İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

SAHADA PROGRAMLANABİLİR KAPI DİZİLERİ İLE LOJİK DEVRE TASARIMI

YÜKSEK LİSANS TEZİ

Müh. Volkan SEZER

Tezin Enstitüye Verildiği Tarih :27 Mayıs 1996

Tezin Savunulduğu Tarih :1 Ekim 1996

Tez Danışmanı : Prof. Dr. Ahmet DERVİŞOĞLU

Diğer Jüri Üyeleri : Prof. Dr. Ali Nur GÖNÜLEREN

Doç. Dr. Bülent ÖRENCİK

EKİM 1996

ÖNSÖZ

Bu konuya beni yönelten ve değerli yardımlarını esirgemeyen sayın hocam Prof. Dr. Ahmet Dervişoğlu'na ve her zaman beni destekleyen çalışma arkadaşım Müh. Sıddıka Berna Örs'e teşekkür ederim.

Eylül, 1996 Volkan Sezer

İÇİNDEKİLER

KISALTMALAR

ASIC:	Aplication Specific Integrated Circuit (Uygulamaya Özgü Tümdevre)
BDD:	Binar Decision Diagram (İkili Karar Diyagramı)
CLB:	Configurable Logic Block (Konfigüre Edilebilir Lojik Blok)
EEPROM:	Electrically Erasable Programmable Read Only Memory (Elektrik ile Silinebilir, Programlanabilir Salt Oku Bellek)
EPROM:	Erasable Programmable Read Only Memory (Silinebilir, Programlanabilir Salt Oku Bellek)
ERA:	Electrically Reconfigurable Array (Elektriksel Biçimlendirilebilir Dizi)
FPGA:	Field Programmable Gate Array (Sahada Programlanabilir Kapı Dizisi)
LABs:	Logic Array Blocks (Lojik Dizi Blokları)
LUT:	Look-up Table (Bir FPGA yapısı)
MPGA:	Mask-Programmable Gate Array (Maske ile Programlanabilir Kapı Dizisi)
MUX:	Multiplexer (Çoklayıcı)
PAL:	Programmable Array Logic (Programlanabilir Dizi Lojiği)
PIA:	Programmable Interconnect Array (Programlanabilir Bağlantı Dizisi)
PLA:	Programmable Logic Array (Programlanabilir Lojik Dizi)
PLD:	Programmable Logic Device (Programlanabilir Lojik Ürün)
PROM:	Programmable Read Only Memory (Programlanabilir Salt Oku Bellek)
RAM:	Random Acces Memory (Rastgele Erişimli Bellek)
TFI(n):	Transitive Fanin (n düğümüne dolaylı veya dolaysız bağlantısı olan düğümlerin oluştuşturduğu küme)
TFO(n):	Transitive Fanout (n düğümünün dolaylı veya dolaysız olarak bağlandığı düğümlerin oluşturduğu küme)
TTL:	Transistor-Transistor Logic (Tranzistör-Tranzistör Lojiği)
VLSI:	Very Large Scale Integration (Çok Büyük Ölçekli Tümleştirme)

ÖZET

Sahada Programlanabilir Kapı Dizileri (Field Programmable Gate Array, FPGA) genel olarak Boole fonksiyonlarının gerçekleştirildiği bloklar, bu bloklar arasındaki bağlantıları sağlayan bağlantı kanalları ve giriş-çıkış bloklarından oluşur. Fonksiyonların gerçekleştirildiği bloklarda yer alan ve "Look-up Table (LUT)" adı verilen devreler m ($m \geq 2$) değişkenli her Boole fonksiyonunu gerçekleyebilirler. Verilen bir fonksiyonun değişken sayısının m veya m'den az olması sonucunda bu fonksiyon bir tane m-LUT ile gerçeklenebilir. Değişken sayısının m'den büyük olması durumunda fonksiyonun birden fazla m-LUT ile gerçeklenmesi gerekir. LUT-tabanlı FPGA'lar kullanılarak yapılan tasarımlarda amaç devrelerin minimum sayıda LUT kullanılarak gerçeklenmesidir. Bu amaçla ilk olarak, verilen fonksiyonlar, herbiri bir m-LUT ile gerçeklenebilecek alt fonksiyonlara ayrıştırılırlar. İkinci işlem olarak bazı uygun alt fonksiyonlar aynı LUT'lar ile gerçeklenerek toplam LUT sayısı azaltılır.

Bu tezde sayısal sistem tasarımında kullanılan Sahada Programlanabilir Kapı Dizilerinin mimarileri, programlama teknolojileri ve özellikleri birbirleriyle karşılaştırılarak incelenmiştir. Piyasada mevcut olan çeşitli FPGA tiplerinin genel yapıları anlatılmıştır.

Ayrıca, Boole fonksiyonlarını minimum sayıda LUT kullanarak bir FPGA'ya yerleştiren "mis-fpga", "chortle-crf", "VISMAP" ve "TechMap" yöntemleri incelenerek bazı hususlar açıklığa kavuşturulmuştur.

Bu tezde "mis-fpga" yönteminin birinci aşaması olan "Ayrıştırma" işleminde kullanılan fonksiyonel ayrıştırma, kutulama ve kofaktör yöntemleri örneklerle karşılaştırılmış ve n+k değişkenli fonksiyonların n değişkenli Karnaugh diyagramı ile indirgenebilmesinden faydalanılarak, fonksiyonel ayrıştırmanın Karnaugh diyagramı ile yapılmasına ilişkin bir yol önerilmiştir. Ayrıca yöntemin ikinci aşamasını oluşturan blok sayısının azaltılması işleminde ilk adım olan m-gerçeklenebilir süper düğümlerin belirlenmesi problemi için bir algoritma geliştirilmiş ve ilk adımda elde edilen süper düğümlerden minimum elemanlı bir örtünün bulunmasına ilişkin bir yol önerilmiştir.

SUMMARY

DIGITAL DESIGN BY FIELD PROGRAMMABLE GATE ARRAY

Modern VLSI technology has made it possible to implement powerful digital circuits at reasonably low costs. The traditional VLSI design techniques, however, require extensive manufacturing efforts taking several months from beginning to end. This results in a high cost for each unit, unless large volumes are produced, due to the high overhead to begin production of such chips. On the other hand, in the electronic industry, it is vital to get new products to the market in the shortest time. Therefore, reduction of the design, development, production and testing times is quite crucial. Furthermore, it is important that the financial risk incurred in the development of a new product be limited so that more new ideas can be prototyped. As a solution of these requirement, programmable logic arrays have been introduced, among which Field Programmable Gate Arrays (FPGA's) are the most promising.

Like an MPGA, an FPGA consists of an array of programmable logic blocks (PLB's) that can be interconnected in a general way. Like a PAL, the interconnections between these blocks are also user-programmable. FPGA's were introduced in 1985 by the Xilinx Company. Since then, many different FPGA' s have been developed by a number of companies.

A typical FPGA consists of a two-dimensional array of logic blocks that can be connected by general interconnection resources. The interconnect comprises segment of wire, where the segment may be of various lengths. Present in the interconnect are programmable switches that serve to connect the logic blocks to the wire segments or one segment to another. Logic circuits are implemented in the FPGA by partitioning the logic into individual logic blocks and then interconnecting the blocks as required via the switches.

FPGA's can be used effectively in a wide variety of applications. In comparison with MPGA's , they have two significant advantages: FPGA's have lower prototype costs and shorter production times.

The two main disadvantages of FPGA's, compared to MPGA's, are their relatively low speed of operation and lower logic density (the amount of logic that can be implemented in a single chip). The propagation delay of an FPGA is adversely affected by the inclusion of programmable switches, which have significant resistance and capacitance, in the connections between logic blocks. Logic density is decreased

because the programmable switches and associated programming circuitry require a great deal of chip area compared to the metal connections in an MPGA.

Over the last few years, several companies have introduced a number of different types of FPGA's. While each product has unique features, they can all be classified into one of four categories: symmetrical array, row-based, hierarchical PLD and sea-of gates.

There are a number of different programming technology of implementing a programming element. Programming technologies that are currently in use commercial products are: static RAM cells, anti-fuses, EPROM transistors and EEPROM transistors. The static RAM programming technology is used in FPGAs which are produced by several companies: Algotronix, Concurrent Logic, Plessey Semiconductors and Xilinx. In these FPGAs, programmable connections are made using pass-transistors, transmission gates or multiplexers that are controlled by SRAM cells. Since the SRAM is volatile, these FPGAs must be configured each time the power is applied to the chip. This implies that a system that includes such chips must have some sort of permanent storage mechanism for the RAM cell bits, such as a ROM or a disk. The major advantage of this technology is that it provides an FPGA that can be re-configured (in-circuit) very quickly. Anti-fuse programming technology is used in FPGAs offered by Actel Corp., QuickLogic and Crosspoint Solutions. An anti-fuse normally resides in a high-impedance state but can be "fused" into a low-impedance state when programmed by a high voltage. EPROM programming technology is used in the FPGAs manufactured by Altera Corp. and Plus Logic. This technology is the same as that used in EPROM memories. One advantage of EPROM transistors is that they are re-programmable but do not require external storage. However, unlike SRAM, EPROM transistors can not be re-programmed in-circuit. The EEPROM approach which is used in the FPGAs offered by Advanced Micro Devices is similar to the EPROM technology except that EEPROM transistors can be re-programmed in-circuit. While each of these technologies are quite different, the programming elements all share the property of being configurable in one of two states: ON or OFF.

Depending on the application in which the FPGA is to be used, it may also be desirable for the programming element to possess other features. For example, a programming element that is non-volatile might be attractive, as well as an element that is re-programmable. Re-programmable elements make it possible to re-configure the FPGA, perhaps without even removing it from circuit board.

There are two popular categories of FPGA block structures, namely Look-up Table (LUT) based and multiplexor based; the resulting architectures are called LUT-based and MUX-based architectures, respectively.

The basic block of an LUT architecture is a look-up table that can implement any Boolean function of up to m inputs. For a given LUT architecture, m is a fixed number. In commercial architectures, m is typically between 3 and 6. An m-LUT is typically implemented by static random access memory (SRAM) that has m address lines and 1 data line. In commercial LUT-based architectures, each basic block has

one or more LUTs, along possibly with other logic elements (such as flip-flops, fast carry logic, etc.).

In the MUX-based architectures, the basic block is a configuration of multiplexors, with possibly a few additional logic gates such as ANDs and ORs.

An important problem in synthesis by using FPGAs is to minimize the cost of a design, where the cost measured by the number of LUTs needed. Minimizing the total number of LUTs allows the implementation of larger logic networks with the fixed number of m-LUT in a given FPGA. In order to solve this problem, first m-infeasible functions should be made m-feasible. Each node function in the resulting network can then be realized by m-LUT. However, it may be possible to reduce the number of m-LUTs if the nodes in the resulting network are small, i.e. more than one can be implemented within the same m-LUT. This called block count minimization (BCM).

There exist a number of LUT technology mappers. All these programs realise a Boolean function by the use of m-LUTs, attempting to minimize either the total number of LUTs or the number of levels of LUTs in the final circuit. Minimizing the total number of LUTs allows the implementation of large number Boolean function with the fixed number of look-up tables available in a given FPGA, and minimizing the number of levels improves the speed-performance of circuits.

In this thesis, LUT technology mappers that are "mis-fpga", "chortle-crf", "VISMAP" and "TechMap" are studied.

The mis-fpga technology mapper minimizes the number of m-LUTs required to implement a Boolean function in two phases. The first phase decomposes the original function to ensure that every node can be implemented by a single m-LUT and the second phases uses heuristic solution to a covering problem to reduce the number of LUTs in the final circuit.

In the first phase, three approaches are used to decompose functions that cannot be implemented by a single LUT. The first approach is functional decomposition that is based on Roth-Karp decomposition, the second adapts the bin-packing approach and the third is based on Shannon cofactoring.

The fist systematic study on decomposition was done by Ashenhurst. He characterized the existence of a simple disjoint decomposition of a function. This work could not be used for functions with more than 10-15 inputs, since it required a construction of a decomposition chart, a modified form of the truth table for a function. Ashenhurst gave necessary and sufficient condition for existence of a simple decomposition of a function f of n variables: The simple disjoint decomposition exists if and only if the corresponding decomposition chart has at most two distinct column patterns. Not every function has a simple disjoint decomposition. Roth and Karp extended the decomposition theory of Ashenhurst by characterizing a general disjoint decomposition, which is of the following form:

$$f(x_1,..,x_s,y_1,..,y_{n-s})=g(\alpha_1(x_1,..,x_s),\ \alpha_2(x_1,..,x_s),\ldots,\ \alpha_t(x_1,..,x_s),\ y_1,..,y_{n-s}).$$

The functional decomposition can be implemented efficiently using BDDs and Karnaugh map.

Cube-packing as a method of decomposition for LUT architectures was first suggested in chortle-crf. The basic idea is to approximate the problem of decomposing a function as that of decomposing an SOP (sum of products) of the function. Cube-packing uses bin-packing. The objective is to pack all items using a minimum number of bins without violating the capacity of any bin. Here, the used capacity of a bin is the sum of the weights of the items in the bin. If each cube in the SOP of the function f is treated as an item with weight equal to its literal count, and an LUT as a bin with capacity m, the problem of decomposing the SOP of f into a minimum number of m-LUTs can be seen as a bin-packing problem.

Cofactoring is a special case of functional decomposition. Cofactoring a function $f(x_1,..,x_n)$ with respect to x_i can also be done using a functional decomposition on f with the input partition $(\{x_1,...,x_{i-1},x_{i+1},..,x_n\}|\{x_i\})$.

After decomposition, m-feasible functions are obtained, which can be implemented straightaway and the LUTs by mapping each node to an LUT. This strategy, however, yields sub-optimal results. There is a transformation that reduce the number of feasible nodes called covering. The covering problem can be stated as: given m-feasible Boolean network N, iteratively collapse nodes into their fanouts such that the resulting network $\tilde{N}$ is m-feasible and the number of nodes in $\tilde{N}$ is minimum. One way to solve this problem is given in the following.

1)Enumerate all possible m-feasible supernodes (A supernode corresponding to a node *n* of the network N is a directed subgraph S of N, with a single root *n* such that S does not contain any primary input node of N).

2) Select a minimum subset of these supernodes that covers the network.

In this thesis, one algorithm is offered for enumerating all possible m-feasible supernodes and one way is offered for selecting a minimum subset of these supernodes that covers the network.

Chortle-crf maps a Boolean function into a circuit of m-LUTs. Its objective is to minimize the number of LUTs in the final circuit. The original network is first partitioned into a forest of trees and then each tree is separately mapped into a subcircuit of m-LUTs. The final circuit is then assembled from the subcircuits implementing the trees. The major innovation of Chortle-crf is that it simultaneously addresses the decomposition and covering problem using a bin-packing approximation algorithm.

The way bin-packing is implemented in Chortle-crf is that it treats the same input going to two gates as different. Better results may be obtained if it is recognized that the same input feeds the two gates. However, placing cubes that share variables in the same LUT may not always yield the optimum solution. Furthermore, there may be several product terms that share variables, but considered together may not fit in the same LUT. So the question of which groups of product terms should be merged into a

single LUT arises. Since the bin-packing algorithm is very fast, the solution chosen by chortle-crf for these problems is to run the algorithm exhaustively on all possible cases, i.e. no merging and all possible mergings of cubes with shared variables. The combination that leads to fewest LUTs is selected.

The VISMAP technology mapper focuses only on the block count minimization problem. It assumes that appropriate decomposition has been performed and an m-feasible network is available. The approach is similar to that of mis-fpga. The main difference is that VISMAP does not generate all the supernodes, but a subset. The basic idea is to label each edge of the network as either visible or invisible. An edge is called visible if it will be implemented as a wire between two LUTs in the circuit. An invisible edge is absorbed within an LUT. Note that a visible edge corresponds to an input to a supernode in the final mapped circuit and an invisible edge to an edge absorbed within a supernode in the mapped circuit. If the network has a large number of edges, it is computationally expensive to go through all possible labelings. The network is then partitioned into sub-networks. The problem is solved optimally on these sub-networks and the solutions are glued together.

The TechMap uses a AND-OR decomposition. In the covering phase, a compatibility graph G(N) is formed whose vertices correspond to the fanins of N. Two fanins i and j of N are said to be compatible if $|\sigma(i) \cup \sigma(j)| \leq m$ ($\sigma(i)$:local support of i). This means that i and j can be combined and a new node, say k, can be created. If i and j are compatible, an edge (i,j) is added to G(N). After all pairs of fanins have been checked for compatibility and edges appropriately added to G(N), the vertices of G(N) are selected for merging.

BÖLÜM 1

GİRİŞ

Programlanabilir lojik ürünler (PLD) sayısal devrelerin tasarlanmasında uzun zamandır önemli kolaylıklar sağlamaktadırlar. 1970 yılında PROM'ların ve 1980 başlarında 4-5 TTL tümdevre yerine geçen PAL tümdevrelerinin Boole fonksiyonlarının gerçeklenmesinde kullanılmasıyla PLD alanında günümüze kadar süren bir yarış başlamıştır. Geçen zaman içinde tümdevrelerin kapı eşdeğerlikleri birkaç yüzden binlere ulaşmıştır. PLD teknolojisinin en önemli uygulaması Sahada Programlanabilir Kapı Dizileri'dir (Field Programmable Gate Array, FPGA).

Günümüzde Çok Büyük Çapta Tümleştirme (VLSI) teknolojisi, düşük maliyetle güçlü sayısal devrelerin tasarlanabilmesine olanak sağlamaktadır. Bu teknoloji sayesinde milyonlarca tranzistörden oluşan devreler yapılabilmektedir. Standart hücreler ve Maske Programlamalı Kapı Dizileri (Mask-Programmable Gate Arrays, MPGAs) gibi yapılar Uygulamaya Özel Tümdevrelerin (Aplication-Specific Integrated Circuits, ASICs) oluşturulmasında büyük kolaylıklar sağlamaktadır. Ancak bu yapılar tasarım esnasında, büyük bir emeğin ve yıllara varan bir zamanın harcanmasına neden olmaktadırlar. Bunun sonucu olarak bir tasarımın maliyeti 20000$ ile 200000$ arasında değişmekte dolayısıyla maliyetin karşılanabilmesi için devreden çok miktarda üretilmesi gerekmektedir [1]. Ayrıca tasarım sırasında çok zaman harcanması da benzer bir devrenin başka bir firma tarafından, daha önce üretilerek piyasaya sürülmesine, dolayısıyla harcanan emek ve paranın boşa gitmesine neden olabilmektedir. FPGA' lar yukarıda anlatılan tasarım risklerinin en aza indirilmesinde önemli rol oynarlar. FPGA'lar kolayca silinebilmeleri ve tekrar programlanabilmeleri ile prototiplerin kısa zamanda tasarlanmasına dolayısıyla tasarım maliyetinin en aza indirilmesine imkan sağlarlar.

Bu tezde kombinezonsal lojik devrelerin LUT-tabanlı (Look-up Table) FPGA' lar ile gerçeklenmesinde kullanılan yöntemler incelenmiş ve yöntemlerde iyileştirmeler yapılmıştır.

Tezin ikinci bölümünde, FPGA' ların mimarileri ve özellikleri incelenerek genel yapıları ve programlama teknolojileri hakkında bilgi verilmiştir. Ayrıca pazarda mevcut olan FPGA' lar ve bu FPGA' ların genel özellikleri belirtilmiştir.

Üçüncü bölümde, kombinezonsal lojik devrelerin LUT-tabanlı (Look-up Table) FPGA' lar ile gerçeklenmesinde kullanılan "mis-fpga", "chortle-crf", "VISMAP" ve "TechMap" yöntemleri incelenmiştir. "mis-fpga" yönteminin birinci aşaması olan "Ayrıştırma" işleminde kullanılan fonksiyonel ayrıştırma, kutulama ve kofaktör yöntemleri örneklerle karşılaştırılmış ve n+k değişkenli fonksiyonların n değişkenli Karnaugh diyagramı ile indirgenebilmesinden faydalanılarak, fonksiyonel ayrıştırmanın Karnaugh diyagramı ile yapılmasına ilişkin bir yol önerilmiştir. Ayrıca yöntemin ikinci aşaması olan blok sayısının azaltılması işleminde ilk adım olan m-gerçeklenebilir süper düğümlerin belirlenmesi problemi için bir algoritma geliştirilmiş ve ilk adımda elde edilen süper düğümlerden minimum elemanlı bir örtünün bulunmasına ilişkin bir yol önerilmiştir.

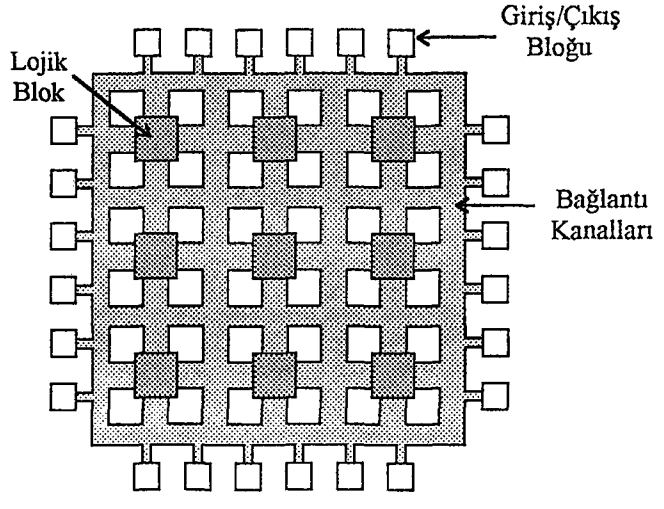
BÖLÜM 2

SAHADA PROGRAMLANABİLİR KAPI DİZİLERİNİN MİMARİSİ VE ÖZELLİKLERİ

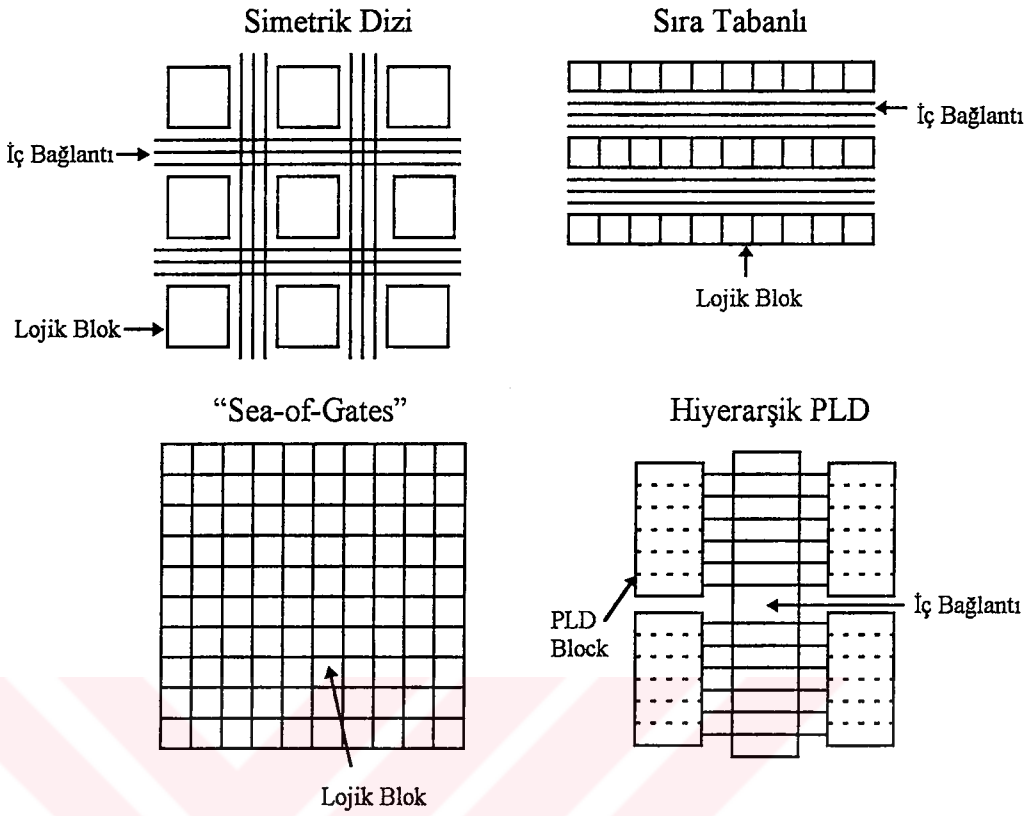
FPGA' lar da MPGA'lar gibi bağlantıları çeşitli yollarla yapılan lojik bloklardan oluşmuşlardır. Bağlantılar PAL'lerde olduğu gibi kullanıcılar tarafından programlanmaktadır. 1985 yılında Xilinx firmasının ilk FPGA' yı üretmesinin ardından Actel, Altera, Plessey, Plus, AMD, QuickLogic, Algotronix gibi firmalar da değişik özellikleri olan FPGA' lar üretmişlerdir.

Şekil 2-1'de de görüldüğü gibi bir FPGA genel olarak Boole fonksiyonlarının gerçekleştirildiği bloklar, bu bloklar arasındaki bağlantıyı sağlayan bağlantı kanalları ve giriş-çıkış bloklarından oluşmaktadır.

Çeşitli firmalar tarafından değişik özelliklere sahip FPGA'lar üretilmesine karşın FPGA' ları genel mimarilerine göre Şekil 2-2'de görüldüğü gibi başlıca 4 sınıfa ayırmak mümkündür [1].



Şekil 2-1 Genel FPGA yapısı



Şekil 2-2 FPGA'ların genel mimarilerine göre sınıflandırılması

2.1 FPGA'ların Programlama Teknolojileri

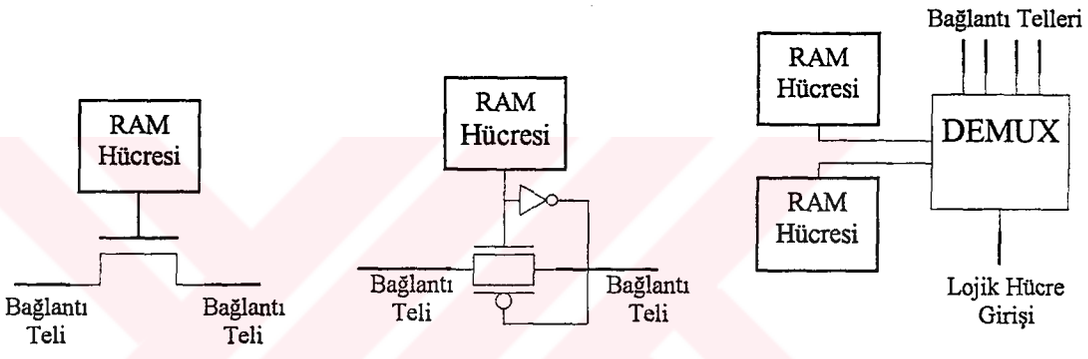
Aşağıda FPGA'larda kullanılan çeşitli programlama teknolojileri incelenmiştir.

2.1.1 Statik RAM Programlama Teknolojisi

Statik RAM programlama teknolojisi Algotronix, Concurrernt Logic, Plessey Semiconductors ve Xilinx firmalarının ürettiği FPGA'larda kullanılmaktadır. Bu FPGA'larda programlanabilir bağlantılar statik RAM hücreleri tarafından kontrol edilen geçiş tranzistörleri, iletim kapıları veya çoklayıcılarla (MUX) yapılmaktadır. Şekil 2-2'de RAM hücresiyle kontrol edilen yapılar görülmektedir. Geçiş tranzistörü kullanılan uygulamalarda, RAM hücresi Şekil 2-3a ve Şekil 2-3b'de görüldüğü gibi tranzistörün iletimde veya kesimde olmasını sağlar. Çoklayıcı kullanılan uygulamalarda RAM hücresi Şekil 2-3c'de de görüldüğü gibi çoklayıcının girişinin hangi çıkışa bağlanacağını kontrol eder. Bu programlama yönteminin kullanıldığı

FPGA'lar, statik RAM'lar uçucu olduğu için entegreye her gerilim uygulandığında yeniden programlanmalıdırlar. Ayrıca programlama verisinin bir dış bellekte saklanması gerekliliği bu tip FPGA' lar kullanılarak tasarlanan devrelerin kolaylıkla kopyalanabilmesine imkan sağlamaktadır.

Diğer programlama yöntemlerinin kullanıldığı FPGA'larla karşılaştırıldığında bu yöntemin kullanıldığı FPGA'ların, statik RAM'ların kapladıkları alan nedeniyle daha büyük oldukları görülür. Ancak kısa sürede programlanabilmeleri bu tip FPGA'ların üstünlüğünü oluşturur.



a) Geçiş Tranzistörü b) İletim Kapısı c) Çoklayıcı (DEMUX)

Şekil 2-3 Statik RAM programlama teknolojisi

2.1.2 "Anti - fuse" Programlama Teknolojisi

"Anti-fuse" programlama teknolojisi Actel Corp., QuickLogic ve Crosspoint Solutions tarafından üretilen FPGA'larda kullanılmaktadır. PAL, PLA, EPROM gibi diğer programlanabilir lojik elemanlarda programlama öncesinde sigortalar mevcuttur. Programlama esnasında gerekli akım ve gerilimlerin uygulanmasıyla kullanılmayan bağlantılar (sigortalar) kesilir. Fakat FPGA'larda kullanılan "Anti -fuse" teknolojisinde programlamadan önce entegrede hiç bağlantı (sigorta) yoktur. Programlama sırasında uygulanan gerilim ve akımlarla gerekli bağlantılar (sigortalar) oluşturulur. Böylece genelde, kullanılan bağlantı sayısının kullanılmayan bağlantı sayısından (attırılmış sigorta) daha az olması sebebiyle programlama süresi kısalır.

Bu teknoloji ile programlanan FPGA'lar sigortaların çok az alan kaplamaları sebebiyle diğerlerine göre daha küçüktürler. Ayrıca programlama verisinin saklanması için kolayca kopyalanabilen bir dış belleğe ihtiyaç duyulmaması, yapılan tasarımların kopyalanmasını kesinlikle önler.Fakat PAL'ler gibi sadece bir kez programlanabilmeleri sebebiyle prototip geliştirme aşamasında maliyeti arttırabilirler.

2.1.3 EPROM ve EEPROM Programlama Teknolojisi

Altera Corp. ve Plus Logic firmaları tarafından üretilen FPGA'larda kullanılan EPROM teknolojisi EPROM hafizalardakine benzemektedir. Bu tip FPGA'ların da tekrar programlanabilmeleri için devreden sökülerek ultraviyole ışınlarına tutulması gerekmektedir. Advanced Micro Devices firması tarafından üretilen FPGA'larda kullanılan EEPROM teknolojisinin EPROM teknolojisine göre üstünlüğü FPGA'ların devreden sökülmeden elektrikle silinebilmesidir. Ancak bu tip FPGA'lar EPROM teknolojisini kullananlara göre iki kat büyüktürler.

Tablo 2-1'de yukarıda anlatılan programlama teknolojileri çeşitli yönlerden karşılaştırılmışlardır.

Tablo 2-1 FPGA' ların programlama teknolojileri

Programlama Teknolojisi	Uçuculuk	Tekrar Programlanabilme	Silikon Alanı
Statik RAM	evet	devre üzerinde	büyük
"Anti - fuse "	hayır	hayır	küçük
EPROM	hayır	devre dışında	küçük
EEPROM	hayır	devre üzerinde	2*EPROM

2.2 FPGA'ların Yapıları

FPGA'lar yapılarına göre başlıca iki gruba ayırılabilir. Bu yapılar aşağıda açıklanmıştır.

2.2.1 "Look-Up Table (LUT)" Tabanlı Yapı

"Look-up Table" tabanlı yapının temel bloğu "Look-up Table (LUT)" adı verilen ve m (m≥2) değişkenli her Boole fonksiyonunu gerçekleştirebilen devredir. Verilen bir LUT yapısı için m genelde 3 ile 6 arasında olan sabit bir sayıdır. Genelde bu yapı m tane adres, 1 tane de veri yolu olan statik RAM ile gerçekleştirilir ve kısaca m-LUT olarak adlandırılır. LUT-tabanlı yapıda her temel blok bir ya da daha fazla LUT ile flip-flop gibi diğer lojik elemanlardan oluşur.

2.2.2 Çoklayıcı (MUX) Tabanlı Yapı

MUX-tabanlı yapının temel bloğu çoklayıcıların çeşitli konfigürasyonlarından ve olabildiğince az VE ve VEYA gibi lojik kapılardan oluşur. Bu yapıdaki FPGA' ların içinde veri tutucu ve flip-flop gibi bellek elemanları bulunmadığından çoklayıcılar ile bu elemanların gerçeklenmesi gerekmektedir.

2.3 Başlıca FPGA Tipleri

Aşağıda bazı özellikleri Tablo 2-2'de karşılaştırmalı olarak verilmiş olan çeşitli FPGA'lar ayrıntılı olarak incelenmiştir.

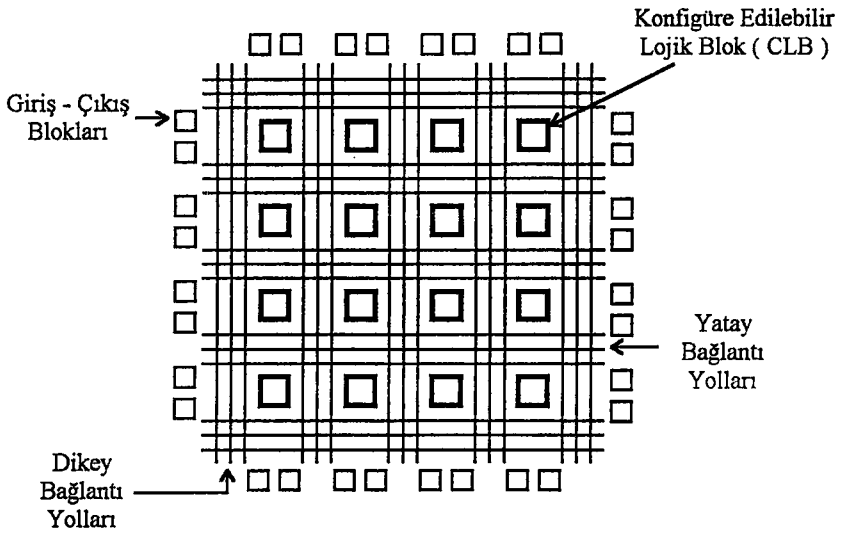
2.3.1 Xilinx Firmasının Ürettiği FPGA'lar

Şekil 2-4'de Xilinx firması tarafından üretilmiş olan LUT-tabanlı FPGA'ların genel yapısı görülmektedir. Bu yapıdan da görülebileceği gibi bu FPGA'lar "Konfigüre Ebilebilir Lojik Blok (Configurable Logic Block -CLB)" adı verilen programlanabilir blok dizileri ile satırlar ve sütunlar arsındaki bağlantı kanallarından oluşmaktadırlar[3].

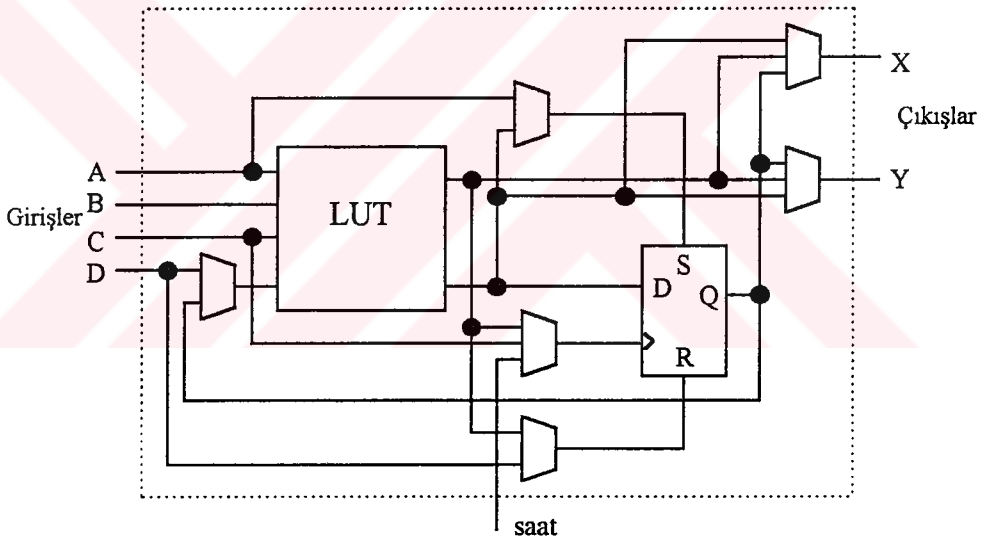
Bu FPGA'larda Statik RAM programlama teknolojisi kullanılmaktadır ve XC2000, XC3000, XC4000 olmak üzere üç modelleri bulunmaktadır. Şekil 2-5'de iç yapısı görülen XC2000 CLB ile 4 değişkenli herhangi bir Boole fonksiyonu veya toplam değişken sayısı en fazla 4 olan 3 değişkenli herhangi iki Boole fonksiyonu gerçekleştirilebilir. CLB'nin çıkışlarının ikisi de doğrudan veya biri bir flip-flop üzerinden olabilir.

Tablo 2-2 Mevcut FPGA' ların özelliklerinin karşılaştırılması

Firma	Genel Mimari	Lojik Blok Tipi	Programlama Teknolojisi
Xilinx	Simetrik Dizi	"Look-up Table"	Statik RAM
Actel	Sıra Tabanlı	Çoklayıcı Tabanlı	"anti-fuse"
Altera	Hiyerarşik-PLD	PLD Blok	EPROM
Plessey	"Sea-of-gates"	TÜVE-Kapısı	Statik RAM
Plus	Hiyerarşik-PLD	PLD Blok	EPROM
AMD	Hiyerarşik-PLD	PLD Blok	EEPROM
QuickLogic	Simetrik Dizi	Çoklayıcı Tabanlı	"anti-fuse"
Algotronix	"Sea-of-gates"	Çoklayıcı& Temel Kapılar	Statik RAM
Concurrent	"Sea-of-gates"	Çoklayıcı&Temel Kapılar	Statik RAM
Crosspoint	Sıra Tabanlı	Tranzistör Çiftleri &Çoklayıcı	"anti-fuse"



Şekil 2-4 Xilinx FPGA'ların genel yapısı



Şekil 2-5 XC2000 CLB

XC3000 serisi FPGA'lar XC2000 serisinin geliştirilmiş modelidir. Bu FPGA'lar ile 5 değişkenli herhangi bir Boole fonksiyonu veya toplam değişken sayısı 5 olan 4 değişkenli herhangi iki Boole fonksiyonu gerçekleştirilebilir.XC3000 serisinin CLB'leri ikisi de kombinezonsal ya da ardışıl olabilecek iki çıkışa sahiptir [3].

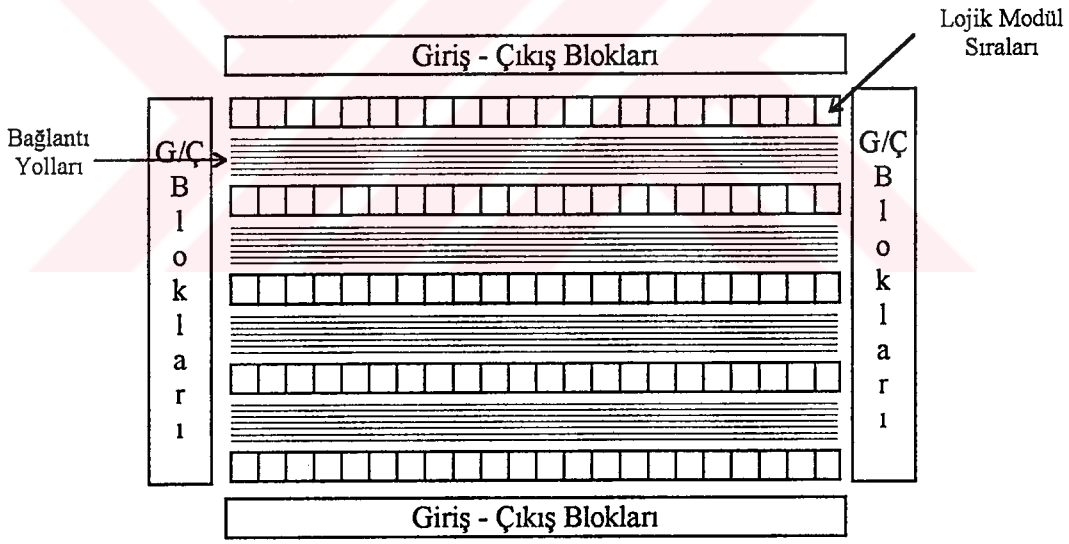
XC4000 serisi, Xilinx firmasının ürettiği en gelişmiş FPGA'lardır. Bu serinin her CLB'si ile beş değişkenli herhangi bir fonksiyon, değişkenleri birbirinde farklı 4

değişkenli herhangi iki fonksiyon ve 9 değişkenliye kadar bazı fonksiyonlar gerçekleştirilebilir [3].

2.3.2 Actel Firmasının Ürettiği FPGA'lar

Şekil 2-6'da Actel firması tarafından üretilmiş olan MUX-tabanlı FPGA'ların genel yapısı görülmektedir. Bu yapıdan da görülebileceği gibi bu FPGA'lar "Lojik Modül" adı verilen programlanabilir bloklar ile bunların arasındaki yatay bağlantı kanallarından oluşmaktadırlar.

Bu FPGA'larda "Anti - Fuse" programlama teknolojisi kullanılmaktadır ve Act - 1, Act - 2 olmak üzere iki modelleri bulunmaktadır. Şekil 2-7'de iç yapısı görülen Act-1 lojik modülü ile 2 değişkenli herhangi bir fonksiyon, 3 değişkenli pek çok fonksiyon ve 4 değişkenli bazı fonksiyonlar gerçekleştirilebilir [1].

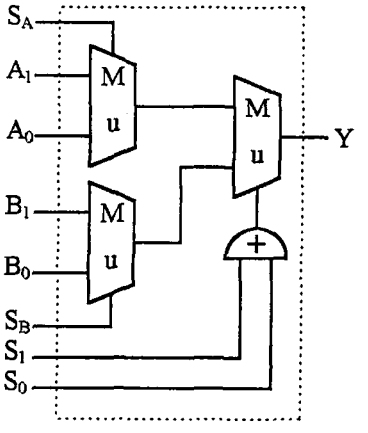


Şekil 2-6 Actel FPGA'ların genel yapısı

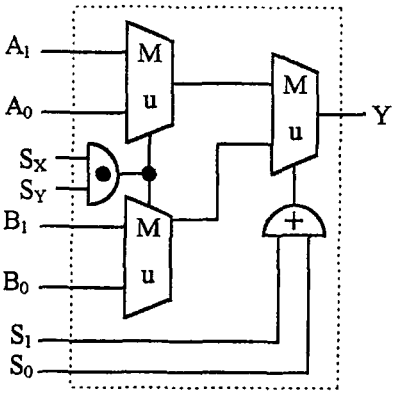
2.3.3 Altera Firmasının Ürettiği FPGA'lar

Altera FPGA' ları PLD gruplarından oluşmaları sebebiyle şimdiye kadar anlatılan FPGA' lardan oldukça farklıdırlar. Genel yapısı Şekil 2-9'da görülen Altera FPGA' larında EPROM programlama teknolojisi kullanılmaktadır. Bu FPGA' lar "Lojik Dizi Blokları (Logic Array Blocks, LABs)" adı verilen programlanabilir bloklardan ve

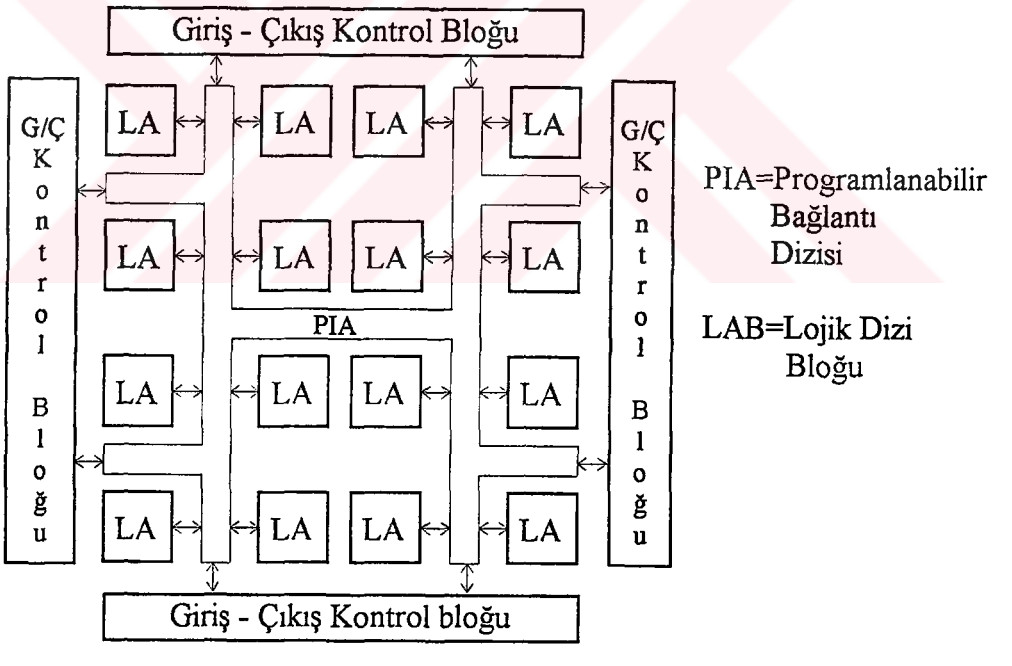
görevi bu bloklar arasındaki bağlantıları sağlamak olan "Programlanabilir Bağlantı Dizisi (Programmable Interconnect Array, PIA)" adı verilen yapıdan oluşur [4].



Şekil 2-7 Act-1 Lojik Modül



Şekil 2-8 Act-2 Lojik Modül

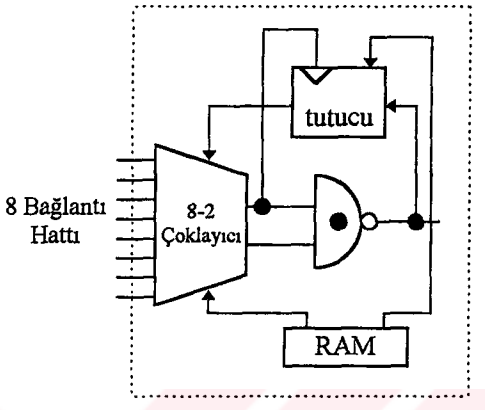


Şekil 2-9 Altera FPGA'ların genel yapısı

2.3.4 Plessey Firmasının Ürettiği FPGA' lar

Plessey FPGA' larına "Elektrik ile Yeniden Düzenlenebilir Dizi (Electrically Reconfigurable Array, ERA)" adı verilir. Statik-RAM programlama teknolojisi

kullanan bu FPGA'lar lojik blok'ların "Sea-of-Gates" mimarisi ile bir araya getirilmelerinden oluşur. Şekil 2-10'da görüldüğü gibi yapıları oldukça basit olan lojik bloklar bir 8-e-2 çoklayıcının çıkışlarına bağlanmış olan bir TÜVE kapısı ve bir tutucudan oluşmaktadır. Statik-RAM tarafından kontrol edilen çoklayıcı, lojik bloğun diğer bloklarla olan bağlantısını denetlemektedir [5].



Şekil 2-10 Plessey Lojik Bloğu

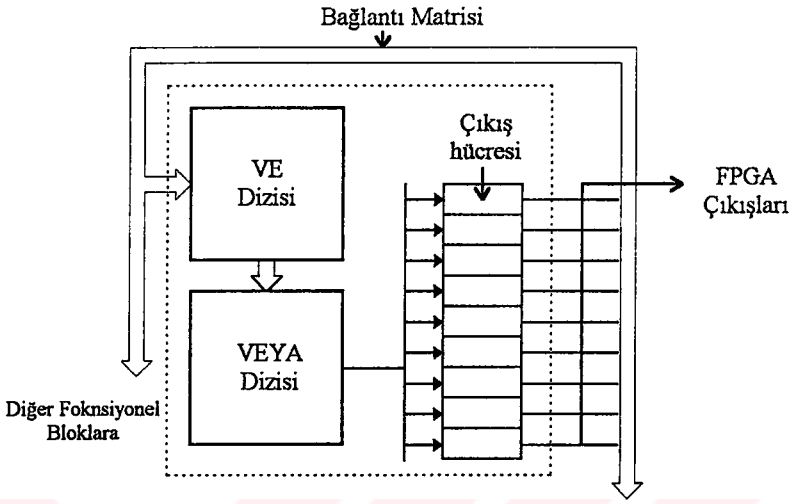
2.3.5 Plus Logic Firmasının Ürettiği FPGA'lar

Plus Logic FPGA'ları, yapıları Şekil 2-11'de görülen Fonksiyonel Bloklar'ın (FB) bir bağlantı matrisiyle birbirlerine bağlanmalarından oluşur. EPROM programlama teknolojisi kullanan bu FPGA' lar Altera FPGA' larına benzemelerine karşın, FB'lar LAB'lara göre daha karmaşıktır. Şekil 2-11'den de görülebileceği gibi FB'ların iç yapısı PLA'larınkine benzemektedir. VE dizisinin çıkışlarına bağlı olan VEYA dizisi FB'un dokuz çıkışını oluşturmaktadır [6].

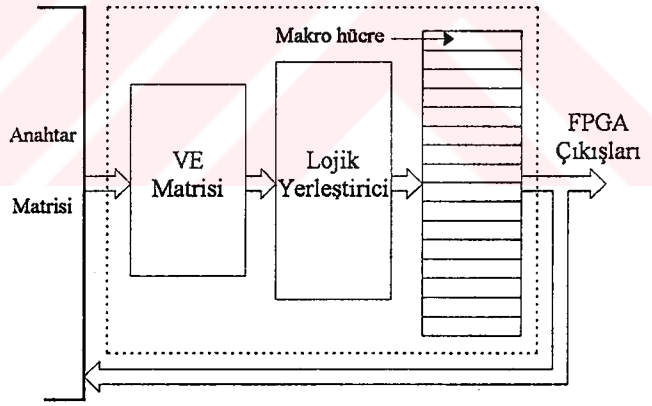
2.3.6 Advanced Micro Devices (AMD) Firmasının Ürettiği FPGA' lar

EEPROM programlama teknolojisi kullanan AMD FPGA' ları, iç yapıları Şekil 2-12'de verilen PAL bloklarının Altera FPGA' larına benzer şekilde, bir anahtar matrisiyle biribirine bağlanmasından oluşur. Her PAL blok bir VE-matrisi, bir lojik yerleştirici (LA) ve 16 elemanlı bir makro hücre kümesi olmak üzere üç temel yapıdan oluşur. LA'lar VE matrisinde üretilen çarpım terimlerinin makro hücrelere uygun

şekilde iletilmesini sağlarlar. Bir PAL bloğunun çıkışları anahtar matrisi yardımıyla diğer bloklara da iletilir [7].



Şekil 2-11 Plus Logic Fonksiyonel Blok

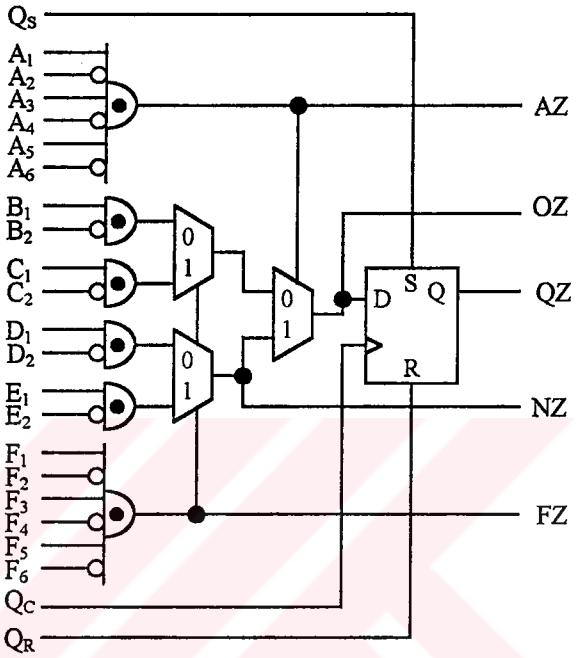


Şekil 2-12 AMD PAL Bloğu

2.3.7 QuickLogic Firmasının Ürettiği FPGA' lar

QuickLogic FPGA' ları, iç yapıları Şekil 2-13'de görülen lojik bloklardan (pASIC Logic Blocks -pLB) oluşmaktadır. Firma tarafından üretilen ilk modellerde 48 -380 lojik blok bulunmaktaydı. Şekil 11'den de görülebileceği gibi 2-girişli dört tane VE kapısını çıkışlarına bağlı olan çoklayıcıların girişleri 6 girişli bir VE kapısı (F_1-F_6) tarafından seçilmektedir. Benzer şekilde ikinci seviyedeki çoklayıcının da girişleri 6-

girişli bir VE kapısı ile seçilmektedir. Ayrıca bu çoklayıcının çıkışı D-tipi bir flip-flop ile tutulabilmektedir. 6-girişli VE kapılarının çıkışları da direk olarak lojik bloğun çıkışı olarak alınabilmektedir. Lojik bloklar arasındaki bağlantılar yatay ve dikey bağlantı kanalları ile sağlanmaktadır [1].



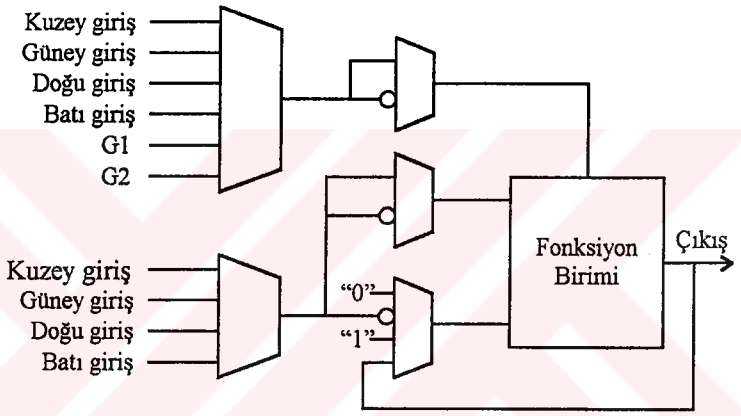
Şekil 2-13 QuickLogic Lojik Bloğu

2.3.8 Algotronix Firmasının Ürettiği FPGA' lar

Algotronix FPGA' ları, konfigüre edilebilir blokların 32*32 lik bir matris şeklinde düzenlenmesi ile oluşmuştur. Matristeki her bloğun dört tarafındaki (sağ, sol, üst, alt) komşu blok ile direk bağlantısı vardır. Şekil 2-14'de bir bloğun iç yapısı görülmektedir. Algotronix FPGA'ların mimarisi, Plessey FPGA' larınınkine benzer şekilde "Sea-of-gate" mimarisidir. Ancak Algotronix FPGA'ların bağlantı kanallarının boyunun sabit olmasına karşın (sadece komşu bloklar arasında bağlantı var) Plessey FPGA' larınki değişkendir. Buna karşın Algotronix FPGA'ların lojik blokları daha komplekstir. Ayrıca Algotronix FPGA'ların her giriş/çıkış bacağı 3-durumlu kapılar sayesinde aynı anda giriş ve çıkış olarak kullanılabilir [1].

2.3.9 Concurrent Logic Firmasının Ürettiği FPGA' lar

Concurrent Logic FPGA' ları özdeş lojik blokların "Sea-of-gates" mimarisine göre bir araya getirilmesinden oluşur. Bir FPGA' da bu bloklardan 3136 tane bulunur. Lojik bloklar, kullanıcı tarafından düzenlenen çoklayıcılardan, temel kapılardan ve bir adet D-tipi flip-flop'tan oluşur. Bu FPGA' lar bir lojik blokla kolaylıkla bir yarı-toplayıcının gerçeklenebilmesi sebebiyle özellikle matematiksel uygulamalar için uygundurlar [1].



Şekil 2-14 Algotronix Fonksiyon Birimi

2.3.10 Crosspoint Solutions Firmasının Ürettiği FPGA' lar

Crosspoint Solutions FPGA' ları tranzistör seviyesinde programlanabilmeleri nedeniyle diğerlerinden oldukça farklıdırlar. Yapı, temel olarak yatay bağlantı yolları ile ayrılmış olan tranzistör satırlarından oluşur. Her satırda biri seri bağlı NMOS diğeri seri bağlı PMOS tranzistörlerden oluşmuş iki sıra vardır. Satırlar arasındaki bağlantıları sağlamak için de düşey bağlantı yolları mevcuttur. Bu tip FPGA' lar "anti-fuse" teknolojisi ile programlanırlar ve mimarileri sıra tabanlıdır [1].

BÖLÜM 3

KOMBİNEZONSAL LOJİK DEVRELERİN LUT TABANLI FPGA' LAR İLE GERÇEKLENMESİNDE KULLANILAN BAZI YÖNTEMLER

Bir lojik devre tasarımındaki en önemli amaç her türlü tasarımda olduğu gibi maliyetin en aza indirilmesidir. Lojik kapılar kullanılarak yapılan tasarımlarda maliyet ölçüsünün kapı ve giriş sayısı olmasına karşılık FPGA'lar kullanılarak yapılan tasarımlarda maliyet ölçüsü gereken lojik blok sayısıdır. FPGA ile tasarımın maliyetini gerçeklenecek fonksiyonun değişken sayısı belirlemektedir. Dolayısıyla lojik kapılar ile tasarlandığında maliyeti fazla olacak bir devre FPGA ile çok daha az bir maliyetle gerçekleştirilebilir. Ya da lojik kapılar ile değişik maliyetleri olan devreler FPGA ile aynı maliyette gerçekleştirilebilir. Örneğin $f_1 = \overline{x}_1\overline{x}_2\overline{x}_3x_4x_5 + \overline{x}_1\overline{x}_2x_3x_4\overline{x}_5$, $f_2 = \overline{x}_1\overline{x}_2\overline{x}_3x_4x_5x_6$, $f_3 = x_1x_2 + \overline{x}_2x_3$ fonksiyonlarının 2 seviyeli VE-VEYA devresi ile gerçeklenmesi durumunda f_1 için 12 giriş ve 3 kapı, f_2 için 6 giriş ve 1 kapı, f_3 için 6 giriş ve 3 kapı gerekmektedir. Aynı fonksiyonların 5-LUT tabanlı bir FPGA ile gerçeklenmesinde ise f_1 için 1 tane 5-LUT, f_2 için 2 tane 5-LUT ve f_3 için 1 tane 5-LUT kullanılmalıdır. Görüldüğü gibi lojik kapılarla gerçeklemede en fazla maliyeti olan f_1 fonksiyonu FPGA ile en az maliyetle gerçekleştirilebilmektedir. Tersine lojik kapılarla en az maliyetle gerçekleştirilen f_2 fonksiyonu ise FPGA ile en fazla maliyet ile gerçekleştirilebilmektedir.

Önceki bölümde de anlatıldığı gibi bir m-LUT ile m değişkenli her Boole fonksiyonu gerçeklenebilir. Gerçeklenmesi istenen fonksiyonun değişken sayısının m'den büyük olması durumunda ise birden fazla m-LUT kullanılması zorunluluğu doğar. Bu bölümde incelenen yöntemlerin ortak amacı fonksiyonların minimum sayıda ya da minimum seviyede LUT kullanılarak gerçeklenmesidir. Fonksiyonların gerçeklenmesi

için minimum sayıda LUT kullanılması, LUT sayıları sabit olan FPGA' lara daha büyük devrelerin yerleştirilmesini, seviyelerin azaltılması ise devrelerin daha hızlı çalışmasını sağlar.

3.1 "mis-fpga" Yöntemi

"mis-fpga" yöntemiyle gerçekleme, 1990 yılında R. Murgai, R.K. Brayton, Y. Nishizaki, N.Shenoy ve A. Sangiovanni-Vincentelli tarafından geliştirilmiştir. m-LUT yapısı için yöntemin iki aşaması vardır. "Ayrıştırma" adı verilen ilk aşamada değişken sayısı m'den fazla olan fonksiyonlar (m-gerçeklenemez fonksiyonlar) m değişkenli alt fonksiyonlara (m-gerçeklenebilir) ayrılmaktadır. "Blok Sayısının Azaltılması (Block Count Minimization, BCM)" adı verilen ikinci aşamada ise uygun bazı alt fonksiyonlar aynı m-LUT'larla gerçeklenerek kullanılan m-LUT sayısı azaltılmaktadır.

3.1.1 Boole Fonksiyonlarının Ayrıştırılması (m-gerçeklenemez Fonksiyonların m-gerçeklenebilir Hale Getirilmesi)

m-gerçeklenemez fonksiyonlar çeşitli yöntemlerle m-gerçeklenebilir alt fonksiyonlara ayrıştırılabilirler. Böylece elde edilen alt fonksiyonların her biri bir m-LUT ile gerçeklenebilir. Aşağıda 3 çeşit ayrıştırma yöntemi verilmiştir.

- Fonksiyonel ayrıştırma
- Kutulama ("Cube-packing")
- Kofaktörlere ayırma

3.1.1.1 Fonksiyonel Ayrıştırma

Ayrıştırma üzerine ilk sistematik çalışma Ashenhurst tarafından yapılmıştır. Bu çalışma sonucunda geliştirilen yöntem en fazla 10-15 girişli fonksiyonlara uygulanabilmektedir. Birkaç yıl sonra Roth-Karp tarafından daha gelişmiş bir yöntem önerilmiştir.

3.1.1.1.1 Ashenhurst Ayrıştırma Yöntemi

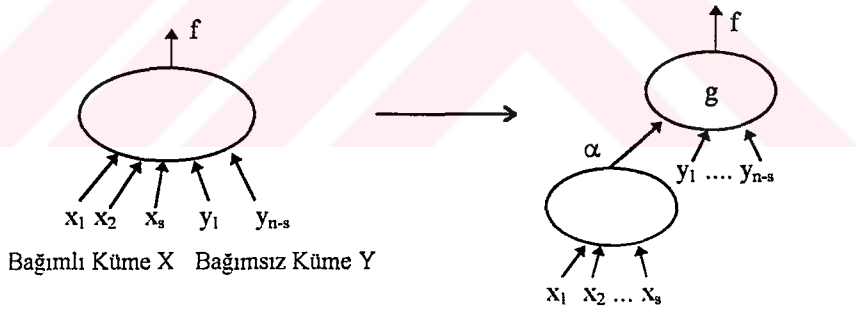
Ashenhurst tarafından önerilen bu yöntem "Ayrıştırma Tablosu" adı verilen ve fonksiyonun doğruluk tablosuna benzer bir tablodan faydalanılarak verilen fonksiyonun,

$$f(x_1,.,x_s, y_1,.,y_{n-s})=g(\alpha(x_1,...,x_s), y_1,.....,y_{n-s}) \quad (\{x_1,.,x_s\}\cap\{y_1,.,y_{n-s}\}=\varnothing) \qquad (3.1)$$

şekline getirilmesini sağlamaktadır. Bu yönteme "Basit Ayrık Ayrıştırma" da denmektedir. (3-1) ifadesi $X = \{ x_1,....,x_s \}$, $Y = \{ y_1,.....,y_{n-s} \}$ olmak üzere,

$$f(X,Y) = g(\alpha(X), Y) \qquad (X \cap Y = \varnothing) \qquad (3.2)$$

şeklinde de yazılabilir. Bu ifadedeki g fonksiyonu "Ayrıştırmanın Görüntüsü", X kümesi "Bağımlı Küme", Y kümesi "Bağımsız Küme" olarak adlandırılır.



Şekil 3-1 Ashenhurst ayrıştırması

Bu yöntemin temelini oluşturan ayrıştırma tablosu D(X|Y), X|Y ayrıştırması için sütunları X kümesinin, satırları Y kümesinin elemanlarının olası bütün kombinezonlarından oluşan matris biçiminde bir doğruluk tablosudur. Şekil 3-2'de $f = \bar{x}_1\bar{x}_3 + x_2\bar{x}_3 + x_1\bar{x}_2x_3$ fonksiyonunun $x_1x_2|x_3$ ayrıştırmasına ilişkin ayrıştırma tablosu görülmektedir. Bir fonksiyonun bu yöntemle ayrıştırılabilmesi için gerek ve yeter koşul ayrıştırma tablosunda en fazla iki farklı sütun vektörünün olmasıdır[2].

x_1x_2 / x_3	00	01	10	11
0	1	1	0	1
1	0	0	1	0

Şekil 3-2 $f = \overline{x}_1\overline{x}_3 + x_2\overline{x}_3 + x_1\overline{x}_2x_3$ Fonksiyonunun Ayrıştırma Tablosu

Ayrıştırma tablosunda aynı vektöre sahip sütunların oluşturduğu kümeye "Eşdeğerlik Sınıfı" adı verilir ve bu sütunlar birbiriyle "Eşdeğerdir" denir . Örneğin Şekil 3-2 de verilen f fonksiyonunun eşdeğerlik sınıfları $C_1(x_1,x_2)=\{00,01,11\}$, $C_2(x_1,x_2)=\{10\}$ dır ve $00 \approx 01 \approx 11$ yazılır.

Bu tezde, eşdeğerlik sınıfları genel olarak aşağıdaki biçimde incelenmiştir. n+k değişkenli bir Boole fonksiyonu, m_i'ler ilk n değişkene ilişkin mintermler olmak üzere,

$$f(x_1,..,x_n,x_{n+1},..,x_{n+k})=m_0f_0(x_{n+1},..,x_{n+k})+....+ m_{2^n-1}f_{2^n-1}(x_{n+1},..,x_{n+k}) \qquad (3.3)$$

şeklinde ifade edilebilir. Eğer X|Y ayrışımı için k tane eşdeğerlik sınıfı oluşuyorsa (3.3) ifadesi,

$$f(x_1,..,x_{n+k}) = \tilde{f}_1(x_{n+1},..,x_{n+k})(\underbrace{\sum \text{bazı } m_i\text{'ler}}_{C_1})+..+\tilde{f}_k(x_{n+1},..,x_{n+k})(\underbrace{\sum \text{bazı } m_i\text{'ler}}_{C_k}) \qquad (3.4)$$
$$\left[C_1+..+C_k=1, \qquad C_i.C_j=0 \quad i\neq j \quad (C_i\cap C_j=\varnothing \quad i\neq j)\right]$$

şeklinde yazılabilir. Özel bir hal olarak 2 tane eşdeğerlik sınıfı varsa (3.4) ifadesi,

$$f(x_1,..,x_{n+k}) = C_1\tilde{f}_1(x_{n+1},..,x_{n+k})+C_2\tilde{f}_2(x_{n+1},..,x_{n+k}) \qquad (3.5)$$
$$C_1+C_2=1 \Rightarrow C_1=\overline{C}_2$$

şekline gelir.

$\alpha(x_1,..,x_n)$ fonksiyonu C_1 veya C_2 fonksiyonuna eşit seçilirse (3.5) ifadesi

$$f(x_1,..,x_{n+k}) = g(\alpha(X),Y) = \alpha \tilde{f}_1(x_{n+1},..,x_{n+k}) + \overline{\alpha}\, \tilde{f}_2(x_{n+1},..,x_{n+k}) \quad (3.6)$$

şeklinde yazılabilir. Örneğin Şekil 3-2'de ayrıştırma tablosu verilen f fonksiyonu (3.4) ifadesine benzer şekilde

$$f(x_1,x_2,x_3) = \overline{x}_1\overline{x}_2\,\overline{x}_3 + \overline{x}_1 x_2\,\overline{x}_3 + x_1\overline{x}_2\,x_3 + x_1 x_2\,\overline{x}_3$$
$$f(x_1,x_2,x_3) = \underbrace{(\overline{x}_1\overline{x}_2 + \overline{x}_1 x_2 + x_1 x_2)}_{C_1}\overline{x}_3 + \underbrace{(x_1\overline{x}_2)}_{C_2}x_3 \quad (3.7)$$

olarak ifade edilebilir. $\alpha=C_2$ olarak seçilirse seçilirse f fonksiyonu $x_1x_2|x_3$ ayrışımı için

$$f(x_1,x_2,x_3) = g(\alpha(x_1,x_2),x_3) = \overline{\alpha}\,\overline{x}_3 + \alpha x_3 \quad (\alpha = x_1\overline{x}_2) \quad (3.8)$$

şeklinde ifade edilebilir. $\alpha=C_1$ seçilmesi durumunda f fonksiyonu

$$f(x_1,x_2,x_3) = g(\alpha(x_1,x_2),x_3) = \alpha\overline{x}_3 + \overline{\alpha}\,x_3 \quad (\alpha = \overline{x}_1 + x_2) \quad (3.9)$$

şeklinde ayrıştırılacaktır. f fonksiyonu 2-LUT'lar kullanılarak ayrıştırılmadan gerçeklenirse 6 tane, ayrıştırılarak gerçeklenirse 2 tane LUT gerekecektir.

3.1.1.1.2 Roth-Karp Ayrıştırma Yöntemi

Şekil 3-3'de ayrıştırma tablosu verilen $f(x_1,x_2,x_3) = \overline{x}_1 x_2\overline{x}_3 + \overline{x}_1\overline{x}_2 x_3 + x_1\overline{x}_2\overline{x}_3$ fonksiyonu için k=3 olduğundan, yani üç eşdeğerlik sınıfı olduğundan bu fonksiyon Ashenhurst yöntemiyle ayrıştırılamaz.

x_1x_2 / x_3	00	01	10	11
0	0	1	1	0
1	1	0	0	0

Şekil 3-3 $f(x_1,x_2,x_3) = \overline{x}_1 x_2\overline{x}_3 + \overline{x}_1\overline{x}_2 x_3 + x_1\overline{x}_2\overline{x}_3$ Fonksiyonunun Ayrıştırma Tablosu

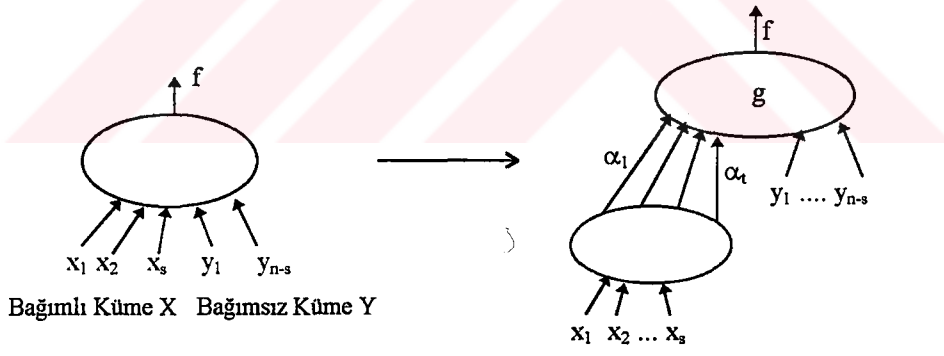
Bu şekildeki fonksiyonların ayrıştırılmasında Roth-Karp tarafından Ashenhurst yönteminin geliştirilmesiyle elde edilmiş olan yeni bir yöntem kullanılmaktadır. Bu yöntemle fonksiyon,

$$f(X,Y) = g(\alpha_1(X), \alpha_2(X),.., \alpha_t(X), Y) = g(\vec{\alpha}(X), Y);\ \vec{\alpha} = (\alpha_1,..,\alpha_t) \quad (3.10)$$

şekline getirilir. $\tilde{X}$, $\tilde{Y}$, $\tilde{Z}$, $\tilde{W}$ sonlu kümeler ve $\tilde{E}$, $\tilde{X} \times \tilde{Y}$ kümesinin bir alt kümesi olsun. $f: \tilde{E} \to \tilde{Z}$ fonksiyonu için $\alpha: \tilde{X} \to \tilde{W}$ olmak üzere $\forall (x,y) \in \tilde{E}$ için

$$f(x,y) = g(\alpha(x), y) \quad (3.11)$$

olacak şekilde bir $g: \tilde{W} \times \tilde{Y} \to \tilde{Z}$ fonksiyonunun olması için gerek ve yeter koşul $\forall x_1, x_2 \in \tilde{X}$ için $\alpha(x_1) = \alpha(x_2) \Rightarrow x_1 \approx x_2$ olmasıdır [2]. Sonlu kümeler için tanımlanmış bu ifade $\tilde{X} = B^{|X|}$, $\tilde{Y} = B^{|Y|}$, $\tilde{W} = B^{|W|}$, $\tilde{Z} = B$ alınarak Boole cebrine uygulanabilir.



Şekil 3-4 Roth-Karp ayrıştırması

$\vec{\alpha}$ ve g Fonksiyonlarının Bulunması:

Herhangi bir f fonksiyonu için belirlenen X bağımlı kümesi k tane eşdeğerlik sınıfına ayrılıyorsa bu k tane eşdeğerlik sınıfı, uzunlukları t ($t \geq \lceil \log_2 k \rceil$) olan kodlarla kodlanabilir. Kodlama değişkenleri α_1,..., α_t olarak seçilirse bu şekilde yapılabilecek

her kodlama değişik $\alpha_1,\ldots,\alpha_t$ fonksiyonları verecektir. g fonksiyonun bulunması için f fonksiyonundaki her terim (x,y), x bağımlı kısım, y bağımsız kısım olmak üzere,

$$\tilde{\alpha}_j = \begin{cases} \alpha_j, & \alpha_j(x)=1 \\ \overline{\alpha}_j, & \alpha_j(x)=0 \end{cases} \qquad (3.12)$$

ifadesinde verilen şarta göre $(\tilde{\alpha}_1\tilde{\alpha}_2\ldots\tilde{\alpha}_t, y)$ şeklinde bir terime dönüştürülür.

Şekil 3-3'de ayrıştırma tablosu verilen $f(x_1,x_2,x_3)=\overline{x}_1x_2\overline{x}_3+\overline{x}_1\overline{x}_2x_3+x_1\overline{x}_2\overline{x}_3$ fonksiyonu için k=3 tane eşdeğerlik sınıfının olduğu görülmektedir. Buna göre her sınıf 2 değişken (α_1, α_2) ile kodlanabilir ($t\geq\lceil\log_2 k\rceil\Rightarrow t\geq\lceil\log_2 3\rceil\Rightarrow t\geq 2\Rightarrow t=2$). Şekil 3-5'de yukarıda verilen f fonksiyonu için farklı kodlamalara karşı düşen α_1, α_2 ve g fonksiyonları görülmektedir.

	α_1'	α_2'	α_1''	α_2''
$C_1=\{00\}=\overline{x}_1\overline{x}_2$	0	1	1	1
$C_2=\{01,10\}=\overline{x}_1x_2+x_1\overline{x}_2$	0	0	0	1
$C_3=\{11\}=x_1x_2$	1	0	1	0

$$\alpha_1' = C_3 = x_1x_2 \qquad \alpha_1'' = C_1+C_3 = \overline{x}_1\overline{x}_2+x_1x_2$$

$$\alpha_2' = C_1 = \overline{x}_1\overline{x}_2 \qquad \alpha_2'' = C_1+C_2 = \overline{x}_1+\overline{x}_2$$

$$g' = \overline{\alpha}_1'\overline{\alpha}_2'\overline{x}_3+\overline{\alpha}_1'\alpha_2'x_3 \qquad g'' = \overline{\alpha}_1''\alpha_2''\overline{x}_3+\alpha_1''\alpha_2''x_3$$

Şekil 3-5 Farklı kodlamalara karşı düşen α ve g fonksiyonları

α fonksiyonlarının bulunmasında eşdeğerlik sınıfları kodlandığı gibi mintermler de kodlanabilir. Bazı durumlarda bu yöntem α ve g fonksiyonlarının daha basit olmasını sağlamaktadır. Ancak X kümesinin eleman sayısının büyük olması halinde mintermlerin sayısı çok artacağı için bu yöntem avantajlı olmamaktadır. Şekil 3-6'da yukarıda verilen $f(x_1,x_2,x_3)=\overline{x}_1x_2\overline{x}_3+\overline{x}_1\overline{x}_2x_3+x_1\overline{x}_2\overline{x}_3$ fonksiyonunun eşdeğerlik

sınıflarının, Şekil 3-7'de mintermlerinin kodlanmasına karşı düşen α ve g fonksiyonları görülmektedir.

	α_1	α_2	
$C_1=\{\overline{x}_1\overline{x}_2\}$	0	0	$\alpha_1 = x_1x_2$
$C_2=\{\overline{x}_1x_2, x_1\overline{x}_2\}$	0	1	$\alpha_2 = \overline{x}_1x_2 + x_1\overline{x}_2$
$C_3=\{x_1x_2\}$	1	0	$g = \overline{\alpha}_1\alpha_2\overline{x}_3 + \overline{\alpha}_1\overline{\alpha}_2x_3$

Şekil 3-6 Eşdeğerlik Sınıflarının kodlanmasına karşı düşen α ve g fonksiyonları

	α_1	α_2	
$\overline{x}_1\overline{x}_2$	0	0	$\alpha_1 = x_1$
$\overline{x}_1x_2$	0	1	$\alpha_2 = \overline{x}_1x_2 + x_1\overline{x}_2$
$x_1\overline{x}_2$	1	1	$g = \alpha_2\overline{x}_3 + \overline{\alpha}_1\overline{\alpha}_2x_3$
x_1x_2	1	0	

Şekil 3-7 Mintermlerin kodlanmasına karşı düşen α ve g fonksiyonları

Roth-Karp Ayrıştırma Yönteminin LUT Yapısına Uygulanması:

Roth-Karp ayrıştırma yönteminin m-LUT yapısına uygulanabilmesi için bağımlı küme X'in değişken sayısının en fazla m olması gerekir. Bir m-LUT ile değişken sayısı m veya daha az olan her fonksiyon gerçeklenebildiği için α fonksiyonlarının mümkün olduğunca basit olmasına çalışılmasının fazla bir anlamı yoktur. m-LUT ile gerçeklemede önemli olan, eşdeğer sınıfların sayısı dolayısıyla bunların kodlanmasında kullanılan kodların uzunluğudur. Çünkü X kümesinin maksimum m değişkenli olması nedeniyle elde edilen α fonksiyonları zaten m-gerçeklenebilir fonksiyonlardır. Ancak kod uzunluğunun bir bit artması α fonksiyonlarının bir tane artması anlamına gelmekte dolayısıyla bir tane daha m-LUT kullanılması gerekmektedir. Bu nedenle m-LUT kullanılarak yapılacak gerçeklemelerde eşdeğer sınıfların sayısının mümkün olduğunca az olmasını sağlıyacak X|Y ayrışımlarının bulunmasına çalışılmalıdır. Ayrıca

eşdeğerlik sınıflarının kodlanmasında kullanılan kodların uzunluğu minimum olmalıdır yani $t = \lceil \log_2 k \rceil$ seçilmelidir.

Teorem 3.1: Eşdeğerlik sınıflarının kodlanmasıyla oluşturulan α_i fonksiyonları ve eşdeğerlik sınıfları birbirleri cinsinden ifade edilebilirler:

$$\alpha_j = g(C_1,...,C_k) \Leftrightarrow C_i = h(\alpha_1,..,\alpha_j) \qquad j=1,2,...,t \quad i=1,2,....,k.$$

İspat:

α_j'ler C_i'ler cinsinden ifade edilebilirler. Çünkü, C_i fonksiyonları belli olduğuna göre, α_j'nin sütunundaki 1'lere karşı düşen C_i'lerin toplamı α_j'yi vermektedir; α_j bu şekilde tanımlanmaktadır.

Öte yandan, $C_i=\alpha_1^*\alpha_2^*...\alpha_t^*$ dir; burada C_i'nin satırındaki 1'lere karşı düşen sütunlardaki α_j'ler, 0'lara düşen sütunlardaki α_j'lerin tümleyenleri alınır. Eşdeğerlik sınıflarının toplamı 1 olduğu için α_j'nin tümleyeni, α_j'nin ifadesinde bulunmayan yani α_i'nin sütunundaki 0'lara karşı düşen eşdeğerlik sınıflarının toplamı olur. Dolayısıyla C_i, ya α_j ifadesinde ya da α_j'nin tümleyeninin ifadesinde bulunur. α_j'lerin eşdeğerlik sınıfları cinsinden ifadelerin C_i'de yerine konulmasıyla elde edilen toplamlar çarpımı biçimindeki ifadede her toplam teriminde sadece C_i ortaktır. Dolayısıyla eşdeğerlik sınıfları için $C_i \cdot C_j=0$ $(i\neq j)$ olduğundan toplamlar çarpımı biçimindeki ifade de çarpma işlemlerinin yapılması sonucunda sadece C_i kalır. Bu ifadeki her toplam teriminde $k\neq i$ olacak şekilde başka bir C_k sınıfının olması için C_i ve C_k'nin aynı koda sahip olması gerekir ki bu da durumda kodlama kurallarına aykırıdır.

Aşağıda m-LUT yapısına Roth-Karp yönteminin uygulanması bir örnek ile anlatılmıştır.

$f(x_1,..,x_5) = x_2x_4 + x_3x_4 + \overline{x}_4x_5 + x_1x_4\overline{x}_5$ fonksiyonunun minimum sayıda 4-LUT ile geçeklenmesi istensin. Bu amaçla ilk olarak $|X|=4$ olacak şekilde minimum sayıda eşdeğerlik sınıfı verecek bir ayrışım bulunmalıdır. Şekil 3-8'deki ayrıştırma

tablosundan $X=\{x_2,x_3,x_4,x_5\}$, $Y=\{x_1\}$ şeklindeki bir ayrışımın aşağıda verilen 3 tane eşdeğerlik sınıfını oluşturduğu görülmektedir.

$$(00)C_1=\{0000,0011,0100,1000,1100\}$$
$$(11)C_2=\{0001,0101,0110,0111,1001,1010,1011,1101,1110,1111\} \quad (3.13)$$
$$(01)C_3=\{0011\}$$

$\frac{x_2x_3x_4x_5}{x_1}$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0	1	0	0	0	1	1	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	0	1	1	1	0	1	1	1

Şekil 3-8 $f(x_1,..,x_5)=x_2x_4+x_3x_4+\overline{x}_4x_5+x_1x_4\overline{x}_5$ Fonksiyonunun Ayrıştırma Tablosu

Eşdeğerlik sınıfları x_i'ler (i=2,3,4,5) cinsinden (3.14) de verilen şekilde ifade edilebilir.

$$C_1=\overline{X}_4\overline{X}_5+\overline{X}_2\overline{X}_3X_4X_5$$
$$C_2=\overline{X}_4X_5+X_3X_4+X_2X_4 \quad (3.14)$$
$$C_3=\overline{X}_2\overline{X}_3X_4\overline{X}_5$$

f fonksiyonu bu eşdeğerlik sınıfları cinsinden Şekil 3-9'da görüldüğü gibi ifade edilebilir.

$\frac{x_2x_3x_4x_5}{x_1}$	C_1	C_2	C_3
0	0	1	0
1	0	1	1

Şekil 3-9 f Fonksiyonunun Eşdeğerlik Sınıfları cinsinden ifadesi

Şekil 3-9'daki tablodan f fonksiyonu,

$$f(x_1,..,x_5)=C_2\overline{x}_1+(C_2+C_3)x_1 \quad (3.15)$$

olarak bulunur. 3 tane eşdeğerlik sınıfı olduğu için bunları kodlamak için $t = \lceil \log_2 3 \rceil \Rightarrow t = 2$ bit uzunluğunda kodlar yeterlidir. Eşdeğerlik sınıfları Şekil 3-10'da verilen şekilde kodlanırsa $\bar{\alpha}$ fonksiyonları,

$$\begin{aligned} \alpha_1 &= C_1 + C_2 \Rightarrow \alpha_1 = x_2 + x_3 + \bar{x}_4 + x_5 \\ \alpha_2 &= C_2 + C_3 \Rightarrow \alpha_2 = \bar{x}_4 x_5 + x_3 x_4 + x_2 x_4 + x_4 \bar{x}_5 \end{aligned} \tag{3.16}$$

olarak bulunur.

	α_1	α_2
C_1	1	0
C_2	1	1
C_3	0	1

Şekil 3-10 Eşdeğerlik Sınıflarının kodlanması

Şekil 3-10 da verilen kodlama tablosundan,

$$\begin{aligned} C_1 = \alpha_1 \bar{\alpha}_2 \quad & (\alpha_1 = C_1 + C_2, \quad \alpha_2 = C_2 + C_3 \Rightarrow \bar{\alpha}_2 = C_1 \\ & \Rightarrow \alpha_1 \bar{\alpha}_2 = (C_1 + C_2) C_1 = C_1 + C_1 C_2 \\ & = C_1 + 0 \quad (C_i . C_j = 0 \quad i \neq j) \\ & = C_1) \end{aligned} \tag{3.17}$$

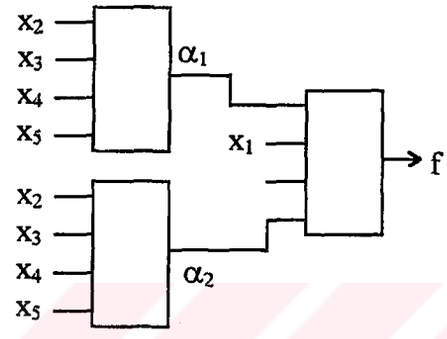
$$\begin{aligned} C_2 = \alpha_1 \alpha_2 \quad & (\alpha_1 = C_1 + C_2, \quad \alpha_2 = C_2 + C_3 \\ & \Rightarrow \alpha_1 \alpha_2 = (C_1 + C_2)(C_2 + C_3)_1 = C_2 + C_1 C_3 + C_2 + C_2 C_3 \\ & = 0 + 0 + C_2 + 0 \quad (C_i . C_j = 0 \quad i \neq j) \\ & = C_2) \end{aligned} \tag{3.18}$$

$$\begin{aligned} C_3 = \bar{\alpha}_1 \alpha_2 \quad & (\alpha_1 = C_1 + C_2 \Rightarrow \bar{\alpha}_1 = C_3, \quad \alpha_2 = C_2 + C_3 \\ & \Rightarrow \bar{\alpha}_1 \alpha_2 = C_3 (C_2 + C_3)_1 = C_2 C_3 + C_3 \\ & = 0 + C_3 \quad (C_i . C_j = 0 \quad i \neq j) \\ & = C_3) \end{aligned} \tag{3.19}$$

bulunur. (3.15) ifadesinde eşdeğerlik sınıfları cinsinden bulunan f fonksiyonu, (3.18) ve (3.19) ifadelerinin (3.15) ifadesinde yerine konulmasıyla α_1 ve α_2 cinsinden,

$$
\begin{aligned}
f(x_1,..,x_5) &= g(\alpha_1(X),\alpha_2(X),Y) = C_2\,\overline{x}_1 + (C_2 + C_3)x_1 \\
f(x_1,..,x_5) &= g(\alpha_1(X),\alpha_2(X),Y) = \alpha_1\alpha_2\,\overline{x}_1 + (\alpha_1\alpha_2 + \overline{\alpha}_1\alpha_2)x_1 \\
f(x_1,..,x_5) &= g(\alpha_1(X),\alpha_2(X),Y) = \alpha_1\alpha_2\,\overline{x}_1 + \alpha_2\,x_1 \\
f(x_1,..,x_5) &= g(\alpha_1(X),\alpha_2(X),Y) = \alpha_1\alpha_2 + \alpha_2\,x_1
\end{aligned}
\qquad (3.20)
$$

olarak bulunur. Bu ayrıştırma sonucunda f fonksiyonu Şekil 3-11'de görüldüğü gibi 3 tane 4-LUT kullanılarak gerçekleştirilebilir.



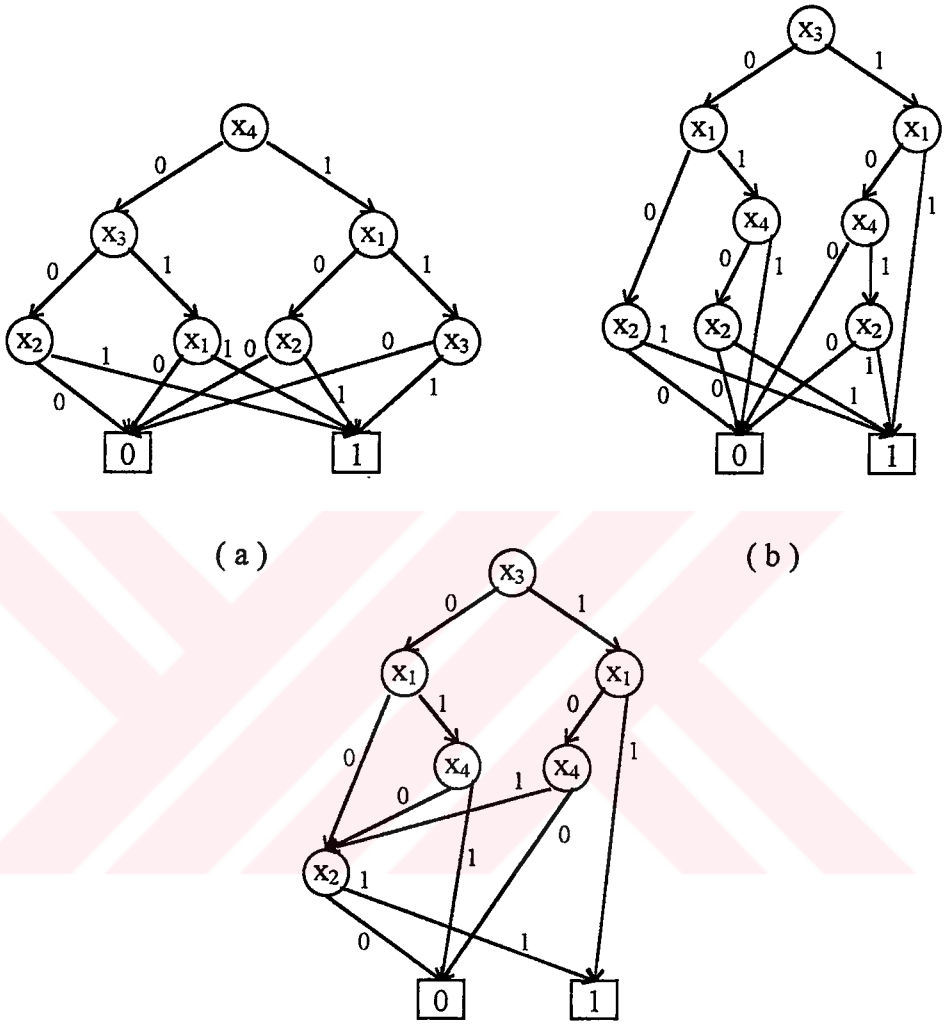
Şekil 3-11 $f(x_1,..,x_5) = x_2x_4 + x_3x_4 + \overline{x}_4x_5 + x_1x_4\overline{x}_5$ Fonksiyonunun 4-LUT'lar ile gerçeklenmesi

3.1.1.1.3 Fonksiyonel Ayrıştırmanın İkili Karar Diyagramlarından Faydalanılarak Yapılması

Boole fonksiyonlarının gösteriliş biçimlerinden biri olan ikili karar diyagramları (Binary Decision Diagram, BDD), bir kaynak düğümü ve iki terminal düğümü olan çevre içermeyen yönlü graflardır. Şekil 3-12a'da görülen $f(x_1,.,x_4) = x_1x_3 + \overline{x}_1x_2x_4 + x_2\overline{x}_3\overline{x}_4$ fonksiyonuna ilişkin BDD'de 0 ve 1 simgeleri ile gösterilen terminal düğümleri fonksiyonun 0 ve 1 değerlerini, terminal-olmayan diğer düğümler ise fonksiyonu gösterirler. Örneğin f fonksiyonunun $(x_1,x_2,x_3,x_4)=(1,0,1,0)$ girişi için değeri 1'dir. Bu sonuç BDD yardımıyla, x_4 düğümünün 0 ile gösterilen dalından x_3 düğümüne, x_3 düğümünün 1 ile gösterilen dalında x_1 düğümüne ve x_1 düğümünün 1 ile gösterilen dalından 1 ile gösterilen terminale ulaşılarak bulunabilir.

Herbir seviyesi Şekil 3-12b'de de görüldüğü gibi aynı değişkenden oluşan ikili karar diyagramlarına "Sıralı İkili Karar Diyagramları (Ordered Binary Decision Diagram) denir. Şekil 3-12c'de görüldüğü gibi aynı düğümlere giden aynı indisli düğümlerin (4.

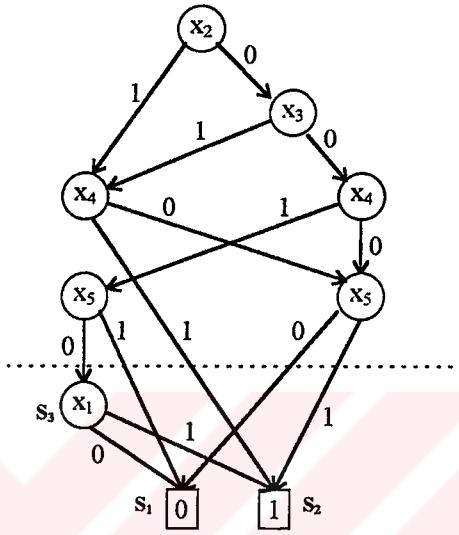
seviyedeki x_2 düğümleri) birleştirilmesiyle oluşan ikili karar diyagramlarına "İndirgenmiş Sıralı İkili Karar Diyagramları (Reduced Ordered Binary Decision Diagram, ROBDD)" adı verilir.



Şekil 3-12 a) $f(x_1,..,x_4) = x_1x_3 + \overline{x}_1x_2x_4 + x_2\overline{x}_3\overline{x}_4$ fonksiyonuna ilişkin BDD, b) Sıralı BDD (Ordered BDD) , c) ROBDD

Bir fonksiyonun Roth-Karp yöntemiyle m-LUT yapısına uygun olarak ayrıştırılması ikili karar diyagramları kullanılarak da yapılabilir. Bir (X|Y) ayrışımı için X bağımlı kümesindeki değişkenler Y bağımsız kümesindeki değişkenlerin üzerinde yer alacak şekilde oluşturulan indirgenmiş sıralı bir karar diyagramı (ROBDD) yardımıyla eşdeğerlik sınıfları belirlenebilir. Bu işlem aşağıda bir örnek ile açıklanmıştır.

$f(x_1,..,x_5) = x_2x_4 + x_3x_4 + \overline{x}_4x_5 + x_1x_4\overline{x}_5$ fonksiyonu önceki bölümde olduğu gibi 4-LUT'lar ile gerçekleştirilebilecek şekilde ayrıştırılmak istensin. $X=\{x_2,x_3,x_4,x_5\}$, $Y=\{x_1\}$ için f fonksiyonunun ROBDD gösterilimi Şekil 3-13'de görüldüğü gibi olur. Kesikli çizgi X kümesi ile Y kümesinin elemanlarını ayırmaktadır.



Şekil 3-13 $f = x_2x_4 + x_3x_4 + \overline{x}_4x_5 + x_1x_4\overline{x}_5$ Fonksiyonunun ROBDD Gösterilimi

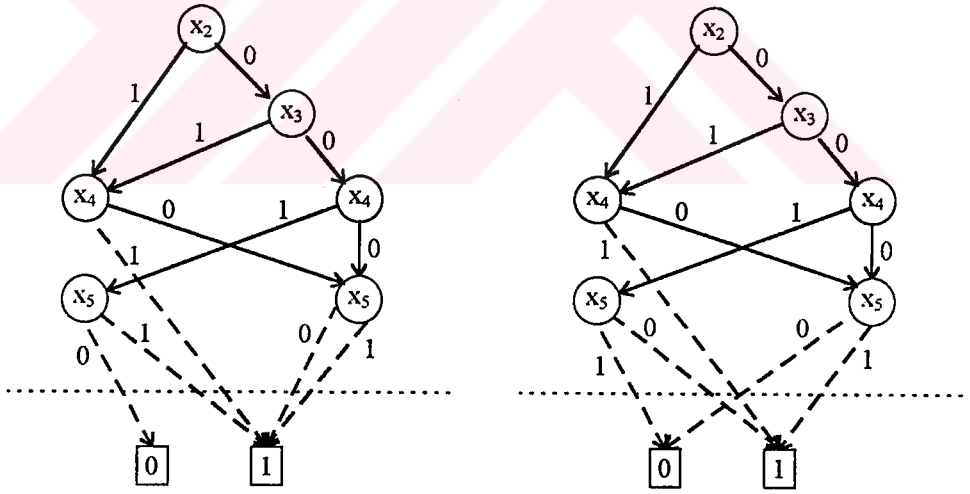
Diyagramdan X kümesinin elemanlarının kesikli çizginin altındaki üç noktaya (S_1, S_2, S_3) gittikleri görülmektedir. Bu noktalara ilişkin S_1, S_2, S_3 fonksiyonları f fonksiyonunun $(x_2x_3x_4x_5|x_1)$ ayrışımı için eşdeğerlik sınıflarını vermektedir.

$$
\begin{aligned}
&S_1 = \overline{x}_2\overline{x}_3\overline{x}_4\overline{x}_5 + \overline{x}_2\overline{x}_3x_4x_5 + x_2\overline{x}_4\overline{x}_5 + \overline{x}_2x_3\overline{x}_4\overline{x}_5 \\
&\Rightarrow S_1 = \overline{x}_4\overline{x}_5 + \overline{x}_2\overline{x}_3x_4x_5 \\
&S_2 = \overline{x}_2\overline{x}_3\overline{x}_4x_5 + x_2x_4 + x_2\overline{x}_4x_5 + \overline{x}_2x_3\overline{x}_4x_5 + \overline{x}_2x_3x_4 \\
&\Rightarrow S_2 = \overline{x}_4x_5 + x_3x_4 + x_2x_4 \\
&S_3 = \overline{x}_2\overline{x}_3x_4\overline{x}_5
\end{aligned}
\qquad (3.21)
$$

(3.14) ve (3.21) ifadeleri karşılaştırılırsa S_1'in C_1 ile, S_2'nin C_2 ile, S_3'ün C_3 ile aynı olduğu görülür. Bu sınıfların kodlaması Şekil 3-10'dakine benzer şekilde yapılırsa aşağıdaki kodlar elde edilir.

	α_1	α_2
S_1	1	0
S_2	1	1
S_3	0	1

α fonksiyonlarına ilişkin BDD'lerin oluşturulması için Şekil 3-12 de verilen ROBDD'nin üst kısmı her α fonksiyonu için tekrar oluşturulur (Şekil 3-14'deki düz çizgili kısımlar). Diyagramın tamamlanması için yukarıdaki kodlamadan ve f için çizilen ROBDD'den faydalanılır. Örneğin f için çizilen ROBDD'de S_3 noktasına giden dal, S_3'ün kodunda α_1=0 olduğundan α_1 için çizilen BDD'de 0 noktasına gitmelidir. Benzer şekilde f için çizilen ROBDD'de S_1 ve S_2 noktalarına giden dallar S_1 ve S_2'nin kodlarında α_1=1 olduğu için 1 noktasına gitmelidir. α_2 için de benzer şekilde bir BDD oluşturulabilir. Şekil 3-14 de α_1 ve α_2 için oluşturulmuş BDD'ler görülmektedir.



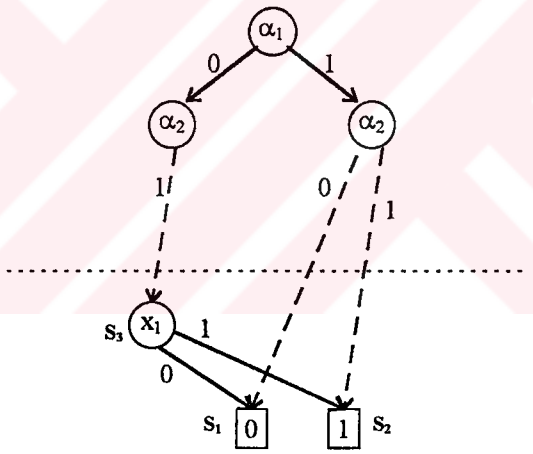
Şekil 3-14 α_1 ve α_2'nin BDD gösterilimi

Şekil 3-14'deki diyagramlardan α_1 ve α_2 nin çarpımlar toplamı biçimindeki ifadeleri,

$$\begin{aligned}
&\alpha_1 = \overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_2x_3x_4 + \overline{x}_2x_3\overline{x}_4 + x_2\overline{x}_4 + x_2x_4 + \overline{x}_2\overline{x}_3x_4x_5 \\
&\Rightarrow \alpha_1 = x_2 + x_3 + \overline{x}_4 + x_5 \\
&\alpha_2 = \overline{x}_2\overline{x}_3\overline{x}_4x_5 + x_2x_4 + x_2\overline{x}_4x_5 + \overline{x}_2x_3\overline{x}_4x_5 + \overline{x}_2x_3x_4 + \overline{x}_2\overline{x}_3x_4\overline{x}_5 \\
&\Rightarrow \alpha_2 = \overline{x}_4x_5 + x_3x_4 + x_2x_4 + x_4\overline{x}_5
\end{aligned} \tag{3.22}$$

olarak bulunur. (3.16) ifadesi ile (3.22) ifadesi karşılaştırılırsa aynı $\tilde{\alpha}$ ifadelerinin bulunduğu görülür.

g fonksiyonuna ilişkin BDD'nin alt parçası f için çizilen ROBDD'deki kesikli çizginin altında kalan kısımdan oluşur. Üst kısım Şekil 3-15'de görüldüğü gibi α_1 ve α_2 nin olası bütün kombinezonlarını gösteren bir ağaçtan oluşur. Üst kısımın alt kısımla olan bağlantılarının belirlenmesi için eşdeğerlik sınıflarının kodlarından faydalanılır. Örneğin $\alpha_1\alpha_2$=10 kombinezonu S_1 sınıfına karşı düşmektedir. Bu sebeple 10 kombinezonuna ilişkin yol S_1 noktasına bağlanmalıdır. Benzer şekilde, $\alpha_1\alpha_2$= 11 kombinezonu S_2 sınıfına karşı düştüğünden bu kombinezona ilişkin yol S_2 noktasına, $\alpha_1\alpha_2$= 01 kombinezonu S_3 sınıfına karşı düştüğünden bu kombinezona ilişkin yol S_3 noktasına bağlanmalıdır. Bu bağlantılar Şekil 3-15'de kesikli çizgiler ile gösterilmiştir.



Şekil 3-15 g Fonksiyonuna ilişkin BDD gösterilimi

Şekil 3-15'de görülen diyagramdan g fonksiyonuna ilişkin çarpımlar toplamı biçimindeki ifade,

$$g = \overline{\alpha}_1\alpha_2 x_1 + \alpha_1\alpha_2$$
$$g = \alpha_1\alpha_2 + \alpha_2 x_1 \qquad (3.23)$$

olarak bulunur. (3.20) ifadesi ile (3.23) ifadesi karşılaştırılırsa aynı g fonksiyonunun bulunduğu görülür.

3.1.1.1.4 Fonksiyonel Ayrıştırmanın Karnaugh Diyagramından Faydalanılarak Yapılması

Bu tezde, n+k değişkenli fonksiyonların n değişkenli Karnaugh diyagramı ile indirgenebilmesinden faydalanılarak, fonksiyonel ayrıştırmanın Karnaugh diyagramı ile yapılmasına ilişkin bir yol önerilmiştir [8]. Bu amaçla, Karnaugh diyagramının değişkenleri bağımlı kümenin elemanları seçilir. Diyagram verilen fonksiyonun geri kalan değişkenleri cinsinden doldurulur ve aynı ifadelere sahip gözler kendi aralarında gruplandırılır. Diyagramdaki her bir grup bir eşdeğerlik sınıfına karşı düşer. Eşdeğerlik sınıflarının ifadelerinin bulunması için her grup kendi içinde, gözlerdeki değişkenler dikkate alınmadan sadece Karnaugh diyagramının değişkenleri cinsinden, Karnaugh diyagramı ile indirgeme kurallarına uyularak indirgenir. Daha sonra fonksiyonun eşdeğerlik sınıfları cinsinden ifadesi diyagramdan bulunur. Eşdeğerlik sınıflarının kodlanmasıyla bulunan α_i fonksiyonları bu ifadede yerine konulur. Aşağıda bu işlemler örnekler üzerinde gösterilmiştir.

Örnek 3.1: $f(x_1,..,x_5) = x_2x_4 + x_3x_4 + \overline{x}_4x_5 + x_1x_4\overline{x}_5$ fonksiyonu m=4 için $\{x_1x_2x_3x_4|x_5\}$ biçiminde ayrıştırılırsa Şekil 3-16'da görülen ayrıştırma tablosu ve eşdeğerlik sınıfları oluşur.

$\frac{x_1x_2x_3x_4}{x_5}$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1

(11) $C_1 = \{ 0011,0101,0111,1011,1101,1111 \} \Rightarrow C_1 = x_3x_4 + x_2x_4$

(10) $C_2 = \{ 1001 \} \Rightarrow C_2 = x_1\overline{x}_2\overline{x}_3x_4$

(00) $C_3 = \{ 0001 \} \Rightarrow C_3 = \overline{x}_1\overline{x}_2\overline{x}_3x_4$

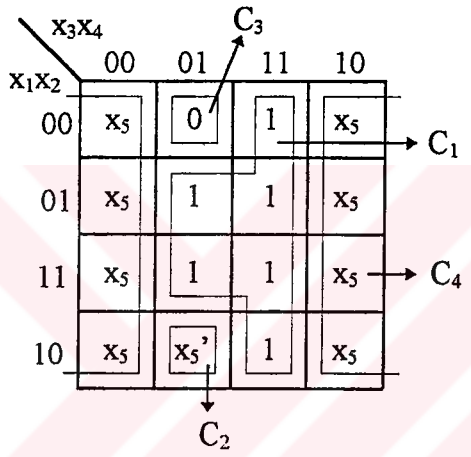
(01) $C_4 = \{ 0000,0010,0100,0110,1000,1010,1100,1110 \} \Rightarrow C_4 = \overline{x}_4$

Şekil 3-16 $\{x_1x_2x_3x_4|x_5\}$ Ayrışımına ilişkin Ayrıştırma Tablosu ve Eşdeğerlik Sınıfları

f fonksiyonu eşdeğerlik sınıfları cinsinden,

$$f = (C_1 + C_2)\overline{x}_5 + (C_1 + C_4)x_5 \Rightarrow f = C_1 + C_2\overline{x}_5 + C_4x_5 \qquad (3\text{-}24)$$

olarak bulunur.Aynı fonksiyonun $\{x_1x_2x_3x_4|x_5\}$ ayrışımına göre Karnaugh diyagramına taşınmasıyla Şekil 3-17'de görüldüğü gibi $\{0\}$, $\{1\}$, $\{x_5\}$ ve $\{\overline{x}_5\}$ olmak üzere dört farklı grup (eşdeğerlik sınıfı) oluşur. Bu grupların kendi içlerinde indirgenmesiyle Şekil 3-16'dakiyle aynı eşdeğerlik sınıflarının bulunduğu görülür.



$C_1 = x_3x_4 + x_2x_4$

$C_2 = x_1\overline{x}_2\overline{x}_3x_4$

$C_3 = \overline{x}_1\overline{x}_2\overline{x}_3x_4$

$C_4 = \overline{x}_4$

Şekil 3-17 $\{x_1x_2x_3x_4|x_5\}$ Ayrışımına ilişkin Karnaugh Diyagramı ve Eşdeğerlik Sınıfları

Şekil 3-17'deki diyagramdan f fonksiyonu eşdeğerlik sınıfları cinsinden,

$$f = C_1 + C_2\overline{x}_5 + C_4x_5 \qquad (3\text{-}25)$$

olarak bulunur. (3-24) ifadesi ile (3-25) ifadesinin aynı olduğu görülmektedir.

Örnek 3.2: Örnek 3.1'de verilen fonksiyon $\{x_2x_3x_4x_5|x_1\}$ biçiminde ayrıştırılırsa Şekil 3-18'de görülen ayrıştırma tablosu ve eşdeğerlik sınıfları oluşur.

$x_2x_3x_4x_5$ / x_1	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0	1	0	0	0	1	1	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	0	1	1	1	0	1	1	1

$$(00)\ C_1 = \{0000, 0011, 0100, 1000, 1100\} \Rightarrow C_1 = \overline{x}_4 x_5 + \overline{x}_2 \overline{x}_3 \overline{x}_4 x_5$$

$$(11)\ C_2 = \{0001, 0101, 0110, 0111, 1001, 1010, 1011, 1101, 1110, 1111\}$$

$$\Rightarrow C_2 = \overline{x}_4 x_5 + x_3 x_4 + x_2 x_4$$

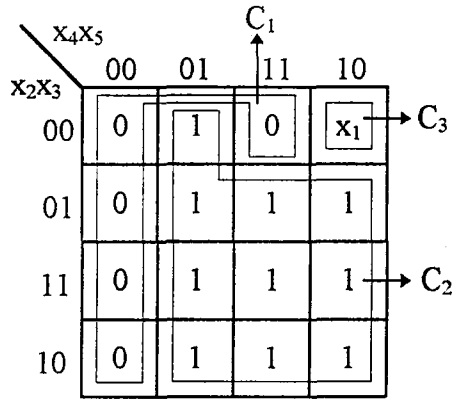
$$(01)\ C_3 = \{0010\} \Rightarrow C_3 = \overline{x}_2 \overline{x}_3 x_4 \overline{x}_5$$

Şekil 3-18 $\{x_2x_3x_4x_5|x_1\}$ Ayrışımına ilişkin Ayrıştırma Tablosu ve Eşdeğerlik Sınıfları

f fonksiyonu eşdeğerlik sınıfları cinsinden,

$$f = C_2 \overline{x}_1 + (C_2 + C_3) x_1 \Rightarrow f = C_2 + C_3 x_1 \qquad (3\text{-}26)$$

olarak bulunur.Aynı fonksiyonun $\{x_2x_3x_4x_5|x_1\}$ ayrışımına göre Karnaugh diyagramına taşınmasıyla Şekil 3-19'da görüldüğü gibi $\{0\}$, $\{1\}$ ve $\{x_1\}$ olmak üzere üç farklı grup (eşdeğerlik sınıfı) oluşur. Bu grupların kendi içlerinde indirgenmesiyle Şekil 3-18'dekiyle aynı eşdeğerlik sınıflarının bulunduğu görülür.



$$C_1 = \overline{x}_4 x_5 + \overline{x}_2 \overline{x}_3 \overline{x}_4 x_5$$

$$C_2 = \overline{x}_4 x_5 + x_3 x_4 + x_2 x_4$$

$$C_3 = \overline{x}_2 \overline{x}_3 x_4 \overline{x}_5$$

Şekil 3-19 $\{x_2x_3x_4x_5|x_1\}$ Ayrışımına ilişkin Karnaugh Diyagramı ve Eşdeğerlik Sınıfları

Şekil 3-15'deki diyagramdan f fonksiyonu eşdeğerlik sınıfları cinsinden,

$$f = C_2 + C_3 x_1 \tag{3-27}$$

olarak bulunur. (3-26) ifadesi ile (3-27) ifadesinin aynı olduğu görülmektedir.

Ayrıca Şekil 3-19'da görülen eşdeğerlik sınıflarının, aynı fonksiyonun ikili karar diyagramlarından faydalanılarak ayrıştırılması sonucunda bulunan (3-21) ifadesindeki eşdeğerlik sınıflarıyla da aynı oldukları görülmektedir. Fonksiyonların Karnaugh diyagramına taşınmasının fonksiyonun ikili karar diyagramının çizilmesinden daha kolay olması ve fonksiyonun eşdeğerlik sınıfları cinsinden ifadesinin bulunması için ayrı bir Karnaugh diyagramının çizilmesine gerek duyulmaması sebebiyle ayrıştırma işleminde Karnaugh diyagramın kullanılması, ikili karar diyagramlarının kullanılmasına göre daha üstündür.

Örnek 3.3: Örnek 3.1'de verilen fonksiyon $\{x_1x_2x_3|x_4x_5\}$ biçiminde ayrıştırılırsa Şekil 3-20'de görülen ayrıştırma tablosu ve eşdeğerlik sınıfları oluşur.

$x_1x_2x_3$ / x_4x_5	000	001	010	011	100	101	110	111
00	0	0	0	0	0	0	0	0
01	1	1	1	1	1	1	1	1
10	0	1	1	1	1	1	1	1
11	0	1	1	1	0	1	1	1

(0100) $C_1 = \{000\} \Rightarrow C_1 = \overline{x}_1\overline{x}_2\overline{x}_3$

(0111) $C_2 = \{001,010,011,101,110,111\} \Rightarrow C_2 = x_2 + x_3$

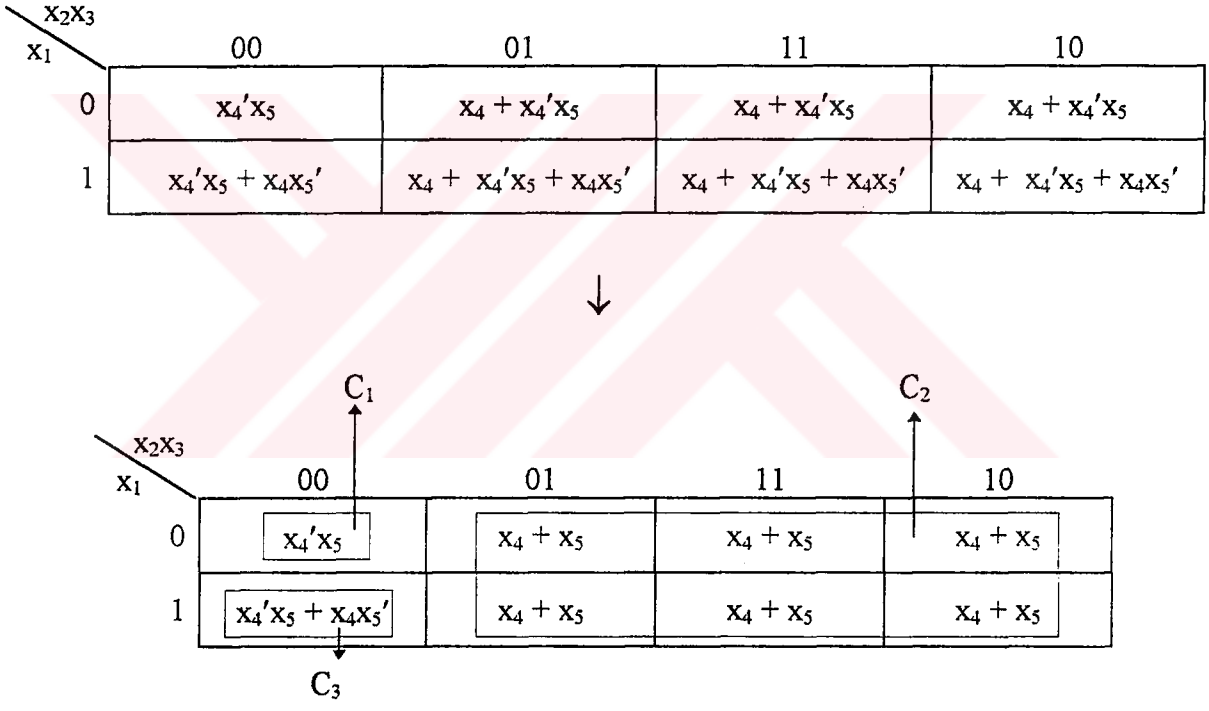
(0110) $C_3 = \{100\} \Rightarrow C_2 = x_1\overline{x}_2\overline{x}_3$

Şekil 3-20 $\{x_1x_2x_3|x_4x_5\}$ Ayrışımına ilişkin Ayrıştırma Tablosu ve Eşdeğerlik Sınıfları

f fonksiyonun eşdeğerlik sınıfları cinsinden,

$$f = (C_1 + C_2 + C_3)\overline{x}_4 x_5 + C_2 x_4 x_5 + (C_2 + C_3)x_4 \overline{x}_5$$
$$\Rightarrow f = \overline{x}_4 x_5 C_1 + (x_4 + x_5)C_2 + (\overline{x}_4 x_5 + x_4 \overline{x}_5)C_3 \qquad (3\text{-}28)$$

olarak bulunur.Aynı fonksiyonun $\{x_1x_2x_3|x_4x_5\}$ ayrışımına göre Karnaugh diyagramına taşınmasıyla Şekil 3-21'de görüldüğü gibi $\{\overline{x}_4x_5\}$, $\{\overline{x}_4x_5 + x_4\overline{x}_5\}$ ve $\{x_4 + x_5\}$ olmak üzere üç farklı grup (eşdeğerlik sınıfı) oluşur. Bu grupların kendi içlerinde indirgenmesiyle Şekil 3-20'dekiyle aynı eşdeğerlik sınıflarının bulunduğu görülür.



$$C_1 = \overline{x}_1\overline{x}_2\overline{x}_3$$
$$C_2 = x_2 + x_3$$
$$C_2 = x_1\overline{x}_2\overline{x}_3$$

Şekil 3-21 $\{x_1x_2x_3|x_4x_5\}$ Ayrışımına ilişkin Karnaugh Diyagramı ve Eşdeğerlik Sınıfları

Şekil 3-21'deki diyagramdan f fonksiyonu eşdeğerlik sınıfları cinsinden,

$$f = \bar{x}_4 x_5 C_1 + (x_4 + x_5) C_2 + (\bar{x}_4 x_5 + x_4 \bar{x}_5) C_3 \qquad (3\text{-}29)$$

olarak bulunur. (3-28) ifadesi ile (3-29) ifadesinin aynı olduğu görülmektedir.

3.1.1.2 Kutulama ("Cube-packing") Yöntemi İle Ayrıştırma

Kutulama ("Cube-packing") LUT-tabanlı FPGA' lar için 1991 yılında önerilmiş olan bir ayrıştırma yöntemidir. Bu yöntemin temeli "Bin-packing" adı verilen bir yönteme dayanmaktadır. "Bin-packing" yönteminde amaç, ağırlıkları verilen parçaları sabit kapasiteli kutulara, kutuların kapasitesini aşmadan ve minimum sayıda kutu kullanarak yerleştirmektir. Çarpımlar toplamı biçimindeki bir fonksiyondaki her çarpım terimi ağırlığı terimdeki değişken sayısı kadar olan parça, bir LUT'da kapasitesi m olan bir kutu olarak düşünülürse verilen fonksiyonun minimum sayıda m-LUT ile gerçekleştiril-mesi, ileride verilen sebepler nedeniyle tam olarak olmasa da bir "bin-packing" problemi olarak ele alınabilir.

Aşağıda "bin-packing" yöntemi için geliştirilmiş olan sezgisel bir algoritmanın değiştirilmesiyle oluşturulmuş bir kutulama algoritmasının adımları verilmiştir.

Adım 1 : Değişken sayısı m'den fazla olan çarpan terimlerinden, değişken sayıları m oluncaya kadar m değişkenli parçalar çıkarılır. Yani m-gerçeklenemez çarpım terimleri m-gerçeklenebilir hale getirilir. Bu işlem için iki metod kullanılabilir:

a) Çarpım terimindeki değişkenlerden rastgele m tanesi çıkarılır.

b) Çarpım terimindeki değişkenlerden, verilen fonksiyonda en az görünen m tanesi çıkarılır. Böylece her çarpım teriminin m-gerçeklenebilir hale geldiği son ifadede herhangi bir değişkenin mümkün olduğunca çok çarpım teriminde görünmesi sağlanır. Bunun sonucunda bir kutuya daha çok çarpım terimi yerleştirilebilir.

Bu parçaların her biri bir kutuya konularak yeni bir değişkenle isimlendirilir ve ait oldukları çarpım terimleri yeniden düzenlenir.

Adım 2 : Çarpım terimleri ağırlıklarına (içerdikleri çarpan sayısına) göre büyükten küçüğe sıralanır.

Adım 3 : En büyük ağırlıklı çarpım terimi daha önce bir kısmı kullanılmış olan bir kutuya yerleştirilmeye çalışılır. Eğer bu terimin yerleştirilebileceği bir kutu yoksa kapasitesi m olan yeni bir kutu açılır (1'inci adım her çarpım teriminin maksimum m değişkenli olmasını sağladığı için her terim mutlaka bir kutuya sığar). Bir kutuya yerleştirilen terim listeden silinir. Bu şekilde bütün çarpım terimleri kutulara yerleştirilir.

Adım 4 : Bütün çarpım terimleri kutulara yerleştirildikten sonra kapasitesi en fazla kullanılmış kutu kapatılır ve yeni bir değişken olarak tamamıyla dolmamış bir kutuya eklenir. Bu adım bütün kutular kapatılıncaya kadar tekrarlanır. Son kutunun kapatılmasıyla oluşan değişken f fonksiyonudur.

Yukarıda verilen adımlar sonucunda oluşan her kutuya bir LUT karşı düşürülür. Aşağıda bu algoritmanın çalışması bir örnek üzerinde gösterilmiştir.

$f(x_1,..,x_{12}) = x_1x_2x_3x_4x_5x_6x_7x_8 + \overline{x}_9\overline{x}_{10}x_{11} + \overline{x}_6\overline{x}_7\overline{x}_8 + x_{10}x_{12}$ fonksiyonunun 5-LUT' lar kullanılarak gerçeklenmesi istensin. Verilen fonksiyonun çarpım terimleri

$T_1 = x_1x_2x_3x_4x_5x_6x_7x_8$,
$T_2 = \overline{x}_9\overline{x}_{10}\overline{x}_{11}$,
$T_3 = \overline{x}_6\overline{x}_7\overline{x}_8$,
$T_4 = x_{10}x_{12}$ dir.

• T_1 terimi 5-gerçeklenemez olduğundan algoritmanın 1'inci adımı gereğince 5-gerçeklenebilir hale getirilmelidir. Bunun için $a=x_1x_2x_3x_4x_5$ seçilerek bir kutuya (1. kutu) yerleştirilir ve $T_1=ax_6x_7x_8$ şeklinde düzenlenir. Böylece fonksiyonun bütün çarpım terimleri 5-gerçeklenebilir hale gelir.

•2'nci adım gereğince terimler ağırlıklarına göre büyükten küçüğe sıralanır.
($w(T_i)$: T_i'inci çarpım teriminin ağırlığı)

$w(T_1)=4$
$w(T_2)=3$
$w(T_3)=3$
$w(T_4)=2$

• 3'üncü adım gereğince en büyük ağırlıklı T_1 terimi daha önceden bir kısmı kullanılmış olan bir kutuya yerleştirilmeye çalışılır. Böyle bir kutu olmadığından yeni bir kutu (2'nci kutu) açılarak T_1 yerleştirilir ve listeden silinir. T_2 terimi, toplam ağırlığın (toplam değişken sayısının) kutu kapasitesi olan 5'i geçmesi nedeniyle 2'nci kutuya yerleştirilemez. Bu nedenle tekrar yeni bir kutu (3'üncü kutu) açılarak T_2 bu kutuya yerleştirilir ve listeden silinir. T_3 terimi toplam ağırlığın 4 olması sebebiyle 2'nci kutuya yerleştirilebilir. Benzer sebeple T_4 terimi de 3'üncü kutuya yerleştirilebilir. Böylece bütün terimler kutulara yerleştirilmiş olur.

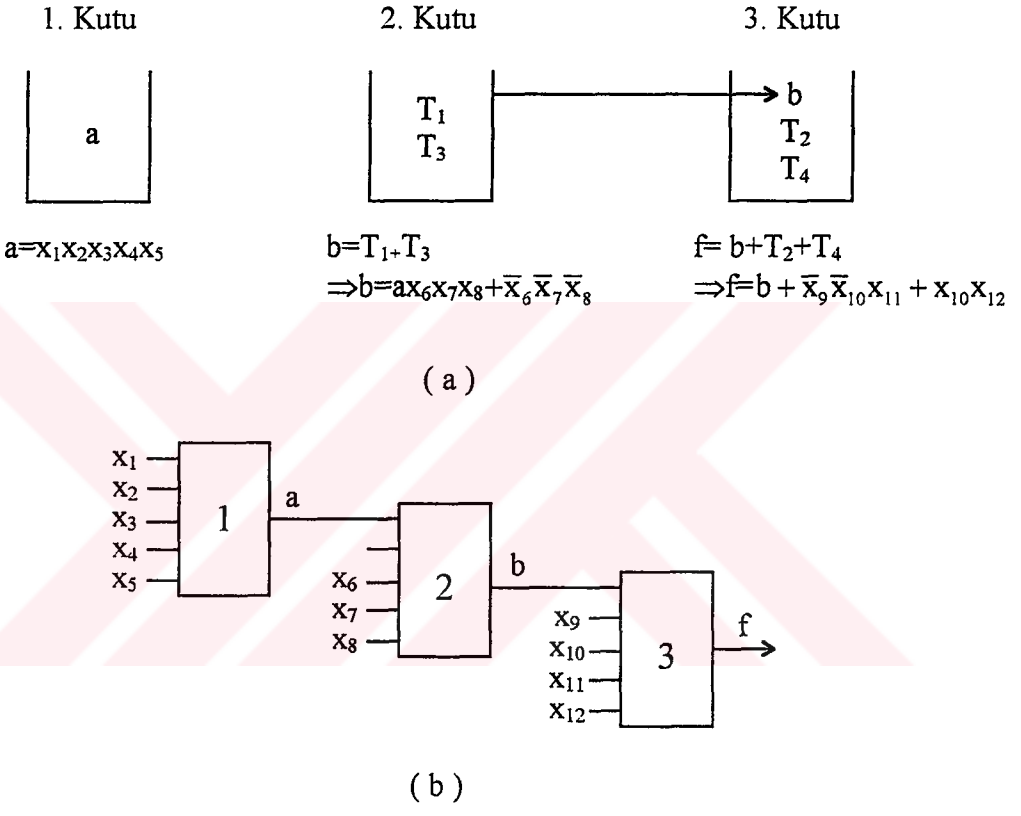
• 4'üncü adımın gereği olarak ilk önce 2'nci kutu kapatılır ve yeni bir değişkenle temsil edilerek 3'üncü kutuya eklenir. 3'üncü kutunun kapatılmasıyla f fonksiyonu elde edilir.

f fonksiyonu her kutuya bir 5-LUT karşı düşürülerek 3 tane 5-LUT ile gerçeklenmiş olur. Şekil 3-22'de yukarıda anlatılan işlemlerin sonucu görülmektedir.

Yapılan işlemlerde iki önemli özellik göze çarpmaktadır:

1) Çarpım terimleri kutu kapasitelerini yani LUT'ların girişlerini paylaşabilirler. Dolayısıyla iki terimin ağırlıkları toplamı bunların birleştirilmesiyle (VEYA işlemi uygulanmasıyla) oluşan terimin ağırlığından fazla olabilir. Örneğin $w(T_3)=3$, $w(T_4)=2$ olmasına karşın $w(T_3+T_4)=4 \neq w(T_3)+w(T_4)$ dür. "Bin-packing" problemlerinde ise $w(T_x+T_y) = w(T_x)+w(T_y)$ dir.

2) Sonunda f fonksiyonunu gerçekleyen kutu haricindeki her kutu, f'nin bir alt fonksiyonunu gerçeklemesi sebebiyle ağırlığı 1 olan yeni bir terim oluşturmaktadır. Bu nedenle bir kutu kapatıldığında ağırlığı 1 olan bir terim diğer bir kutuya eklenmelidir. Örneğin yukarıda incelenen örnekte 2'nci kutunun kapatılmasıyla oluşan 1 ağırlıklı b terimi 3'üncü kutuya eklenmiştir.



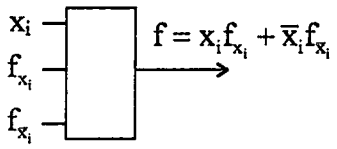
Şekil 3-22 (a) $f(x_1,..,x_{12}) = x_1x_2x_3x_4x_5x_6x_7x_8 + x_9\bar{x}_{10}x_{11} + \bar{x}_6\bar{x}_7\bar{x}_8 + x_{10}x_{12}$ Fonksiyonunun Kutulama yöntemi ile ayrıştırılması, (b) 5-LUT'lar ile gerçeklenmesi

3.1.1.3 Kofaktör Ayrıştıma Yöntemi

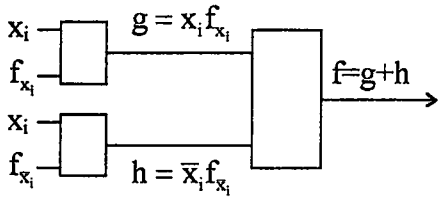
Bir fonksiyon kofaktör yöntemi ile ayrıştırılıp,

$$f(x_1,..,x_n) = x_i f_{x_i} + \bar{x}_i f_{\bar{x}_i} \quad 1 \le i \le n \qquad (3\text{-}30)$$

şekline getirilerek m-LUT'lar ile gerçeklenebilir. n-değişkenli bir fonksiyonun kofaktör yöntemi ile ayrıştırılması sonucunda elde edilen f_{x_i} ve $f_{\bar{x}_i}$ fonksiyonları en fazla n-1 değişkenli fonksiyonlardır. Bu fonksiyonlar, m-gerçeklenebilir fonksiyonlar ise ayrıştırma işlemi tamamlanmış olur. Eğer m-gerçeklenemez fonksiyonlar iseler ayrıştırma işlemine devam edilir. (3-24) ifadesi m>2 durumu için bir tane LUT, m=2 durumu için 3 tane LUT ile gerçeklenebilir. Şekil 3-23a ve Şekil 3-23b'de bu durumlar görülmektedir.



Şekil 3-23a m>2

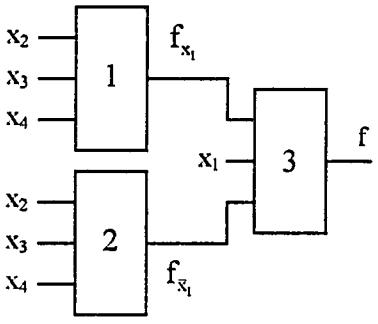


Şekil 3-23b m=2

Örneğin $f(x_1,..,x_4) = x_1x_2\bar{x}_3 + x_1\bar{x}_2x_4 + \bar{x}_1x_3x_4 + x_2x_3\bar{x}_4$ fonksiyonu Shannon kofaktör yöntemi ile 3-LUT'lar ile gerçeklenecek şekilde ayrıştırılırsa Şekil 3-24'de görüldüğü gibi,

$$
\begin{aligned}
f_{x_1} &= x_2\bar{x}_3 + \bar{x}_2x_4 + x_2x_3\bar{x}_4 \\
f_{\bar{x}_1} &= x_3x_4 + x_2x_3\bar{x}_4 \\
f &= x_1f_{x_1} + \bar{x}_1f_{\bar{x}_1}
\end{aligned}
\qquad (3\text{-}31)
$$

olmak üzere, toplam 3 tane 3-LUT kullanılarak gerçeklenebilecektir.



Şekil 3-24 Shannon Kofaktör ayrışımı

Kofaktör ayrıştırma yöntemi Roth-Karp ayrıştırma yönteminin özel bir şeklidir. n-değişkenli bir $f(x_1,....,x_n)$ fonksiyonunun x_i değişkenine göre Kofaktör yöntemiyle ayrıştırılması, bu fonksiyonun Roth-Karp yöntemiyle $\{x_1,...,x_{i-1},x_{i+1},..,x_n\}|\{x_i\})$ ayrışımına göre ayrıştırılmasıyla aynı anlama gelmektedir. Bağımsız kümenin eleman sayısı 1 olduğundan oluşturulacak ayrıştırma tablosundaki farklı sütun vektörü sayısı (eşdeğerlik sınıflarının sayısı) Şekil 3-25'de de görüldüğü gibi en fazla 4 olacaktır. Bu 4 eşdeğerlik sınıfı α_1 ve α_2 gibi iki değişkenle Şekil 3-26'da görüldüğü gibi kodlanabilir. Ayrıştırma işlemi tamamlanırsa $f_{\bar{x}_i} = \alpha_1$, $f_{x_i} = \alpha_2$ olarak bulunur.

$\frac{x_1..x_{i-1}x_{i+1}..x_n}{x_i}$	C_0	C_1	C_2	C_3
0	0	0	1	1
1	0	1	0	1

Şekil 3-25 $(\{x_1,...,x_{i-1},x_{i+1},..,x_n\}|\{x_i\})$ ayrışımında olabilecek eşdeğerlik sınıfları

Eşdeğerlik sınıfları	α_1	α_2
C_0	0	0
C_1	0	1
C_2	1	0
C_3	1	1

Şekil 3-26 $(\{x_1,...,x_{i-1},x_{i+1},..,x_n\}|\{x_i\})$ ayrışımında olabilecek eşdeğerlik sınıflarının kodlanması

3.1.1.4 Ayrıştırma Yöntemlerinin Karşılaştırılması

Yukarıda incelenen ayrıştırma yöntemlerinin hiçbiri, herhangi bir fonksiyon için optimal çözüme ulaşılacağını garanti edememektedir. Ancak belirli tiplerdeki fonksiyonlar için belirli ayrıştırma yöntemlerinin daha iyi sonuçlar vereceği söylenebilmektedir. Örneğin simetrik fonksiyonlar için, iyi bir ayrışımın (eşdeğer sınıfların sayısının minimum olmasını sağlıyacak bir ayrışım) bulunmasının kolay olması sebebiyle fonksiyonel ayrıştırma daha iyi sonuçlar vermektedir. m+1 değişkenli fonksiyonların m-LUT'lar ile gerçeklenmesinde kofaktör ayrıştırma yöntemi ve fonksiyonel ayrıştırma optimal veya optimale yakın çözüme ulaşabilmektedirler.

Verilen bir fonksiyonu gerçekleştirmek için gerekli LUT sayısı, yöntemden yönteme çok değişebilmektedir. Aşağıda verilmiş olan örnekler ayrıştırma yöntemlerinin özelliklerini mümkün olduğunca açığa çıkaracak şekilde geliştirilmiştir.

Örnek 3.4: $f(x_1,..,x_4)=\overline{x}_2\overline{x}_3\overline{x}_4+\overline{x}_1x_2\overline{x}_3+x_2\overline{x}_3x_4+\overline{x}_2x_3x_4+x_2x_3\overline{x}_4$ fonksiyonunun m=3 için $\{x_1x_2x_3|x_4\}$ ayrışımı sonucunda elde edilen ayrıştırma tablosu ve eşdeğerlik sınıfları Şekil 3-27'de verilmiştir.

$x_1x_2x_3$ / x_4	000	001	010	011	100	101	110	111
0	1	0	1	1	1	0	0	1
1	0	1	1	0	0	1	1	0

(01) $C_1=\{001,101,110\} \Rightarrow C_1=\overline{x}_2x_3+x_1x_2\overline{x}_3$

(10) $C_2=\{000,011,100,111\} \Rightarrow C_2=\overline{x}_2\overline{x}_3+x_2x_3$

(11) $C_3=\{010\} \Rightarrow C_3=\overline{x}_1x_2\overline{x}_3$

Şekil 3-27 $\{x_1x_2x_3|x_4\}$ Ayrışımına ilişkin ayrıştırma tablosu ve eşdeğerlik sınıfları

f fonksiyonu bu eşdeğerlik sınıfları cinsinden Şekil 3-28'de görüldüğü gibi ifade edilebilir.

$x_1x_2x_3$ / x_4	C_1	C_2	C_3
0	0	1	1
1	1	0	1

Şekil 3-28 f Fonksiyonunun Eşdeğerlik Sınıfları cinsinden ifadesi

Şekil 3-28'deki tablodan f fonksiyonu,

$$f = \overline{x}_4(C_2 + C_3) + x_4(C_1 + C_3) \tag{3-32}$$

olarak bulunur. Eşdeğerlik sınıflarının Şekil 3-29'de görüldüğü gibi kodlanması sonucunda α_i (i=1,2) fonksiyonları,

$$\begin{aligned} \alpha_1 = C_2 + C_3 &\Rightarrow \alpha_1 = \overline{x}_2\overline{x}_3 + x_2x_3 + \overline{x}_1x_2 \\ \alpha_2 = C_1 + C_3 &\Rightarrow \alpha_2 = \overline{x}_2x_3 + x_2\overline{x}_3 \end{aligned} \tag{3-33}$$

olarak bulunur.

	α_1	α_2
C_1	0	1
C_2	1	0
C_3	1	1

Şekil 3-29 Eşdeğerlik Sınıflarının kodlanması

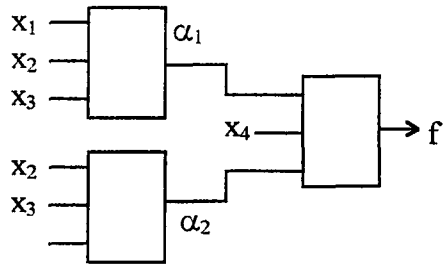
Şekil 3-29'deki kodlama tablosundan,

$$\begin{aligned} C_1 &= \overline{\alpha}_1\alpha_2 \\ C_2 &= \alpha_1\overline{\alpha}_2 \\ C_3 &= \alpha_1\alpha_2 \end{aligned} \tag{3-34}$$

bulunur. (3-34) ifadesinin (3-32) ifadesinde yerine konulmasıyla f fonksiyonu,

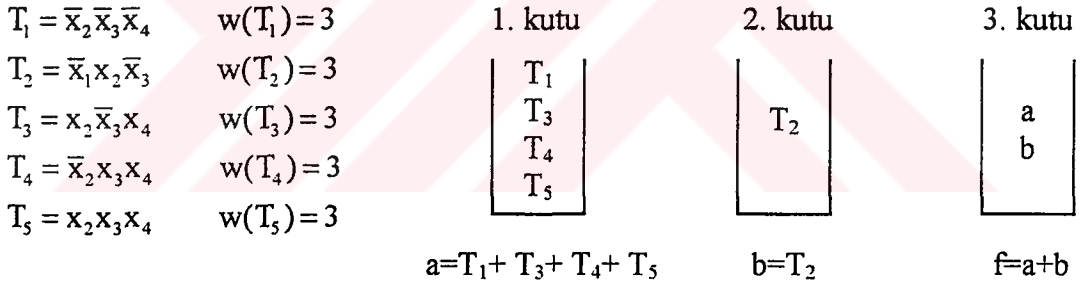
$$f = \alpha_1\overline{x}_4 + \alpha_2x_4 \tag{3-35}$$

olarak bulunur. Bu ifade Şekil 3-30'da görüldüğü gibi 3 tane 3-LUT ile gerçeklenebilir.

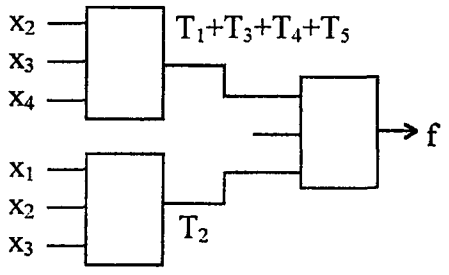


Şekil 3-30 $f(x_1,..,x_4)=\overline{x}_2\overline{x}_3\overline{x}_4+\overline{x}_1x_2\overline{x}_3+x_2\overline{x}_3x_4+\overline{x}_2x_3x_4+x_2x_3\overline{x}_4$ Fonksiyonunun Fonksiyonel Ayrıştırma ile 3-LUT'lar kullanılarak gerçeklenmesi

Aynı $f(x_1,..,x_4)$ fonksiyonu kutulama yöntemiyle ayrıştırılırsa Şekil 3-31a'da görüldüğü gibi 3 tane kutu gerekecektir. Bunun sonucunda fonksiyon Şekil 3-31b'de görülen şekilde 3 tane 3-LUT ile gerçeklenebilecektir.



(a)



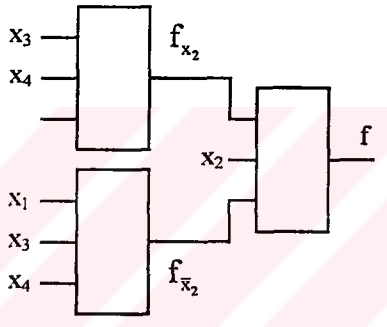
(b)

Şekil 3-31 a) $f(x_1,..,x_4)=\overline{x}_2\overline{x}_3\overline{x}_4+\overline{x}_1x_2\overline{x}_3+x_2\overline{x}_3x_4+\overline{x}_2x_3x_4+x_2x_3\overline{x}_4$ Fonksiyonunun Kutulama Yöntemi ile ayrıştırılması ve b) 3-LUT'lar kullanılarak gerçeklenmesi

Aynı fonksiyon, kofaktör ayrıştırma yöntemiyle,

$$f_{x_2} = \overline{x}_3 x_4 + \overline{x}_1 x_3 + x_3 \overline{x}_4$$
$$f_{\overline{x}_2} = \overline{x}_3 \overline{x}_4 + x_3 x_4 \qquad (3\text{-}36)$$
$$f = x_2 f_{x_2} + \overline{x}_2 f_{\overline{x}_2}$$

olacak şekilde, toplam 3 tane 3-LUT kullanılarak gerçeklenebilecektir. Bu yapı Şekil 3-32'de görülmektedir.



Şekil 3-32 $f(x_1,..,x_4) = \overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_1 x_2 \overline{x}_3 + x_2\overline{x}_3 x_4 + \overline{x}_2 x_3 x_4 + x_2 x_3 \overline{x}_4$ Fonksiyonunun Kofaktör Yöntemi ile ayrıştırılarak 3-LUT'lar ile gerçeklenmesi

Bu örnekte verilen fonksiyona uygulanan ayrıştırma yöntemlerinin hepsinin optimal çözümü verdikleri gösterilebilir.

Örnek 3.5: $f(x_1,..,x_4) = \overline{x}_1\overline{x}_2\overline{x}_3 x_4 + \overline{x}_1\overline{x}_2 x_3\overline{x}_4 + \overline{x}_1 x_2\overline{x}_3\overline{x}_4 + \overline{x}_1 x_2 x_3 x_4 + x_1 x_2\overline{x}_3 x_4$ $+x_1 x_2 x_3\overline{x}_4 + x_1\overline{x}_2\overline{x}_3\overline{x}_4 + x_1\overline{x}_2 x_3 x_4$ fonksiyonu 1. Kanonik açınım biçimindedir ve aynı zamanda daha fazla indirgenememektedir. Fonksiyonunun m=3 için $\{x_1x_2x_3|x_4\}$ ayrışımı sonucunda elde edilen ayrıştırma tablosu ve eşdeğerlik sınıfları Şekil 3-33'de verilmiştir.

$x_1x_2x_3$ / x_4	000	001	010	011	100	101	110	111
0	0	1	1	0	1	0	0	1
1	1	0	0	1	0	1	1	0

(01) $C_1=\{000,011,101,110\} \Rightarrow C_1=\overline{x}_1\overline{x}_2\overline{x}_3+\overline{x}_1x_2x_3+x_1\overline{x}_2x_3+x_1x_2\overline{x}_3$

(10) $C_2=\{001,010,100,111\} \Rightarrow C_2=\overline{x}_1\overline{x}_2x_3+\overline{x}_1x_2\overline{x}_3+x_1\overline{x}_2\overline{x}_3+x_1x_2x_3$

Şekil 3-33 $\{x_1x_2x_3|x_4\}$ Ayrışımına ilişkin Ayrıştırma Tablosu ve Eşdeğerlik Sınıfları

f fonksiyonu bu eşdeğerlik sınıfları cinsinden Şekil 3-34'de görüldüğü gibi ifade edilebilir.

$x_1x_2x_3$ / x_4	C_1	C_2
0	0	1
1	1	0

Şekil 3-34 f Fonksiyonunun Eşdeğerlik Sınıfları cinsinden ifadesi

Şekil 3-34'deki tablodan f fonksiyonu,

$$f=\overline{x}_4C_2+x_4C_1 \tag{3-37}$$

olarak bulunur. Eşdeğerlik sınıflarının Şekil 3-35'de görüldüğü gibi kodlanması sonucunda α fonksiyonu,

$$\alpha=\overline{x}_1\overline{x}_2x_3+\overline{x}_1x_2\overline{x}_3+x_1\overline{x}_2\overline{x}_3+x_1x_2x_3 \tag{3-38}$$

olarak bulunur.

	α_1
C_1	0
C_2	1

Şekil 3-35 Eşdeğerlik Sınıflarının kodlanması

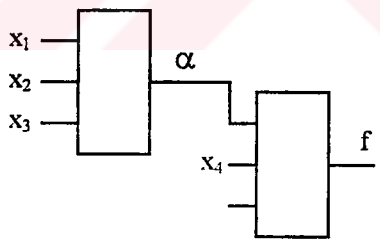
Şekil 3-35'deki kodlama tablosundan

$$C_1 = \overline{\alpha}$$
$$C_2 = \alpha \qquad (3\text{-}39)$$

bulunur. (3-39) ifadesinin (3-37) ifadesinde yerine konulmasıyla f fonksiyonu,

$$f = \overline{x}_4\alpha + x_4\overline{\alpha} \qquad (3\text{-}40)$$

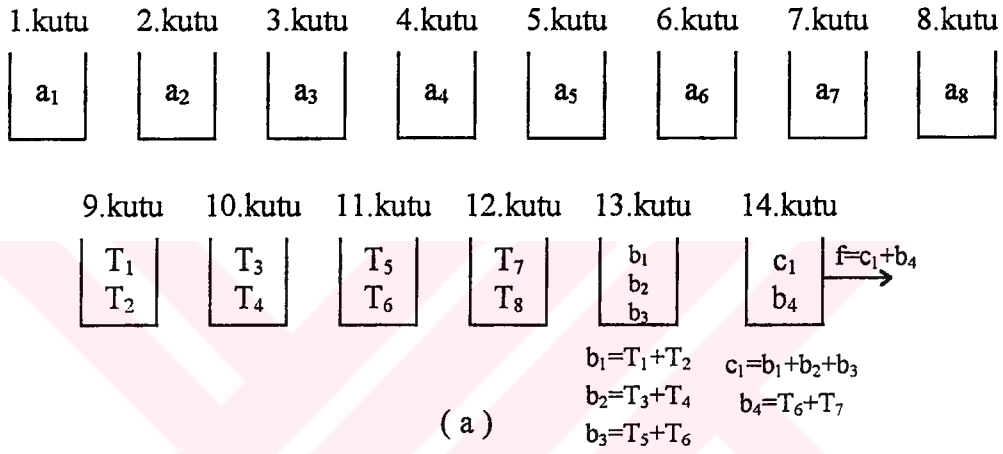
olarak bulunur. Bu ifade Şekil 3-36'da görüldüğü gibi 2 tane 3-LUT kullanılarak gerçeklenebilir.



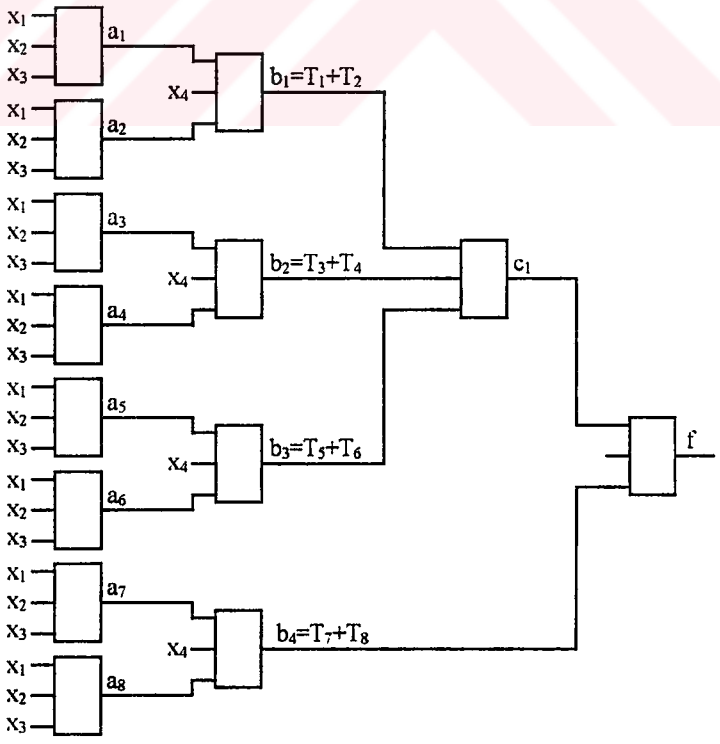
Şekil 3-36 $f(x_1,..,x_4) = \overline{x}_1\overline{x}_2\overline{x}_3x_4 + \overline{x}_1\overline{x}_2x_3\overline{x}_4 + \overline{x}_1x_2\overline{x}_3\overline{x}_4 + \overline{x}_1x_2x_3x_4 + x_1x_2\overline{x}_3x_4 + x_1x_2x_3\overline{x}_4 + x_1\overline{x}_2\overline{x}_3\overline{x}_4 + x_1\overline{x}_2x_3x_4$ Fonksiyonunun Fonksiyonel Ayrıştırma ile 3-LUT'lar kullanılarak gerçeklenmesi

Verilen $f(x_1,..,x_4)$ fonksiyonunun kutulama yöntemiyle m=3 olacak şekilde ayrıştırılması için ilk olarak ağırlıkları 3'den büyük olan çarpım terimlerinin yeniden düzenlenmesi gerekir. (3-41) ifadesinde yeniden düzenlenmiş çarpım terimleri görülmektedir. Bu işlem sonucunda elde edilen yeni terimler kapasiteleri 3 olan 14 tane kutuya Şekil 3-37'de görüldüğü gibi yerleştirilebilirler. Bunun sonucunda fonksiyon Şekil 3-38'de görülen şekilde 14 tane 3-LUT ile gerçeklenebilecektir

$$
\begin{aligned}
T_1 &= \bar{x}_1\bar{x}_2\bar{x}_3x_4 \Rightarrow a_1 = \bar{x}_1\bar{x}_2\bar{x}_3,\ T_1 = a_1x_4\\
T_2 &= \bar{x}_1\bar{x}_2x_3\bar{x}_4 \Rightarrow a_2 = \bar{x}_1\bar{x}_2x_3,\ T_2 = a_2\bar{x}_4\\
T_3 &= \bar{x}_1x_2\bar{x}_3\bar{x}_4 \Rightarrow a_3 = \bar{x}_1x_2\bar{x}_3,\ T_3 = a_3\bar{x}_4\\
T_4 &= \bar{x}_1x_2x_3x_4 \Rightarrow a_4 = \bar{x}_1x_2x_3,\ T_4 = a_4x_4\\
T_5 &= x_1x_2\bar{x}_3x_4 \Rightarrow a_5 = x_1x_2\bar{x}_3,\ T_5 = a_5x_4\\
T_6 &= x_1x_2x_3\bar{x}_4 \Rightarrow a_6 = x_1x_2x_3,\ T_6 = a_6\bar{x}_4\\
T_7 &= x_1\bar{x}_2\bar{x}_3\bar{x}_4 \Rightarrow a_7 = x_1\bar{x}_2\bar{x}_3,\ T_7 = a_7\bar{x}_4\\
T_8 &= x_1\bar{x}_2x_3\bar{x}_4 \Rightarrow a_8 = x_1\bar{x}_2x_3,\ T_8 = a_8x_4
\end{aligned}
\qquad (3\text{-}41)
$$



(a)



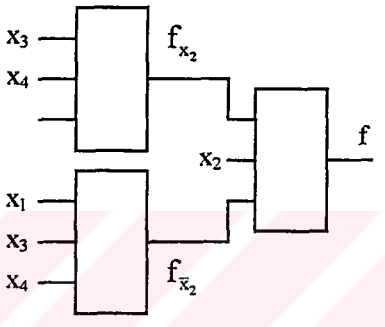
(b)

Şekil 3-38 a)$f(x_1,..,x_4) = \bar{x}_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_1\bar{x}_2x_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3\bar{x}_4 + \bar{x}_1x_2x_3x_4 + x_1x_2\bar{x}_3x_4 + x_1x_2x_3\bar{x}_4 + x_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3x_4$ Fonksiyonunun Kutulama Yöntemi ile ayrıştırılması ve b) 3-LUT'lar kullanılarak gerçeklenmesi

Aynı fonksiyon, kofaktör ayrıştırma yöntemiyle,

$$f_{\bar{x}_4} = \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2\bar{x}_3 + x_1x_2x_3 + x_1\bar{x}_2\bar{x}_3$$
$$f_{x_4} = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2x_3 + x_1x_2\bar{x}_3 + x_1\bar{x}_2x_3 \qquad (3\text{-}42)$$
$$f = f_{\bar{x}_4}\bar{x}_4 + f_{x_4}x_4$$

olacak şekilde, toplam 3 tane 3-LUT kullanılarak gerçeklenebilecektir. Bu yapı Şekil 3-39'da verilmiştir.



Şekil 3-39 $f(x_1,..,x_4) = \bar{x}_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_1\bar{x}_2x_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3\bar{x}_4 + \bar{x}_1x_2x_3x_4 + x_1x_2\bar{x}_3x_4 + x_1x_2x_3\bar{x}_4 + x_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3x_4$ Fonksiyonunun Kofaktör Yöntemi ile ayrıştırılarak 3-LUT'lar ile gerçeklenmesi

Yukarıdaki sonuçlardan da görüldüğü gibi verilen fonksiyonun simetrik olması ve m+1 değişkenli olması sebebiyle fonksiyonel ayrıştırma optimal sonuç, kofaktör ayrıştırma yöntemi optimale yakın sonuç vermiştir. Kutulama yöntemi ise optimale çok uzak bir sonuç vermiştir.

Örnek 3.6: $f(x_1,..,x_5) = x_1x_2x_3 + \bar{x}_2\bar{x}_4x_5 + \bar{x}_1x_2\bar{x}_3$ fonksiyonunun m=3 için $\{x_1x_2x_3|x_4x_5\}$ ayrışımı sonucunda elde edilen ayrıştırma tablosu ve eşdeğerlik sınıfları Şekil 3-40'da verilmiştir.

$x_1x_2x_3$ / x_4x_5	000	001	010	011	100	101	110	111
00	0	0	1	0	0	0	0	1
01	1	1	1	0	1	1	0	1
10	0	0	1	0	0	0	0	1
11	0	0	1	0	0	0	0	1

(0000) $C_1=\{011,110\} \Rightarrow C_1=\overline{x}_1x_2x_3+x_1x_2\overline{x}_3$

(0100) $C_2=\{000,001,100,101\} \Rightarrow C_2=\overline{x}_2$

(1111) $C_3=\{010,111\} \Rightarrow C_3=\overline{x}_1x_2\overline{x}_3+x_1x_2x_3$

Şekil 3-40 $\{x_1x_2x_3|x_4x_5\}$ Ayrışımına ilişkin Ayrıştırma Tablosu ve Eşdeğerlik Sınıfları

f fonksiyonu bu eşdeğerlik sınıfları cinsinden Şekil 3-41'de görülen şekilde ifade edilebilir.

$x_1x_2x_3$ / x_4x_5	C_1	C_2	C_3
00	0	0	1
01	0	1	1
10	0	0	1
11	0	0	1

Şekil 3-41 f Fonksiyonunun Eşdeğerlik Sınıfları cinsinden ifadesi

Şekil 3-41'deki tablodan f fonksiyonu,

$$f = C_3 + C_2\overline{x}_4x_5 \tag{3-43}$$

olarak bulunur. Eşdeğerlik sınıflarının Şekil 3-42'de görülen şekilde kodlanması sonucunda α_i (i=1,2) fonksiyonları,

$$\begin{aligned} \alpha_1 = C_3 &\Rightarrow \alpha_1 = \overline{x}_1x_2\overline{x}_3 + x_1x_2x_3 \\ \alpha_2 = C_2 &\Rightarrow \alpha_2 = \overline{x}_2 \end{aligned} \tag{3-44}$$

olarak bulunur.

	α_1	α_2
C_1	0	0
C_2	1	1
C_3	1	0

Şekil 3-42 Eşdeğerlik Sınıflarının kodlanması

Şekil 3-42'deki kodlama tablosundan bulunan

$$\begin{aligned} C_1 &= \overline{\alpha}_1\overline{\alpha}_2 \\ C_2 &= \alpha_1\alpha_2 \\ C_3 &= \alpha_1\overline{\alpha}_2 \end{aligned} \qquad (3\text{-}45)$$

ifadelerinin (3-43) ifadesinde yerine konulmasıyla f fonksiyonu,

$$\begin{aligned} f &= \alpha_1\overline{\alpha}_2 + \alpha_1\alpha_2\overline{x}_4x_5 \\ \Rightarrow f &= \alpha_1\overline{\alpha}_2 + \alpha_1\overline{x}_4x_5 \end{aligned} \qquad (3\text{-}46)$$

olarak bulunur. Bu ifade 3-gerçeklenemez olduğundan tekrar ayrıştırılması gerekir. $\{\alpha_2x_4x_5|\alpha_1\}$ ayrışımı için Şekil 3-43'de görülen ayrıştırma tablosu ve eşdeğerlik sınıfları elde edilir.

$\alpha_2x_4x_5$ / α_1	000	001	010	011	100	101	110	111
0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	1	0	0

(00) $D_1=\{100,110,111\} \Rightarrow D_1=\alpha_2\overline{x}_5+\alpha_2x_4$
(01) $D_2=\{000,001,010,011,101\} \Rightarrow D_2=\overline{D}_1$

Şekil 3-43 $\{\alpha_2x_4x_5|\alpha_1\}$ Ayrışımına ilişkin Ayrıştırma Tablosu ve Eşdeğerlik Sınıfları

f fonksiyonu bu eşdeğerlik sınıfları cinsinden Şekil 3-44'de görüldüğü gibi ifade edilebilir.

$\alpha_2 x_4 x_5$ / α_1	D_1	D_2
0	0	0
1	0	1

Şekil 3-44 f Fonksiyonunun Eşdeğerlik Sınıfları cinsinden ifadesi

Şekil 3-44'deki tablodan f fonksiyonu,

$$f=D_2\,\alpha_1 \tag{3-47}$$

olarak bulunur. Eşdeğerlik sınıflarının Şekil 3-45'de görüldüğü gibi kodlanması sonucunda β fonksiyonu,

$$\beta = D_1 \Rightarrow \beta = \alpha_2 \overline{x}_5 + \alpha_2 x_4 \tag{3-48}$$

olarak bulunur.

	β
D_1	1
D_2	0

Şekil 3-45 Eşdeğerlik Sınıflarının kodlanması

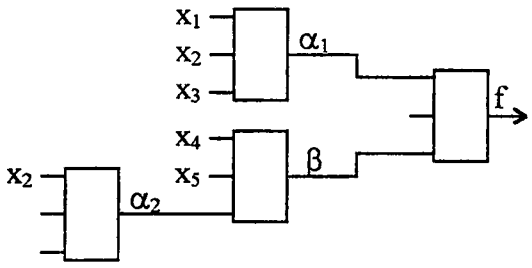
Şekil 3-45'deki kodlama tablosundan bulunan

$$\begin{aligned} D_1 &= \beta \\ D_2 &= \overline{\beta} \end{aligned} \tag{3-49}$$

ifadelerinin (3-47) ifadesinde yerine konulmasıyla f fonksiyonu,

$$f = \overline{\beta}\alpha_1 \tag{3-50}$$

olarak bulunur. Şekil 3-46'da, yapılan ayrıştırmanın toplam 4 tane 3-LUT ile gerçeklenmesi görülmektedir.



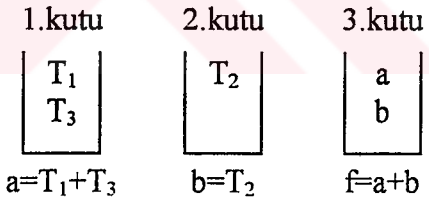
Şekil 3-46 $f(x_1,..,x_5)=x_1x_2x_3+\overline{x}_2\overline{x}_4x_5+\overline{x}_1x_2\overline{x}_3$ Fonksiyonunun Fonksiyonel Ayrıştırma ile 3-LUT'lar kullanılarak gerçeklenmesi

Verilen $f(x_1,..,x_5)$ fonksiyonu m=3 olacak şekilde kutulama yöntemiyle ayrıştırılırsa Şekil 3-47'de görülen şekilde toplam 3 kutu yani 3 tane 3-LUT kullanılarak gerçeklenebilir.

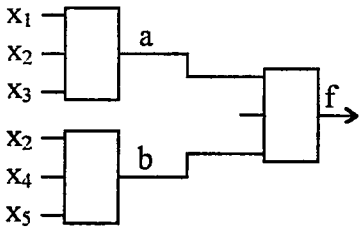
$T_1=x_1x_2x_3 \quad w(T_1)=3$

$T_2=\overline{x}_2\overline{x}_4x_5 \quad w(T_2)=3$

$T_3=\overline{x}_1x_2\overline{x}_3 \quad w(T_3)=3$



(a)



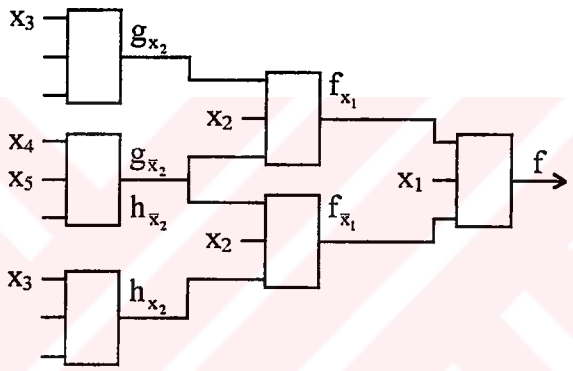
(b)

Şekil 3-47 a) $f(x_1,..,x_5)=x_1x_2x_3+\overline{x}_2\overline{x}_4x_5+\overline{x}_1x_2\overline{x}_3$ Fonksiyonunun Kutulama Yöntemiyle ayrıştırılması ve b) 3-LUT'lar kullanılarak gerçeklenmesi

Aynı $f(x_1,..,x_5)$ fonksiyonu m=3 olacak şekilde kofaktör yöntemiyle ayrıştırılırsa

$$\begin{aligned}
&\bullet g_{x_2} = x_3 \qquad \bullet g_{\bar{x}_2} = \bar{x}_4 x_5 \\
&\bullet h_{x_2} = \bar{x}_3 \qquad \bullet h_{\bar{x}_2} = \bar{x}_4 x_5 \\
&\bullet f_{x_1} = x_2 g_{x_2} + \bar{x}_2 g_{\bar{x}_2} \\
&\bullet f_{\bar{x}_1} = x_2 h_{x_2} + \bar{x}_2 h_{\bar{x}_2} \\
&f = x_1 f_{x_1} + \bar{x}_1 f_{\bar{x}_1}
\end{aligned} \tag{3-51}$$

olmak üzere Şekil 3-48'de görüldüğü gibi 5 tane 3-LUT kullanılarak gerçeklenebilir.



Şekil 3-48 $f(x_1,..,x_5) = x_1x_2x_3 + \bar{x}_2\bar{x}_4x_5 + \bar{x}_1x_2\bar{x}_3$ Fonksiyonunun Kofaktör Yöntemi ile ayrıştırılması sonucunda 3-LUT'lar kullanılarak gerçeklenmesi

Yukarıdaki sonuçlardan görüldüğü gibi bu örnekte en iyi sonucu kutulama yöntemi vermiştir. Ancak fonksiyonel ayrıştırma ve kofaktör yöntemiyle ayrıştırma sonucunda elde edilen fonksiyonlara 3.1.2'de anlatılacak olan işlemlerin uygulanmasıyla kullanılan LUT sayısı azaltılabilmektedir.

Yukarıdaki örneklerden de görüldüğü gibi verilen fonksiyonları gerçekleştirmek için gerekli LUT sayısı yöntemden yönteme çok değişebilmektedir. Örnek 3.4'de incelenen fonksiyon için bütün yöntemler aynı sonucu vermiş ve fonksiyon 3 tane 3-LUT ile gerçeklenmiştir. Örnek 3.5'de incelenen fonksiyonun fonksiyonel ayrıştırma yöntemiyle ayrıştırılması sonucunda optimal çözüm bulunmuş ve fonksiyon 2 tane

3-LUT ile gerçeklenmiştir. Fonksiyonun kutulama yöntemiyle ayrıştırılması sonucunda optimalden çok uzak bir sonuca ulaşılmış ve fonksiyonun gerçeklenmesi için 14 tane 3-LUT kullanılmıştır. Ayrışımın kofaktör yöntemiyle yapılması sonucunda optimale yakın bir çözüm bulunmuş ve fonksiyon 3 tane 3-LUT ile gerçeklenmiştir. Örnek 3-6'da incelenen fonksiyon için en iyi sonucu kutulama yöntemi vermiştir ve fonksiyon 3 tane 3-LUT ile gerçeklenmiştir. Fonksiyonun fonksiyonel ayrıştırma yöntemiyle ayrıştırılması sonucunda 4 tane 3-LUT, kofaktör yöntemiyle ayrıştırılması sonucunda 6 tane 3-LUT kullanılmıştır.

3.1.2 Blok Sayısını Azaltılması

Bir fonksiyonun, fonksiyonel ayrıştırma, kutulama veya kofaktör yöntemleriyle ayrıştırılması sonucunda her biri bir m-LUT kullanılarak gerçeklenebilecek m-gerçeklenebilir alt fonksiyonlar elde edilir. Örnek 3.6'da verilen $f(x_1,..,x_5)=x_1x_2x_3+\overline{x}_2\overline{x}_4x_5+\overline{x}_1x_2\overline{x}_3$ fonksiyonu fonksiyonel ayrıştırma işlemi sonucunda her biri 3-gerçeklenebilir olan,

$$\begin{aligned} f &= \overline{\beta}\alpha_1 \\ \beta &= \alpha_2\overline{x}_5+\alpha_2x_4 \\ \alpha_1 &= \overline{x}_1x_2\overline{x}_3+x_1x_2x_3 \\ \alpha_2 &= \overline{x}_2 \end{aligned} \qquad (3\text{-}52)$$

fonksiyonlarına ayrıştırılarak toplan 4 tane 3-LUT ile gerçeklenmişti.Diğer yandan α_2 fonksiyonunun β fonksiyonunda yerine konulmasıyla elde edilen,

$$\begin{aligned} f &= \overline{\beta}\alpha_1 \\ \beta &= \overline{x}_2\overline{x}_5+\overline{x}_2x_4 \\ \alpha_1 &= \overline{x}_1x_2\overline{x}_3+x_1x_2x_3 \end{aligned} \qquad (3\text{-}53)$$

fonksiyonları da 3-gerçeklenebilir fonksiyonlardır ve bu fonksiyonlar üzerinden f fonksiyonunun gerçeklenmesi için toplam 3 tane 3-LUT yetmektedir. Görüldüğü gibi α_2 ifadesi β ifadesinde yerine konulduğunda 3-gerçeklenebilir 4 fonksiyondan yine

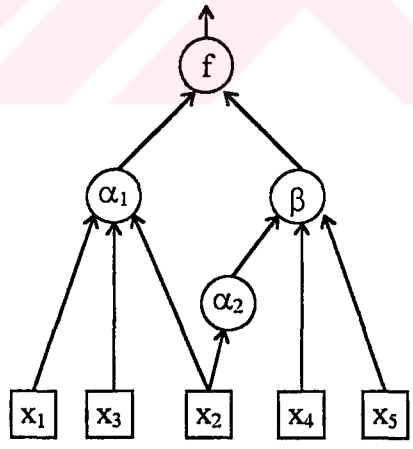
3-gerçeklenebilir 3 fonksiyon elde edilmekte dolayısıyla bir tane LUT kazanılmış olmaktadır.

m-gerçeklenebilirlik şartını bozmadan, bir fonksiyonun diğer fonksiyonlarda yerine konulmasıyla yapılan bu işlem "Örtme (covering)" olarak adlandırılmaktadır.

3.1.2.1 Örtme Adımlarının Belirlenmesi

Örtme işleminde amaç, herbiri m-gerçeklenebilir k fonksiyondan, bazılarının diğerlerinde yerine konularak elenmesi sonucunda, $\tilde{k} \leq k$ olmak üzere m-gerçeklenebilir $\tilde{k}$ fonksiyon elde etmektir.

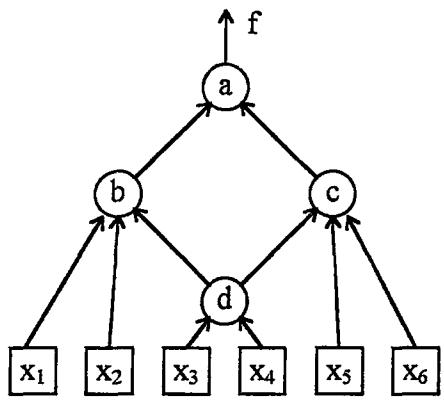
Tanım 3.1: Verilen fonksiyonun ayrıştırılması sonucunda elde edilen m-gerçeklenebilir her bir fonksiyona bir düğüm, her giriş ve çıkışa da yönlü bir graf elemanı karşı düşürülerek elde edilen grafa **fonksiyon grafı** denir. Şekil 3-49'da (3-52) ifadesinde verilen fonksiyonlar cinsinden f fonksiyonuna ilişkin graf görülmektedir.



Şekil 3-49 Fonksiyon Grafi

Tanım 3.2: Fonksiyon grafindaki herhangi bir düğümün kök düğüm olarak alınmasıyla elde edilen ve serbest girişleri içermeyen alt grafa **"Süper düğüm"** adı verilir [2]. Fonksiyon grafindaki bir düğüm birden fazla süper düğümün kök düğümü olabilir. Örneğin Şekil 3-50'de görülen fonksiyon grafinda kök düğümü a olan {a},

{a,b}, {a,c}, {a,b,d}, {a,c,d}, {a,b,c} ve {a,b,c,d} süper düğümleri vardır. Aynı şekildeki {a,d}, a ve d gibi iki kök düğüm olması sebebiyle bir süper düğüm değildir.



Şekil 3-50 Bir çıkışlı, 4 düğümlü ve 6 girişli bir devrenin grafi

Tanım 3.3: Girişlerinin sayısı m'den az olan süper düğümlere **m-gerçeklenebilir süper düğüm** denir (S:süper düğüm, σ(S):S'nin giriş değişkenlerinin oluşturduğu küme, $|\sigma(S)| \leq m$). Aşağıda Şekil 3-50'deki grafta kök düğümü a olan 5-gerçeklenebilir süper düğümler görülmektedir.

süper düğüm	girişler
{a}	{b,c}
{a,b}	{d,c,x_1,x_2}
{a,c}	{b,d,x_5,x_6}
{a,b,d}	{c, x_3, x_4,x_5,x_6}
{a,c,d}	{b,x_1,x_2, x_3, x_4}
{a,b,c}	{d,x_1,x_2, x_5, x_6}

{a,b,c,d} süper düğümü 6 girişli (x_1, x_2, x_3, x_4, x_5, x_6) olduğu için 5-gerçeklenebilir değildir.

m-gerçeklenebilir k tane fonksiyondan, $\tilde{k} \leq k$ olmak üzere $\tilde{k}$ tane m-gerçeklenebilir fonksiyon aşağıda verilen iki adımla elde edilebilir[2].

1) Graftaki bütün m-gerçeklenebilir süper düğümler belirlenir.
2) Bu süper düğümlerden minimum elemanlı bir örtü seçilir.

3.1.2.1.1 m-gerçeklenebilir Süper Düğümlerin Belirlenmesi

Kök düğümü n olan bütün m-gerçeklenebilir süper düğümlerin belirlenmesi için ilk olarak $|\sigma| \leq m$ olan bütün $\sigma \subseteq TFI(n)$ kümeleri belirlenir. Daha sonra herbir kümenin bir süper düğümün giriş kümesi olup olmadığı kontrol edilir. Bu işlem Şekil 3-51'de verilen algoritma ile yapılabilir[2]. Bir graftaki bütün süper düğümlerin belirlenebilmesi için her bir düğüm kök düğüm olarak seçilerek bu işlemler tekrarlanır. Her bir düğüm için çok sayıda σ kümesi olması nedeniyle algoritmanın sonuç vermesi uzun zaman almaktadır.

```
/* verilen bir η devresindeki n kök düğümü ve giriş
/* kümesine karşı düşen S süper düğümünün bulunması */
/* başlangıç koşulu S={n}*/
/* baslangıçta bütün düğümlerin işaretlenmemiş olduğu
varsayılıyor*/
determine_supernode_from_support(n,σ)
{
     FI(n)=set of fanins(n);
     for each F∈FI(n)
     {
          if (F∈σ) continue;
          if (F∈PI(η)) continue;
          if F is marked continue;
          S=S∪{F};
          mark F;
          determine_supernode_from_support (F,σ);
     }
}
```

Şekil 3-51 Bir süper düğümü, girişlerinden ve kök düğümünden faydalanarak bulan yöntem

Şekil 3-52'de verilen algoritma sadece kök düğüm olarak alınan düğümün giriş düğümlerinden faydalanarak o kök düğüme ilişkin m-gerçeklenebilir süper düğümleri belirlemesi sebebiyle daha kısa sürede sonuca ulaşır.

```
/* verilen bir η devresindeki n kök düğümüne karşı*/
/* düşen S süper düğümünün bulunması */
/* başlangıç koşulu S={n}*/
determine_m_feasible_supernode(S)
{
     FI(S)=set of fanins(S);
     for each F∈FI(S)
     {
          if (F∈PI(η)) continue;
          if (|FI(S)∪FI(F)|-1≤m)
          {
               S=S∪{F};
               determine_m_feasible_supernode(S);
          }
     }
}
```

Şekil 3-52 Bir m-gerçeklenebilir süper düğümü kök düğümünden faydalanarak bulan yöntem

Örneğin Şekil 3-50'de verilen fonksiyon grafında a kök düğümüne karşı düşen 5-gerçeklenebilir süper düğümlerin Şekil 3-52'de verilen alt program yardımıyla bulunması için ilk olarak S={a} seçilerek S'nin giriş kümesi FI(S)={b,c} belirlenir. Daha sonra FI(S) kümesinden bir eleman seçilir ve ilk olarak serbest giriş olup olmadığına bakılır. Eğer bu eleman bir serbest girişse, bir süper düğümde serbest girişler olmayacağı için, herhangi bir işlem yapılmadan FI(S) kümesindeki diğer elemana geçilir. Serbest giriş olmayan bir eleman bulunduğunda ($|FI(S)\cup FI(F)|-1\leq m$ olup olmadığına bakılır. Eğer bu koşul sağlanıyorsa bu eleman S kümesine eklenir ve yeni S kümesi için yukarıda anlatılan işlemler tekrarlanır. Yukarıda verilen FI(S) kümesinden ilk olarak b düğümü seçilirse serbest giriş olmadığı için ($|FI(S)\cup FI(b)|-1\leq 5$ şartı incelenir. $FI(S)\cup FI(b)=\{x_1,x_2,b,c,d\}$ ve ($|\{x_1,x_2,d,b,c\}|-1=4\leq 5$) olması nedeniyle b düğümü S kümesine eklenir. Böylece yeni S kümesi {a,b} olarak bulunur ve aynı işlemler tekrarlanır. $FI(\{a,b\})=\{x_1,x_2,c,d\}$ kümesinde x_1 ve x_2 serbest girişler olduğu için herhangi bir işlem yapılmadan d düğümüne geçilir. $FI(S)\cup FI(d)=\{x_1,x_2,d,c,x_3,x_4\}$ kümesi için ($|\{x_1,x_2,d,c,x_3,x_4\}|-1=5\leq 5$) olduğundan d düğümü S kümesine eklenerek

S={a,b,d} için işlemler tekrarlanır. Sonuçta kök düğümü a olan 5-gerçeklenebilir süper düğümler {a}, {a,b}, {a,c}, {a,b,d}, {a,b,c} ve {a,c,d} olarak bulunur.

Bir graftaki bütün m-gerçeklenebilir süper düğümlerin bulunabilmesi için graftaki her düğüm sırayla kök olarak seçilerek yukarıda anlatılan işlemler tekrarlanmalıdır. Örneğin Şekil 3-50'deki graftan 5-gerçeklenebilir {a}, {a,b}, {a,c}, {a,b,d}, {a,b,c}, {a,c,d}, {b}, {b,d}, {c}, {c,d} ve {d} süper düğümleri bulunabilir.

3.1.2.1.2 Minimum Elemanlı Örtünün Seçilmesi

Fonksiyon grafindaki m-gerçeklenebilir bütün süper düğümlerin belirlenmesinden sonra ikinci adım olarak bu düğümlerden aşağıdaki iki koşulu sağlıyan minimum elemanlı bir M örtüsü belirlenmelidir. [2].

1) Graftaki her çıkış düğümü için kökü bu düğüm olan bir süper düğüm seçilmelidir.

2) Örtüdeki süper düğümlerin girişleri ya başka bir süper düğümün çıkışı ya da bir serbest giriş olmalıdır.

Aşağıda bu şartları sağlayan örtülerin belirlenmesi için bir yöntem verilmiştir.

İlk olarak graftaki bütün m-gerçeklenebilir süper düğümler bir sütun olarak sıralanır ve herbir süper düğümün karşısına serbest olmayan girişleri yazılır. Bütün girişleri serbest giriş olan süper düğümlerin karşısına "-" konur. Örtünün ilk elemanı olarak kökü, grafin çıkış düğümü olan bir süper düğüm seçilir. Daha sonra kök düğümleri, seçilen süper düğümün girişleri olan süper düğümlerde örtüye eklenir. Bu işlem bütün girişleri serbest düğümler olan süper düğümlerin örtüye eklenmesiyle sona erer. Bu yöntemle olası bütün örtüler belirlenerek aralarından minimum elemanlı olanı seçilir. Süper düğümlerin seçilmesi sırasında maksimum düğümlü minimum girişli süper düğümlerin seçilmesi daha avantajlıdır.

Bu yöntemle elde edilecek örtülerdeki eleman sayılarının alt ve üst sınırları, M_δ 2.şartı sağlaması gerekmeyen minimum elemanlı herhangi bir örtü, M_e fonksiyon grafindaki düğümlerin oluşturduğu küme olmak üzere,

$$|M_\delta| \leq |M| \leq |M_e| \qquad (3\text{-}54)$$

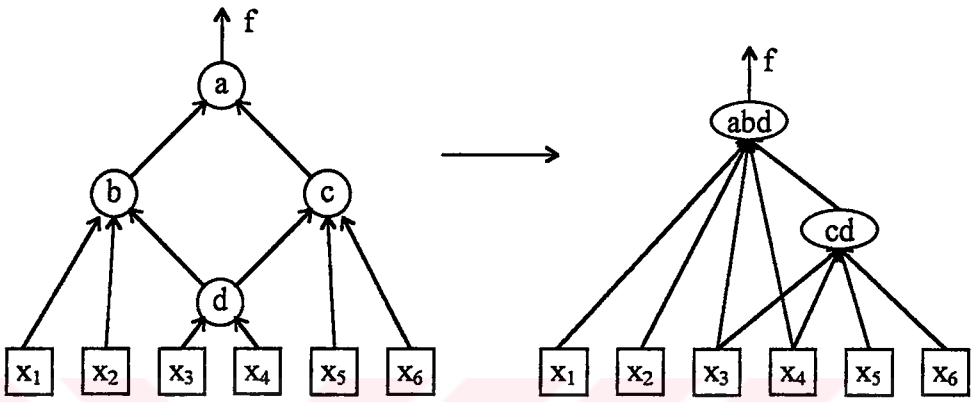
ile belirlenir. Aşağıda bu yöntem bir örnek üzerinde incelenmiştir.

Örnek 3.7: Şekil 3-50'deki graf için $M_\delta=\{\{a,b\},\{c,d\}\}$ ve $M_e=\{a,b,c,d\}$ olduğu için bu graftan elde edilecek örtülerin eleman sayıları $2\leq M \leq 4$ olacaktır. Graftan elde edilen 5-gerçeklenebilir süper düğümler ve bunların girişlerinden aşağıdaki tablo elde edilir.

süper düğümler	girişler
{a}	b,c
{a,b}	c,d
{a,c}	b,d
{a,b,d}	c
{a,c,d}	b
{a,b,c}	d
{b}	d
{b,d}	-
{c}	d
{c,d}	-
{d}	-

Örtünün ilk elemanı olarak, grafin çıkış düğümü olan a düğümününü kapsayan {a}, {a,b}, {a,c}, {a,b,d}, {a,c,d}, {a,b,c} süper düğümlerden biri seçilmelidir. Yukarıdaki tablodan, maksimum düğümlü minimum izleyicili {a,b,d}, {a,c,d}, {a,b,c} süper düğümlerinden birinin seçilmesinin daha avantajlı olacağı görülmektedir. Eğer {a,b,d} süper düğümü seçilirse bunun girişi olan c düğümünün kökü olduğu {c} ve {c,d} süper düğümlerinden biri alınmalıdır. Bütün girişleri serbest giriş olan {c,d} düğümünün alınmasıyla işlem bitmiş olur. Böylece $M_1=\{\{a,b,d\},\{c,d\}\}$ örtüsü bulunmuş olur ve alt sınıra ulaşıldığı için başka örtülerin

aranmasına gerek kalmaz. Şekil 3-53'de yeni fonksiyon grafi görülmektedir. Eğer {c,d} düğümü yerine {c} düğümü seçilseydi bunun girişi olan {d} düğümününde örtüye eklenmesi gerekecekti. Sonuçta 3 elemanlı olan dolayısıyla minimal olmayan M_2={{a,b,d},{c},{d}} örtüsü bulunacaktı. Dolayısıyla minimum elemanlı başka örtülerin araştırılması gerekecekti.



Şekil 3-53 4 Düğümlü devreden örtme işlemiyle 2 düğümlü devre elde edilmesi

Şekil 3-50'de verilen fonksiyon grafının gerçeklenmesi için toplam 4 tane 5-LUT gerekirken örtme işlemi sonucunda iki tane LUT kazanılarak iki tane 5-LUT'tan oluşan devre elde edilmiştir.

3.2 "Chortle-crf" Yöntemi

"Chortle-crf" yöntemi 1991 yılında R.F.Francis, J.Rose ve Z.Vranesic tarafından geliştirilmiştir. Yöntemin amacı Boole fonksiyonlarının minimum sayıda m-LUT ile gerçeklenmesidir. Bu amaçla, verilen fonksiyon ilk olarak herbir parça bir m-LUT ile gerçeklenecek şekilde parçalara ayrılır. Daha sonra bu alt parçalar uygun şekilde birleştirilerek minimum sayıda m-LUT'dan oluşan devre elde edilir. Bu yöntemin avantajı ayrıştırma ve örtme aşamalarını birleştirmesidir.

Fonksiyonların bu yöntemle m-LUT'lar kullanılarak gerçeklenebilmesi için ayrıştırma ağacı adı verilen bir yapı oluşturulur. Ayrıştırma ağacının oluşturulmasında 3.1.1.2'de verilen kutulama yönteminin temelini oluşturan "bin-packing" algoritması kullanılır. Bu yöntemde de kutulama yönteminde olduğu gibi ilk olarak çarpım terimleri sabit

kapasiteli kutulara yerleştirilir. Ancak bu işlem sırasında kutulama yönteminin tersine $w(T_i+T_j)=w(T_i)+w(T_j)$ olmaktadır. Yani bu yöntemde aynı değişkenler farklı değişken olarak kabul edilmektedir. Daha sonra kutulama yönteminde olduğu gibi "bin-packing" algoritmasında da kutular kapatılarak fonksiyon ayrıştırılmış olmaktadır. Örneğin ,

$$f(x_1,....,x_9)=x_1x_2+ x_3x_4+ x_4x_5+ x_6x_7+ x_8x_9 \qquad (3\text{-}55)$$

fonksiyonu "bin-packing" algoritmasıyla m=5 olacak şekilde ayrıştırılırsa

$$\begin{aligned} g_1&= x_1x_2+ x_3x_4 \\ g_2&= x_4x_5+ x_6x_7+g_1 \\ f&= x_8x_9+g_2 \end{aligned} \qquad (3\text{-}56)$$

olmak üzere toplam 3 tane 5-LUT ile gerçeklenebilecektir. Diğer taraftan (3-55) ifadesi kutulama yöntemiyle ayrıştırılmış olsaydı,

$$\begin{aligned} h_1 &= x_1x_2 + x_3x_4 + x_4x_5 \\ f &= x_6x_7 + x_8x_9 + h_1 \end{aligned} \qquad (3\text{-}57)$$

olmak üzere toplam 2 tane 5-LUT ile gerçeklenecekti.

Ancak ortak değişkenleri olan çarpım terimlerinin aynı kutuya yerleştirilmesi her zaman optimum sonucu vermez. Ayrıca ortak değişkenleri olan fakat birlikte aynı kutuya sığamayacak sayıda çarpım terimleri olabilir. Örneğin $x_1x_2x_3$, $x_3x_4x_5$ ve $x_1x_5x_6$ terimlerinin ortak değişkenleri vardır fakat üçü birlikte toplam ağırlığın 6 olması sebebiyle 5 kapasiteli bir kutuya yerleştirilemezler. Bu sebeple 3 çarpım terimi arasından 2 tanesinin seçilmesi gerekmektedir. "Chortle-crf" yönteminde bu problem, "bin-packing" algoritmasının çok hızlı olması sebebiyle, olası bütün kombinezonların denenmesiyle çözülür. Yani herhangi bir birleştirme yapılmadan ve olabilecek bütün birleştirmeler yapıldıktan sonra alınan sonuçlardan minimum LUT gerektireni seçilir. Ancak ortak değişkenleri olan çarpım terimlerinin çok olması durumunda bu işlem

uzun sürmektedir. Bu durumda "bin-packing" algoritması, bütün olasılıklar yerine en uygun çarpım terimlerinin birleştirilmeyle oluşturulmuş yeni çarpım terimlerine uygulanır. En uygun çarpım terimlerinin belirlenmesinde 3 kriterden faydalanılır: Buna göre maksimum sayıda çarpan içeren, mevcut kutularla ve kalan çarpım terimleriyle maksimum sayıda değişken paylaşan terim seçilmelidir. Seçilen terim maksimum sayıda değişkenini paylaştığı bir kutuya, kutunun kapasitesini aşmamak şartıyla yerleştirilir. Böyle bir kutu yoksa yeni bir kutu açılır[1].

3.3 "VISMAP" Yöntemi

VISMAP yöntemi sadece blok sayısının azaltılması aşaması için geliştirilmiş olup, verilen fonksiyondan m-gerçeklenebilir alt fonksiyonların herhangi bir ayrıştırma yöntemiyle elde edildiği kabul edilmektedir. Blok sayısının azaltılması "mis-fpga" yöntemine benzemektedir. Ancak bu yöntemde bütün süper düğümlerin elde edilmesine gerek yoktur.

Yöntem, fonksiyon grafindaki yönlü graf elemanlarının yani düğümler arasındaki bağlantıların "görünür" ve "görünmez" olarak etiketlenmesiyle çalışır. Bir bağlantının görünür olarak etiketlenmesinin anlamı, devrenin gerçeklenmesi aşamasında bu bağlantının uçlarındaki düğümlere ayrı LUT'lar karşı düşürüleceğidir. Görünmez olarak etiketlenmiş bir bağlantı bir LUT'un içinde gerçeklenecektir. Görünür olarak etiketlenmiş bir bağlantı sonuçtaki devrede bir süper düğümün girişine karşılık gelir. Görünmez olarak etiketlenen bağlantılar süper düğümlerin içinde gerçeklenmiş olur.

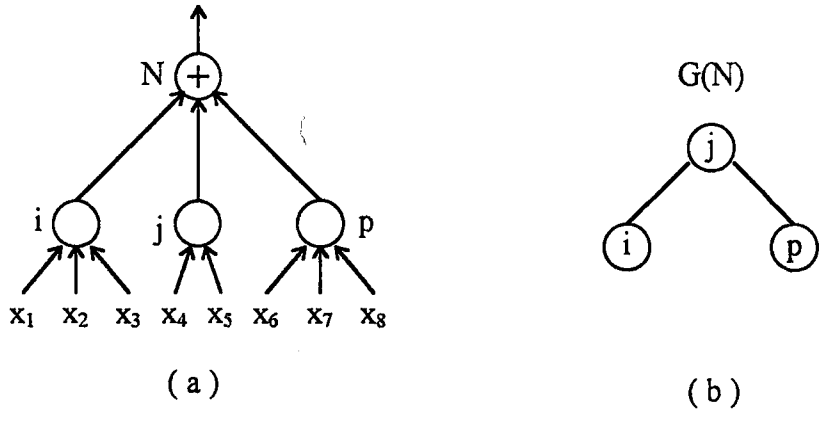
K tane bağlantısı olan bir fonksiyon grafinda, herbir bağlantının görünür yada görünmez olmasına karşılık gelen 2^K tane değişik etiketleme yapılabilir. Bağlantılar serbest girişlerden çıkışa doğru ilk olarak görünür daha sonra görünmez olarak etiketlenirler. Serbest girişleri düğümlere bağlayan bağlantılar daima görünür olarak etiketlenir. Eğer bir bağlantı görünmez olarak etiketlendiyde bu bağlantının uçlarındaki düğümler birleştirilir. Birleştirilmiş düğümlerin toplam giriş sayılarının m'den büyük olması durumunda bu düğümleri daha yukarılardaki düğümlere bağlayan bağlantılar, m-gerçeklenebilir bir yapı oluşturamayacakları için kontrol edilmezler. Bu

işlem yapılacak kontrol sayısını sayısını azaltmaktadır.yapılan bütün etiketlemelerin sonucunda en az sayıda düğüm üreteni seçilir.

Fonksiyon grafında çok sayıda bağlantının olması durumunda yukarıda anlatılan işlemler çok zaman alır. Bu gibi durumlarda graf, alt graflara ayrılır ve herbir alt graf kendi içinde kontrol edilir. Daha sonra çözümler birleştirilir.

3.4 "TechMap" Yöntemi

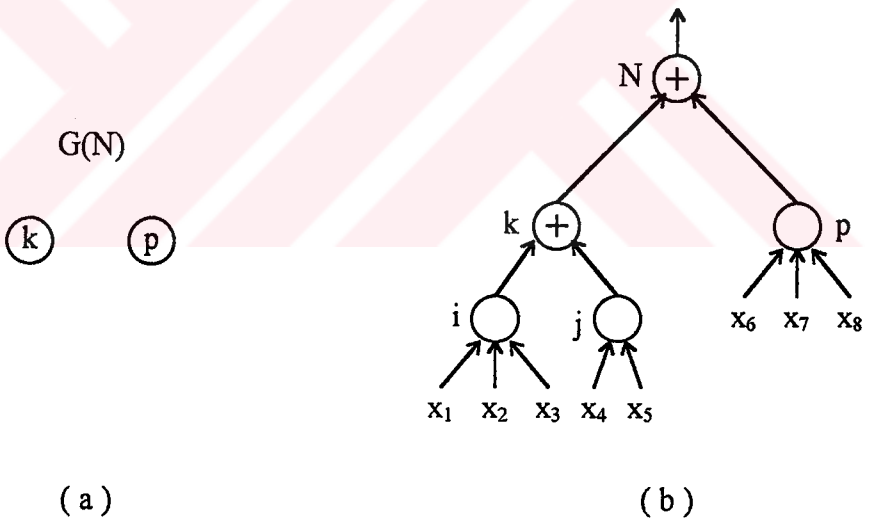
"TechMap" yöntemi P. Sawkar ve D. Thomas tarafından geliştirilmiştir. Yöntemin ayrıştırma aşamasında VE-VEYA ayrışımı kullanılır. Blok sayısının azaltılması amacıyla fonksiyon grafindaki her bir düğüm için **uyumluluk grafı** adı verilen bir graf oluşturulur. Bir N düğümünün girişi olan i ve j düğümleri için $|\sigma(i)\cup\sigma(j)|\leq m$ şartı sağlanıyorsa i ve j uyumludur denir. Bu i ve j düğümlerinin birleştirilerek yeni bir m-gerçeklenebilir k düğümü oluşturabilecekleri anlamına gelir. i ve j düğümleri uyumlu oldukları için N düğümü için çizilen uyumluluk grafina G(N) konularak uyumlu olduklarını belirtmek amacıyla bir çizgiyle birleştirilirler. Bu işlem bütün uyumlu düğümler bulununcaya kadar sürdürülür. Şekil 3-54b'de Şekil 3-54a'da verilen fonksiyon grafindaki N düğümüne karşı düşen uyumluluk grafi görülmektedir.



Şekil 3-54 m=5 için N düğümü için çizilen uyumluluk grafi

Uyumluluk grafından birleştirilecek düğümlerin seçilmesi amacıyla her uyumlu (i,j) çifti için uyumsuzluk sayısı adı verilen bir parametre belirlenir. Bu parametre i ve j

düğümlerinin birleştirilmesi sonucunda uyumluluk grafından silinmek zorunda kalınacak bağlantıların sayısıdır. Graftaki bir p düğümü i veya j düğümüyle uyumlu olmasına karşın, i ve j düğümlerinin birleştirilmesiyle oluşan k düğümüyle uyumsuz olabilir. Bunun sonucunda i ve j düğümlerinin birleştirilmesi durumunda p düğümünün bunlarla olan bağlantısı silinmelidir. Örneğin Şekil 3-54a'da görülen grafta p düğümü j düğümüyle uyumlu olmasına rağmen i ve j düğümlerinin birleştirilmesiyle oluşacak k düğümüyle uyumsuzdur. Her adımda uyumsuzluk sayısı minimum olan çift seçilerek birleştirilir ve bu düğümlerle olan bağlantılar silinir. Uyumluluk grafindaki diğer düğümlerin yeni oluşan k düğümüyle olan uyumlulukları belirlenerek yeni bağlantılar yapılır. Uyumsuzluk sayısı minimum olan çiftlerin seçimine uyumluluk grafinda bağlantı kalmayıncaya kadar devam edilir. Şekil 3-55a'da i ve j düğümlerinin birleştirilmesiyle oluşan yeni uyumluluk grafi, Şekil 3-55b'de yapılan işlemler sonucunda elde edilen yeni fonksiyon grafi görülmektedir.



Şekil 3-55 a)m=5 için N düğümüiçin çizilen yeni uyumluluk grafi ve
b) yeni fonksiyon grafi

SONUÇLAR

Bu tezde Boole fonksiyonlarının m-LUT'lar ile gerçeklenmesinde kullanılan "mis-fpga", "chortle-crf", "VISMAP" ve "TechMap" yöntemleri incelenerek bazı hususlar açıklığa kavuşturulmuştur. Ayrıca üçüncü bölümde verilen Teorem 3.1, bu tezde ortaya konulmuş ve ispatı verilmiştir.

3.1.1.1.4'de fonksiyonel ayrıştırmanın Karnaugh diyagramlarından faydalanılarak yapılmasına ilişkin bir yol önerilmiş ve Örnek 3.2'de ayrıştırmanın bu şekilde yapılmasının, ikili karar diyagramları ile yapılmasından daha hızlı sonuca ulaştığı gösterilmiştir.

Bu tezde, ayrıştırma yöntemlerinin verdikleri sonuçlar geliştirilen örnekler üzerinde incelenmiştir. Örnek 3.4'de verilen fonksiyon için bütün ayrıştırma yöntemleri aynı sonucu vermiş ve fonksiyon 3 tane 3-LUT ile gerçeklenmiştir. Örnek 3.5'de incelenen fonksiyonun fonksiyonel ayrıştırma yöntemiyle ayrıştırılması sonucunda optimal çözüm bulunmuş ve fonksiyon 2 tane 3-LUT ile gerçeklenmiştir. Fonksiyonun kutulama yöntemiyle ayrıştırılması sonucunda optimalden çok uzak bir sonuca ulaşılmış ve fonksiyonun gerçeklenmesi için 14 tane 3-LUT kullanılmıştır. Ayrışımın kofaktör yöntemiyle yapılması sonucunda optimale yakın bir çözüm bulunmuş ve fonksiyon 3 tane 3-LUT ile gerçeklenmiştir. Örnek 3-6'da incelenen fonksiyon için en iyi sonucu kutulama yöntemi vermiştir ve fonksiyon 3 tane 3-LUT ile gerçeklenmiştir. Fonksiyonun fonksiyonel ayrıştırma yöntemiyle ayrıştırılması sonucunda 4 tane 3-LUT, kofaktör yöntemiyle ayrıştırılması sonucunda 6 tane 3-LUT kullanılmıştır.

Yukarıda da görüldüğü gibi fonksiyonların gerçeklenmesi için gerekli LUT sayısı kullanılan ayrıştırma yöntemine göre değişiklikler göstermektedir. Verilen bir fonksiyonu minimum sayıda LUT ile gerçekleştirecek yöntemin peşinen belirlenmesi

Bu tezde "mis-fpga" yönteminin ikinci aşamasını oluşturan blok sayısının azaltılması işleminde ilk adım olan m-gerçeklenebilir bütün süper düğümlerin belirlenmesi problemi için bir algoritma geliştirilmiştir. Ayrıca ikinci adım olan minimum sayıda süper düğümün seçilerek bir örtü oluşturulması işlemi için bir yöntem önerilmiştir.

3.1.1.1'de incelenen fonksiyonel ayrıştırma işleminin optimal veya optimale yakın bir sonuç vermesi minimum sayıda eşdeğerlik sınıfı verecek bir ayrışımın belirlenmesine bağlıdır. Bu tezin devamı şeklinde ele alınabilecek noktalardan biri minimum sayıda eşdeğerlik sınıfı verecek bir ayrışımın bulunması problemidir. Ayrıca, eşdeğerlik sınıflarının kodlanmasının, fonksiyonun gerçeklenmesinde kullanılacak LUT sayısına etkisi de incelenebilir.

KAYNAKLAR

[1] BROWN, S.D., FRANCIS, R.J., ROSE, J., VRANESIC, Z.G., Field-Programmable Gate Arrays, Kluwer Academic Publishers, 1992

[2] MURGAI, R., BRAYTON, R.K., VINCENTELLI, A.S., Logic Synthesis For Field-Programmable Gate Arrays, Kluwer Academic Publishers, 1995

[3] The Programmable Gate Array Data Book, Xilinx Co. 1989

[4] The Maximalist Handbook, Altera Corp. 1990

[5] Plessey Semiconductor ERA 60100 Advance Information, 1989

[6] Plus Logic FPGA 2020 Preliminary Data Sheet, 1990

[7] MACH1 and MACH2 Device Families Preliminary Data Sheets, AMD Co., 1990

[8] DERVİŞOĞLU, A., Lojik Devre Tasarımı Ders Notları, İ.T.Ü. Elektrik-Elektronik Fakültesi

ÖZGEÇMİŞ

Volkan SEZER, 1971 yılında İstanbul'da doğdu. İlk öğrenimine 1977 yılında başlayarak 1988 yılında Kayseri Fen Lisesi'nden mezun oldu. Aynı yıl İ.T.Ü. Elektrik-Elektronik Fakültesi, Elektronik ve Haberleşme Mühendisliği Bölümü'ne girdi. 1992 yılında Elektronik ve Haberleşme Mühendisi ünvanını aldı. 1993 yılında aynı Fakülte'nin Elektronik ve Haberleşme Mükendisliği Bölümü'nde Yüksek Lisans öğrenimine başladı.

1994 Şubat ayında İ.T.Ü. Elektrik-Elektronik Fakültesi, Elektronik ve Haberleşme Mühendisliği Bölümü, Devreler ve Sistemler Anabilim Dalı'nda Araştırma Görevlisi olarak çalışmaya başladı.