEYA kapısından ve ilgili bazı çıktı mantığından oluşur. Makroseller, belirli PAL veya GAL türüne bağlı olarak karmaşıklık bakımından değişir. Bir makrosel kombinasyonel mantık, register mantığı veya her ikisinin kombinasyonu için yapılandırılabilir. Register mantığı, sıralı mantık fonksiyonlarını sağlamak için makroselde bir flip-flop olduğu anlamına gelir. Şekil 10.7, birleşik mantığa sahip üç temel makrosel tipini göstermektedir. Kısım (a), OR kapısı ile basit bir makrosel ve çıkışı tamamen kesmek için sürücüyü açık devre gibi yapabilen üç durumlu kontrollü bir invertör göstermektedir. Bölüm (b), bir giriş veya çıkış olabilen bir makroseldir. Giriş olarak kullanıldığında, üç durumlu invertör bağlantısı kesilir ve giriş AND dizisine bağlı tampon belleğe gider. Kısım (c), bir aktif HIGH veya aktif LOW çıkışa sahip olacak şekilde programlanabilen bir makroseldir veya bir giriş olarak kullanılabilir. EXOR kapısının girişi HIGH veya LOW olabilir. Programlanabilir EXOR girişi HIGH

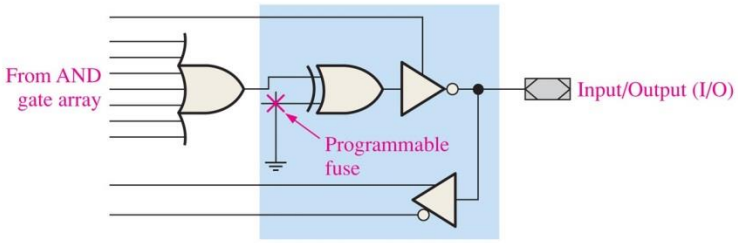
olduğunda, OR kapısı çıkışı ters çevrilir. Benzer şekilde, programlanabilir EXOR girişi LOW olduğunda, OR kapısı çıkışı ters çevrilmez.



(a) Combinational output (active-LOW). An active-HIGH output would be shown without the bubble on the tristate gate symbol.



(b) Combinational input/output (active-LOW)



(c) Programmable polarity output

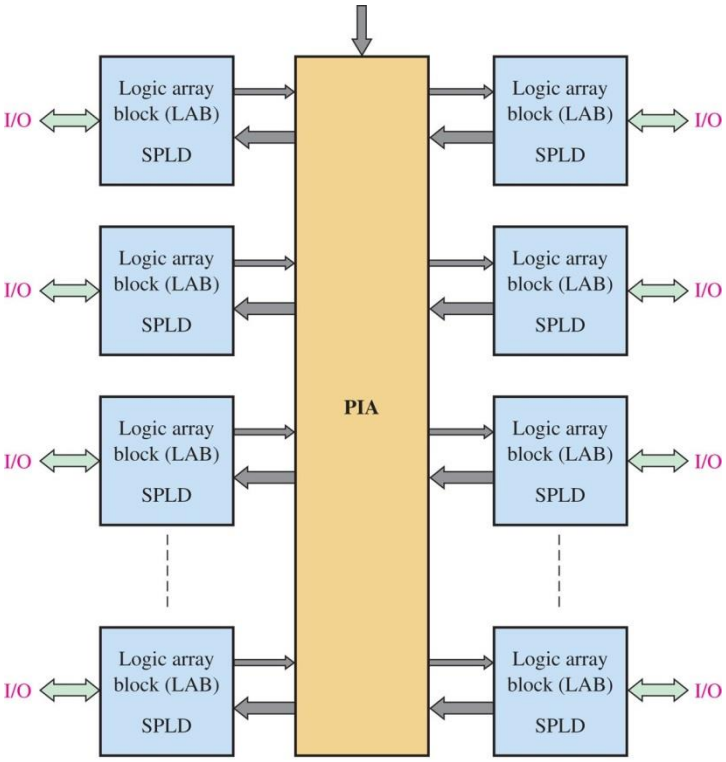
Şekil 10.7

Karmaşık Programlanabilir Mantık Cihazları (CPLD- Complex Programmable Logic Devices)

Karmaşık programlanabilir mantık cihazı (CPLD) temel olarak birden fazla SPLD içeren ve daha büyük mantık tasarımları için daha fazla kapasite sağlayan tek bir cihazdır.

CPLD

Bir CPLD (karmaşık programlanabilir mantık cihazı) temelde programlanabilir ara bağlantılara sahip çoklu SPLD dizilerinden oluşur. CPLD'lerin dahili olarak bağlantı şekli üreticiye göre değişmekle birlikte, Şekil 10-8 genel bir CPLD'yi göstermektedir.



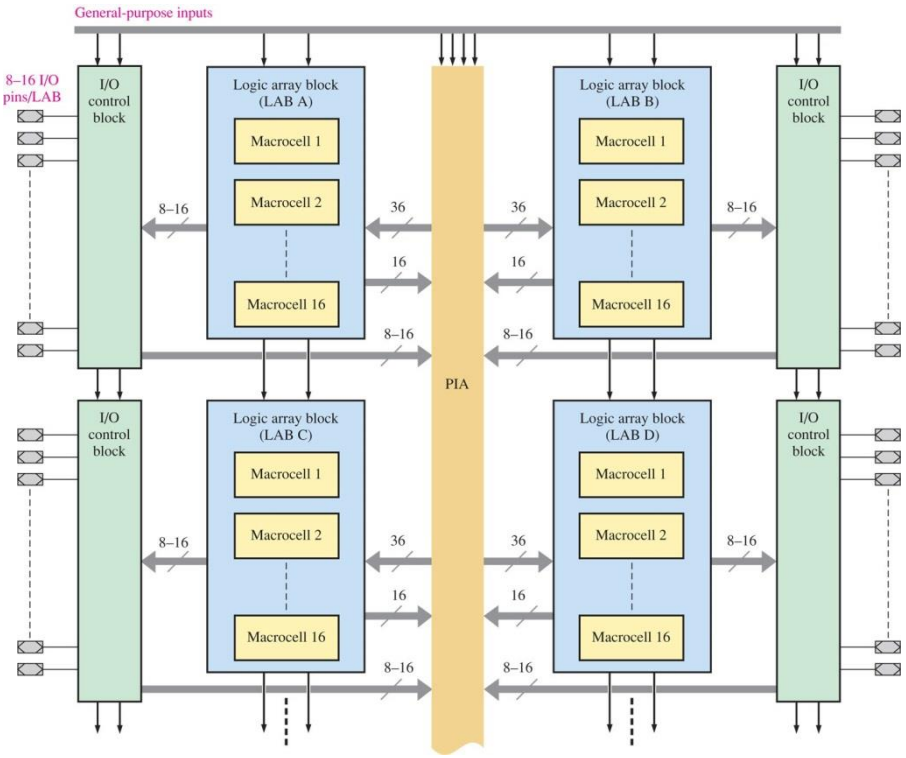
Şekil 10.8 Genel bir CPLD blok

CPLD'deki her SPLD dizisini LAB (logic array block-mantık dizisi bloğu) olarak isimlendireceğiz. Bazen fonksiyon bloğu, mantık bloğu veya genel blok gibi başka tanımlamalar da kullanılır. Programlanabilir ara bağlantılara genel olarak PIA (programmable interconnect array-programlanabilir ara bağlantı dizisi) adı verilir, ancak Xilinx gibi bazı üreticiler AIM (advanced interconnect matrix-gelişmiş ara bağlantı matrisi) veya benzer bir adlandırma kullanmaktadır. LAB'ler ve LAB'ler arasındaki ara bağlantılar yazılım kullanılarak programlanır. Bağımsız LAB'lerin (aslında SPLD'ler) SOP yapısına dayanarak bir CPLD karmaşık mantık fonksiyonları olarak programlanabilir. Girişler herhangi bir LAB'e bağlanabilir ve çıkışlar, PIA üzerinden diğer LAB'lere bağlanabilir. Çoğu programlanabilir mantık üreticisi, farklı yoğunluk, proses teknolojisi, güç tüketimi, besleme gerilimi ve hız gibi değişik CPLD'ler üretmektedir. Üreticiler genellikle CPLD yoğunluğunu makrosel veya mantık dizisi blokları cinsinden belirtir. Yoğunluklar, yüzlerce bağlantı ucuna sahip paketlerde, onlarca makroselden 1500'den fazla makrosel'e kadar değişebilir. PLD'ler daha karmaşık hale geldikçe, maksimum yoğunluk miktarı artacaktır. Çoğu CPLD yeniden programlanabilir ve programlanabilir bağlantılar için EEPROM veya SRAM işlem teknolojisini kullanır. Güç tüketimi birkaç miliwatt'tan birkaç yüz miliwat'a kadar değişik değerler alabilir. DC besleme voltajları, belirli bir cihaza bağlı olarak tipik olarak 2,5 V ila 5 V arasındadır. Bazı üreticiler (örneğin, Altera, Xilinx, Lattice ve Atmel) CPLD üretmektedir. CPLD'ler ve diğer programlanabilir mantık cihazları gerçekten bir donanım ve yazılım birleşimidir.

Klasik CPLD Mimarisi

Bir CPLD mimarisi, iç öğelerin düzenlenme şeklidir. Özel CPLD'lerin mimarisi, genel bir CPLD'nin blok diyagramına benzer (Şekil 10–8). SOP fonksiyonlarını yapan klasik PAL / GAL yapısına sahiptir. Yoğunluk, serideki belirli bir cihaza bağlı olarak 2 LAB ila 16 LAB arasında değişmektedir. Bir LAB

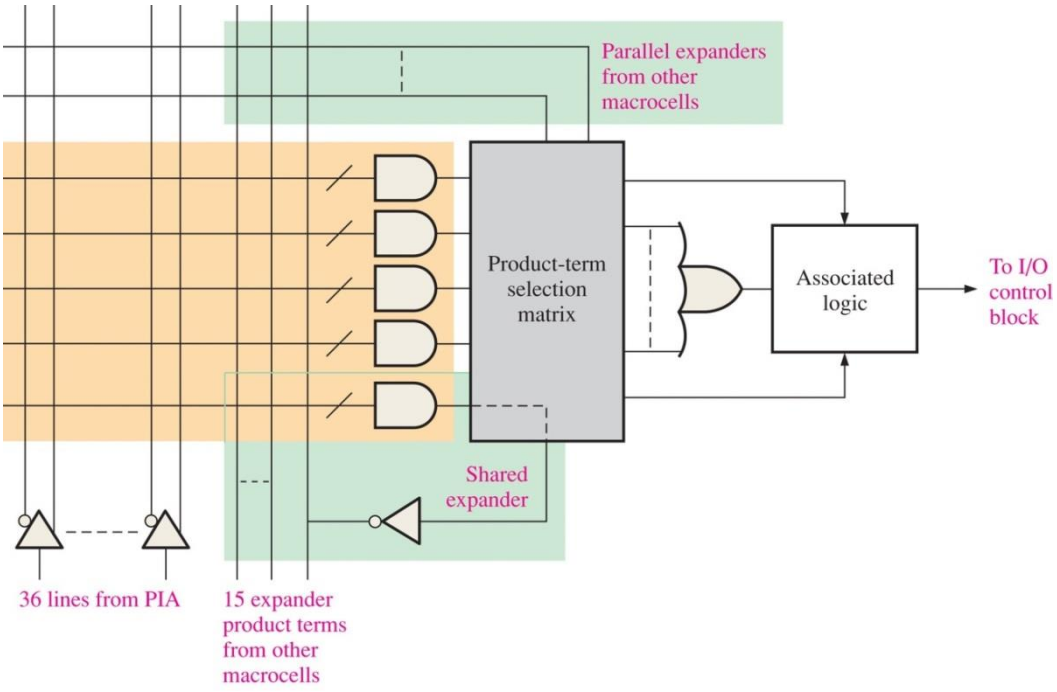
kabaca bir SPLD'ye eşdeğerdir ve CPLD'ler için paket boyutları 44 pin ila 208 pin arasında değişir. Bazı CPLD dizileri EEPROM tabanlı işlem teknolojisini kullanır. Sistem içi programlanabilir (ISP-In-system programmable) sürümleri, JTAG standart ara yüzünü kullanır. Şekil 10–9, tipik bir CPLD'nin genel blok şemasını göstermektedir. Şekilde dört LAB gösterilmektedir, ancak bir serideki belirli cihaza bağlı olarak on altıya kadar olabilir. Dört LAB'den her biri on altı makroselden oluşur ve çoklu LAB'ler, genel amaçlı girişlerin, I/O 'lerin ve makrosellerin girdiği programlanabilir bir global (tüm LAB'lara gider) veriyolu yapısı olan PIA aracılığıyla birbirine bağlanır.



Şekil 10.9

Makrosel

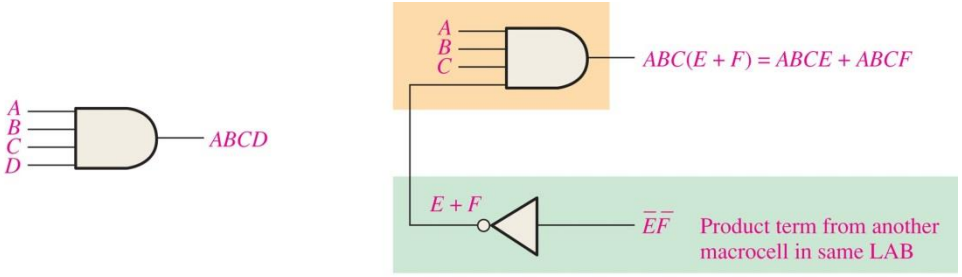
Tipik bir makroselin basitleştirilmiş bir diyagramı Şekil 10–10'da gösterilmektedir. Makrosel, beş AND kapılı küçük bir programlanabilir AND dizisi, bir OR kapısı, AND kapısı çıkışlarını OR kapısına bağlamak için bir çarpımların toplamı seçim matrisi ve giriş, birleşik mantık çıkışı veya programlanabilen mantık içerir. Aynı konsepte dayanmasına rağmen, bu makrosel, SPLD'ler ile ilgili olarak Bölüm 10–1'de tartışılan makroselden biraz farklıdır çünkü programlanabilir AND dizisinin bir kısmını ve çarpımların toplamı seçim matrisini içerir. Şekil 10.10'dan görüleceği gibi PIA'dan gelen ve çarpım terimleri seçim matrisine giden bağlantılar 5 adet VE kapısından gelmektedir. Alttan AND kapısına gelen çarpım terimi, diğer makroseller tarafından kullanılmak üzere paylaşılan bir genişletici olarak programlanabilir diziye geri döndürülebilir. Paralel genişletici girişleri, kullanılmayan çarpım terimlerinin bir SOP ifadesini genişletmek için diğer makro hücrelerden ödünç alınmasına izin verir. Çarpım terim seçim matrisi, AND çıkışından ve genişletici girişlerinden VEYA çıkışına seçilen çıkışları bağlamak için kullanılan bir programlanabilir bağlantı dizisidir.



Şekil 10.10

Paylaşılan Genişleticiler

Bir SOP ifadesindeki çarpım terimlerinin sayısını artırmak için kullanılabilecek bir tamamlanmış çarpım terimi, bir LAB'deki her bir makroselden elde edilebilir. Şekil 10.11, ek çarpım terimleri oluşturmak için başka bir makroselden paylaşılan bir genişletici terimin nasıl kullanılabileceğini gösterir.



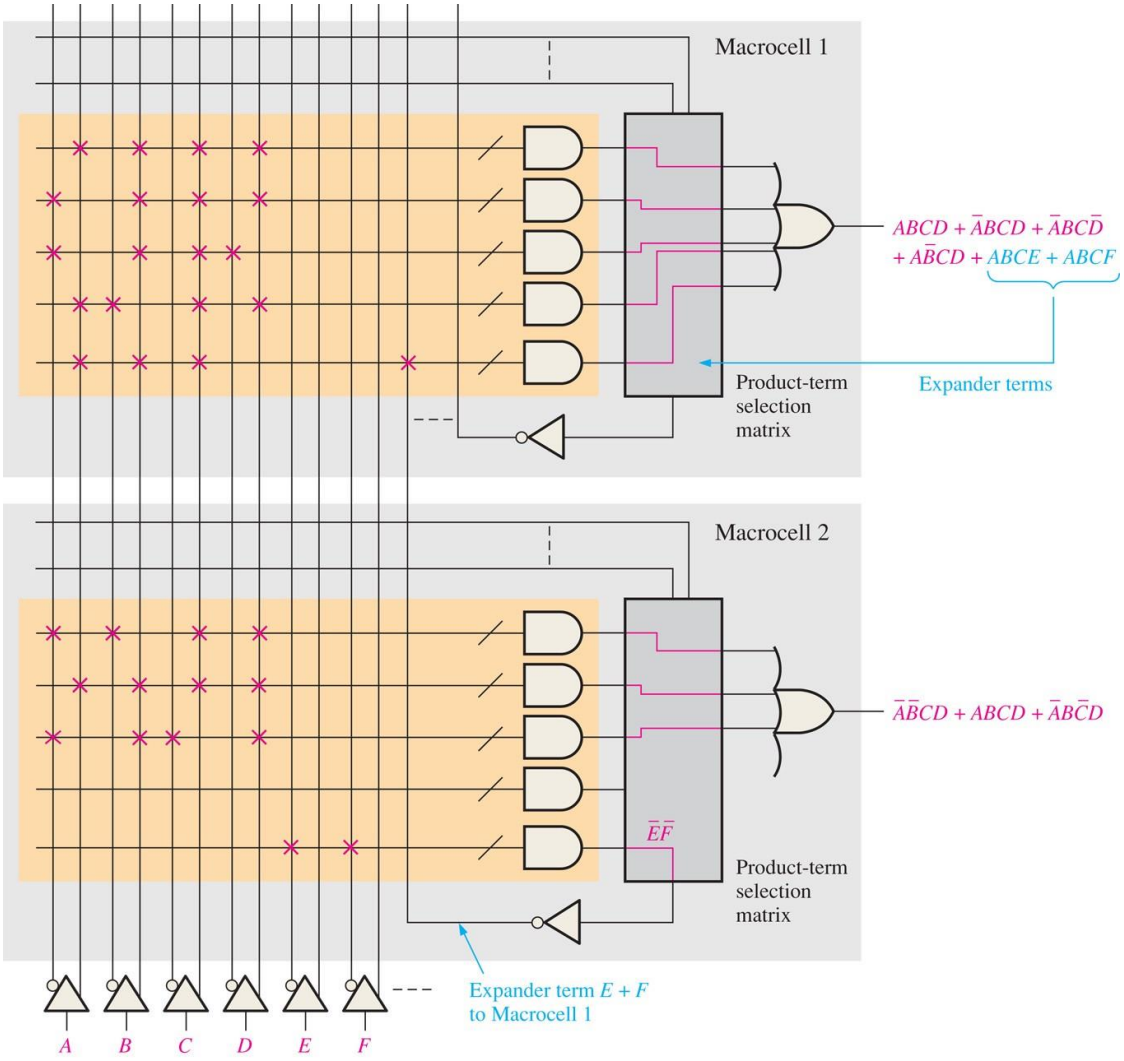
(a) A 4-input AND array gate can produce one 4-variable product term.

(b) AND gate is expanded to produce two product terms.

Şekil 10.11

Bu durumda, bir makrosel dizisindeki beş AND kapısının her biri dört girişle sınırlıdır ve bu nedenle (a) bölümünde gösterildiği gibi 4 değişkenli bir çarpım terimi üretebilir. Şekil 10.11 (b) iki çarpım terimine genişlemeyi gösterir. Her makrosel, AND dizisinden oluşturulan beş çarpım terimini üretebilir. Bir makro hücrenin SOP çıktısı için beş çarpım terimine ihtiyacı varsa, başka bir makro hücresinden bir genişletici terim kullanabilir.

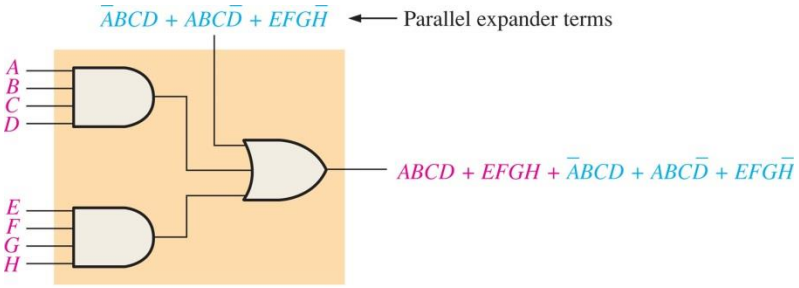
Bir tasarımın altı çarpım terimi içeren bir SOP ifadesi gerektirdiğini varsayalım. Şekil 10.12, bir SOP çıkışını arttırmak için başka bir makroselden gelen bir çarpımın nasıl kullanılabileceğini gösterir. Az kullanılmakta olan Macrocell 2, altı çarpım terimiyle bir SOP ifadesi üretmek için macrocell 1'deki beşinci AND geçidine bağlanan ortak bir genişletici terim (E + F) üretir.



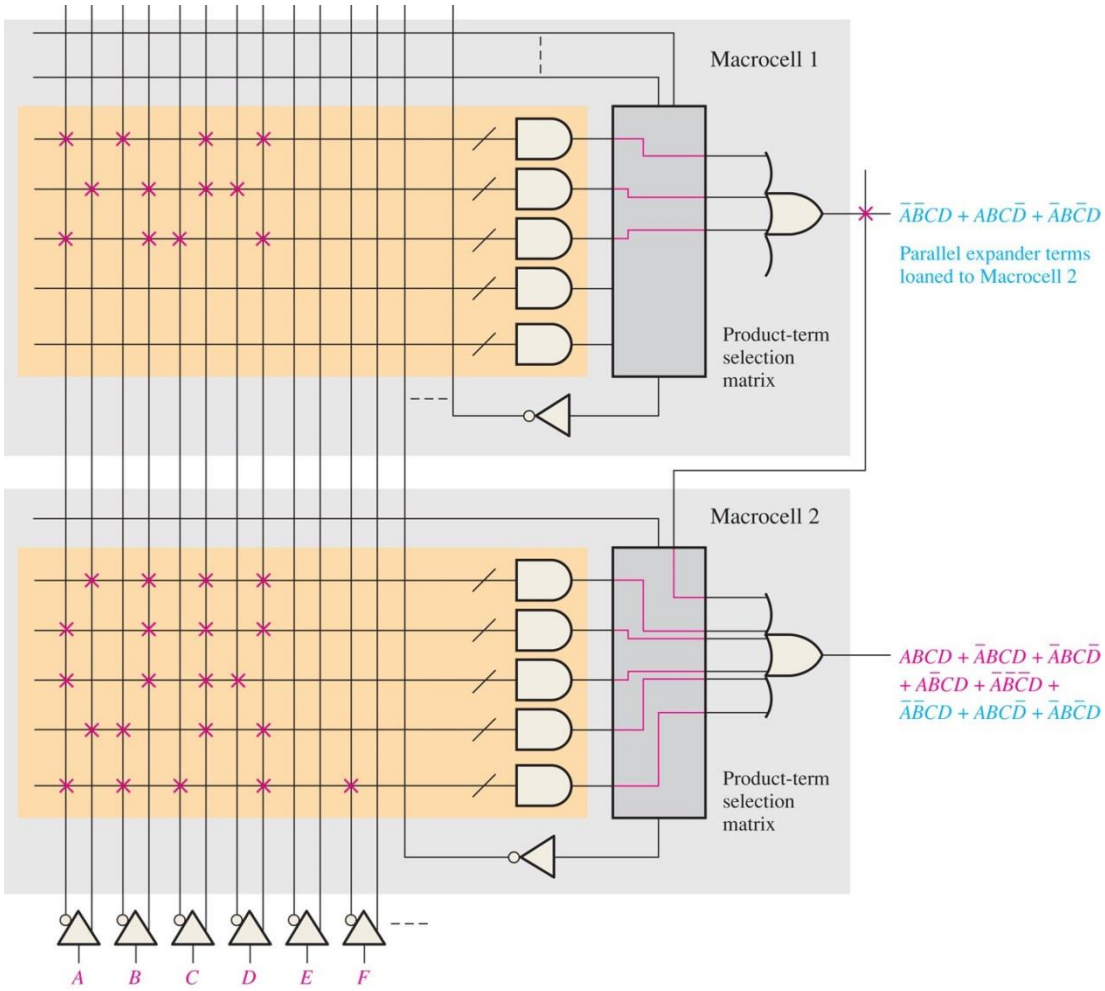
Şekil 10.12

Paralel Genişleticiler

Bir makrosel için çarpım terimlerinin sayısını artırmanın bir başka yolu, ek çarpım terimlerinin, ortak dizilimde olduğu gibi AND dizisinde birleştirilmek yerine bir makrosel tarafından üretilen terimlerle VEYA olduğu paralel genişleticiler kullanmaktır. Belirli bir makrosel, kullanılmamış çarpım terimlerini komşu makrosellerden ödünç alabilir. Temel kavram, iki çarpım terimini üretebilen basitleştirilmiş bir devrenin üç ek çarpım terimini ödünç aldığı Şekil 10.13'te gösterilmektedir. Şekil 10-14, SOP çıkışını artırmak için bir makroselin paralel genişletici terimleri başka bir makroselden nasıl ödünç alabildiğini göstermektedir. Macrocell 2, sekiz terim SOP ifadesi üretmek için macrocell 1'den üç çarpım terimi kullanır.



Şekil 10.13

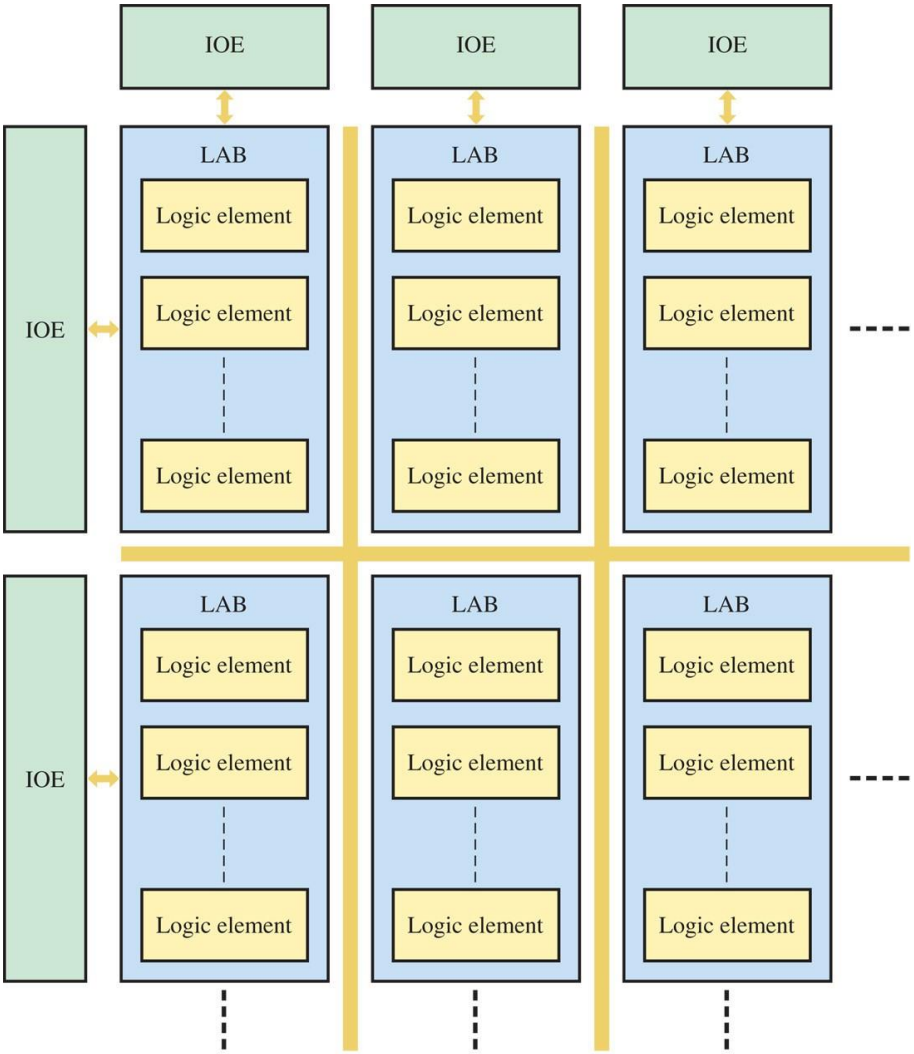


Şekil 10.14

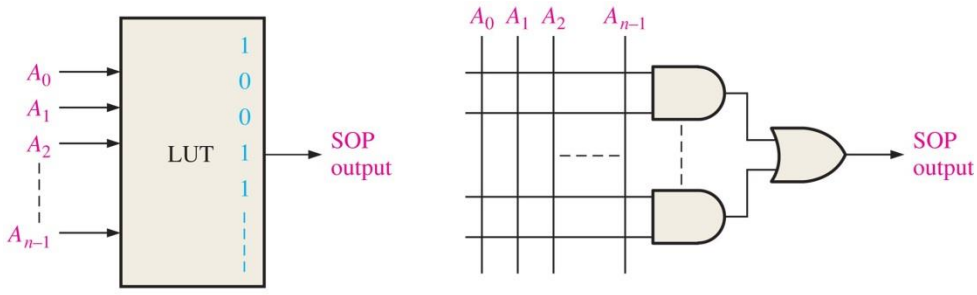
LUT CPLD Mimarisi

Bu mimari daha önce tartışılan klasik CPLD'den farklıdır. Şekil 10 - 15'teki blok şemada gösterildiği gibi, bu cihaz her biri birden fazla mantık elemanına (LEs- logic elements) sahip olan mantık dizisi bloklarını (LAB- logic array blocks) içerir. Bir LE, temel mantık tasarım birimidir ve makrosel ile aynıdır. Programlanabilir ara bağlantılar, LAB'ler arasında sıralı ve sütun şeklinde düzenlenmiştir ve giriş / çıkış

elemanları (IOE'ler- input/output elements) çevresine yerleştirilmiştir. Bu tür CPLD ve daha önce tartışılan klasik AND / OR dizisi CPLD arasındaki temel fark, bir mantık fonksiyonunun geliştirilme şeklidir. AND / OR dizileri yerine arama tabloları (LUT- Look-up tables) kullanılır. LUT temel olarak SOP işlevlerini üretmek için programlanabilen bir hafıza türüdür. Bu iki yaklaşım, Şekil 10-16'da karşılaştırılmaktadır. Belirtildiği gibi, LUT CPLD, klasik CPLD'lerin çoğunda bulunan kanal tipi ara bağlantılar yerine ara bağlantıların satır / sütun düzenine sahiptir. Bu iki yaklaşım, Şekil 10-17'de karşılaştırılmıştır ve Şekil 10-9 ve Şekil 10-15 karşılaştırılarak anlaşılabilir. Çoğu CPLD, programlanabilir bağlantılar için geçici olmayan bir işlem teknolojisi kullanır. Bununla birlikte, LUT CPLD, geçici olan SRAM tabanlı bir işlem teknolojisi kullanır - güç kapatıldığında programlanan tüm mantık kaybolur. Çip üzerine yerleştirilmiş bellek, geçici olmayan bellek teknolojisini kullanarak program verilerini depolar ve açılışta CPLD'yi yeniden yapılandırır.



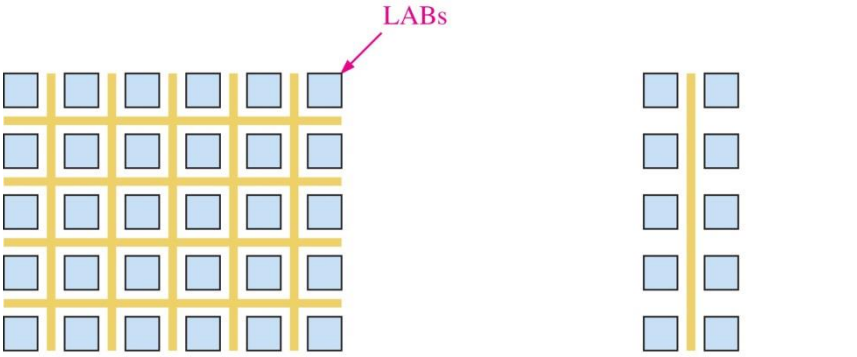
Şekil 10.15



(a) Look-up table logic. A 1 is stored at each product term address.

(b) AND/OR array logic

Şekil 10.16



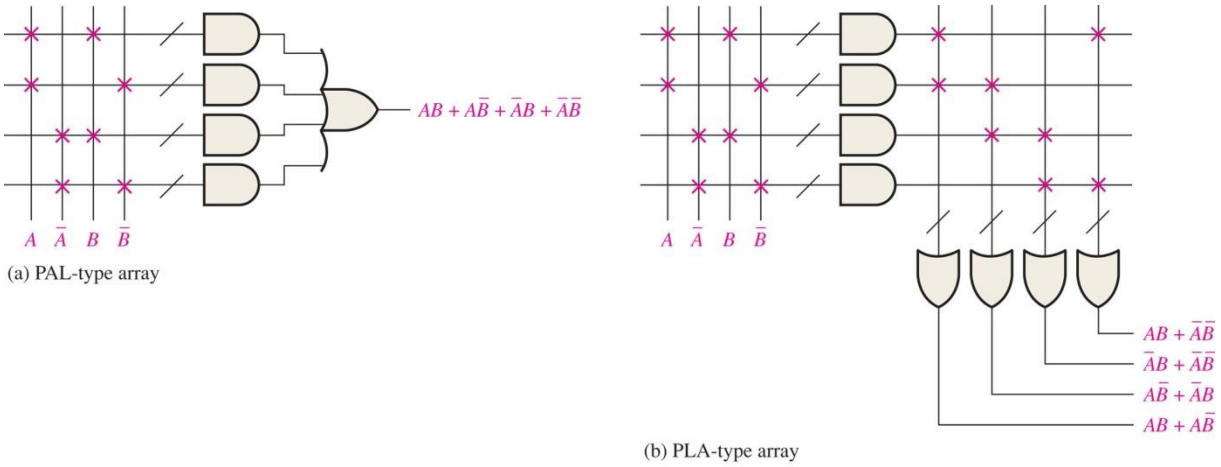
(a) Row/column interconnects

(b) Channel-type interconnect

Şekil 10.17

PLA (Programlanabilir Mantık Dizisi)

Bir CPLD mimarisi, iç öğelerin düzenlenme şeklidir. Bazı PLD'lerin mimarisi, anlatılan PAL (programlanabilir dizi mantığı) yapısından ziyade bir PLA (programlanabilir mantık dizisi) yapısına dayanmaktadır. Şekil 10–18, basit bir PAL yapısını basit bir PLA yapısıyla karşılaştırmaktadır. PAL, sabit bir OR dizisinin ardından programlanabilir bir AND dizisine sahiptir ve Şekil 10-18 (a) 'daki örnekte gösterildiği gibi bir SOP ifadesi üretir. PLA, Şekil 10–18 (b) 'deki örnekte gösterildiği gibi programlanabilir bir OR dizisinin ardından programlanabilir bir VE dizisine sahiptir.



(a) PAL-type array

(b) PLA-type array

Şekil 10.18

Özel CPLD Cihazları

Birkaç üretici CPLD üretir. Tablo 10–1 seçilen şirketlerden cihaz ailelerini listeler. Zaman geçtikçe, bir dizi eskimiş hale gelebilir veya yeni bir dizi eklenebilir. En güncel bilgiler için web sitelerini kontrol edebilirsiniz. CPLD'ler karmaşıklık bakımından büyük farklılıklar gösterir. Tablo 10-2, mevcut olan bazı parametre aralıklarını listeler. Teknoloji ilerledikçe bu sayıların değişebilir.

TABLE 10–1

CPLD manufacturers.

Manufacturer	Series Name	Design Software	Website
Altera	MAX	Quartus II	Altera.com
Xilinx	Coolrunner	ISE Design Suite	Xilinx.com
Lattice	ispMACH	ispLEVER classic	Latticesemi.com
Atmel	ATF	ProChip Designer	Atmel.com

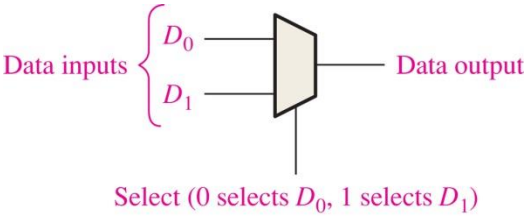
TABLE 10–2

Selected CPLD parameters.

Feature	Range
Number of macrocells	10–1700
Number of LABs	10–221
Maximum operating frequency	20.4 MHz–400 MHz
Number of I/Os	10–1156
DC operating voltage	1.8 V, 2.5 V, 3.3 V, 5 V

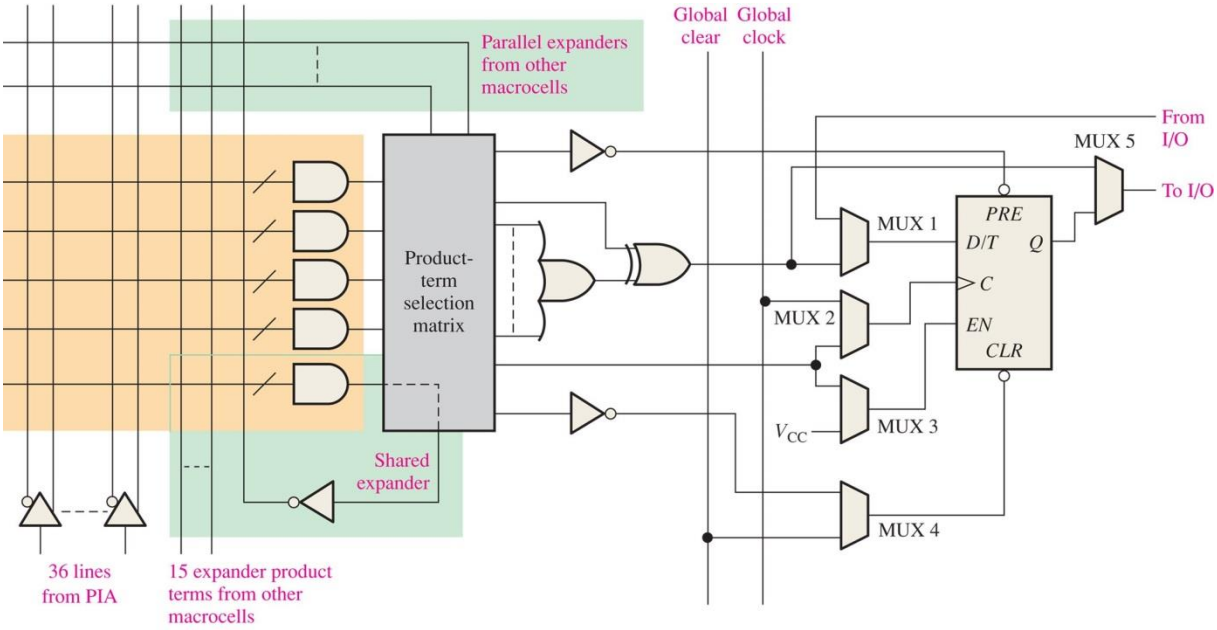
Makrosel Modları

Bir makrosel, kombinasyonel mantık veya kayıtlı (registered) mantık çıkışları ve girişleri programlama yoluyla yapılandırılabilir. Kayıtlı (registered) terimi, flip-flop kullanımı anlamına gelir. Her ne kadar makrosel mimarisi farklı CPLD'ler arasında değişse de, gösterim için genel bir makrosel mimarisi kullanılır. Mantık şemaları, genellikle, data çoğullayıcıyı (multiplexer) temsil etmek için Şekil 10–19'da gösterilen sembolü kullanır.



Şekil 10.19

Bu durumda, data çoğullayıcıda (multiplexer) iki veri girişi ve programlanabilir seçim sağlayan bir seçim girişi bulunur; Seçim girişi genellikle bir mantık şemasında gösterilmez.



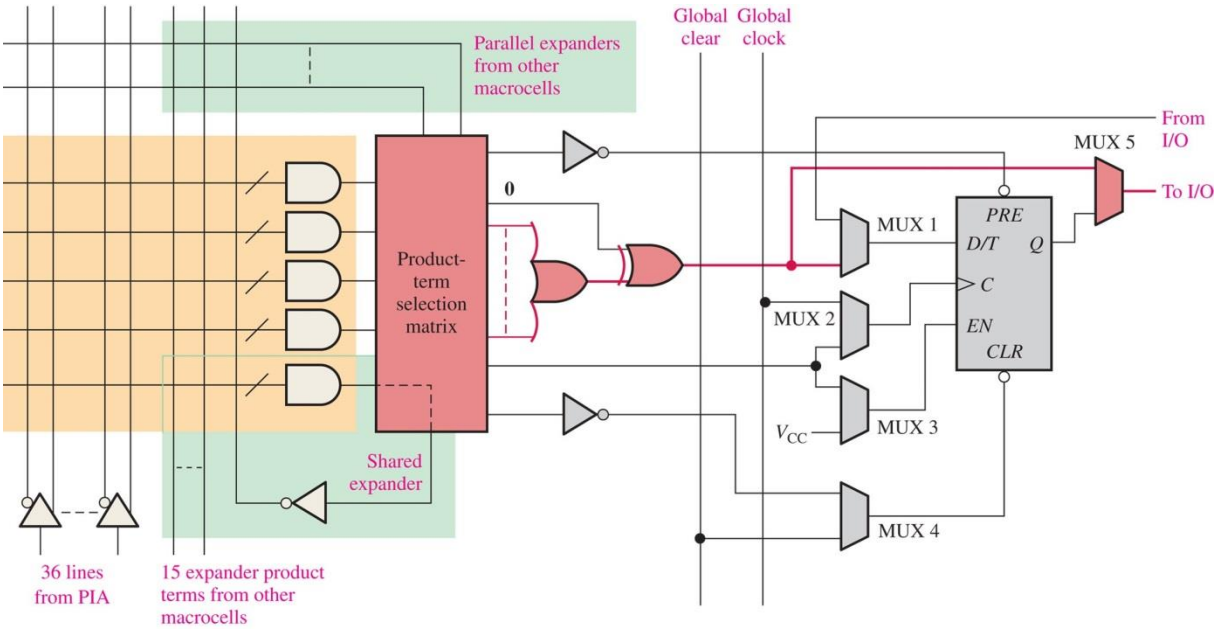
Şekil 10.20

Şekil 10-20, flip-flop (register) dahil olmak üzere eksiksiz bir makrosel göstermektedir. XOR geçidi, POS formunda bir işlev üretmek için SOP işlevini OR kapısından tamamlamayı sağlar. XOR kapısının üst girişinin 1 olması OR kapısının çıkışınının tümleyeni ve 0 olması, OR çıkışının terslendirilmemiş olarak geçmesini sağlar. MUX 1, XOR çıktısının ya da G / Ç'den gelen bir girişin seçimini sağlar. MUX 2 genel saati veya çarpım terimine dayalı bir saat sinyalini seçmek için programlanabilir. MUX 3, çarpım

terimini enable yapmak yada Vcc'yi seçmek (lojik 1 ile) için programlanabilir. etkinliğini seçmek üzere programlanabilir. MUX 4, genel temizliği veya çarpım terimini temizlemeyi seçebilir. MUX 5, flip-flop'u atlamak ve birleşik mantık çıkışını G / Ç'ye bağlamak veya kayıtlı çıkışı G / Ç'ye bağlamak için kullanılır. Flip-flop, D, T (toggle) veya J-K flip-flop olarak programlanabilir.

Kombinasyonel Mod

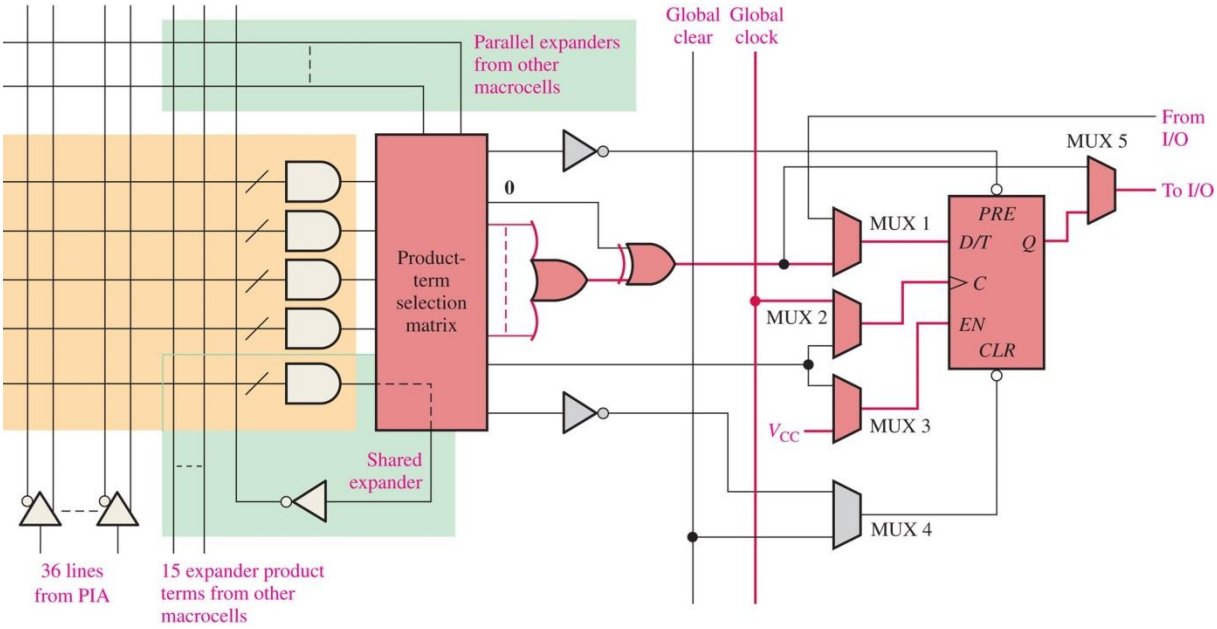
Bir makrosel bir SOP kombinasyonel mantık fonksiyonu üretmek üzere programlandığında, veri izlediği mantık elemanları Şekil 10–21'de kırmızı ile gösterilmiştir. Görüldüğü gibi, sadece bir mux kullanılmış ve register (flip-flop) atlanmıştır.



Şekil 10.21

Kayıtlı(registered) Mod

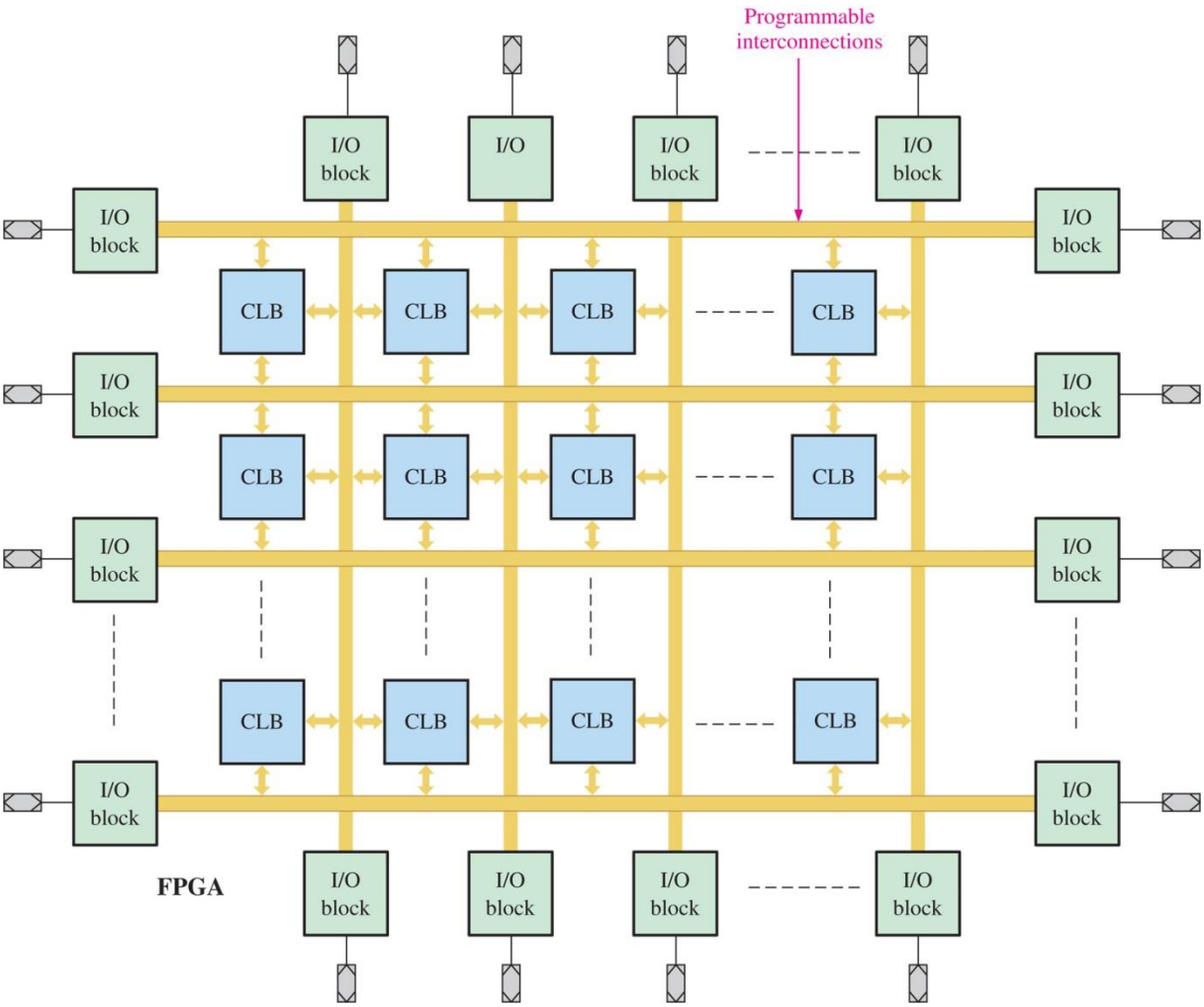
Makrosel register modunda programladığında SOP kombinasyonel mntık çıkışı registera bilgi gönderir. Ayrıca saat sinyali genel saat sinyalidir. Bilgi geçiş yolu Şekil 10–22'de kırmızı ile gösterilmiştir. Görülldüğü gibi, dört multiplexer (mux) kullanılır ve register (flip-flop) etkindir.



Şekil 10.22

FPGA (Field-Programmable Gate Arrays)

Klasik CPLD mimarisi, programlanabilir ara bağlantılara sahip PAL / GAL veya PLA tipi mantık bloklarından oluşur. Temel olarak, FPGA (sahada programlanabilir kapı dizisi) mimaride farklılık gösterir, PAL / PLA tipi dizileri kullanmaz ve CPLD'lerden çok daha fazla yoğunluğa sahiptir. Tipik bir FPGA, tipik bir CPLD'den çok daha fazla sayıda eşdeğer kapıya sahiptir. FPGA'lardaki mantık üreten unsurlar genellikle CPLD'lerden çok daha küçüktür ve bunlardan çok daha fazlası vardır.



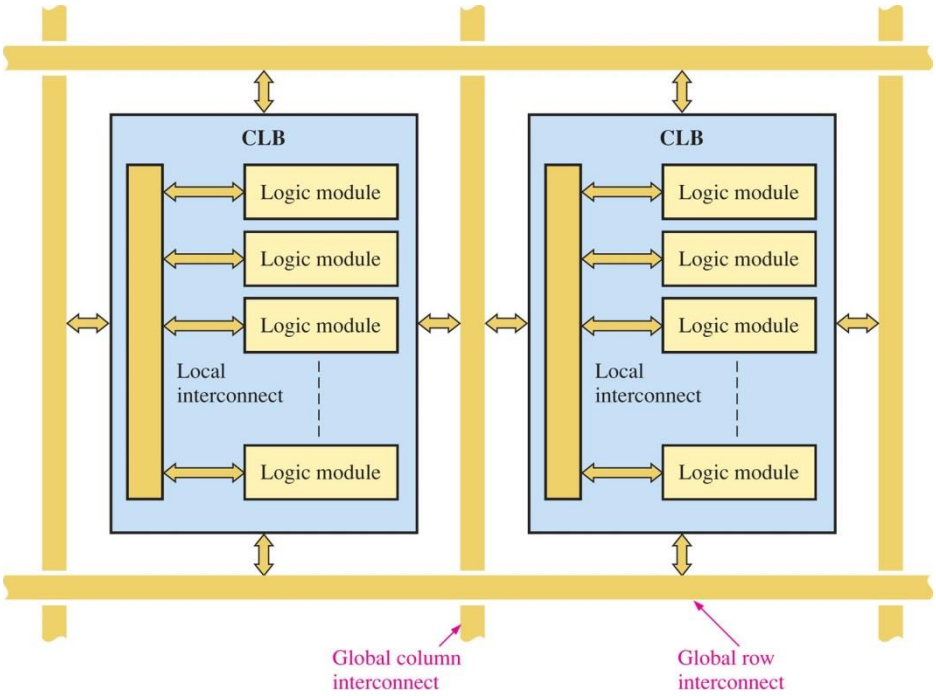
Şekil 10.23

Ayrıca, programlanabilir ara bağlantılar genellikle FPGA'larda bir sıra ve sütun düzeninde düzenlenir. Bir FPGA'daki üç temel eleman, Şekil 10-23'te gösterildiği gibi yapılandırılabilir mantık bloğu (CLB-configurable logic blocks), ara bağlantılar ve giriş / çıkış (I / O) bloklarıdır. Bir FPGA'daki konfigüre edilebilir mantık blokları (CLB'ler) bir CPLD'deki LAB'ler veya fonksiyon blokları (FB'ler) kadar karmaşık değildir, fakat genellikle bunlardan daha fazlası vardır. CLB'ler nispeten basit olduğunda, FPGA mimarisine ince taneli denir. CLB'ler daha büyük ve daha karmaşık olduğunda, mimariye kaba taneli denir. Yapının çevresinde bulunan G / Ç blokları dış dünyaya ayrı ayrı seçilebilir girdi, çıktı veya çift yönlü erişim sağlar. Programlanabilir ara bağlantıların matrisi, CLB'lerin ara bağlantılarını ve giriş ve çıkışlara bağlantılarını sağlar. Büyük FPGA'larda bellek ve diğer kaynaklara ek olarak onbinlerce CLB bulunabilir. Çoğu programlanabilir mantık üreticisi, yoğunluk, güç tüketimi, besleme voltajı, hız ve mimaride farklı derecelerde değişen FPGA'ler üretir.

FPGA'ler yeniden programlanabilir. Ayrıca, programlanabilir bağlantılar için SRAM veya antifuse proses teknolojisini kullanır. Yoğunluklar, yüzlerce mantık modülünden yüzbinlerce mantık modülüne kadar ve 1.000'den fazla pin içeren paketlerde değişebilir. DC besleme voltajları, belirli cihaza bağlı olarak 1.8 V ila 5 V arasındadır.

Yapılandırılabilir Mantık Blokları

Tipik olarak bir FPGA mantık bloğu, CPLD'deki makrosellere benzer şekilde temel yapı birimleri olan birkaç küçük mantık modülünden oluşur.



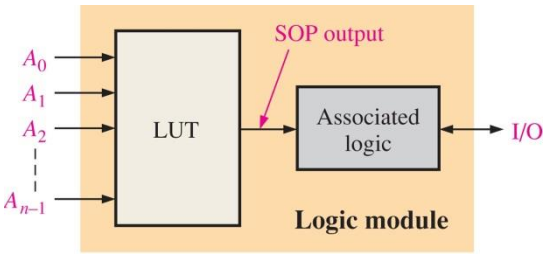
Şekil 10.24

Şekil 10–24, mantık bloklarını bağlamak için kullanılan genel satır / sütun programlanabilir ara bağlantılarında temel yapılandırılabilir mantık bloklarını (CLB'ler) göstermektedir.

Her CLB (aynı zamanda mantık dizisi bloğu, LAB olarak da bilinir), çoklu küçük mantık modüllerinden ve CLB içindeki mantık modüllerini bağlamak için kullanılan yerel bir programlanabilir bağlantılardan oluşur.

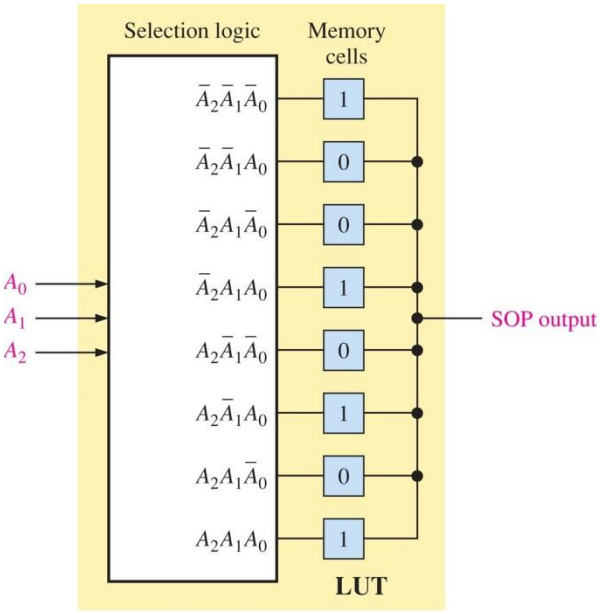
Mantık Modülleri

Bir FPGA mantık bloğundaki bir mantık modülü, kombinasyonel mantık, kayıtlı(register) mantık veya her ikisinin bir kombinasyonu için yapılandırılabilir. Genel bir LUT-tabanlı mantık modülünün bir blok şeması, Şekil 10–25'te gösterilmektedir. Bir LUT (arama tablosu) programlanabilir ve SOP kombinasyonel mantık fonksiyonlarını oluşturmak için kullanılan bir hafıza türüdür. LUT, esasen PAL veya PLA ile aynı işi yapar. Genel olarak, bir LUT'nin organizasyonu 2^n ye eşit bir n bellek hücresinden oluşur, burada n, giriş değişkenlerinin sayısıdır.



Şekil 10.25

Örneğin, üç giriş, sekiz taneye kadar bellek hücresi seçebilir, böylece üç giriş değişkenli bir LUT, sekiz adede kadar çarpım terimine sahip bir SOP ifadesi üretebilir. Belirtilen bir SOP işlevi için Şekil 10–26'da gösterildiği gibi, LUT bellek hücrelerine 1 ve 0'lık bir model programlanabilir.



Şekil 10.26

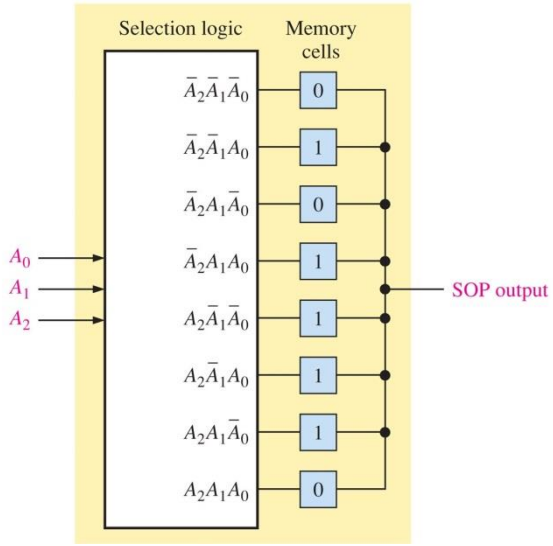
Her 1, ilişkili çarpım teriminin SOP çıktısında göründüğü anlamına gelir ve her 0, ilgili çarpım teriminin SOP çıktısında görünmediği anlamına gelir.

Sonuçta ortaya çıkan SOP çıktı ifadesi: $\bar{A}_2\bar{A}_1\bar{A}_0 + \bar{A}_2A_1A_0 + A_2\bar{A}_1A_0 + A_2A_1A_0$

Örnek: Aşağıdaki SOP işlevini oluşturmak için programlanmış 3 değişkenli bir LUT gösterin?

$$A_2A_1\bar{A}_0 + A_2\bar{A}_1\bar{A}_0 + \bar{A}_2A_1A_0 + A_2\bar{A}_1A_0 + \bar{A}_2\bar{A}_1A_0$$

Çözüm:



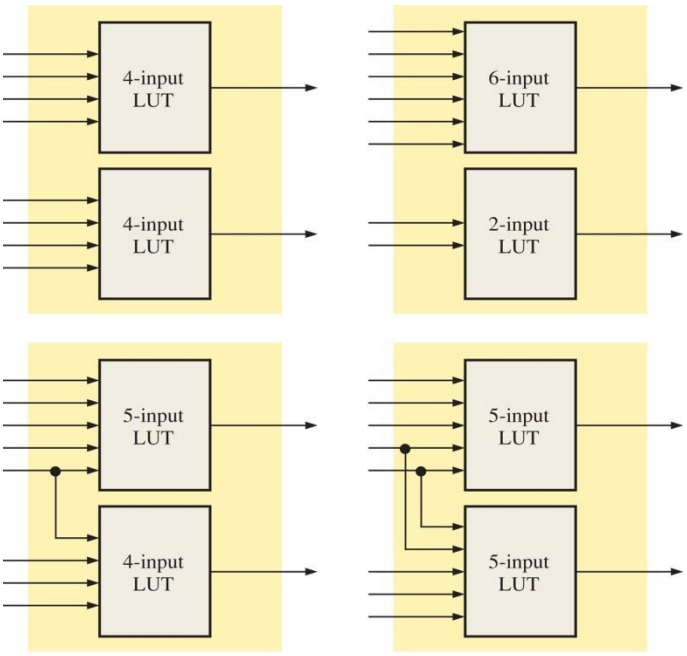
Mantık Modülünün Çalışma Modları

Tipik olarak, aşağıdaki çalışma modları için bir mantık modülü (LM) programlanabilir:

Normal mod
Genişletilmiş LUT modu
Aritmetik modu
Paylaşılan aritmetik modu

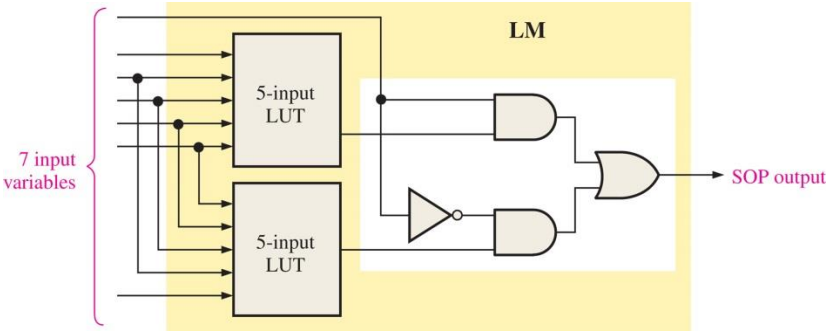
Bu dört moda ek olarak, sayıcılar ve shift register oluşturmak için bir kayıt zinciri olarak bir mantık modülü kullanılabilir. Bu bölümde normal modu ve genişletilmiş LUT modunu öğreneceğiz. Normal mod öncelikle birleşimsel mantık fonksiyonları oluşturmak için kullanılır. Bir mantık modülü iki LUT ile bir veya iki kombinasyon çıkış fonksiyonunu uygulayabilir. Dört LUT konfigürasyonunun örnekleri, Şekil 10-28'de gösterilmektedir. Genel olarak, her biri dört değişkenli veya daha az olan iki SOP işlevi, girişleri paylaşmadan LM'ye uygulanabilir.

Örneğin, iki adet 4 değişkenli fonksiyon, bir adet 4-değişkenli fonksiyon ve bir adet 3-değişkenli fonksiyon olabilir. Girişleri paylaşarak, her LUT için maksimum altı girişe kadar toplam sekiz giriş kombinasyonunu kullanabilirsiniz. Normal modda, 6 değişkenli SOP işlevleri ile sınırlandırılırsınız.



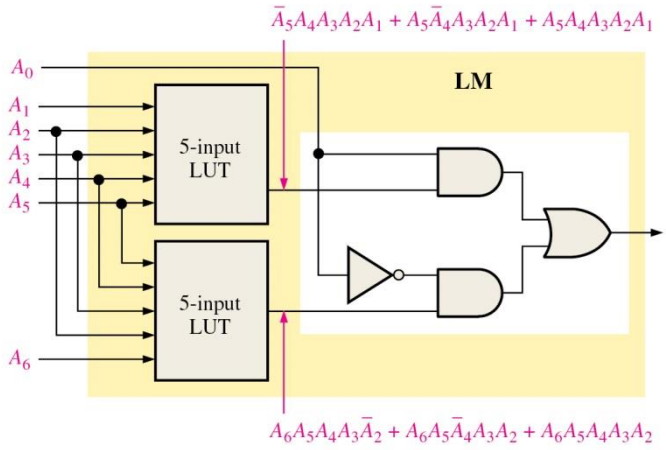
Şekil 10.28

Genişletilmiş LUT modu, Şekil 10-29'da gösterildiği gibi 7 değişkenli bir fonksiyona genişlemeye izin verir.



Şekil 10.29

Örnek: Genişletilmiş LUT modunda, Şekil 10–30'da gösterildiği gibi bir mantık modülü yapılandırılmıştır. Gösterilen belirli LUT çıkışları için nihai SOP çıkışını belirleyin.

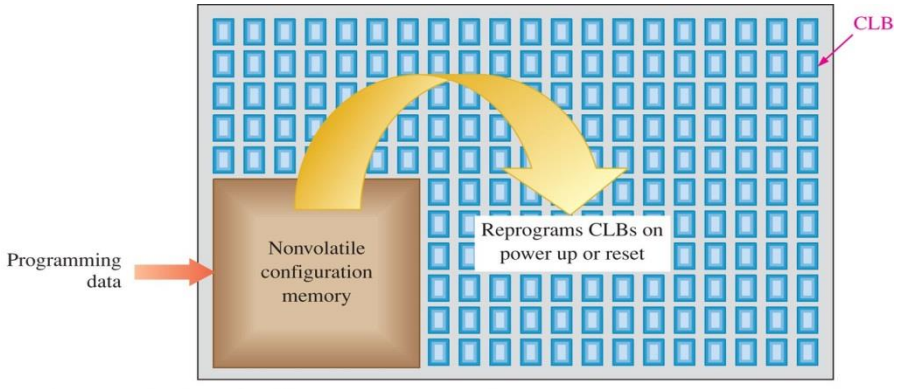


Şekil 10.30

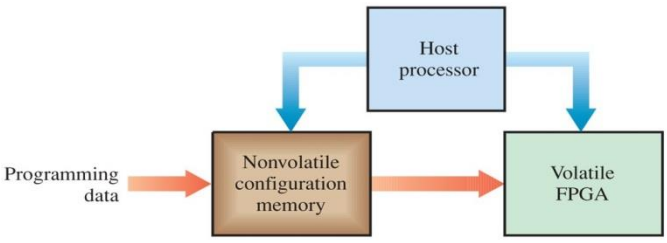
$$\overline{A}_5A_4A_3A_2A_1A_0 + A_5\overline{A}_4A_3A_2A_1A_0 + A_5A_4A_3A_2A_1A_0 + A_6A_5A_4A_3\overline{A}_2\overline{A}_0 + A_6A_5\overline{A}_4A_3A_2\overline{A}_0 + A_6A_5A_4A_3A_2\overline{A}_0$$

SRAM Tabanlı FPGA'lar

Bazı FPGA'lar ya geçici hafızaya sahip değildir, çünkü bunlar antifuse teknolojisine dayanır bazıları ise geçici hafızaya sahiptir çünkü SRAM teknolojisine dayanır. Bu nedenle, SRAM tabanlı FPGA'lar, program verilerini depolamak ve güç her açıldığında cihazı yeniden yapılandırmak için yonga üzerine yerleştirilmiş kalıcı bir yapılandırma belleği içerir veya ana işlemci tarafından kontrol edilen veri aktarımıyla harici bir bellek kullanır. Çipte bellek kavramı Şekil 10-31 (a) 'da gösterilmektedir. Ana bilgisayar işlemci yapılandırması kavramı, (b) bölümünde gösterilmektedir.



(a) Volatile FPGA with on-the-chip nonvolatile configuration memory



(b) Volatile FPGA with on-board memory and host processor

Şekil 10.31

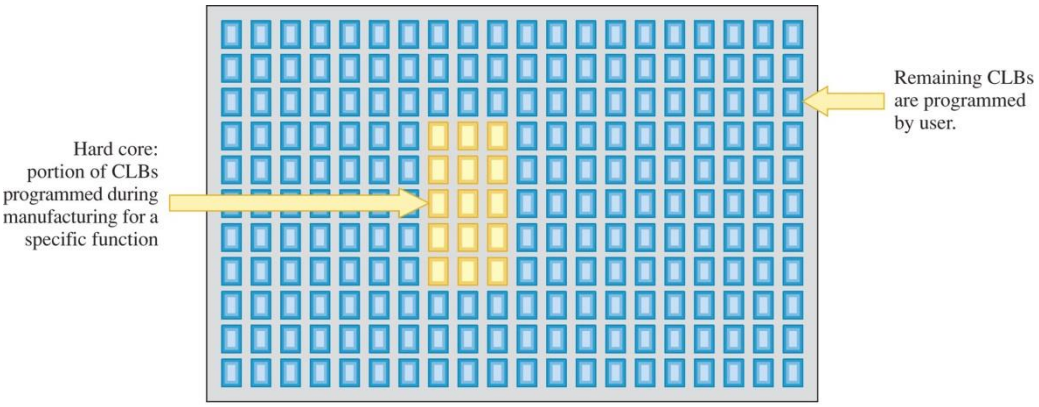
FPGA Çekirdekleri

FPGA'lar, esasen son kullanıcının herhangi bir mantık tasarımı için programlayabileceği “boş sayfalar” gibidir. Sert çekirdekli mantık içeren FPGA'lar da mevcuttur. Sert bir çekirdek, belirli bir işlevi sağlamak için üretici tarafından yerleştirilen ve yeniden programlanamayan bir FPGA'daki bir mantık parçasıdır. Örneğin, bir müşteri sistem tasarımının bir parçası olarak küçük bir mikroişlemciye ihtiyaç duyuyorsa, müşteri tarafından FPGA'ya programlanabilir veya üretici tarafından sert çekirdek olarak sağlanabilir. Gömülü fonksiyon bazı programlanabilir özelliklere sahipse, yumuşak çekirdek işlevi olarak bilinir.

Sert çekirdekli yaklaşımın bir avantajı, aynı tasarımın, FPGA'nın mevcut kapasitesinin çok daha azını kullanarak, kullanıcının sahada programladığından daha verimli olarak gerçekleştirilebilmesidir; bu da kullanıcı için daha az geliştirme süresi demektir. Ayrıca, sert çekirdekli işlevler tamamen test edilmiştir. Sert çekirdeğin dezavantajı, teknik özelliklerin üretim sırasında sabitlenmesi ve müşterinin, sert çekirdekli mantığı “olduğu gibi” kullanması gerektiğidir. Daha sonra değiştirilemez.

Sert çekirdekler, genellikle mikroişlemci, standart giriş / çıkış arabirimleri ve dijital sinyal işlemcileri gibi dijital sistemlerde yaygın olarak kullanılan fonksiyonlar için kullanılabilir. Bir FPGA'da birden fazla sert çekirdek fonksiyonu programlanabilir.

Şekil 10–32, kullanıcı tarafından programlanan yapılandırılabilir mantıkla çevrili bir sert çekirdek kavramını göstermektedir. Bu, temel bir gömülü sistemdir, çünkü sert çekirdekli fonksiyon, kullanıcı tarafından programlanan mantığa gömülmüştür.



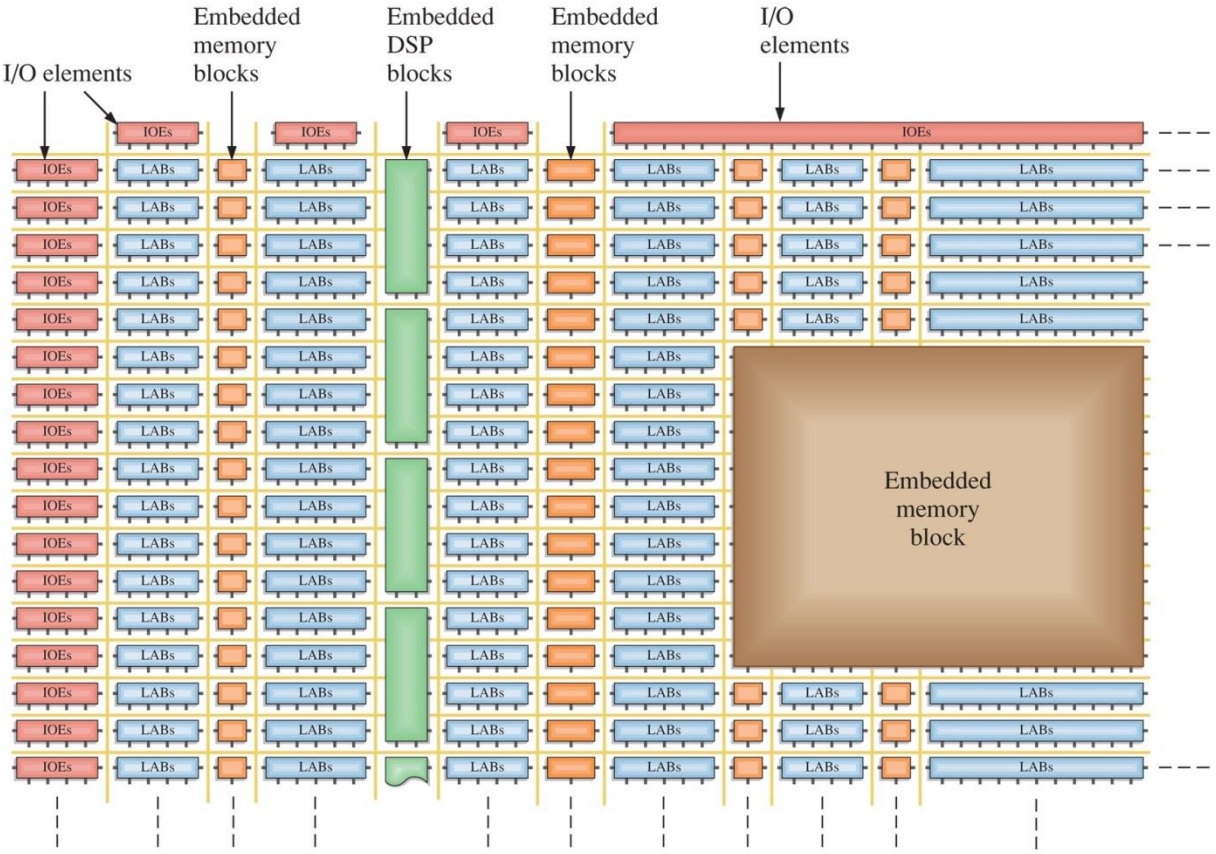
Şekil 10.32

Sert çekirdekli tasarımlar genellikle FPGA üreticisinin mülküdür ve geliştirilir. Üreticinin sahip olduğu tasarımlar fikri mülkiyet (IP-intellectual property) olarak adlandırılır. Bir şirket genellikle web sitesinde bulunan fikri mülkiyet türlerini listeler. Bazı entelektüel özellikler sert çekirdek ve yumuşak çekirdek karışımıdır. Kullanıcı tarafından belirli parametrelerin seçilmesinde ve ayarlanmasında biraz esnekliğe sahip bir işlemci buna bir örnektir. Sert çekirdekli ve yumuşak çekirdekli yerleşik işlemcileri veya diğer işlevleri içeren veya bu işlevleri içeren FPGA'lar, platform FPGA'ları olarak bilinir, çünkü harici bir

destek cihazlarına ihtiyaç duymadan tüm bir sistemi uygulamak için kullanılabilirler.

Gömülü Fonksiyonlar

Tipik bir FPGA'nın blok şeması, Şekil 10–33'te gösterilmektedir. FPGA, dahili sinyal fonksiyonlarının yanı sıra dijital sinyal işleme (DSP) fonksiyonlarını da içerir. Dijital filtreler gibi DSP işlevleri çoğu sistemde yaygın olarak kullanılır. Blok diyagramda görebileceğiniz gibi, gömülü bloklar FPGA ara bağlantı matrisi boyunca düzenlenir ve giriş / çıkış elemanları (IOE'ler) FPGA çevresine yerleştirilir.



Şekil 10.33

Özel FPGA Cihazları

Bazı üreticiler CPLD'lerın yanı sıra FPGA'lar da üretmektedir. Tablo 10–3 bazı şirketlerin cihaz ailelerini listeler. En güncel bilgiler için web sitesini kontrol edin. FPGA'lar karmaşıklık açısından büyük farklılıklar gösterir. Tablo 10–4 mevcut olan bazı parametre aralıklarını listeler. Teknoloji ilerledikçe bu sayıların değişebilir.

TABLE 10–3

FPGA manufacturers.

Manufacturer	Series Name(s)	Design Software	Website
Altera	Stratix Aria Cyclone	Quartus II	Altera.com
Xilinx	Spartan Artix Kintex Virtex	ISE Design Suite	Xilinx.com
Lattice	iCE40 MachX02 Lattice ECP3 LatticeXP2 LatticeGC/M	Lattice Diamond iCEcube2	Latticesemi.com
Atmel	AT40	IDS	Atmel.com

TABLE 10–4

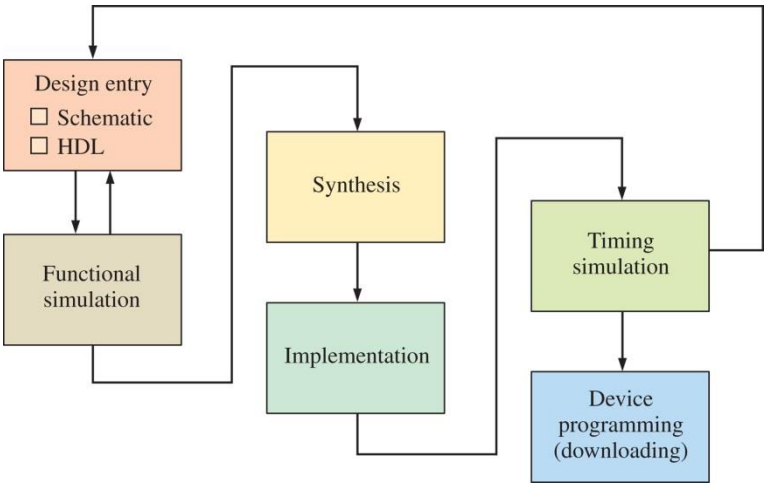
Selected FPGA parameters.

Feature	Range
Number of LEs	1,500–813,000
Number of CLBs	26–359,000
Embedded memory	26 kb–63 Mb
Number of I/Os	18–1200
DC operating voltage	1.8 V, 2.5 V, 3.3 V, 5 V

Programlanabilir Mantık Yazılımı

Yararlı olması için, programlanabilir mantığın fonksiyonel bir birimde birleşen hem donanım hem de yazılım bileşenlerine sahip olması gerekir. Tüm SPLD, CPLD ve FPGA üreticileri, her donanım aygıtı için yazılım desteği sağlar. Bu yazılım paketleri bilgisayar destekli tasarım (CAD) olarak bilinen bir yazılım kategorisindedir.

Web sitesinde iki tür yazılım eğitimi vardır bunlar; Altera Quartus II ve Xilinx ISE dir. Programlama işlemi genellikle tasarım akışı olarak adlandırılır. Mantıksal bir tasarımın programlanabilir bir cihazda uygulanması için temel bir tasarım akış şeması Şekil 10–34'te gösterilmektedir.



Şekil 10.34

Özel yazılım paketlerinin çoğu bu öğeleri bir biçimde veya başka şekilde birleştirir ve bunları otomatik olarak işler. Programlanan cihaz genellikle hedef cihaz olarak adlandırılır. Bir cihazı programlamaya başlamak için dört şeye sahip olmalısınız: bilgisayar, geliştirme yazılımı, programlanabilir bir mantık cihazı (SPLD, CPLD veya FPGA) ve cihazı bilgisayara bağlamanın bir yolu. Bu esaslar Şekil 10–35'te gösterilmektedir.

(a) Computer

(b) Software (CD or Website download)

(c) Device

(d) Programming hardware (programming fixture or development board with cable for connection to computer port)

Şekil 10.35

Kısım (a), kullandığınız yazılımın sistem gereksinimlerini karşılayan bir bilgisayarı gösterir. Kısım (b), cihaz üreticisinin CD'sinde edinilen veya aygıt üreticisinin web sitesinden indirilen yazılımı gösterir. Çoğu üretici, sınırlı bir süre için indirilip kullanılabilecek ücretsiz bir yazılım sağlar (Örnekler Altera Quartus II ve Xilinx ISE.). Bölüm (c) programlanabilir bir mantık cihazı göstermektedir. Kısım (d), cihazın içine yerleştirildiği programlama tertibatını veya cihazın üzerine monte edildiği geliştirme panelini kullanarak cihazı bilgisayara kabloyla fiziksel olarak bağlamanın iki yolunu göstermektedir.

Yazılım bilgisayarınıza yüklendikten sonra, bir aygıtı bağlayıp programlamaya başlamadan önce belirli yazılım araçlarına aşina olmalısınız.

Tasarım girişi

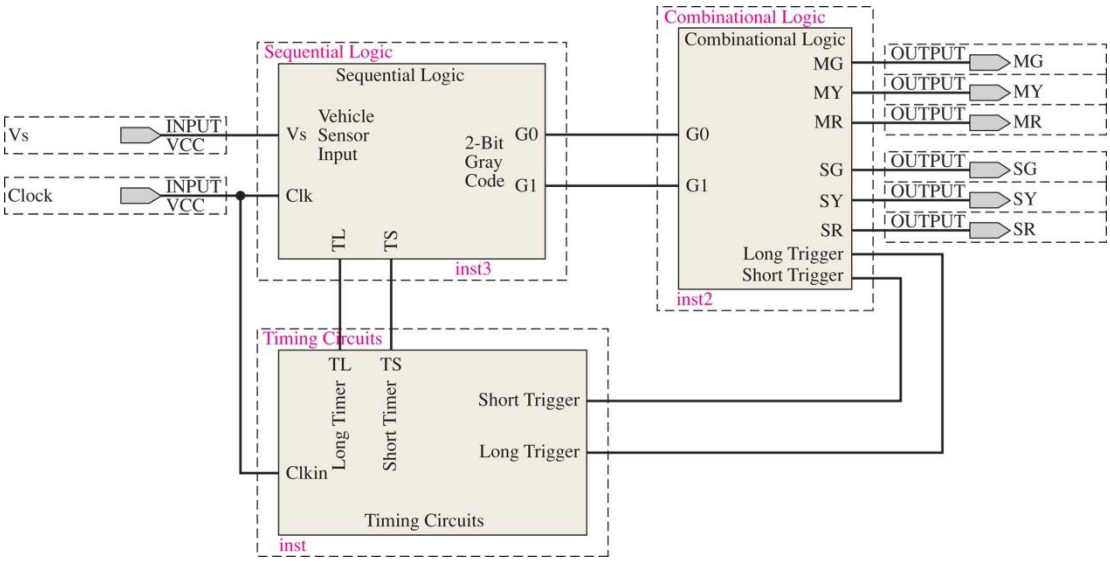
Programlanabilir bir cihazda uygulamak istediğiniz bir mantık devresi tasarımınız olduğunu varsayalım. Tasarımı bilgisayarınıza iki temel yoldan biriyle girebilirsiniz: şematik giriş veya metin girişi. Metin girişini kullanmak için, VHDL, Verilog veya AHDL gibi bir HDL diline aşina olmalısınız. Çoğu programlanabilir mantık üreticisi, standart HDL'ler oldukları için VHDL ve Verilog'u destekleyen yazılım paketleri sağlar. Bazıları ayrıca AHDL, ABEL veya diğer tescilli HDL'leri de destekler.

Şematik giriş, mantık kapılarının sembollerini ve diğer mantık fonksiyonlarını bir kütüphaneden ekrana yerleştirmenize ve bunları tasarımınızın gerektirdiği şekilde bağlamanıza izin verir. Şematik giriş için bir HDL bilgisi gerekmez.

Mantık Tasarımı Kurmak

VHDL ve Verilog gibi programlama dillerine ek olarak, PLD geliştirmede şematik tasarım da kullanılabilir. Ekrana şematik bir mantık devresi girdiğinizde, buna "düz" bir şematik denir. Karmaşık mantık devrelerinin ekrana sığması ve okunması zor olabilir.

Lojik devreleri kısımlara bölünebilir, her kısım bir blok sembolü olarak kaydedebilir ve daha sonra trafik sinyal denetleyicisinin Şekil 10–36'da gösterildiği gibi karmaşık bir devre oluşturmak için blok sembollerini grafiksel olarak bağlayabilirsiniz. Buna hiyerarşik yaklaşım denir.



Şekil 10.36

Trafik sinyali kontrol cihazının sıralı mantık bölümü, şematik yerleştirme kullanılarak oluşturulur ve VHDL kullanılarak oluşturulan aynı uygulama ile karşılaştırılır.

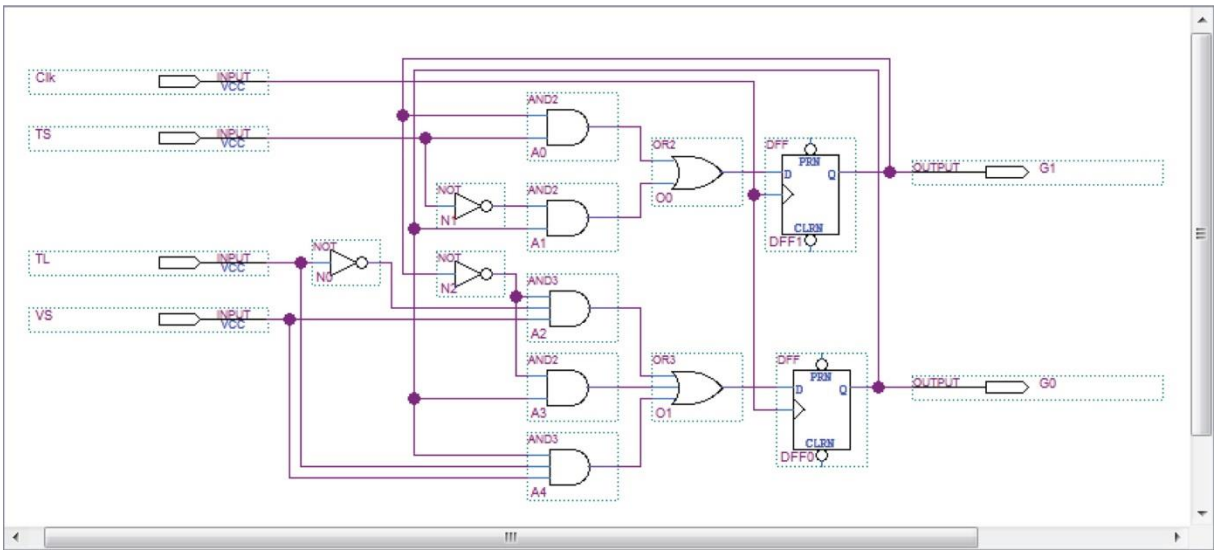
Şekil 10–37, sistemin sıralı mantık bileşenini oluşturmak için VHDL kullanımını göstermektedir.

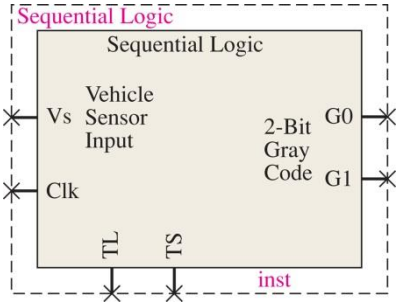
```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity SequentialLogic is
5  port(VS, TL, TS, Clk: in std_logic;
6       G0, G1: inout std_logic);
7  end entity SequentialLogic;
8
9  architecture SequenceBehavior of SequentialLogic is
10 component dff is
11 port(D,Clk: in std_logic;
12      Q: out std_logic);
13 end component dff;
14
15 signal D0, D1: std_logic;
16 begin
17   D1 <= (G0 and not TS) or
18         (G1 and TS);
19
20   D0 <= (not G1 and not TL and VS)  or
21         (not G1 and G0) or
22         (G0 and TL and VS);
23
24 DFF0: dff port map(D => D0, Clk => Clk, Q => G0);
25 DFF1: dff port map(D => D1, Clk => Clk, Q => G1);
26 end architecture SequenceBehavior;
```

Şekil 10.37

D0 ve D1'e atanan ifadeler için kod doğrudan Boolean ifadelerinden oluşturulur.

Şekil 10–38 (a), şematik giriş teknikleri kullanılarak oluşturulan sıralı mantık bloğunu gösterir.





Şekil 10.38

Şemayı ayrı mantık devrelerine bölmek, işlevsel bölümlendirme ve daha kolay gelişim sağlar. Boolean ifadeleri, bunları bağlamak için gereken tellerin ve I / O bileşenlerinin grafiksel gösterimini içeren ayrı mantık kapıları kullanılarak uygulanır. Tamamlanan ve test edilen modül, Şekil 10–38 (b) 'de gösterildiği gibi basit bir blok sembolüne indirgenir ve Şekil 10–36'da gösterildiği gibi bir bileşen olarak yerleştirilebilir. VHDL kodu kullanılarak bir blok sembolü de oluşturulabilir.

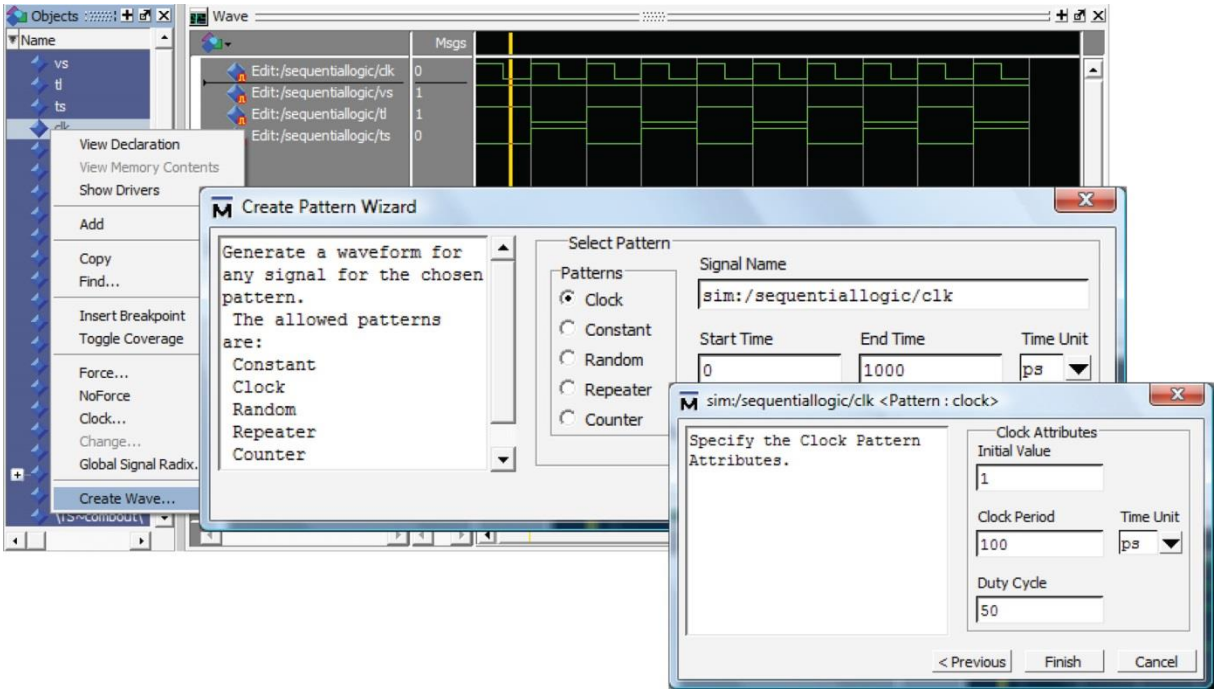
Fonksiyonel Simülasyon

Tasarım akışındaki fonksiyonel simülasyonun amacı, girdiğiniz tasarımın bir donanım tasarımına sentezlenmeden önce, mantıksal çalışması bakımından gerektiği gibi çalıştığından emin olmaktır. Temel olarak, bir mantık devresi derlendikten sonra, giriş dalga formları uygulanarak ve olası tüm giriş kombinasyonları için çıkış kontrol edilerek simüle edilebilir. Fonksiyonel simülasyon, bir dalga biçimi editörü kullanılarak grafik olarak veya bir test cihazı kullanarak programlı olarak gerçekleştirilir. Grafik dalga formu editörleri, dalga şekli çizim özelliklerini ve sürükle/bırak tekniklerini kullanarak test uyarısının çizilmesine izin verir.

Grafiksel Yaklaşım

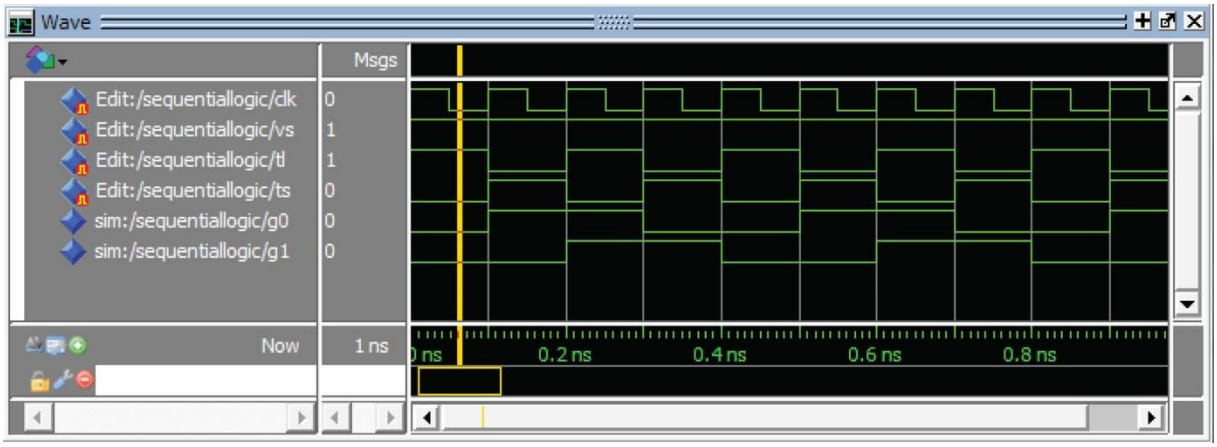
Grafiksel oluşturma araçları, basit test uygulamaları için çizilmiş uyaran dalga formlarının kolayca oluşturulmasına izin verir. Trafik sinyal kontrol sisteminin sıralı mantık bileşeni için giriş uyarısını

örnek olarak vermek üzere grafik dalga formları oluşturulmuştur. Clk, TL, TS ve VS girişleri grafiksel araçlar kullanılarak oluşturulacaktır. G0 ve G1 çıkış tanımlayıcıları giriş uyarıcısı gerektirmez ve basitçe Wave penceresine sürüklenip bırakılır. Saat tanımı, sistem saatini Clk sürmek ve simülasyon çalışma zamanını sınırlandırmak için Saati Tanımla özelliği kullanılarak oluşturulur. Ofset, görev döngüsü, periyot, mantık değerleri, iptal ve ilk kenar sağlanır. VS, TL ve TS girişleri aynı grafik teknikleri kullanılarak oluşturulmuştur. Çizilmiş uyaran dalga formlarını simülasyondan önce görüntüleyebilirsiniz. Tipik pencereler Şekil 10-39'da gösterilmektedir.



Şekil 10.39

Giriş dalga formlarını belirledikten sonra simülasyon çalışmaya hazırdır. Simülasyon başlatıldığında, G0 ve G1 için çıkış dalga formları, Şekil 10–40'ta gösterildiği gibi gösterilecektir.



Şekil 10.40

Bu, tasarımın iyi olduğunu veya doğru çalıştığını doğrulamanıza olanak tanır. Bu durumda, çıkış dalga formu seçilen giriş dalga formlarına düzenlenir. Yanlış bir çıkış dalga formu, mantığın işlevselliğindeki

bir kusuru belirtir; geri dönüp orijinal tasarımı kontrol edip gözden geçirilmiş tasarıma tekrar girmelisiniz.

Test Bench Yaklaşımı

Tasarım simülasyonuna programatik bir yaklaşım, test bench olarak adlandırılan ek bir program dosyası oluşturmaktır. Bir test bench, yapım program koduna benzer ve tipik olarak orijinal HDL ile aynı HDL'ye yazılır. Test bench programı, orijinal program kadar karmaşık olabilir.
Bu örnekte, trafik sinyali kontrol cihazının sıralı mantık bileşeni için giriş uyarıcısını sağlamak üzere bir test bench programı yazılmıştır. Aşağıdaki test bench programı, sıralı mantık modülü için giriş dalga formlarını simüle etmek üzere VHDL'de yazılmıştır.

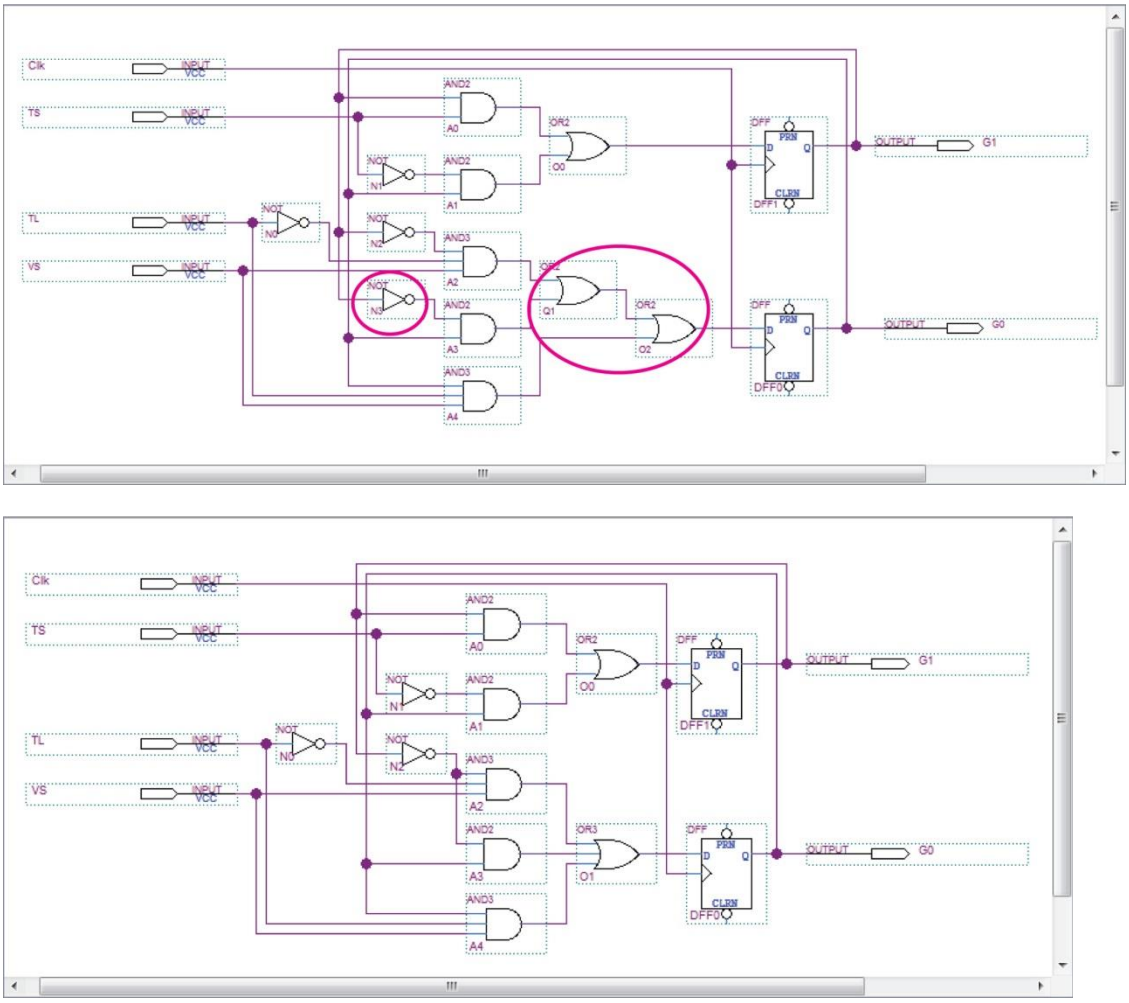
```
library IEEE;
use IEEE.std_logic_1164.all;
entity TestSL is
end entity TestSL;
architecture TestSLBehavior of TestSL is
component SequentialLogic is
port(VS, TL, TS, CLK: in std_logic;
G0, G1: inout std_logic);
end component SequentialLogic;
signal VS, TL, TS, Clk, G0, G1: std_logic;
begin
Clk_process:process
begin
for iterate in 1 to 10000
loop
CLK6='1';
wait for 50 us;
CLK6='0';
wait for 50 us;
end loop;
wait;
end process;
TLS_process: process
begin
TL 6='0';
TS 6='1';
wait for 100 us;
TL 6='1';
TS 6='0';
wait for 100 us;
end process;
stim_proc: process
begin
VS 6='0';
wait for 100 us;
VS 6='1';
wait;
end process;
UUT: SequentialLogic port map
(VS =7 VS, TL =7 TL, TS =7 TS, Clk =7 Clk, G0 =7 G0, G1 =7 G1);
end architecture TestSLBehavior;
```

Test bench simülasyonu çalıştırıldıktan sonra, dalga formu editörü ekranındaki çıkış dalga formu mantığın düzgün çalıştığını göstermelidir.

Sentez

Tasarım girildikten ve mantıksal çalışmasının doğru olduğunu doğrulamak için fonksiyonel olarak simüle edildikten sonra, derleyici otomatik olarak hedef cihaza indirilecek tasarımı hazırlamak için birkaç aşamadan geçer. Tasarım akışının bu sentez aşamasında, tasarım, kapı sayısını en aza indirmek, mantık öğelerini aynı fonksiyonu daha verimli bir şekilde yerine getirebilen diğer mantık öğeleriyle değiştirmek ve gereksiz mantığı ortadan kaldırmak için optimize edilmiştir.

Sentez aşamasından elde edilen son çıktı, mantık devresinin optimize edilmiş versiyonunu tanımlayan bir ağ listesidir. Tasarım optimizasyon sürecini göstermek için, sistemin sıralı mantık bölümünün şematik tasarlama versiyonu, Şekil 10–41 (a) 'da gösterildiği gibi yedekli ORGates ve NotGates ile sunulmaktadır. Şekil 10–41 (a) 'da gösterilen tasarım giriş aşamasına girilen AND-OR mantığı, Şekil 10-41 (b)' de gösterilen optimize edilmiş devre ile sonuçlanabilir. Bu şekilde, derleyici iki adet 2 girişli OR kapısını çıkardı ve onları bir adet 3 girişli OR kapısıyla değiştirdi. Ayrıca, gereksiz invertörlerden biri elimine edildi.



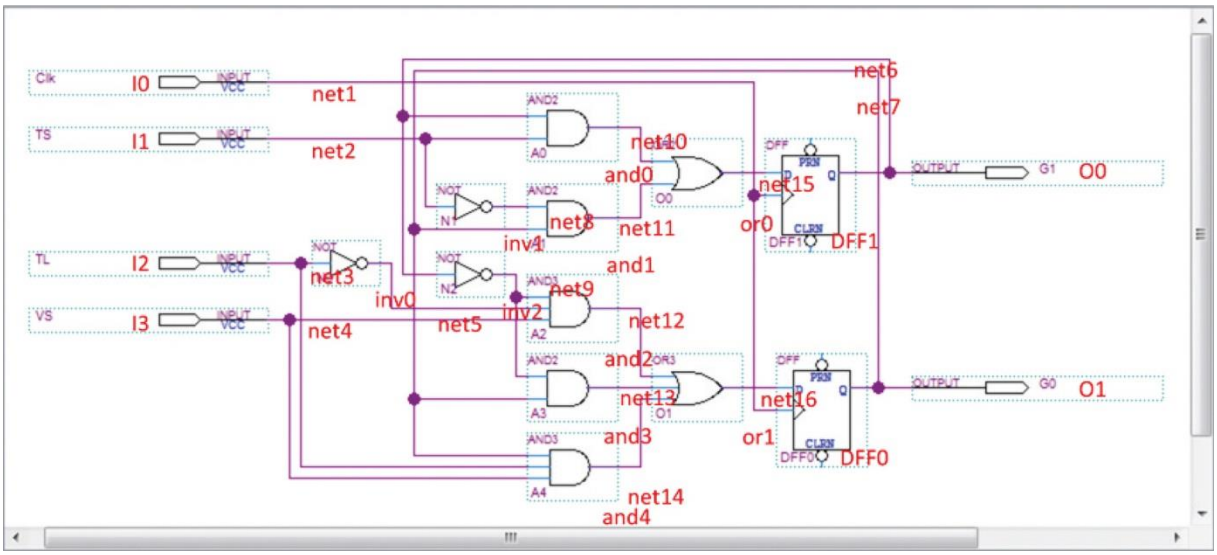
Şekil 10.41

Netlist

Netlist, bileşenleri ve bunların nasıl birbirine bağlandıklarını açıklayan bir bağlantı listesidir. Genellikle, bir netlist kullanılan bileşenlerin veya öğelerin açıklamalarına referanslar içerir. Bir mantık kapısı gibi bir bileşen netlistede her kullanıldığında, buna örnek denir. Her örnek, bu tür bir bileşene yapılabilecek bağlantıları ve bu bileşenin bazı temel özelliklerini listeleyen bir tanımlamaya sahiptir. Bu bağlantı noktalarına bağlantı noktaları veya pinler denir. Genellikle, her örneğin benzersiz bir adı olur; örneğin, iki VE kapısının örneğiniz varsa, biri "ve1" diğeri "ve2" olabilir. İsimleri dışında, aynı olabilirler. Ağlar, devrede bir şeyleri birbirine bağlayan "tellerdir". Ağ tabanlı ağ listeleri genellikle tüm örnekleri ve özelliklerini açıklar, ardından her ağı açıklar ve her bir durumda hangi bağlantı noktasına bağlanacaklarını belirtir. Sentez yazılımı, Şekil 10-42 (a) 'da gösterildiği gibi bir ağ listesi oluşturur. Netlist, bir devreyi tanımlamak için gerekli olan bilgi tipini belirtir. Ağ listeleri için kullanılan bir biçim EDIF'tir (Elektronik Tasarım Değişim Biçimi- Electronic Design Interchange Format). Netlist kullanılarak, yazılım, Şekil 10–42 (b) 'de gösterildiği gibi, net görevlerinin şematik bir gösterimini oluşturur.

```
Netlist(SequentialLogic)
Net<name>: instance<name>,<from>,<to>;
Instances: and0,and1,and2,and3,and4,or0,or1,inv0,inv1,inv2,DFF0,DFF1;
Input/outputs:I1,I2,I3,I4,O1,O2
net1:   DFF0, inport2; DFF1, inport2; I1;
net2:   and0, inport2; inv1, outport1; I2;
net3:   inv0, outport1; and4, inport2; I3;
net4:   and2, inport3, and4, inport4; I4;
net5:   and2, inport2;
net6:   DFF1, outport1; and0, inport1; inv2, output1; O0;
net7:   DFF0, outport1; and1, inport2; and3, inport2; and4, inport1; O1;
net8:   and1, inport1;
net9:   and2, inport1; and3, inport1;
net10: or0, inport1;
net11: or0, inport2;
net12: or1, inport1;
net13: or1, inport2;
net14: or1, inport3;
net15: DFF1, inport1;
net16: DFF0, inport1;
end;
```

(a) Netlist



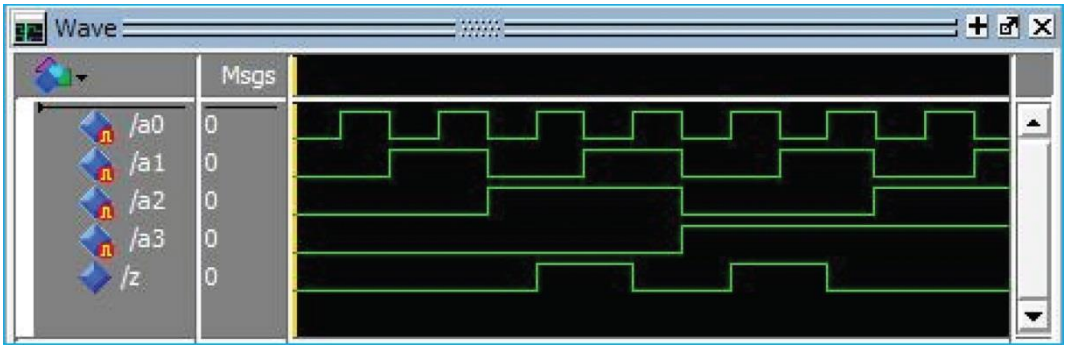
Şekil 10.42

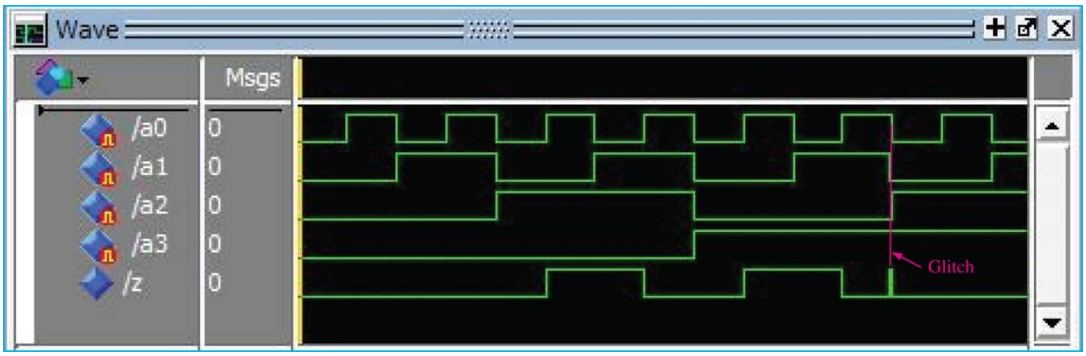
Uygulama (Yazılım)

Tasarım sentezlendikten sonra, derleyici, temel olarak tasarımın “eşlemesi” olan tasarımı uygular; Bu işleme fitting veya yönlendirme denir. Tasarım akışının uygulama aşamasını gerçekleştirmek için, yazılım belirli cihaz hakkında “bilgi sahibi” olmalı ve ayrıntılı pin bilgisine sahip olmalıdır. Tüm potansiyel hedef cihazlardaki eksiksiz veriler genellikle yazılım kütüphanesinde saklanır.

Zamanlama Simülasyonu

Tasarım akışının bu kısmı uygulamadan sonra ve hedef cihaza indirilmeden önce gerçekleşir. Zamanlama simülasyonu, devrenin tasarım frekansında çalıştığını ve genel çalışmayı etkileyecek zamanlama problemleri olmadığını doğrular. Fonksiyonel bir simülasyon zaten yapıldığından, devre mantıksal açıdan düzgün çalışmalıdır. Geliştirme yazılımı, tasarımın zamanlama simülasyonunu gerçekleştirmek için kapıların yayılma gecikmeleri gibi belirli hedef cihaz hakkındaki bilgileri kullanır. Fonksiyonel simülasyon için, hedef cihazın spesifikasyonu gerekli değildi; ancak zamanlama simülasyonu için hedef cihaz seçilmelidir. Dalga Biçimi Düzenleyicisi, zamanlama simülasyonunun sonucunu, Şekil 10–43'te gösterildiği gibi, fonksiyonel simülasyonda olduğu gibi görüntülemek için kullanılabilir.



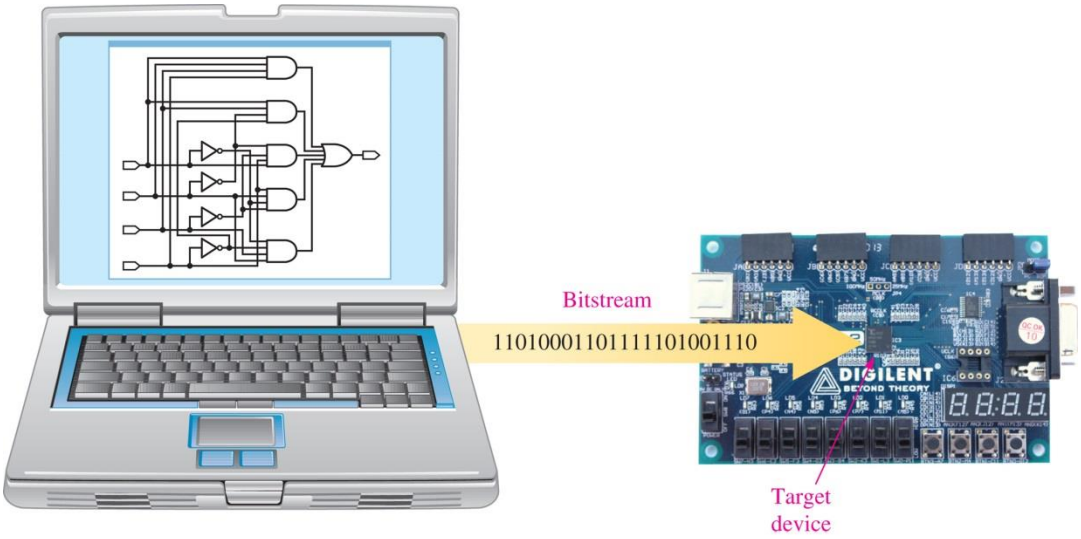


Şekil 10.43

Zamanlama ile ilgili herhangi bir sorun yoksa, (a) bölümünde gösterildiği gibi, tasarım indirilmeye hazırdır. Bununla birlikte, zamanlama simülasyonunun, Şekil 10–43 (b) 'de gösterildiği gibi yayılma gecikmesi nedeniyle bir "aksaklık" ortaya çıkardığını varsayalım. Bir aksaklık, dalga biçiminde çok kısa süreli bir ani yükselmedir. Bu durumda, nedenin tasarımını dikkatlice analiz etmeniz, ardından değiştirilmiş tasarıma tekrar girmeniz ve tasarım akış işlemini tekrarlamanız gerekir.

Cihaz Programlama (İndirme)

Fonksiyonel ve zamanlama simülasyonları tasarımın düzgün çalıştığını doğruladıktan sonra, indirme sırasını başlatabilirsiniz. Nihai tasarımı temsil eden bir bit akımı oluşturulur ve otomatik olarak yapılandırmak için hedef cihaza gönderilir. Tamamlandığında, tasarım aslında donanımda ve devrede test edilebilir. Şekil 10-44, temel indirme kavramını gösterir.



Şekil 10.44