T.C.
NİĞDE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK - ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

BİR ASANSÖR KONTROL SİSTEMİNİN PETRİ NETLER YARDIMIYLA MODELLENMESİ VE XILINX XC2S200E FPGA'SI İLE GERÇEKLEŞTİRİLMESİ

B. Hakan AKSEBZECİ

Yüksek Lisans Tezi

Danışman
Doç. Dr. Murat UZAM

Temmuz 2005

Fen Bilimleri Enstitüsü Müdürlüğüne:

Bu çalışma jürimiz tarafından ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Başkan : Prof. Dr. Kenan DANIŞMAN (Erciyes Üniversitesi) .....

Üye : Doç. Dr. Murat UZAM (Niğde Üniversitesi, Danışman) .....

Üye : Yrd. Doç. Dr. Halis ALTUN (Niğde Üniversitesi) .....

ONAY:

Bu tez 13 / 07 / 2005 tarihinde, Fen Bilimleri Enstitüsü Yönetim Kurulunca belirlenmiş olan yukarıdaki jüri üyeleri tarafından uygun görülmüş ve Enstitü Yönetim Kurulu'nun 22/07/2005 tarih ve 2005/14-12 sayılı kararıyla kabul edilmiştir.

18/08/2005

Doç. Dr. Meysun İBRAHİM

Enstitü Müdürü

ÖZET

BİR ASANSÖR KONTROL SİSTEMİNİN PETRİ NETLER YARDIMIYLA MODELLENMESİ VE XILINX XC2S200E FPGA'SI İLE GERÇEKLEŞTİRİLMESİ

AKSEBZECİ, Bekir Hakan

Niğde Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik - Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Murat UZAM

Temmuz 2005, 125 Sayfa

Bu çalışmanın amacı, dört katlı bir model asansör için gerekli olan kontrol sisteminin, Petri netler yardımıyla modellenmesi ve bir Xilinx XC2S200E FPGA'sı ile gerçekleştirilmesidir.

Bu çalışmada bir asansör kontrol sistemini incelemek amacıyla, dört katlı bir model asansör kullanılmıştır. Bu model asansör için gerekli olan kontrol sisteminin modellenmesinde Otomasyon Petri netlerin bir alt kümesi olan Basit Otomasyon Petri netlerden (BOPN) faydalanılmıştır. BOPN'lerin tercih edilmesinin sebebi; bir kontrol sisteminin görsel bir biçimde modellenebilmesi ve elde edilen BOPN modellerinin kolaylıkla gerçekleştirilebilmesidir.

Bu çalışmada, model asansörün toplamasız ve toplamalı olarak kontrolü için iki ayrı BOPN modeli tasarlanmıştır. Tasarlanan bu BOPN modelleri, VHDL diline çevrilerek, Xilinx XC2S200E FPGA'sında ayrı ayrı sentezlenmiş ve model asansör başarıyla kontrol edilmiştir.

Anahtar sözcükler: Asansör kontrolü, Petri net, FPGA, VHDL

SUMMARY

MODELLING AN ELEVATOR CONTROL SYSTEM USING PETRI NETS AND ITS IMPLEMENTATION BY MEANS OF A XILINX XC2S200E FPGA

AKSEBZECİ, Bekir Hakan

Niğde University
Institute of Natural and Applied Sciences
Department of Electrical and Electronics Engineering

Supervisor: Assoc. Prof. Dr. Murat UZAM

July 2005, 125 pages

The aim of this work is to model a control system necessary for a four storey elevator model and to implement it by using a Xilinx XC2S200E FPGA.

In this work, in order to study an elevator control system, a four storey elevator model is utilized. To model the control system, necessary for controlling this elevator model, simple automation Petri nets (SAPN), which are a sub-set of automation Petri nets, are used. The first reason for chosing SAPNs is that they allow us to visualize the control system. The second one is that it is easy to implement SAPN models.

In this study, two SAPN models are designed to control the elevator model for two different operation scenarios, namely traditional and collective type. These two SAPN models are then synthesized seperately with a Xilinx XC2S200E FPGA by using VHDL language. Finally, the elevator model is controlled successfully.

Keywords: Elevator control, Petri net, FPGA, VHDL

TEŞEKKÜR

Çalışmam boyunca, yönlendirici ve yol gösterici olan ve değerli görüş, öneri ve deneyimlerini benden esirgemeyen tez danışmanım Doç. Dr. Murat UZAM'a, desteklerinden dolayı Arş. Gör. İ. Burak KOÇ'a, maddi ve manevi desteğini her zaman yanımda hissettiğim değerli aileme en içten teşekkürlerimi sunarım.

İÇİNDEKİLER

ÖZET iii
SUMMARY iv
TEŞEKKÜR v
İÇİNDEKİLER DİZİNİ vi
TABLOLAR DİZİNİ ix
ŞEKİLLER DİZİNİ x
FOTOĞRAFLAR DİZİNİ xiii
KISALTMALAR ve SİMGELER xiv
BÖLÜM I. GİRİŞ 1
BÖLÜM II. BASİT OTOMASYON PETRİ NETLER 6
2.1. Basit Otomasyon Petri Netler (BOPN) 8
2.1.1. Bir BOPN'nin tetiklenmesi 11
2.1.2. Yasaklama oklu (Inhibitor arc) BOPN 12
2.1.3. Yetkileme oklu (Enabling arc) BOPN 13
2.1.4. Temel BOPN modülleri 15
2.1.4.1. Başlangıç değeri atama (Initial marking) 15
2.1.4.2. Bağlama (Join) 15
2.1.4.3. Birleştirme (Merge) 16
2.1.4.4. Çatal (Fork) 17
2.1.4.5. İkilem (Conflict) 17
2.1.4.6. Toggle 18
2.1.4.7. Seçim (Select) 18
2.1.4.8. Zamanlı geçişli BOPN 19
2.1.5. BOPN'lerin farklı gösterim şekilleri 20
BÖLÜM III. XILINX SPARTAN-IIE FPGA SERİSİ ve DIGILENT D2-SB UYGULAMA GELİŞTİRME KARTI 22
3.1. XILINX Spartan-IIE FPGA Serisi 24
3.1.1. Spartan-IIE mimarisi 25
3.1.1.1. Giriş/Çıkış bloğu 26

3.1.1.2. Yapılandırılabilir lojik bloklar 28
3.1.1.3. Blok RAM 30
3.2. Digilent D2-SB Uygulama Geliştirme Kartı 31
3.2.1. JTAG portları 33
3.2.2. Güç kaynakları 33
3.2.3. Osilatörler 34
3.2.4. Buton ve LED 34
3.2.5. Genişletme konnektörleri 34
3.2.6. Sistem hattı 36
BÖLÜM IV. VHDL DONANIM TANIMLAMA DİLİ 37
4.1. Simülasyon 37
4.2. Sentezleme 38
4.3. VHDL'de Kullanılan Temel Yapılar 38
4.3.1. Entity 39
4.3.2. Mimariler 40
4.3.2.1. Davranışsal mimari 40
4.3.2.2. Veri akışı mimarisi 42
4.3.2.3. Yapısal mimari 43
4.3.2.4. Farklı mimarilerin karşılaştırılması 47
4.3.3. Paketler 47
4.3.4. Konfigürasyon 48
4.3.5. Tasarım kütüphaneleri 49
4.4. VHDL Dilinin Şematik Gösterime Göre Üstünlükleri 50
BÖLÜM V. BOPN'LERİN VHDL DİLİ KULLANILARAK XILINX XC2S200E FPGA'SI İLE GERÇEKLEŞTİRİLMESİ 52
5.1. Temel BOPN Modellerinin Xilinx XC2S200E FPGA'sı ile Gerçekleştirilmesi 52
5.1.1. Başlangıç değeri atama (Initial marking) 52
5.1.2. Yasaklama oku (Inhibitor arc) 54
5.1.3. Yetkileme oku (Enabling arc) 55
5.1.4. Bağlama (Join) 56
5.1.5. Birleştirme (Merge) 57
5.1.6. Çatal (Fork) 59
5.1.7. İkilem (Conflict) 60

5.1.8. Toggle 61
5.1.9. Seçim (Select) 62
5.1.10. Zamanlı geçişli BOPN 64
5.2. Geçişlerin Tetiklenmesi 66
5.2.1. Yükselen kenar algılama 66
5.2.2. Düşen kenar algılama 68
5.2.3. Geçişlerin yükselen kenar tetikleme şartı ile tetiklenmesi 69
BÖLÜM VI. DÖRT KATLI MODEL ASANSÖR ve I/O TAMPON KARTI 73
6.1. Model Asansör 73
6.1.1. Ara bağlantı kartı 78
6.2. I/O Tampon Kartı 78
6.2.1. I/O tampon kartı besleme devresi 79
6.2.2. I/O tampon kartı temel giriş devreleri 80
6.2.2.1. Debouncer devresi 81
6.2.3. I/O tampon kartı temel çıkış devreleri 82
BÖLÜM VII. ASANSÖR KONTROLÜ İÇİN ELDE EDİLEN BOPN MODELLERİ 86
7.1. Toplamasız BOPN Modeli 87
7.2. Toplamalı BOPN Modeli 95
BÖLÜM VIII. SONUÇLAR 104
KAYNAKLAR 106
EK-A TOPLAMASIZ ASANSÖR KONTROL SİSTEMİNİ OLUŞTURAN VHDL MODÜLLERİ 108
EK-B TOPLAMALI ASANSÖR KONTROL SİSTEMİNİ OLUŞTURAN VHDL MODÜLLERİ 115

TABLOLAR DİZİNİ

Tablo 3.1. Spartan-IIE serisine ait FPGA'lar 24
Tablo 3.2. XC2S200E FPGA'sının giriş ve çıkışları için desteklenen standartlar 27
Tablo 3.3. Spartan-IIE Blok RAM miktarları 30
Tablo 4.1. Bir bitlik tam toplayıcı için doğruluk tablosu 41
Tablo 4.2. Bir bitlik tam toplayıcının, farklı mimarilerde XC2S200E FPGA'sında kullandığı sistem kaynakları 47
Tablo 6.1. Model asansörde kullanılan sembollerin tanımları 75
Tablo 6.2. I/O tampon kartlarının, D2-SB kartındaki A2 ve C2 konnektörleri ve XC2S200E FPGA'sının pinleri ile olan bağlantıları 85
Tablo 7.1. Toplamasız BOPN modelindeki geçişler ve bu geçişlerin tetikleme şartları 89
Tablo 7.2. Toplamasız BOPN modelindeki mevkilere atanan aksiyonlar 90
Tablo 7.3. Toplamalı kontrol sisteminde kabinin hareketi 96
Tablo 7.4. Toplamalı BOPN modelindeki mevkiler ve bu mevkilere atanan aksiyonlar 99
Tablo 7.5. Toplamalı BOPN modelindeki geçişler ve bu geçişlerin tetikleme şartları 100

ŞEKİLLER DİZİNİ

Şekil 1.1. Tasarlanan asansör kontrol sistemini oluşturan yapılar 4
Şekil 2.1. Bir Petri netin yapısı 7
Şekil 2.2. Bir motorun durumunu gösteren OPN modeli 8
Şekil 2.3. (a) Tipik bir ayrık olay kontrol sistemi (DECS) (b) Basit Otomasyon Petri net (BOPN) (c) Bir DECS'te denetleyici olarak BOPN 9
Şekil 2.4. Bir BOPN'de : (a) ilk işaretleme (b) t_1'in tetiklenmesinden sonraki işaretleme (c) t_2'nin tetiklenmesinden sonraki işaretleme 12
Şekil 2.5. Yasaklama oklu bir BOPN : (a) t_1 geçişi açık değil (b) t_1 geçişi açık değil (c) (t_1 geçişi açık) tetiklemeden önce (d) Tetiklemeden sonra 13
Şekil 2.6. Yetkileme oklu bir BOPN : (a) t_1 geçişi açık değil (b) t_1 geçişi açık değil (c) (t_1 geçişi açık) tetiklemeden önce (d) Tetiklemeden sonra 14
Şekil 2.7. (a) Yetkileme oklu bir BOPN (b) Eşdeğeri (c) İkilem yokken yetkileme oklu bir BOPN (d) İkilem varken, sıradan bir BOPN 14
Şekil 2.8. Başlangıç değeri atama örneği 15
Şekil 2.9. Bağlama (join) modeli 16
Şekil 2.10. Birleştirme (merge) modeli 16
Şekil 2.11. Çatal (fork) modeli 17
Şekil 2.12. İkilem (conflict) modeli 18
Şekil 2.13. Toggle modeli 18
Şekil 2.14. Seçim (select) modeli 19
Şekil 2.15. Zamanlı geçişli bir BOPN (a) İlk işaretleme (b) χ_2 tetikleme şartı gerçekleştiğindeki durum (c) χ_3 tetikleme şartı gerçekleştiğindeki durum (d) zaman gecikmesi (T_d) dolduğundaki durum 20
Şekil 2.16. Bir sınırlı p_1 mevkisinin; (a) kapasite ifadesi ile gösterimi (b) gölge mevkisi kullanılarak gösterimi (c) yasaklama oku kullanılarak gösterimi (d) basitleştirilmiş yasaklama oklu gösterimi 21
Şekil 3.1. Temel Spartan-IIE serisi FPGA bloğu 26
Şekil 3.2. Spartan-IIE giriş/çıkış bloğu (IOB) 28
Şekil 3.3. Spartan-IIE CLB dilimi (her bir CLB içinde iki dilim) 29

Şekil 3.4. Çift-port Blok RAM 30
Şekil 3.5. D2-SB kartı blok diyagramı 32
Şekil 3.6. D2-SB kartındaki sinyal yolları 33
Şekil 3.7. Buton ve LED 34
Şekil 3.8. Genişletme konnektörü pin yerleşimleri 35
Şekil 3.9. Genişletme konnektörlerinin sinyal dağılımı 35
Şekil 4.1. Bir bitlik tam toplayıcı 39
Şekil 4.2. Bir bitlik tam toplayıcının lojik kapılarla gösterimi 42
Şekil 4.3. Tam toplayıcının alt sistemlere ayrılışı 44
Şekil 4.4. Bir bitlik tam toplayıcının iki yarı toplayıcı ve bir OR kapısı ile gösterimi 44
Şekil 4.5. Bir yarı toplayıcının lojik kapılarla gösterimi 45
Şekil 5.1. Bir BOPN için başlangıç değeri atama örneği 53
Şekil 5.2. Yasaklama oku modeli 54
Şekil 5.3. Yetkileme oku modeli 56
Şekil 5.4. Bağlama (join) modeli 57
Şekil 5.5. Birleştirme (merge) modeli 58
Şekil 5.6. Çatal (fork) modeli 59
Şekil 5.7. İkilem (conflict) modeli 60
Şekil 5.8. Toggle modeli 62
Şekil 5.9. Seçim (select) modeli 63
Şekil 5.10. Zamanlı geçişli bir BOPN 64
Şekil 5.11. Yükselen kenar algılamak için kullanılan devre 66
Şekil 5.12. Yükselen kenar algılamak için kullanılan devredeki sinyallerin dalga şekilleri 67
Şekil 5.13. Düşen kenar algılamak için kullanılan devre 68
Şekil 5.14. Düşen kenar algılamak için kullanılan devredeki sinyallerin dalga şekilleri 68
Şekil 5.15. Geçişleri yükselen kenarla tetiklenen bir BOPN modeli 70
Şekil 5.16. Yükselen kenarda tetiklenen başlangıç değer atama modeli 70
Şekil 6.1. Dört katlı model asansör 74
Şekil 6.2. I/O tampon kartı besleme devresindeki a) 3.3 V'luk regüle devresi b) 5 V'luk regüle devresi 79
Şekil 6.3. Bir pine ait giriş devresi 80

Şekil 6.4. Bir anahtarlama gürültüsüne ait dalga şekli 81
Şekil 6.5. Örnek bir giriş sinyali için çıkış sinyali 82
Şekil 6.6. Temel giriş devresi 82
Şekil 6.7. Temel çıkış devresi 83
Şekil 7.1. Gerçekleştirilen toplamasız BOPN modeli 88
Şekil 7.2. Gerçekleştirilen toplamalı BOPN modeli 97

FOTOĞRAFLAR DİZİNİ

Fotoğraf 3.1. Digilent D2-SB uygulama geliştirme kartı 31
Fotoğraf 6.1. Model asansörün 3. katında kullanılan elemanlar 76
Fotoğraf 6.2. Dört katlı model asansörün görünümü 78
Fotoğraf 6.3. Tasarlanan I/O tampon kartı besleme devresi 80
Fotoğraf 6.4. Tasarlanan I/O tampon kartının görünümü 83
Fotoğraf 6.5. Tasarlanan I/O tampon kartlarının, D2-SB kartı ile olan bağlantıları 84

KISALTMALAR ve SİMGELER

KISALTMA/SİMGE	
µC	Microcontroller (Mikrodenetleyici)
BOPN	Basit Otomasyon Petri Net
CAD	Computer Aided Design (Bilgisayar Destekli Tasarım)
CLB	Configurable Logic Block (Yapılandırılabilir Lojik Blok)
CPLD	Complex Programmable Logic Device (Karmaşık Programlanabilir Lojik Cihaz)
DEC	Discrete Event Controller (Ayrık Olay Denetleyicisi)
DECS	Discrete Event Control System (Ayrık Olay Kontrol Sistemi)
DES	Discrete Event System (Ayrık Olay Sistemi)
DLL	Delay Locked Loop
FPGA	Field Programmable Gate Array (Alan Programlanabilir Kapı Dizisi)
FSM	Finite State Machine (Sonlu Durum Makinesi)
HDL	Hardware Description Language (Donanım Tanımlama Dili)
IEEE	Institute of Electrical and Electronics Engineers
IOB	I/O Block (Giriş/Çıkış Bloğu)
ISE	Integrated Software Environment (Tümleşik Yazılım Ortamı)
JTAG	Joint Test Advisory Group
LC	Logic Cell (Lojik Hücre)
LUT	Look Up Table
LVTTL	Low Voltage Transistor-Transistor Logic
OPN	Otomasyon Petri Net
OTP	One Time Programmable
PAL	Programmable Array Logic (Programlanabilir Dizi Mantığı)
PC	Personal Computer (Kişisel Bilgisayar)
PLA	Programmable Logic Array (Programlanabilir Lojik Diziler)

PLC	Programmable Logic Controller (Programlanabilir Lojik Denetleyici)
PLD	Programmable Logic Device (Programlanabilir Lojik Aygıt)
PROM	Programmable Read Only Memory (Programlanabilir ROM)
RAM	Random Access Memory (Rasgele Erişimli Bellek)
ROM	Read Only Memory (Salt Okunabilir Bellek)
SRAM	Static RAM (Statik RAM)
TTL	Transistor Transistor Logic
USB	Universal Serial Bus
VHDL	Very High Speed Integrated Circuit Hardware Description Language (Çok Hızlı Tümleşik Devre Donanım Tanımlama Dili)
p	Mevki
t	Geçiş
χ	Tetikleme Şartı
q	Aksiyon

BÖLÜM I

GİRİŞ

Bilim ve teknolojideki hızlı gelişmeler alışılagelmiş diferansiyel veya fark denklemleriyle açıklanamayan bir çok sistemi de beraberinde getirmiştir. Üretim sistemleri, bilgisayar sistemleri, haberleşme sistemleri vb. bu sistemlere örnek olarak verilebilir. Bu sistemlerin davranışları, çoğunlukla kendi içlerinde işlemekte olan ayrık (discrete) olaylarla belirlenir. Bu sistemlere Ayrık Olay Sistemleri (Discrete Event System - DES) adı verilir. Eş zamanlılık, asenkron işlem yapabilme, olay sürümlülük (event-driven) ve belirsizlik (Non-determinism) ayrık olay sistemlerinin karakteristikleridir.

Ayrık olay sistemlerinin modellenmesinde genellikle iki teknik kullanılır. Bunlar Sonlu Durum Makineleri (Finite State Machines - FSMs) ve Petri netlerdir. Sonlu durum makineleri kullanılarak yapılan modellemelerde durum patlaması (state explosion) sorunu ortaya çıkmaktadır. Sonlu durum makineleri, sadece sıralı işlem yapılan modellemelerde kullanılabilir. Ayrıca çıkarılan modelin grafiksel gösteriminin takip edilebilmesi de güçtür. Petri netler kullanılarak yapılan modellemeler ise, tüm bu olumsuzlukların üstesinden gelmektedir. Kolay anlaşılabilen grafiksel gösterimleri ve matematiksel olarak ifade edilebilmeleri, Petri netlerin en önemli üstünlüklerindendir.

Grafiksel ve matematiksel bir araç olarak Petri netler; ayrık olay sistemlerinin modellenmesinde, analizinde, tasarımında ve kontrolünde giderek artan bir hızla kullanılmaktadır. Petri netler, Alman matematikçi Carl A. Petri'den sonra bu ismi almıştır (Petri, 1962). O zamandan bu yana çeşitli sistemlerin modellenmesinde Petri netlerden faydalanılmaktadır. Petri netlerin en önemli özelliği; eş zamanlılık, senkronizasyon ve kaynak paylaşımını içeren durumların modellenmesini ve göz önünde canlandırılmasını mümkün kılmasıdır.

İmalat sistemleri kompleks hale geldikçe, ayrık olay kontrol sistemlerinin (Discrete Event Control System - DECS) tasarımı için etkili bir otomasyon aracına

duyulan ihtiyaç çok daha önemli hale gelmektedir. Petri netler bu tip tasarım çalışmasını kolaylaştırmak için kullanılabilecek en uygun model olarak ortaya çıkmaktadır. Normal Petri netler sensörleri ve aksiyonları içermediğinden, Petri net kavramları bu faaliyetleri içeren Otomasyon Petri netlere dönüştürülür. Otomasyon Petri netler, normal Petri netlerde geçişlere sensör sinyalleri yerleştirmek ve mevkilere aksiyonlar atamak yoluyla elde edilir.

Otomasyon Petri netler kontrol edilecek sistem için gerekli kontrol mekanizmasının modelini temsil ederler [1]. Bu kontrol mekanizmasının gerçekleştirilmesi için günümüzde yaygın olarak kullanılan aygıtlar kişisel bilgisayarlar (PC), mikrodenetleyiciler (μC) ve programlanabilir lojik denetleyicilerdir (PLC). Bu adı geçen aygıtlarda gerçekleştirilen otomasyon Petri net tabanlı kontrol programları, yapıları gereği sıralı bir şekilde çalışmak zorundadır ve ayrıca program tarama süreleri kullanılan işlemcinin hızıyla sınırlıdır. Bu şekilde çalışan programlar ayrık olay sistemlerinde bulunan asenkron işlem yapabilme karakteristiğini tam olarak yerine getirememektedirler. Çünkü yapıları gereği bu programlar bir saat sinyali ile senkron çalışırlar. Bununla beraber artık günümüzde çok hassas zaman ölçeklerinde hızlı şekilde çalışması gereken sistemler mevcuttur (time-critical-systems). Başka bir deyişle; bir kişisel bilgisayar, bir mikrodenetleyici ya da bir PLC'den daha hızlı çalışacak kumanda sistemlerine ihtiyaç duyulmaktadır. Yüksek hızda çalışan kontrol mekanizmaları elde etmek için PC, μC ve PLC gibi yazılımla çalışan aygıtlar yerine donanımla çalışan aygıtlar araştırma konusu olmuştur. Bu kapsamda Petri netlerin donanımla gerçekleştirilmesi ile ilgili çalışmalar son zamanlarda ortaya konmuştur [2]. Bu yüksek lisans tez çalışmasında [2]'de ortaya konan yöntem, yüksek hız ve asenkron çalışmaya imkan vermesi sebebiyle VHDL (Very High Speed Integrated Circuit Hardware Description Language - Çok Hızlı Tümleşik Devre Donanım Tanımlama Dili) dili kullanılarak bir FPGA (Field Programmable Gate Array - Alan Programlanabilir Kapı Dizisi) ile gerçekleştirilmiştir.

FPGA'lar, programlanabilir lojik ürünlerdir. FPGA'ların sayısal ürün pazarındaki payı, özellikle yüksek kapı miktarlarına sahip olması ve karmaşık fonksiyonların bu yapı içerisinde kolaylıkla gerçekleştirilebilmesi sayesinde, hızlı bir biçimde artmaktadır.

FPGA'yı, mikrodenetleyici ve PLC ile kıyasladığımızda şu üstünlüklere sahip olduğunu görmekteyiz:

- PLC'lerin en yaygın kullanıldığı alanlar, endüstriyel otomasyon sistemleridir. FPGA'lar ise çoğunlukla elektronik cihazlarda kullanılmaktadır. Ancak istenildiği takdirde PLC'lerin kullanıldığı alanlarda da kullanılabilirler.
- FPGA'lar, mikrodenetleyiciler ve PLC'lerle karşılaştırıldığında kullanıcıya daha fazla serbestlik sağlamaktadırlar. Örneğin kullanıcı FPGA'da istediği özelliklere sahip bir mikroişlemciyi gerçekleştirebilir.
- Mikrodenetleyiciler ve PLC'ler belleklerinde yüklü bulunan programı birinci komuttan başlayarak son program komutuna kadar bütün komutların sırayla yürütülmesi biçiminde gerçekleştirirler. FPGA'lar ise hem sıralı (sequential) hem de paralel (concurrent) işlem yapabilme özelliğine sahiptirler.

Günümüzün karmaşık elektronik sistemlerinde, FPGA'ların içerisinde bulunan mantık devrelerini şematik giriş yöntemiyle tasarlamak kolay değildir. Şematik girişe alternatif olarak, Donanım Tanımlama Dilleri (Hardware Description Languages - HDLs) geliştirilmiştir [3]. Donanım tanımlama dilleri bir elektronik sistemin tanımlanmasında, test edilmesinde ve sentezlenmesinde kullanılırlar. Pek çok donanım tanımlama dillerinin arasında VHDL en yaygın kullanılanıdır [4].

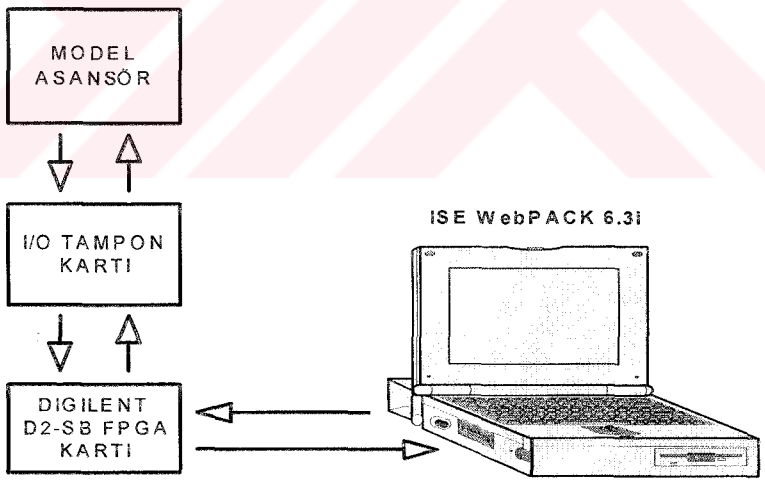
Bu yüksek lisans tez çalışmasında, otomasyon Petri netlerin bir alt kümesi olan Basit Otomasyon Petri netler (BOPN) kullanılmıştır. Basit otomasyon Petri netlerde, her mevkide en fazla bir jeton bulunur. Ayrıca kullanılan okların ağırlık değeri de birdir.

Son yıllarda, Petri netlerin FPGA uygulamalarında kullanılması amacıyla çeşitli akademik çalışmalar yapılmaktadır [5,6,7]. Bu yüksek lisans tez çalışmasında, basit otomasyon Petri netlerin endüstriyel ortamlarda FPGA'lar ile kullanılabileceğini göstermek amacıyla dört katlı bir model asansör kullanılmıştır. Bu model mümkün olduğunca ideale yakın, yani günlük hayatta kullanılan şekle en uygun biçimdedir. Gerçek bir asansörde kullanılan elemanların minimize edilmiş hallerini içermektedir. Örneğin katlarda bekleyen kişilerin güvenliğini sağlamak maksadıyla her katta birer selenoid valf, yine her katta meşgul lambasına ya da kabinin hareketini göstermek

amacıyla kullanılan lambalara karşılık birer tane LED, kapıların açık veya kapalı konumda olduğunu belirtmek maksadıyla kapı anahtarları vb. elemanlar bulunmaktadır.

Model Asansörün kumandası, Digilent D2-SB FPGA uygulama geliştirme kartı kullanılarak gerçekleştirilmiştir [8]. Bu FPGA uygulama geliştirme kartı, üzerinde Spartan-IIE XC2S200E - PQ208 FPGA entegresi bulunan bir karttır [9]. FPGA'lı kartın giriş/çıkış (I/O) gerilimleri model asansörde kullanılan gerilim seviyesinden farklı olduğu için ayrı bir I/O tampon kartı tasarlanmıştır. Şekil 1.1'de, gerçekleştirilen asansör kontrol sistemini oluşturan kısımlar görülmektedir.

Asansör kontrolü, toplamasız ve toplamalı olmak üzere iki farklı şekilde gerçekleştirilmiştir. İki farklı çalışma şekli için iki ayrı BOPN modeli oluşturulmuştur. Daha sonra bu BOPN modelleri VHDL'e dönüştürülerek, iki VHDL programı yazılmıştır. Bu VHDL programları, Xilinx firmasının ISE WebPack 6.3i yazılımı kullanılarak hazırlanmıştır. Yazılan bu programlar, FPGA'ya ayrı ayrı yüklenerek uygulanmıştır.



Şekil 1.1. Tasarlanan asansör kontrol sistemini oluşturan yapılar

Bu tezin bundan sonraki bölümlerinin içerikleri şu şekildedir:

İkinci bölümde Petri netler ve otomasyon Petri netler hakkında açıklamalar yapılmıştır. Daha sonra basit otomasyon Petri netler (BOPN) açıklanmıştır. Bir BOPN'nin tetiklenmesi, yasaklama ve yetkileme oklu BOPN'ler hakkında bilgiler verilmiştir. Ayrıca bazı temel BOPN modelleri incelenmiştir.

Üçüncü bölümde FPGA'lar hakkında genel bir açıklama yapılmıştır. FPGA'ların kullanılmasının sağladığı avantajlar belirtilmiştir. Xilinx XC2S200E'nin ait olduğu Spartan-IIE serisine ait FPGA'ların özellikleri ve mimari yapıları açıklanmıştır. Daha sonra Digilent D2-SB FPGA uygulama geliştirme kartının özellikleri üzerinde durulmuştur.

Dördüncü bölümde FPGA'ların programlanmasında kullanılan VHDL donanım tanımlama dili açıklanmıştır. VHDL'de kullanılan temel yapılar örnekler verilerek belirtilmiştir. VHDL'in şematik gösterimle karşılaştırılması yapılmış ve üstünlükleri ifade edilmiştir.

Beşinci bölümde basit otomasyon Petri netlerin VHDL dili kullanılarak gerçekleştirilmesi üzerinde durulmuştur. İkinci bölümde açıklanan temel BOPN modellerinin VHDL karşılıkları verilmiştir. Bu modellerin geçişlerinin tetiklenmesi için geçişlere tetikleme şartları atanmıştır. Bu tetikleme şartları FPGA'ya dışarıdan giriş olarak verilmektedir. Girişlerdeki yükselen veya düşen kenar değişikliklerini algılamak için ayrı ayrı VHDL program modülleri yazılmıştır. Temel BOPN modelleri ve geçişlerin tetiklenmesi için yazılan VHDL modüllerinin yapısal olarak birleştirilmesi açıklanmıştır.

Altıncı bölümde bu yüksek lisans tez çalışmasında kullanılan dört katlı model asansör hakkında bilgi verilmiştir. Daha sonra model asansör ile Digilent D2-SB kartı arasındaki sinyal seviyelerini düzenleyen I/O tampon kartı hakkında açıklamalar yapılmıştır.

Yedinci bölümde dört katlı model asansörün toplamasız ve toplamalı olarak kontrolü için tasarlanan BOPN modelleri incelenmiştir.

Sekizinci ve son bölümde, elde edilen sonuçlar hakkında bilgiler verilmiştir.

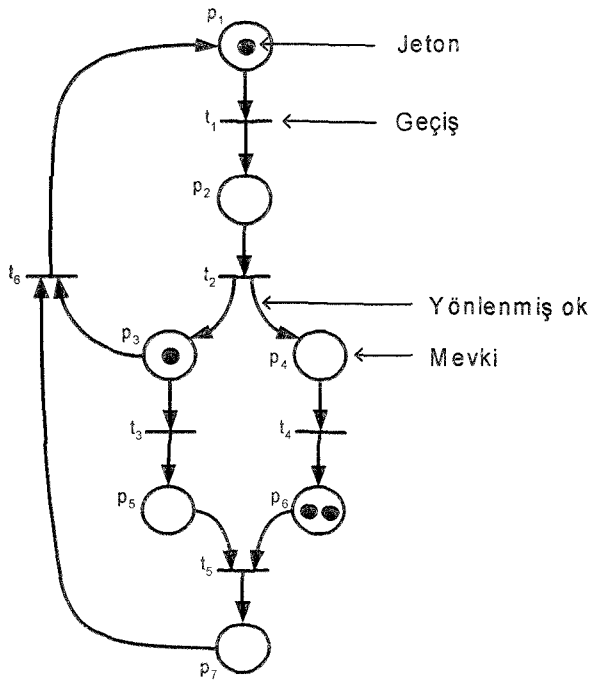
BÖLÜM II

BASİT OTOMASYON PETRİ NETLER

Petri netler, Alman matematikçi Carl A. Petri'nin 1962 yılında hazırladığı, doktora çalışmasında tanımlamış olduğu, matematiksel modelden dolayı bu ismi almıştır. O zamandan bu yana imalat, haberleşme, bilgisayar vb. sistemlerin modellenmesinde Petri netlerden yararlanılmaktadır [10,11,12]. Petri netler ayrık olay sistemlerinin modellenmesinde, analizinde, tasarımında ve kontrolünde giderek artan bir şekilde kullanılmaktadır [13]. Petri netler eş zamanlılık, senkronizasyon ve kaynak paylaşımını içeren durumların modellenmesini ve göz önünde canlandırılmasını mümkün kılmaktadır [14].

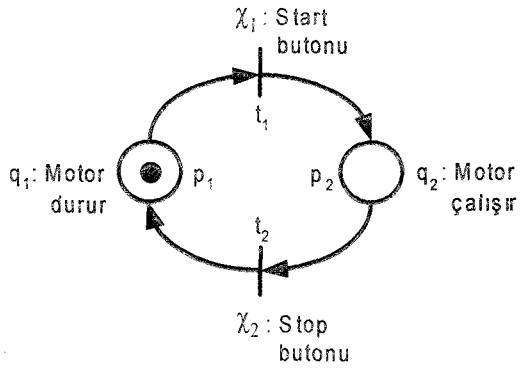
Belirli bir sistemin nasıl çalıştığını açıklamaya yardımcı olması açısından, Petri netler grafiksel olarak gösterilebilir. Bir Petri net grafiğinde, mevkileri ve geçişleri göstermek için sırasıyla daireler ve çubuklar kullanılır. Mevkiler ve geçişler, birbirlerine yönlenmiş oklarla bağlıdırlar. Ayrıca işaretli bir Petri nette, mevkilerde jetonlar bulunur. Jetonlar oklar boyunca hareket ederler ve netlerdeki akışları geçişlerle kontrol edilmektedir. Petri netlerle modellenen bir sistemin durumu, mevkilerde jeton bulunup bulunmaması ile belirlenir. Şekil 2.1'de bir Petri netin yapısı görülmektedir.

Bir Petri netin çalışması, jetonların netteki sayısı ve dağılımıyla kontrol edilir. Bir geçiş, giriş mevkilerinin her biri en az bir jeton içerdiği zaman tetiklemeye açıktır. Şekil 2.1'deki Petri nette, t_1 ve t_3 geçişlerinin tetiklemeye açık olduğu görülmektedir. Bir geçiş, giriş mevkilerinin her birinden bir jeton alınıp, çıkış mevkilerinin her birine bir jeton depolanmasıyla tetiklenir. Geçişlerin tetiklenmesi, jetonların nette akmasına neden olur.



Şekil 2.1. Bir Petri netin yapısı

Normal Petri netler tetikleme şartları ve aksiyonlar içermediğinden, Petri net kavramları bu ifadeleri içeren Otomasyon Petri netlere (OPN) dönüştürülür. Otomasyon Petri netler, normal Petri netlerde geçişlere tetikleme şartları ve mevkilere aksiyonlar atamak yoluyla elde edilir. Örneğin Şekil 2.2'deki Otomasyon Petri net modeli, bir motorun durumunu göstermektedir. Başlangıçta p_1 mevkisinde jeton bulunmasından dolayı, q_1 aksiyonu gerçekleştirilmekte ve motor durmaktadır. Start butonuna basıldığında, χ_1 tetikleme şartı gerçekleşir ve t_1 geçişi tetiklenir. Bu durumda p_1'de bulunan jeton p_2 mevkisine depolanır. p_2 mevkisinde jeton bulunduğu müddetçe, q_2 aksiyonu gerçekleştirilir ve motor çalışır. Stop butonuna basıldığında, χ_2 tetikleme şartı gerçekleşir ve t_2 geçişi tetiklenir. Bu durumda p_2'de bulunan jeton p_1 mevkisine depolanır. p_1 mevkisinde jeton bulunduğu müddetçe, q_1 aksiyonu gerçekleştirilir ve motor durur.



Şekil 2.2. Bir motorun durumunu gösteren OPN modeli

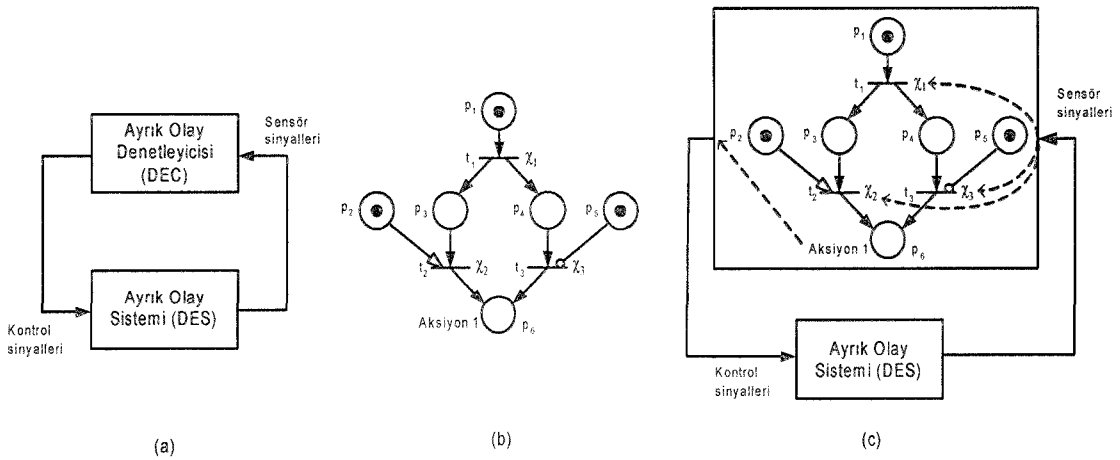
Otomasyon Petri netler, kontrol edilecek sistemin gerekli kontrol mekanizmasını modellemek için kullanılır. OPN'ler, ilk olarak [1]'de tanımlanmıştır. Basit Otomasyon Petri netler (BOPN), OPN'lerden farklı olarak her mevkide en fazla bir jeton içerirler ve kullanılan okların ağırlık değeri de birdir. Bu bölümde BOPN'ler hakkında bilgi verilmiştir. Bir BOPN'nin tetiklenmesi anlatılmıştır. Ayrıca yasaklama ve yetkileme oklu BOPN'ler ve temel BOPN modelleri açıklanmıştır.

2.1. Basit Otomasyon Petri Netler (BOPN)

Bu yüksek lisans tez çalışması kapsamında Basit Otomasyon Petri netler, ayrık olay kontrol sistemlerinin (Discrete Event Control Systems - DECS) tasarımı için yeni bir metot olarak önerilmektedir. Tipik bir ayrık olay kontrol sistemi Şekil 2.3 (a)'da görülmektedir. Kontrol edilecek bir ayrık olay sistemi (Discrete Event System – DES) ve bir ayrık olay denetleyicisinden (Discrete Event Controller – DEC) oluşmaktadır. Sensör sinyalleri DES'ten DEC'e girişler olarak kabul edilmektedir ve kontrol sinyalleri DEC'ten DES'e çıkışlar olarak düşünülmektedir. DEC'in esas fonksiyonu arzu edilen DES işlemini yönetmek ve yasaklanmış işlemlerden kaçınmaktır. Bunu yapmak için DEC sensör sinyallerini işler ve daha sonra DES'i kontrol sinyalleri ile arzu edilen şartlara uymaya zorlar.

DEC'leri tasarlamak için Petri netler kullanılabilir. Ancak daha önce de belirtildiği gibi normal Petri netler aksiyonlar ve sensörler içermez. Bu yüzden, genişletilmiş bir Petri net yapısında hem aksiyonları hem de sensörleri kullanabilen, Petri net tabanlı denetleyicileri (Basit Otomasyon Petri neti) tanımlamak gereklidir. Bu amaçla Şekil 2.3 (b)'de bir BOPN görülmektedir. BOPN'de sensör sinyalleri, geçişlerde

tetikleme şartları olarak kullanılabilir. Mevkilere atanan aksiyonlar da, kontrol sinyalleri olarak kullanılabilir. Bu ilave özelliklerle ayrık olay kontrol sistemlerini tasarlamak mümkündür. Şekil 2.3 (c)'de bir BOPN'nin bir DECS'te nasıl DEC olarak kullanıldığı görülmektedir.



Şekil 2.3. (a) Tipik bir ayrık olay kontrol sistemi (DECS) (b) Basit Otomasyon Petri net (BOPN) (c) Bir DECS'te denetleyici olarak BOPN

Bir BOPN aşağıdaki gibi tanımlanabilir:

BOPN = (P, T, Pre, Post, In, En, χ, Q, M_0)

Burada,

- P = $\{p_1, p_2, ..., p_n\}$ Sonlu sayıda boş olmayan mevkiler kümesidir.
- T = $\{t_1, t_2, ..., t_m\}$ Sonlu sayıda boş olmayan geçişler kümesidir. (P Y T = Ø ve P I T = Ø)
- Pre: (P×T) → {0,1} mevkilerden geçişlere yönlenmiş sıradan okları ifade eden bir giriş fonksiyonudur.
- Post: (T×P) → {0,1} geçişlerden mevkilere yönlenmiş sıradan okları ifade eden bir çıkış fonksiyonudur.
- In: (P×T) → {0,1} mevkilerden geçişlere olan yasaklama oklarını ifade eden bir yasaklama giriş fonksiyonudur.
- En: (P×T) → {0,1} mevkilerden geçişlere olan yetkileme oklarını ifade eden bir yetkileme giriş fonksiyonudur.
- χ = $\{\chi_1, \chi_2, ..., \chi_m\}$ geçişlere atanmış, sonlu sayıda boş olmayan tetikleme şartları kümesidir.

- $Q = \{q_1, q_2, ..., q_n\}$ mevkilere atanabilir, sonlu sayıda aksiyonlar kümesidir.
- $M_0 : P \rightarrow \{0,1\}$ ilk işaretlemedir.

Bir BOPN iki tip düğümden oluşur; dairelerle (○) gösterilen mevkiler ve çubuklarla (—) gösterilen geçişler. BOPN'de üç tip ok kullanılır. Bunlar yönlenmiş bir okla (—►) gösterilen sıradan oklar, sonu daire olan bir okla (—○) gösterilen yasaklama okları ve son olarak, ucu boş yönlenmiş bir okla (—▷) gösterilen yetkileme oklarıdır. Yetkileme okları ve yasaklama okları sadece mevkileri geçişlere bağlarken, yönlenmiş sıradan oklar mevkileri geçişlere ve geçişleri de mevkilere bağlarlar. Mevkiler sistemin durumunu ve geçişler de tetikleme şartlarını simgelerler. Her geçiş, geçişin ön koşullarını ve sonraki koşullarını gösteren bir giriş ve çıkış mevkileri kümesine sahiptir. Mevkilere atanmış aksiyonlar (Q), impuls aksiyonları ya da seviye aksiyonları olabilir. Impuls aksiyonları bir geçiş tetiklendiğinde aktiftir ve seviye aksiyonları mevkide bir jeton bulunduğu sürece aktiftir. Bir mevkiye birden çok aksiyon atanmış olabilir. BOPN'deki tetikleme şartları, sensör sinyalleri gibi dış olaylar olarak kabul edilmektedir. Bir *t* geçişiyle ilişkilendirilmiş olan bir χ tetikleme şartı, bir Boolean değişkenidir ve ilgili *t* geçişinin tetiklenmesine izin verilmediği bir durumda 0 olur veya ilgili geçişin tetiklenmesine izin verildiği bir durumda 1 olur. BOPN'nin işareti, her mevkideki jeton sayılarıyla gösterilmektedir. Jetonlar siyah noktalarla (●) gösterilmektedir. Jetonların mevkiler arasındaki hareketi BOPN'nin gelişimini gösterir ve açık geçişlerin tetiklenmesiyle yapılır. Jetonların akışını yönetmek için aşağıdaki kurallar kullanılmaktadır.

Geçişin tetiklemeye açık olması durumu : BOPN'de, bir geçişin tetiklenmek için açık olduğunu tanımlayan esas olarak üç kural vardır.

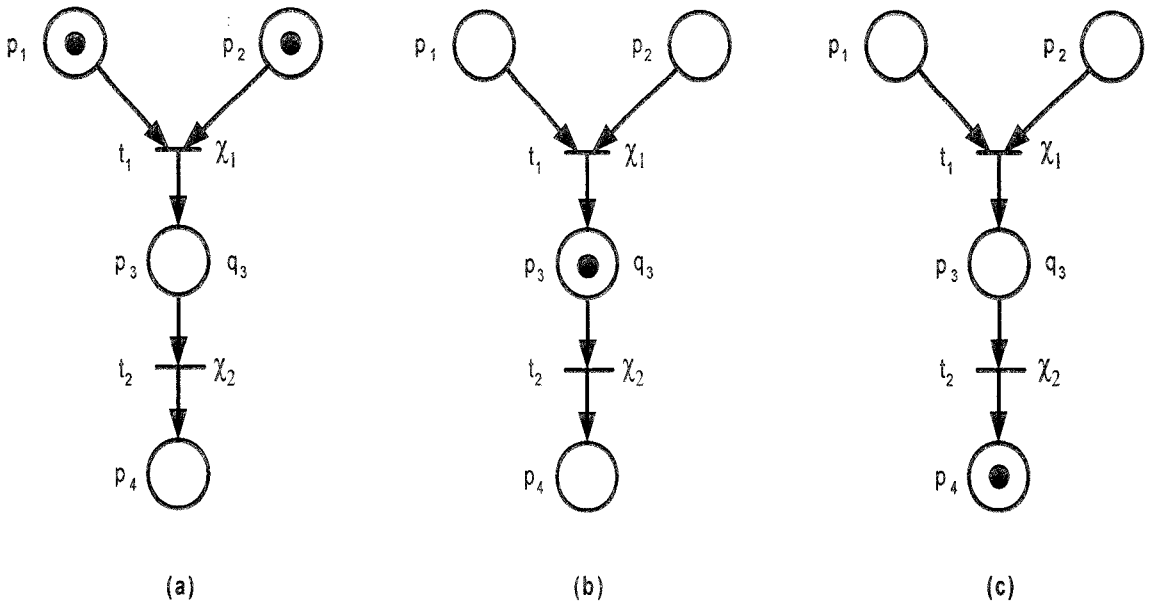
1-) Eğer bir *t* geçişinin giriş mevkisi geçişe yönlenmiş sıradan bir okla bağlıysa giriş mevkisi p, 1 jeton içerdiğinde, *t* geçişi tetiklemeye açıktır denir.
2-) Eğer bir *t* geçişinin giriş mevkisi geçişe bir yasaklama okuyla bağlıysa giriş mevkisi p, jeton içermiyorsa, *t* geçişi tetiklemeye açıktır denir.
3-) Eğer bir *t* geçişinin giriş mevkisi geçişe bir yetkileme okuyla bağlıysa giriş mevkisi p, 1 jeton içerdiğinde, *t* geçişi tetiklemeye açıktır denir.

Bir geçiş, bu giriş oklarının birden fazlasına sahipse, geçişin tetiklemeye açık olması durumu, bu kuralların tümü göz önüne alınarak analiz edilmelidir.

Tetikleme şartları : BOPN'de tetiklemeye açık bir t geçişi, t'nin harici tetikleme şartı olan χ'e bağlı olarak tetiklenebilir ya da tetiklenmeyebilir. Bu tetikleme şartları, bir sensör sinyalinin pozitif seviyesi, sıfır seviyesi veya seviye değişimine bağlı olabilir. Seviye değişiminde, sensör sinyalinin yükselen kenar veya düşen kenar değişimine göre tetikleme şartı gerçekleşir. Genellikle bir tetikleme şartı olan χ 'AND', 'OR' ve 'NOT' lojik operatörleriyle birden çok sensör sinyali içerebilir. Birden çok sensör sinyaliyle ilgilenildiğinde, tetikleme şartlarının lojik operatörleri dikkate alınır. χ = 1 olduğu özel durumda, t geçişi tetiklemeye açık olduğu her zaman tetiklenmesine izin verilmektedir. Tetiklemeye açık bir t geçişi tetiklendiğinde, her p_i giriş mevkisinden bir jeton eksilir. Aynı anda her p_o çıkış mevkisine bir jeton yerleştirilir. Tetiklemeye açık bir t geçişinin tetiklenmesinin, t geçişine sadece yetkileme veya yasaklama oklarıyla bağlı olan giriş mevkilerinin işaretlerini değiştirmediğine dikkat edilmelidir.

2.1.1. Bir BOPN'nin tetiklenmesi

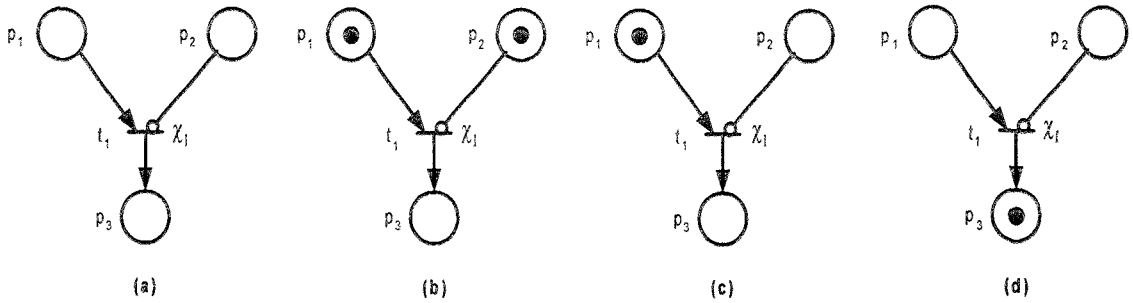
Bir BOPN'nin tetiklenmesi Şekil 2.4'te görülmektedir. Burada dört mevki, P= $\{p_1, p_2, p_3, p_4\}$ ve iki geçiş T = $\{t_1, t_2\}$ vardır. İlk olarak Şekil 2.4 (a)'da görüldüğü gibi, t_1 geçişi tetiklemeye açıktır. Çünkü $M(p_1) = 1$, $Pre(p_1, t_1) = 1$ ve $M(p_2) = 1$, $Pre(p_2, t_2) = 1$'dir. t_2 geçişi tetiklemeye açık değildir. Çünkü $M(p_3) = 0$, $Pre(p_3, t_2) = 1$'dir. Bu durumda $\chi_1 = 1$ olduğunda, t_1 geçişi tetiklenir ve p_1 ve p_2 mevkilerinin her birinden bir jeton alır ve Şekil 2.4 (b)'de görüldüğü gibi p_3 mevkisine bir jeton depolar. p_3 mevkisinde jeton bulunduğu müddetçe p_3'e bağlı q_3 aksiyonu gerçekleştirilir. Bu durumda t_2 geçişi tetiklemeye açıktır. Çünkü $M(p_3) = 1$, $Pre(p_3, t_2) = 1$'dir. Böylece $\chi_2 = 1$ olduğunda, t_2 geçişi tetiklenir ve p_3 mevkisinden bir jeton alır ve Şekil 2.4 (c)'de görüldüğü gibi p_4 mevkisine bir jeton depolar.



Şekil 2.4. Bir BOPN'de : (a) ilk işaretleme (b) t_1'in tetiklenmesinden sonraki işaretleme (c) t_2'nin tetiklenmesinden sonraki işaretleme

2.1.2. Yasaklama oklu (Inhibitor arc) BOPN

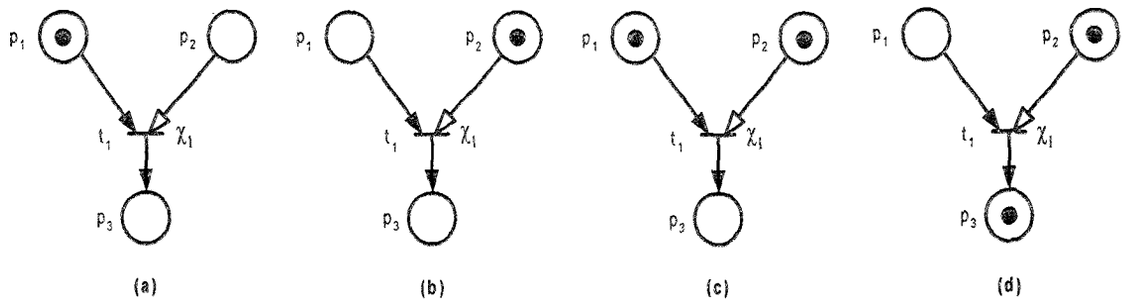
Bir mevkide jeton bulunmadığını test etmek, yasaklama oku kullanılarak yapılmaktadır. Yasaklama oku, bir giriş mevkisini bir geçişe bağlar ve ucu küçük bir daireyle işaretli bir okla gösterilir. Bir giriş mevkisini bir geçişe bağlayan bir yasaklama oku, geçişin sadece giriş mevkisinde jeton bulunmaması halinde tetiklemeye açık olduğunu temsil eder. Bir geçişin tetiklenmesi, bir yasaklama oku ile geçişe bağlı bulunan mevkinin işaretini değiştirmez. Yasaklama oklu bir BOPN Şekil 2.5'te görülmektedir. Burada üç mevki P = {p_1, p_2, p_3} ve bir geçiş T = {t_1} vardır. BOPN'de $p_2 \rightarrow t_1$ oku bir yasaklama okudur, yani: In(p_2, t_1). Şekil 2.5 (a)'daki BOPN'de t_1 geçişi tetiklemeye açık değildir. Çünkü, Pre(p_1, t_1) = 1 ve M(p_1) = 0 olmakla beraber In(p_2, t_1) = 1 ve M(p_2) = 0'dır. Benzer olarak Şekil 2.5 (b)'deki t_1 geçişi tetiklemeye açık değildir. Çünkü In(p_2, t_1) = 1 ve M(p_2) = 1 olmakla beraber Pre(p_1, t_1) = 1 ve M(p_1) = 1'dir. Ancak Şekil 2.5 (c)'deki t_1 geçişi tetiklemeye açıktır. Çünkü Pre(p_1, t_1) = 1 ve M(p_1) = 1 ve In(p_2, t_1) = 1 ve M(p_2) = 0'dır. Bu durumda χ_1 = 1 olduğunda, t_1 geçişi tetiklenir ve p_1 mevkisinden bir jeton alır ve Şekil 2.5 (d)'de görüldüğü gibi p_3 mevkisine bir jeton depolar. t_1 geçişinin tetiklenmesinden sonra, p_2 mevkisinin işaretinin değişmediği görülmektedir.



Şekil 2.5. Yasaklama oklu bir BOPN : (a) t_1 geçişi açık değil (b) t_1 geçişi açık değil (c) (t_1 geçişi açık) tetiklemeden önce (d) Tetiklemeden sonra

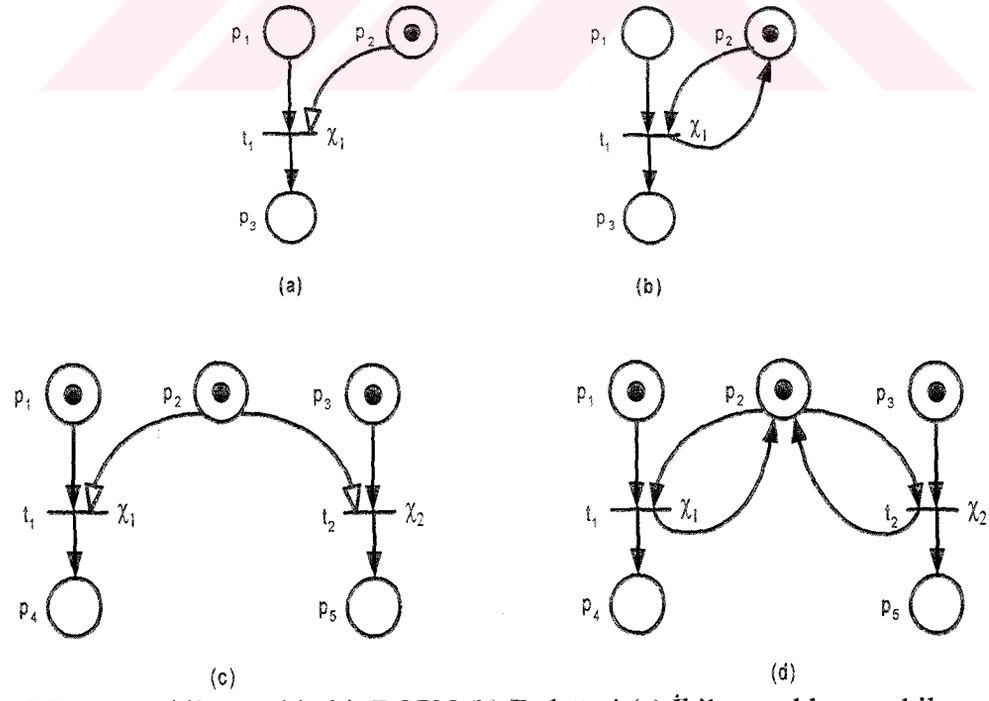
2.1.3. Yetkileme oklu (Enabling arc) BOPN

Bir mevkide jeton bulunduğunu test etmek, yetkileme oku kullanılarak yapılmaktadır. Yetkileme oku, bir giriş mevkisini bir geçişe bağlar ve ucu boş bir okla gösterilir. Bir giriş mevkisini bir geçişe bağlayan bir yetkileme oku, geçişin sadece giriş mevkisinde jeton bulunduğunda tetiklemeye açık olduğunu temsil eder. Bir geçişin tetiklenmesi, bir yetkileme oku ile geçişe bağlı bulunan mevkinin işaretini değiştirmez. Yetkileme oklu bir BOPN Şekil 2.6'te görülmektedir. Burada üç mevki P = $\{p_1, p_2, p_3\}$ ve bir geçiş T = $\{t_1\}$ vardır. BOPN'de $p_2 \rightarrow t_1$ oku bir yetkileme okudur, yani: En(p_2, t_1). Şekil 2.6 (a)'daki BOPN'de t_1 geçişi tetiklemeye açık değildir. Çünkü, En(p_2, t_1) = 1 ve M(p_2) = 0 olmakla beraber Pre(p_1, t_1) = 1 ve M(p_1) = 1'dir. Benzer olarak Şekil 2.6 (b)'deki t_1 geçişi tetiklemeye açık değildir. Çünkü Pre(p_1, t_1) = 1 ve M(p_1) = 0 olmakla beraber En(p_2, t_1) = 1 ve M(p_2) = 1'dir. Ancak Şekil 2.6 (c)'deki t_1 geçişi tetiklemeye açıktır. Çünkü Pre(p_1, t_1) = 1 ve M(p_1) = 1 ve En(p_2, t_1) = 1 ve M(p_2) = 1'dir. Bu durumda χ_1 = 1 olduğunda, t_1 geçişi tetiklenir ve p_1 mevkisinden bir jeton alır ve Şekil 2.6 (d)'de görüldüğü gibi p_3 mevkisine bir jeton depolar. t_1 geçişinin tetiklenmesinden sonra, p_2 mevkisinin işaretinin değişmediği görülmektedir.



Şekil 2.6. Yetkileme oklu bir BOPN : (a) t_1 geçişi açık değil (b) t_1 geçişi açık değil (c) (t_1 geçişi açık) tetiklemeden önce (d) Tetiklemeden sonra

Şekil 2.7 (a)'da görülen yetkileme oklu BOPN modeli, Şekil 2.7 (b)'de görüldüğü gibi sıradan iki okla gösterilebilir. Ancak yetkileme okları, bir BOPN'yi ikileme götürmemek anlamında, sıradan oklardan ayırt edici bir şekilde farklıdır. Örneğin Şekil 2.7 (c)'de görülen BOPN modelinde, t_1 ve t_2 geçişleri ikilemsiz olarak herhangi bir zamanda tetiklenebilirler. Ancak En(p_2, t_1) ve En(p_2, t_2) yetkileme okları, Şekil 2.7 (d)'de görüldüğü gibi sıradan oklarla yer değiştirilirse, potansiyel bir ikilem durumu ortaya çıkmaktadır. Bunun nedeni; χ_1 ve χ_2 tetikleme şartları aynı anda gerçekleştiğinde, p_2'de bulunan jetonun hem t_1 hem de t_2 geçişlerinin tetiklenmesi için kullanılmak istenmesidir.



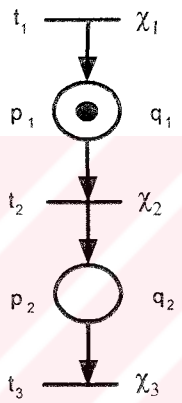
Şekil 2.7. (a) Yetkileme oklu bir BOPN (b) Eşdeğeri (c) İkilem yokken yetkileme oklu bir BOPN (d) İkilem varken, sıradan bir BOPN

2.1.4. Temel BOPN modelleri

Aşağıda bazı temel BOPN modelleri incelenmiştir. Bu modellerden, hem yüksek seviyeli hem de düşük seviyeli sistem modellemesinde faydalanılır.

2.1.4.1. Başlangıç değeri atama (Initial marking)

Bazı BOPN modellerinde, belirli mevkilerde başlangıçta jeton bulunması gerekebilir. Örneğin Şekil 2.8'de görülen BOPN modelinde başlangıçta p_1'de bir jeton depolanmalıdır. Böylece bu BOPN modelinin temsil ettiği sistemde, başlangıçta p_1 mevkisine atanan q_1 aksiyonu gerçekleştirilir.



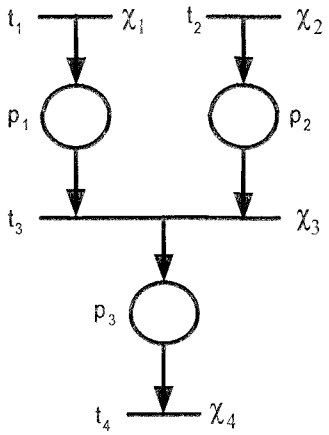
Şekil 2.8. Başlangıç değeri atama örneği

Şekil 2.8'deki BOPN modelinde görüldüğü gibi, her mevkiye bir aksiyon atanmıştır. Bu mevkilerde jeton bulunduğu müddetçe o mevkiye ait aksiyon gerçekleştirilir. Başlangıçta p_1 mevkisinde jeton bulunmasından dolayı q_1 aksiyonu gerçekleştirilir. χ_2 tetikleme şartı gerçekleştiğinde, t_2 geçişi tetiklenerek p_1'de bulunan jeton p_2 mevkisine depolanır. Bu durumda q_2 aksiyonu gerçekleştirilir.

2.1.4.2. Bağlama (Join)

Bağlama modeli, iki ayrı işlemin tek bir senkronize işleme dönüşmesini temsil eder. Şekil 2.9'da bağlama modeli görülmektedir. Burada iki ayrı mevki aynı geçişe bağlanmıştır. Böylece p_1 ve p_2'de birer jeton bulunduğunda, t_3 geçişi tetiklemeye açılır. χ_3 tetikleme şartı gerçekleştiğinde t_3 geçişi tetiklenerek, p_1 ve p_2'deki jetonlar alınır ve p_3'e bir jeton depolanır. Eğer senkronize edilecek işlem sayısı ikiden fazla olursa, bu durumda t_3 geçişine bağlanacak olan giriş mevkilerinin sayısı ihtiyaca göre arttırılır.

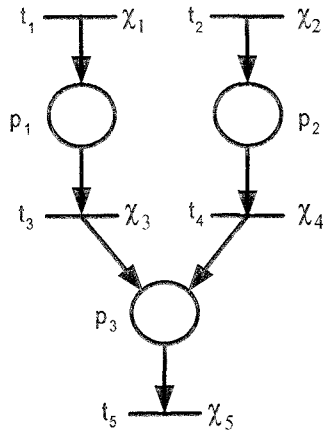
Örneğin bir üretim sisteminde iki farklı parçanın birleşerek tek bir parça haline gelmesi bağlama modeli ile ifade edilebilir.



Şekil 2.9. Bağlama (join) modeli

2.1.4.3. Birleştirme (Merge)

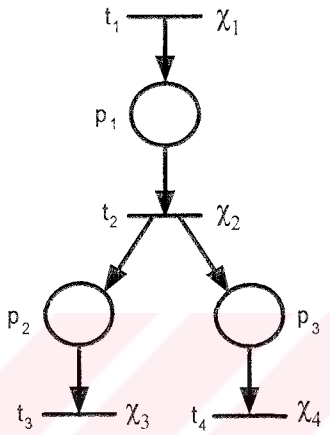
Birleştirme modeli, paralel olarak oluşan olası iki ayrı işlemden herhangi birinin gerçekleşmesi sonucunda üçüncü bir işlemin meydana gelmesini temsil etmektedir. Şekil 2.10'da birleştirme modeli görülmektedir. p_1'de jeton var ise t_3 geçişi, p_2'de jeton var ise t_4 geçişi tetiklemeye açılır. χ_3 veya χ_4 tetikleme şartlarından birinin gerçekleşmesiyle t_3 veya t_4 geçişlerinden hangisi tetiklenirse, bağlı bulunduğu giriş mevkisinden bir jeton alır ve p_3'e bir jeton depolar. Tüm mevkilerin jeton kapasitesi 1 olduğu için, p_3'te jeton olması durumunda t_3 ya da t_4 geçişi tetiklenemez. Eğer paralel olarak çalışacak işlemlerin sayısı ikiden fazla olursa, bu durumda p_3 mevkisine bağlanacak olan geçişlerin sayısı ihtiyaca göre arttırılır.



Şekil 2.10. Birleştirme (merge) modeli

2.1.4.4. Çatal (Fork)

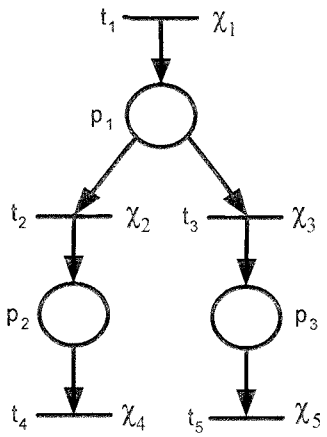
Çatal modeli, aynı anda paralel olarak başlatılan iki işlemi temsil etmektedir. Şekil 2.11'de çatal modeli görülmektedir. p_1'de jeton var ise t_2 geçişi tetiklemeye açılır. χ_2 tetikleme şartı gerçekleşirse t_2 geçişi tetiklenerek p_1'deki jetonu alır, p_2 ve p_3 mevkilerine birer jeton depolar. Eğer aynı anda paralel olarak başlatılacak işlemlerin sayısı ikiden fazla olursa, bu durumda t_2 geçişine bağlanacak olan mevkilerin sayısı ihtiyaca göre arttırılır.



Şekil 2.11. Çatal (fork) modeli

2.1.4.5. İkilem (Conflict)

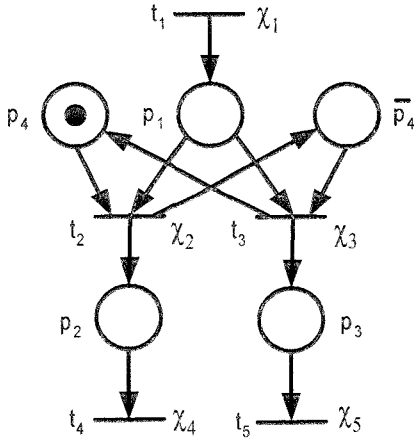
İkilem, bir kaynağın aynı anda iki kullanıcı tarafından talep edilmesi sonucu ortaya çıkar. Petri netler, modellenen bir sistemde ikilem bulunup bulunmadığının tespitinde yararlı olurlar. Eğer sistemde ikilem varsa, bu ikilemin giderilmesi gerekir. Şekil 2.12'de ikilem modeli görülmektedir. p_1'de jeton var ise, t_2 ve t_3 geçişleri tetiklemeye açılır. χ_2 ve χ_3 tetikleme şartlarından hangisi daha önce gerçekleşirse o geçiş tetiklenir ve p_1'deki jeton o geçişin bağlı bulunduğu çıkış mevkisine depolanır. Ancak t_2 ve t_3 geçişleri aynı anda tetiklenirse, bu modelde sorun ortaya çıkmaktadır. Bu nedenden dolayı bu modele "ikilem" adı verilmiştir. Eğer bir işlem sonucunda başlatılabilecek işlemlerin sayısı ikiden fazla olursa, bu durumda t_2 mevkisine bağlanacak olan mevkilerin sayısı ihtiyaca göre arttırılır.



Şekil 2.12. İkilem (conflict) modeli

2.1.4.6. Toggle

Toggle modeli, BOPN'lerde ortaya çıkan ikilem sorununun çözülmesi için önerilmiş bir modeldir. Şekil 2.13'te toggle modeli görülmektedir. Başlangıçta p_4'te bir jeton bulunmaktadır. p_1 ve p_4'te jeton bulunurken, χ_2 tetikleme şartı gerçekleşirse t_2 geçişi tetiklenir. Bunun sonucunda bir jeton p_2'ye, bir jetonda $\overline{p}_4$'e depolanır. p_1'e tekrar bir jeton depolandığında, χ_3 tetikleme şartı gerçekleşirse t_3 geçişi tetiklenir. Bunun sonucunda p_3 ve p_4'e birer jeton depolanır. Bu BOPN modelinde t_2 ve t_3 geçişlerinin aynı anda tetiklenmesi önlenmiştir. Bu modelde sırasıyla önce t_2 daha sonra t_3 geçişlerinin sürekli olarak tetiklenmesi sağlanmıştır.

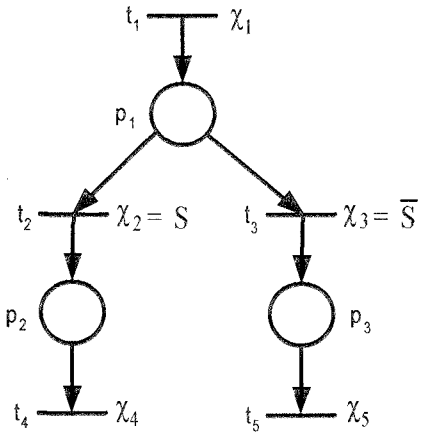


Şekil 2.13. Toggle modeli

2.1.4.7. Seçim (Select)

Seçim modeli, ikilem sorununun çözümü için önerilmiş olan bir başka yöntemdir. Şekil 2.14'te seçim modeli görülmektedir. Bu modelde t_2 ve t_3 geçişlerine

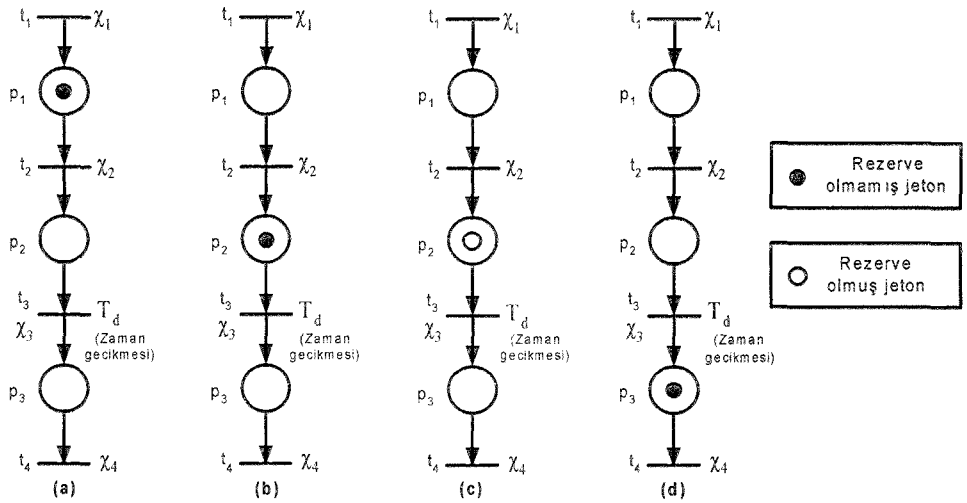
sırasıyla S ve $\overline{S}$ tetikleme şartları atanmıştır. Buna göre p_1 mevkisinde bir jeton bulunması durumunda S tetikleme şartının '1' ya da '0' olmasına göre ya t_2 ya da t_3 geçişlerinden yalnızca bir tanesi tetiklenebilecektir. Böylece ikileme konu olan geçişlerden birisi tetiklenmek üzere seçilmiş olmaktadır.



Şekil 2.14. Seçim (select) modeli

2.1.4.8. Zamanlı geçişli BOPN

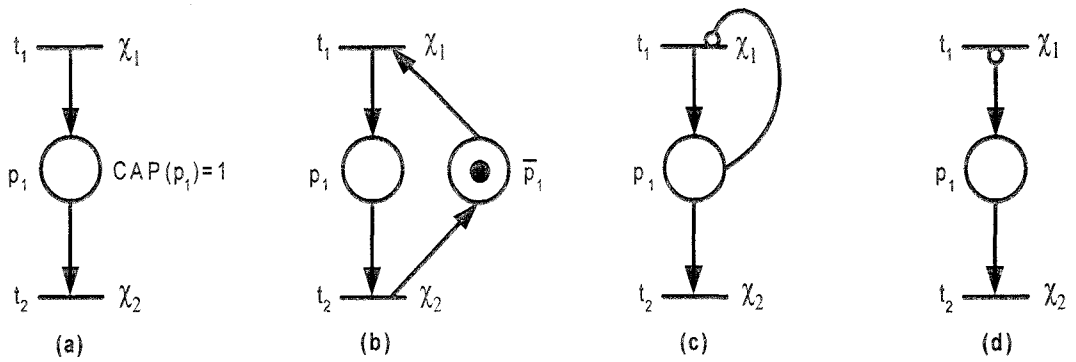
Zaman gecikmesi kullanma ihtiyacı zamanlı Petri netlerin ortaya konulmasını sağlamıştır. Bir Petri nette zaman gecikmesi mevkiye ya da geçişe atanabilir. Burada amaç bir jetonun belirli bir süre sonunda kullanılmasına müsaade edilmesidir. Bu çalışmada zaman gecikmesi elde etmek için, zaman gecikmeleri geçişlere atanmıştır. Şekil 2.15'de zamanlı geçişli bir BOPN modeli görülmektedir. Başlangıçta p_1 mevkisinde bir jeton bulunmaktadır. χ_2 tetikleme şartı gerçekleşirse, t_2 geçişi tetiklenir ve p_1'de bulunan jeton p_2'ye depolanır. p_2'de jeton varken χ_3 tetikleme şartı gerçekleşirse, p_2'deki jeton t_3 zamanlı geçişinin tetiklenmesi için rezerve (reserved) edilir ve gecikme süresi (T_d) başlar. Sürenin dolması ile birlikte, t_3 geçişi tetiklenir ve p_2'deki rezerve edilmiş jeton alınarak, p_3'e rezerve edilmemiş bir jeton depolanır.



Şekil 2.15. Zamanlı geçişli bir BOPN (a) İlk işaretleme (b) χ_2 tetikleme şartı gerçekleştiğindeki durum (c) χ_3 tetikleme şartı gerçekleştiğindeki durum (d) zaman gecikmesi (T_d) dolduğundaki durum

2.1.5. BOPN'lerin farklı gösterim şekilleri

Önceki kısımlarda açıklanan BOPN modellerinin mevkileri 1 sınırlıdır. Yani bir mevkiye en fazla bir jeton depolanabilir. BOPN modellerinde her mevkinin jeton kapasitesi Şekil 2.16 (a)'da görüldüğü gibi belirtilebilir. Bir mevkinin 1 sınırlı olduğunu belirtmek için Şekil 2.16 (b)'de görülen gölge mevkili model de kullanılabilir. Bu model $\mu_i + \overline{\mu_i} = 1$ işaret değişmezi ifadesini sembolize etmektedir. Yani p_i ve $\overline{p_i}$ mevkilerindeki jeton sayılarının toplamı daima 1 olmaktadır. Şekil 2.16 (c)'de görülen model de $In(p_i, t_i)$ yasaklama oku kullanılarak, p_i mevkisinde en fazla bir jeton depolanması sağlanmaktadır. Şekil 2.16 (d)'de görülen model, Şekil 2.16 (c)'deki modelin basitleştirilmiş bir gösterim şeklidir. Bundan sonraki kısımlarda açıklanacak olan BOPN modellerinde, Şekil 2.16 (d)'deki gösterim esas alınacaktır.



Şekil 2.16. Bir sınırlı p_1 mevkisinin; (a) kapasite ifadesi ile gösterimi (b) gölge mevkisi kullanılarak gösterimi (c) yasaklama oku kullanılarak gösterimi (d) basitleştirilmiş yasaklama oklu gösterimi

BÖLÜM III

XILINX SPARTAN-IIE FPGA SERİSİ ve DIGILENT D2-SB UYGULAMA GELİŞTİRME KARTI

Bu yüksek lisans tez çalışmasında Basit Otomasyon Petri netlerin FPGA (Field Programmable Gate Array) ile gerçekleştirilmesi için Xilinx firmasının Spartan-IIE serisine ait 200 bin kapı içeren XC2S200E FPGA'sı kullanılmıştır [9]. Bu amaçla, adı geçen FPGA'yı içeren Digilent firmasının D2-SB FPGA uygulama geliştirme kartı kullanılmıştır [8]. Bu bölümde Xilinx Spartan-IIE FPGA serisi ve Digilent D2-SB FPGA uygulama geliştirme kartı ile ilgili açıklamalar yer almaktadır.

Programlanabilir lojik ürünlerin gelişim sürecini inceleyecek olursak, sırasıyla PLA (Programmable Logic Array), PAL (Programmable Array Logic), PLD (Programmable Logic Device), CPLD (Complex Programmable Logic Device) ve FPGA'lar geliştirilmiştir. PLA ürünleri, hem VE hem de VEYA düzleminin programlanabildiği lojik ürünlerdir. PAL ürünlerinde, sadece VE düzlemi programlanabilir, VEYA düzlemi ise sabittir. PAL'ler PLA'lardan daha hızlıdırlar. PLD'ler, PAL'lerin VE-VEYA düzlemindeki çıkışına programlanabilir kaydedici eklenmesi ile elde edilmiştir. CPLD'ler, büyük ve karmaşık devreler için birbirine bağlanmış PLD'lerden oluşmaktadır. CPLD'ler PAL ve PLD'lerden daha geniş kapasiteye sahiptirler. FPGA'lar programlanabilir lojik ürün sınıfının en gelişmiş üyesidir. FPGA'lar, yüksek kapı miktarlarına sahip programlanabilir lojik ürünlerdir. Ayrıca FPGA'lar, yüksek hızda çalışan ve aynı anda birçok işlemi yapabilecek şekilde programlanabilen entegrelerdir. Karmaşık fonksiyonların FPGA içerisinde kolaylıkla gerçekleştirilebilmesinden dolayı, FPGA'lar günümüzde yaygın bir şekilde kullanılmaktadır.

Farklı üretici firmalar tarafından piyasaya sunulan FPGA'lar, değişik yapılara sahiptirler. İçerdikleri programlanabilen bloklar ve bu bloklar arasındaki bağlantıların yerleşim mimarisine göre çeşitlilik gösterirler. Bunun yanı sıra, bloklar arası bağlantıların hangi teknoloji ile programlanacağı da FPGA'lar için önemli bir

sınıflandırma özelliğidir. Günümüzde kullanılan FPGA'lar iki çeşittir [15].

- **Statik RAM (SRAM):** SRAM FPGA'lar kullanıcı tarafından defalarca programlanabilirler. Bu tür FPGA'lar RAM tabanlı olduğu için, enerjileri kesilip tekrar verildiğinde yeniden programlanmaları gerekmektedir. Bunu önlemek için seri bir PROM kullanılarak program bilgisinin burada depolanması sağlanabilir.
- **OTP (One Time Programmable):** OTP FPGA'lar sadece bir kez programlanabilirler. Bu yüzden seri bir PROM kullanmaya gerek yoktur. Ancak tasarımda yapılacak herhangi bir değişiklikte daha önce programlanmış entegrenin yerine yeni bir entegrenin kullanılması gerekmektedir.

Sayısal sistemlerde FPGA kullanılması sonucunda sağlanan avantajlar şu şekilde sıralanabilir [3]:

- FPGA'lar yüksek hızda çalıştığından aynı anda birçok işlemi yapabilecek şekilde programlanabilirler.
- FPGA tasarım süreci kısadır. Prototipin üretimi için beklemek gerekmez, entegre hazır olarak tasarımcının elinde mevcuttur.
- FPGA'lar defalarca programlanabilmektedir. Tasarımı geliştirmek için yeniden programlama yeterli olmakta, entegreyi değiştirmek gerekmemektedir. Bu durumda FPGA, maliyet ve zamanlama yönünden avantaj sağlamaktadır.
- FPGA ile sağlanan hızlı tasarım ve doğrulama süreçleri, ürünlerin piyasaya daha hızlı sunulmasını sağlamaktadır.
- FPGA'lar üretildikten sonra yapısal olarak test edilirler. Bu nedenle, test için ekstra bir düzeneğe gereksinim duyulmamaktadır. Sadece fonksiyonel olarak, istenilen şekilde çalışıp çalışmadığının incelenmesi yeterli olmaktadır.
- FPGA'lar, firmaların ürün olarak piyasaya sundukları entegrelerdir. Üretici firmaların yıllık FPGA üretimi ve satışı milyonlar seviyesindedir. Üretilen

FPGA sayısının fazla olması, üretim maliyetini düşürmekte, dolayısıyla FPGA kullanıcısına fiyat açısından bir avantaj sağlamaktadır.

- FPGA'lar, entegre değişikliği yapılmadan sistem içinde yeniden programlanabilirler. FPGA'ların bu özelliğinden faydalanarak, değişik zamanlarda değişik fonksiyonları yerine getirecek bir tasarım söz konusu ise, FPGA sistem içinde gerekli zamanlarda yeniden programlanarak kullanılabilmektedir.
- Baskı devre kartında değişiklik yapmaya gerek kalmadan aynı paket tipindeki daha büyük ya da küçük kapasiteli FPGA'lar birbirinin yerine kullanılabilmektedir. Bu da, kart değişikliğinin getireceği maliyet ve zamanlama sorunlarını ortadan kaldırmaktadır.

3.1. XILINX Spartan-IIE FPGA Serisi

Spartan-IIE (1.8V) FPGA serisi, kullanıcıya yüksek performansı çok sayıda lojik kaynağı ve zengin özellik seti ile birlikte sağlar. Spartan-IIE serisi, 50000 ile 600000 arasında değişen sayıda sistem kapılarına sahip yedi farklı üründen oluşmaktadır. Spartan-IIE serisine ait ürünler ve özellikleri Tablo 3.1'de görülmektedir. Ayrıca sistem performansı 200 MHz için desteklenmektedir [9].

Tablo 3.1. Spartan-IIE serisine ait FPGA'lar

FPGA	Lojik Hücre	Sistem Kapı Sayısı (Lojik ve RAM)	CLB Matrisi (RxC)	Toplam CLB	Maksimum I/O Sayısı	Maksimum Farklı I/O Çifti	Toplam Dağıtılmış RAM Bitleri	Blok RAM Bitleri
XC2S50E	1.728	23.000 - 50.000	16 x 24	384	182	83	24.576	32K
XC2S100E	2.700	37.000 - 100.000	20 x 30	600	202	86	38.400	40K
XC2S150E	3.888	52.000 - 150.000	24 x 36	864	265	114	55.296	48K
XC2S200E	5.292	71.000 - 200.000	28 x 42	1.176	289	120	75.264	56K
XC2S300E	6.912	93.000 - 300.000	32 x 48	1.536	329	120	98.304	64K
XC2S400E	10.800	145.000 - 400.000	40 x 60	2.400	410	172	153.600	160K
XC2S600E	15.552	210.000 - 600.000	48 x 72	3.456	514	205	221.184	288K

Özellikleri

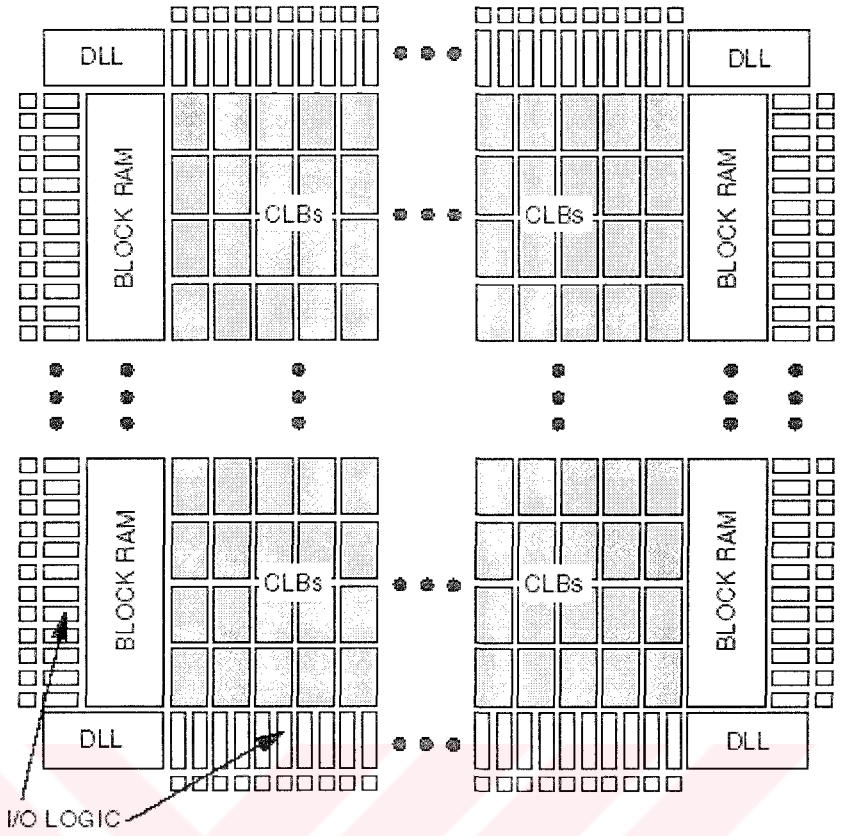
- Virtex-E mimarisi tabanlı, modernize edilmiş özelliklere sahiptir.
- Sınırsız yeniden programlanabilme
- Çok düşük fiyat
- Tamamen PCI uyumlu
- Aynı paket tipine sahip Spartan-IIE FPGA'ların bacakları birbirine uygundur.
- 19 yüksek performanslı arabirim standardını destekler. (LVTTL, LVCMOS2, PCI,... vb.)

3.1.1. Spartan-IIE mimarisi

Şekil 3.1'de temel bir Spartan-IIE FPGA'sına ait blok diyagram görülmektedir. Spartan-IIE FPGA'ların merkezinde CLB'ler (Configurable Logic Blocks) ve kenarlarında programlanabilir giriş/çıkış blokları (IOBs) bulunur. Toplam 4 tane olmak üzere köşelerinde DLL'ler (Delay Locked Loop) bulunmaktadır. FPGA'nın karşılıklı iki kenarında, CLB'lerle IOB'ler arasında blok RAM'lar bulunur.

Şekil 3.1'deki FPGA bloğu beş önemli şekillendirilebilir elementten oluşmaktadır.

- IOB'ler, kılıf pinleri ile dahili lojik arasında bağlantıyı sağlar.
- CLB'ler, daha çok lojik ifadeyi oluşturmak için fonksiyonel elementler sağlar.
- Herbiri 4096 bitten oluşan ayrılmış blok RAM hafızaları içerir.
- Clock DLL'leri clock-yayılma gecikme kompanzasyonu ve clock domeninde kontrol içindir.
- Çok yönlü çoklu-seviyeli alt bağlantı yapısı.



Şekil 3.1. Temel Spartan-IIE serisi FPGA bloğu

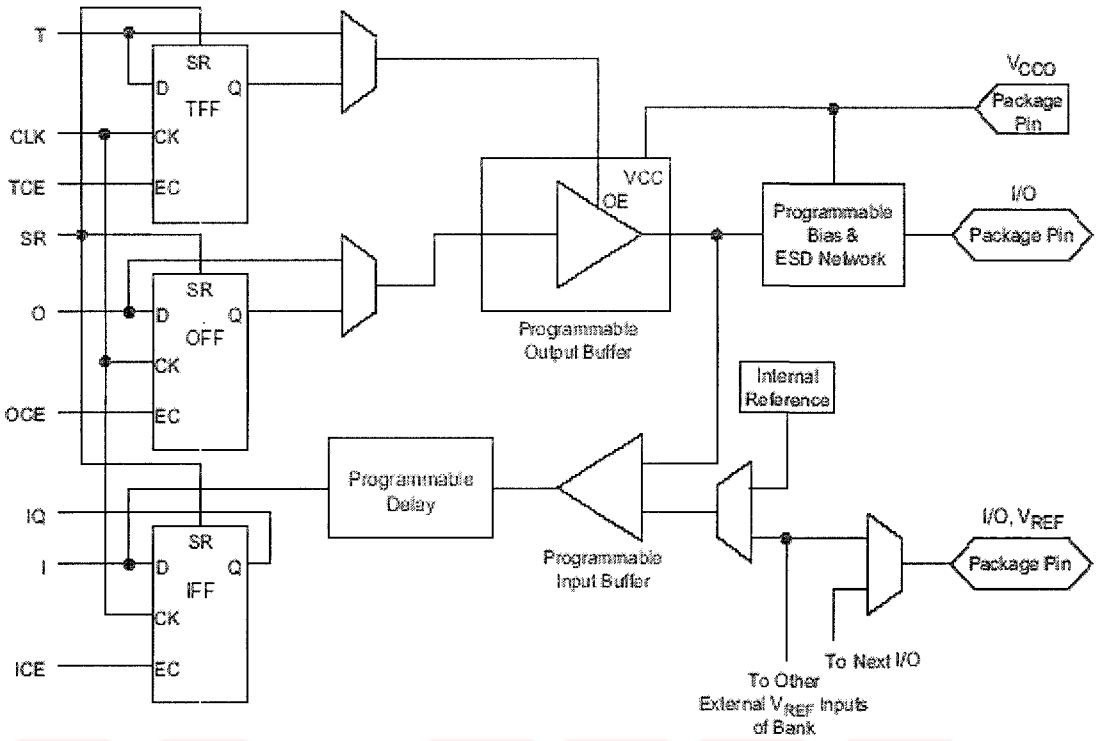
3.1.1.1. Giriş/Çıkış bloğu (IOB)

Spartan-IIE IOB'leri, çeşitli I/O sinyal standartlarını destekleyen girişlere ve çıkışlara sahiptir. Bu yüksek hızlı girişler ve çıkışlar çeşitli durumlardaki hafıza ve hat (bus) bağlantılarını destekleme kabiliyetine sahiptir. Tablo 3.2'de giriş ve çıkışlar için desteklenen standartlar görülmektedir.

Tablo 3.2. XC2S200E FPGA'sının giriş ve çıkışları için desteklenen standartlar

I/O STANDARDI	Giriş Referans Gerilimi (Vref)	Giriş Gerilimi (Vcco)	Çıkış Kaynak Gerilimi (Vcco)	Sonlandırma Gerilimi (Vtt)
LVTTL (2-24 mA)	N/A	3.3	3.3	N/A
LVCMOS2	N/A	2.5	2.5	N/A
LVCMOS18	N/A	1.8	1.8	N/A
PCI(3V, 33 MHz/66 MHz)	N/A	3.3	3.3	N/A
GTL	0.8	N/A	N/A	1.2
GTL+	1.0	N/A	N/A	1.5
HSTL Class I	0.75	N/A	1.5	0.75
HSTL Class III	0.9	N/A	1.5	1.5
HSTL Class IV	0.9	N/A	1.5	1.5
SSTL3 Class I and II	1.5	N/A	3.3	1.5
SSTL2 Class I and II	1.25	N/A	2.5	1.25
CTT	1.5	N/A	3.3	1.5
AGP	1.32	N/A	3.3	N/A
LVDS, Bus LVDS	N/A	N/A	2.5	N/A
LVPECL	N/A	N/A	3.3	N/A

Şekil 3.2'de Spartan-IIE giriş/çıkış bloğunun yapısı görülmektedir. Üç IOB register fonksiyonu, ya kenar-tetiklemeli D-tipi flip-floplar ya da seviye seçici latchler olabilir. Her bir IOB üç register tarafından paylaşılan clock sinyali (CLK) ve her bir register için bağımsız olarak da clock enable sinyallerine sahiptir.

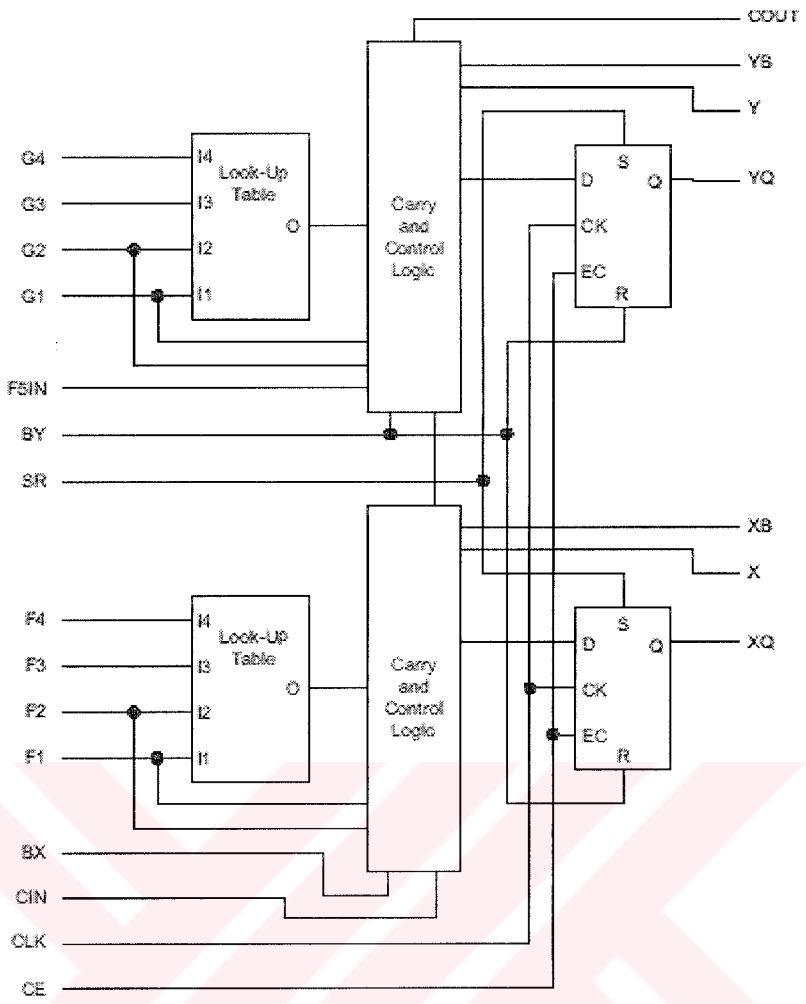


Şekil 3.2. Spartan-IIE giriş/çıkış bloğu (IOB)

CLK ve CE kontrol sinyallerine ek olarak, üç register Set/Reset (SR) sinyallerini de paylaşır. Her bir register için bu sinyal senkron set, senkron reset, asenkron preset veya asenkron clear olarak bağımsız bir şekilde şekillendirilebilir.

3.1.1.2. Yapılandırılabilir lojik bloklar (CLB)

Şekil 3.3'te Spartan-IIE CLB diliminin yapısı görülmektedir. Spartan-IIE CLB temel bloğu, iojik hücreden (Logic Cell, LC) oluşur. Bir LC dört girişli fonksiyon üreteci, elde lojiği ve depolama elementi içerir. Her bir LC'deki fonksiyon üretecinin çıkışı hem CLB çıkışını ve hem de D flip-flop girişini sürer. Her bir Spartan-IIE CLB'si 4 tane LC içerir.



Şekil 3.3. Spartan-IIE CLB dilimi (her bir CLB içinde iki dilim)

Spartan-IIE fonksiyon üreteci 4-girişli look-up tabloları (LUT) şeklinde tasarlanabilir. Her bir LUT 16x1-bit senkron RAM'ı sağlayabilir. Bir dilim içindeki iki LUT 16x2-bit, 32x1-bit senkron RAM veya 16x1-bit dual-port senkron RAM oluşturmak üzere birleştirilebilir. Spartan-IIE LUT tabloları 16-bit shift register oluşturmak için de kullanılabilir.

Spartan-IIE dilimindeki depolama elemanı kenar-tetiklemeli D-tipi flip-floplar ya da seviye seçici latchler şeklinde şekillendirilebilir. Clock ve Clock Enable sinyallerine ek olarak her bir dilim, senkron set ve reset (SR ve BY) sinyallerine sahiptir. SR depolama elementini başlangıç durumuna zorlar. BY ise karşı duruma zorlar. Alternatif olarak, bu sinyaller asenkron çalışacak şekilde şekillendirilebilir.

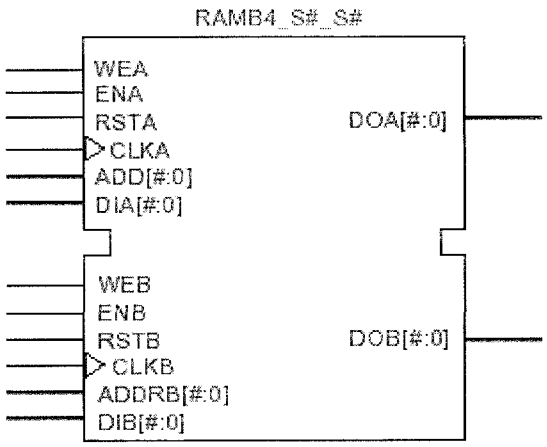
3.1.1.3. Blok RAM

Spartan-IIE FPGA'ları birkaç büyük Blok RAM hafızalarını birleştirir. Blok RAM hafıza blokları sütünlar şeklinde organize edilmiştir. Spartan-IIE FPGA'sı, her biri düşey kenarda bulunmak üzere toplam iki sütun içermektedir. Her hafıza bloğu dört CLB uzunluğundadır. Bir sütunda iki hafıza bloğu bulunduğundan toplam uzunluk 8 CLB kadardır. Tablo 3.3'te Spartan-IIE Blok RAM miktarları görülmektedir.

Tablo 3.3. Spartan-IIE Blok RAM miktarları

Spartan-IIE FPGA	Blok Sayısı	Toplam Blok RAM Bitleri
XC2S50E	8	32K
XC2S100E	10	40K
XC2S150E	12	48K
XC2S200E	14	56K
XC2S300E	16	64K
XC2S400E	40	160K
XC2S600E	72	288K

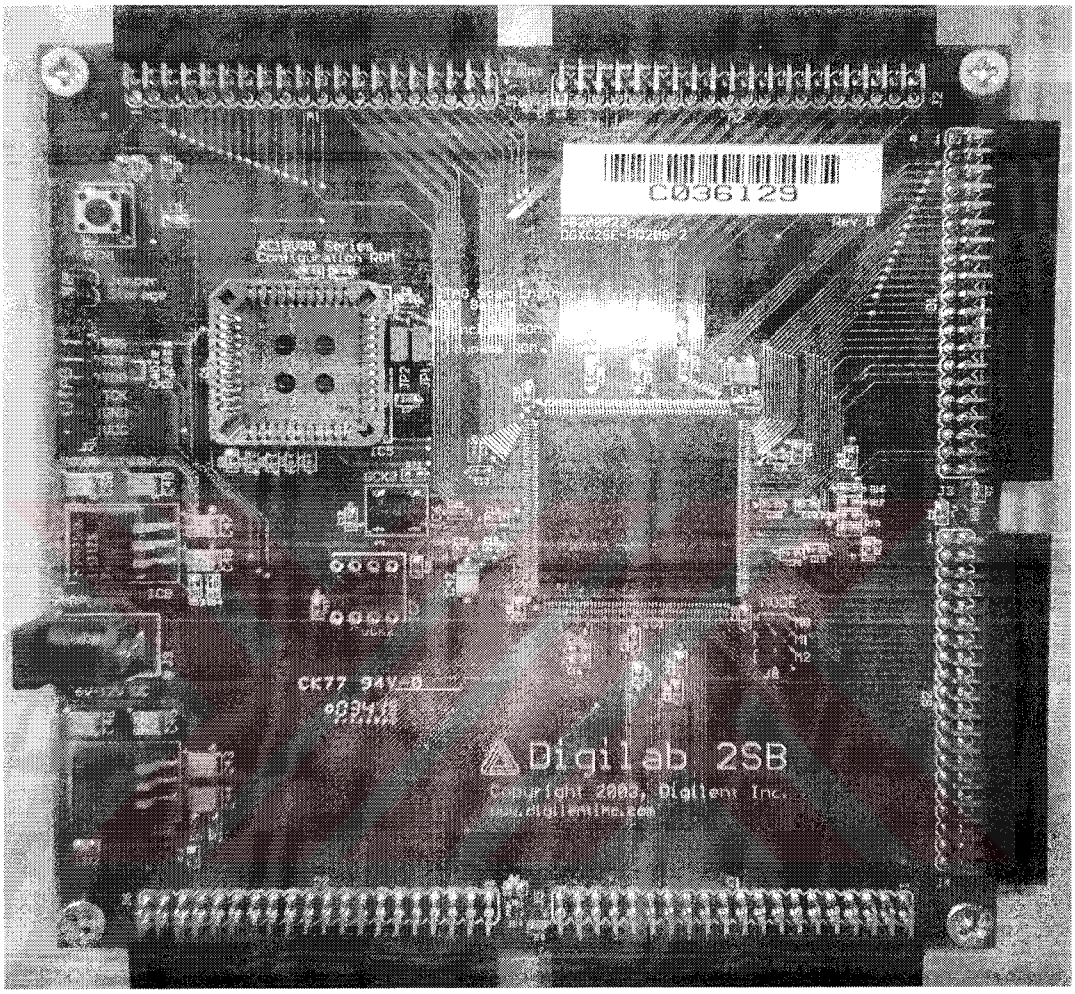
Her bir Blok RAM hücresi, Şekil 3.4'te de görüldüğü gibi, her bir port için bağımsız kontrol sinyalleri ile tam olarak senkron, çift-port 4096-bit'lik RAM dır.



Şekil 3.4. Çift-port Blok RAM

3.2. Digilent D2-SB Uygulama Geliştirme Kartı

Digilent D2-SB uygulama geliştirme kartı, bir Spartan-IIE FPGA'sı içeren bir karttır. Bu kartın resmi Fotoğraf 3.1'de görülmektedir.

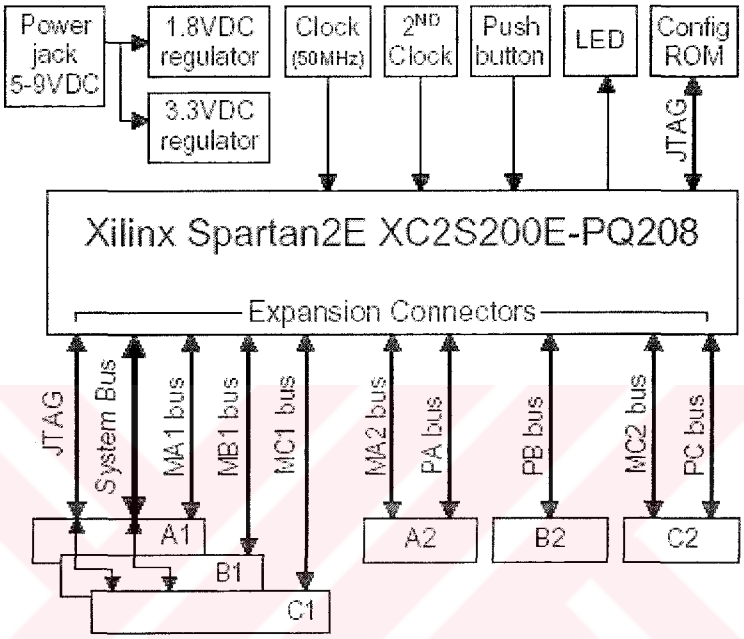


Fotoğraf 3.1. Digilent D2-SB uygulama geliştirme kartı

Bu kartın blok diyagramı Şekil 3.5'te görülmektedir. Kartın özellikleri aşağıda açıklanmıştır [8].

- Bu kart üzerinde Xilinx XC2S200E FPGA'sı bulunmaktadır. Bu FPGA, 200 bin lojik kapı içermektedir ve işlem hızı 200 MHz'dir.
- 143 tane giriş/çıkışa sahiptir ve bunlar 40 pinli genişletme konnektörlerine yönlendirilmiştir.
- Kart üzerinde Flash ROM için bir soket bulunmaktadır.
- 1.8 V ve 3.3 V olmak üzere iki adet 1.5 A'lik güç regülatörü bulunmaktadır.

- Kartta 50 MHz'lik bir osilatör bulunmaktadır. Ayrıca ikinci bir osilatör için de soket bulunmaktadır.
- JTAG (Joint Test Advisory Group) programlama portuna sahiptir.
- Çalışma durumunu gösteren bir LED ve basit giriş/çıkış işlemlerinde kullanılmak üzere bir buton bulunmaktadır.



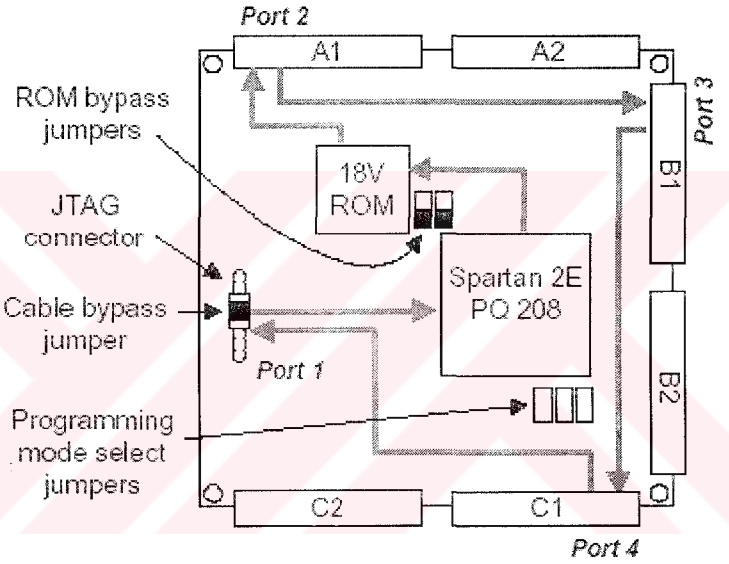
Şekil 3.5. D2-SB kartı blok diyagramı

D2-SB kartı, ücretsiz WebPack'i de içeren Xilinx ISE CAD programları ve bunların tüm versiyonları ile sorunsuz çalışabilecek şekilde tasarlanmıştır. Analog ve dijital giriş/çıkışlara sahip, Ethernet ve USB gibi data portlarının bulunduğu, genişletme kartları D2-SB ile birlikte kullanılabilir.

D2-SB kartındaki altı genişletme konnektörünün her birinde regülesiz kaynak gerilimi (VU), 3.3 V, GND ve 32 adet FPGA'ya giriş/çıkış sinyalleri bulunmaktadır. 18 pinli sistem hattı (system bus) A1, B1 ve C1 konnektörlerine bağlıdır. B genişletme konnektörünün tamamı kullanılmamıştır. Ayrıca JTAG sinyalleri de A1, B1 ve C1 genişletme konnektörlerine bağlanmıştır.

3.2.1. JTAG portları

D2-SB kartı üzerindeki Spartan-IIE FPGA'sını, kullanılan ROM'u ve D2-SB kartına bağlı çevresel kartlardaki programlanabilir aygıtların programlanması JTAG portu tarafından sağlanır. Kart üzerindeki Port 1, 6 pinli JTAG konnektörden oluşmaktadır. Bu konnektöre Digilent'in JTAG 3 kablosu yerleştirilerek programlama işlemi gerçekleştirilir. Ayrıca A1, B1 ve C1 genişletme konnektörleri de JTAG programlama portları olarak kullanılabilir. Bu portlar çift yönlüdürler (bi-directional). Şekil 3.6'da D2-SB kartındaki JTAG sinyal yolları ve programlamada kullanılan jumper ayarları görülmektedir.



Şekil 3.6. D2-SB kartındaki sinyal yolları

3.2.2. Güç kaynakları

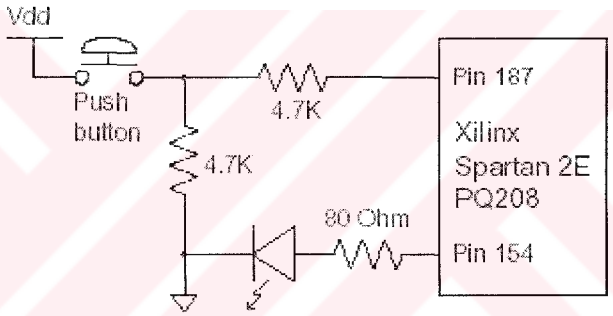
D2-SB kartı iki adet LM317 gerilim regülatörü içermektedir. Birisi Spartan-IIE'nin çekirdek (core) gerilimi için 1.8 V DC'ye ayarlanmıştır. Diğeri de Spartan-IIE'nin giriş/çıkışlarında kullanılmak üzere 3.3 V DC'ye ayarlanmıştır. Kartın besleme gerilimi 6 V DC ile 12 V DC arasında olmalıdır. Genişletme konnektörlerinin 2. pininde bu gerilim bulunmaktadır.

3.2.3. Osilatörler

D2-SB kartı, 50 MHz'lik bir osilatör ve ikinci bir osilatör için de soket içermektedir. 50 MHz'lik osilatör, Spartan-IIE'nin GCLK2 girişine (pin 182) bağlıdır. Diğer osilatör de, Spartan-IIE'nin GCLK3 girişine (pin 185) bağlıdır.

3.2.4. Buton ve LED

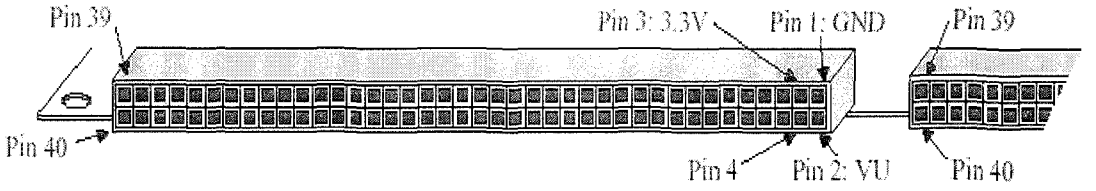
Buton ve LED yardımıyla, basit durum ve kontrol fonksiyonları çevresel kartlara gerek kalmadan gerçekleştirilebilir. Örneğin LED, FPGA'nın çıkış sinyaline bağlanarak, sinyal durumunu göstermede kullanılabilir. Buton da diğer girişlerden bağımsız olarak reset fonksiyonu gibi amaçlarla kullanılabilir. Buton ve LED'in FPGA pinlerine olan bağlantısı Şekil 3.7'de görülmektedir.



Şekil 3.7. Buton ve LED

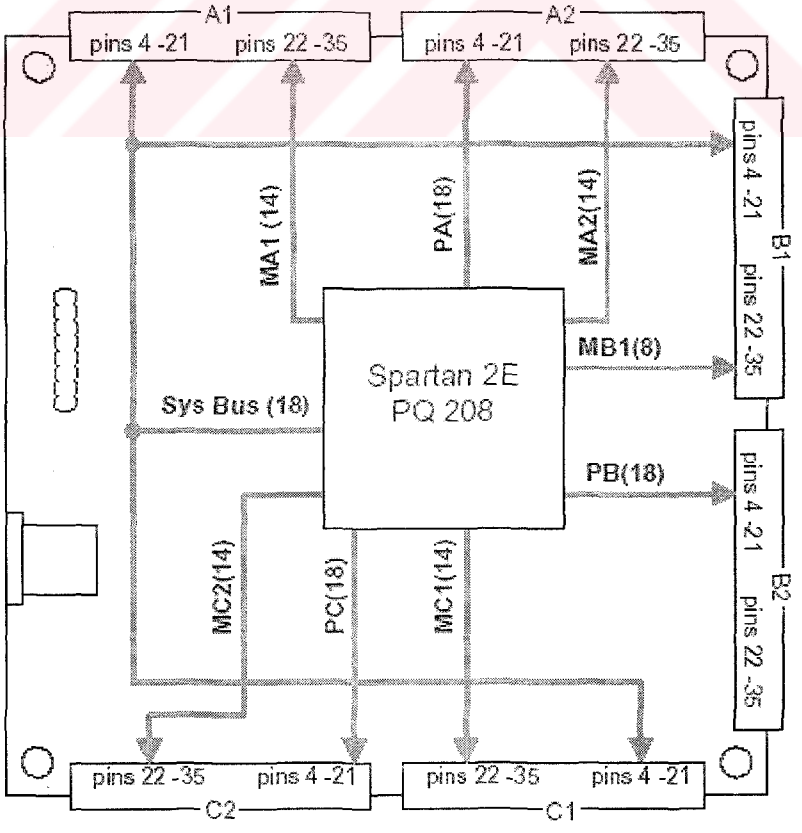
3.2.5. Genişletme konnektörleri

D2-SB kartı üzerinde, A1-A2, B1-B2 ve C1-C2 adları verilen altı adet genişletme konnektörü bulunmaktadır. Bu genişletme konnektörleri 2 × 20 pinlidir. Şekil 3.8'de bir genişletme konnektörünün pin yerleşimleri görülmektedir. Her konnektörün 1. pini GND, 2. pini VU (regülesiz gerilim) ve 3. pini de 3.3 V'tur. Konnektörlerin 4-35 pinleri FPGA'nın giriş/çıkış sinyalleri için ve 36-40 pinleri de JTAG ve/veya clock sinyalleri için ayrılmıştır.



Şekil 3.8. Genişletme konnektörü pin yerleşimleri

Spartan-IIE - PQ208 FPGA'sı sadece 143 giriş/çıkışa sahip olmasına rağmen, genişletme konnektörlerinde 192 sinyal bağlantısı vardır. Bunun nedeni; bazı FPGA sinyallerinin birden fazla konnektöre bağlanmış olmasıdır. A1, B1 ve C1 konnektörlerinin 4-21 pinlerinin hepsi FPGA'nın aynı 18 pinine bağlıdır. Bu 18 pin sistem hattı olarak tasarlanmıştır. Bu 18 pinin dışındaki diğer FPGA sinyalleri, bireysel olarak genişletme konnektörlerine bağlıdır. A2, B2 ve C2 konnektörlerinin 4-21 pinleri çevresel hatlar (peripheral busses) olarak tasarlanmıştır ve bunlara PA, PB ve PC adları verilmiştir. A1, A2, C1 ve C2 konnektörlerinin 22-35 pinleri modül hatları (module busses) olarak belirlenmiştir. Şekil 3.9'da genişletme konnektörlerinin sinyal dağılımı görülmektedir.



Şekil 3.9. Genişletme konnektörlerinin sinyal dağılımı

3.2.6. Sistem hattı

Sistem hattı, özellikle D2-SB kartının diğer genişletme kartları ile haberleşmesinde kullanılmak amacıyla tasarlanmıştır. Sistem hattı, 8 veri hattı, 6 adres hattı, WE (write enable), OE (output enable), CS (chip select) ve CLK (clock) sinyallerinden oluşmaktadır.

BÖLÜM IV

VHDL DONANIM TANIMLAMA DİLİ

VHDL (Very High Speed Integrated Circuit Hardware Description Language), donanımı soyut bir tanımlamadan somut bir yapıya dönüştürmede kullanılan endüstri standardı haline gelmiş bir donanım tanımlama dilidir. Amerika Birleşik Devletleri Savunma Bakanlığı'nın 1970'ler ve 1980'lerin başlarındaki çalışmalarının sonucu olarak ortaya çıkmıştır. Daha sonra VHDL, IEEE tarafından standart bir donanım tanımlama dili (Hardware Description Language - HDL) olarak kabul edilmiştir. VHDL dilinin kökeni, ADA programlama diline dayanmaktadır [16].

VHDL en çok kullanılan yapısal tanımlama dilidir. HDL'ler sayısal sistemin yapısını veya davranışını tanımlamada kullanılır. HDL'de yapılan devre tanımı kullanılarak, bir simülasyon programı ile yazılan tanımlamanın istenilen fonksiyonu sağlayıp sağlamadığı kontrol edilir. Tanımlamanın doğruluğu görüldükten sonra, bir sentezleme programı ile devrenin fiziksel yapısı elde edilir.

HDL tabanlı tasarım yöntemlerinin geleneksel kapı seviyesinde tasarım yöntemlerine göre bir takım üstünlükleri vardır. İlk olarak, tasarımcının daha kısa sürede ve devre tasarımı hakkında çok fazla bilgi sahibi olmadan tasarım yapmasını sağlamaktadırlar. İkinci olarak, aynı tanımlama ile devrenin farklı FPGA'lardaki yapısı elde edilebilmektedir. Devrenin hangi FPGA kullanılarak gerçekleneceği gözönünde bulundurulmaksızın HDL tanımlaması yapılır. Bir sentezleme programından yararlanarak, HDL ile yapılan tanımlama her defasında farklı FPGA'lara uygulanabilir.

4.1. Simülasyon

VHDL'de yazılan bir tasarım tanımı, bir VHDL simülasyon programından faydalanılarak test edilebilir. Tanımlamanın simulasyonunu yapabilmek için simulasyon programına bir takım test girişleri verilir, program daha önceden belirlenen aralıklarla

bu test girişlerini tasarımın modeline uygular ve çıkışları üretir. Bu sonuçlar tasarımcı tarafından gözlenerek modelin istenildiği gibi çalışıp çalışmadığına karar verilir [4].

Simülasyon işlemi tasarımın her aşamasında kullanılabilir. Tasarımın yüksek seviyelerinde yapılan simülasyonda sadece tasarımın fonksiyonel davranışı hakkında bilgi elde edilir. Bu aşamada simülasyon çok hızlıdır, fakat simülasyon sonuçlarından tasarımın gerçek devre elemanları ile çalışması ve zamanlaması konusunda çok fazla bilgi elde edilemez. Daha düşük tasarım aşamalarına gidildikçe simülasyon daha uzun sürecektir. Fakat simülasyon sonuçlarından tasarımın çalışması ve zamanlaması konusunda daha fazla bilgi elde edilir [4].

4.2. Sentezleme

Sentezleme, tasarım tanımlamasının bir seviyeden daha düşük başka bir seviyeye çevrilmesi olarak tanımlanabilir. Bu çevirme işlemi, C programlama dilinde yazılmış bir programın makina diline çevrilmesine benzetilebilir. Sentezleme programına HDL tanımlama verilir. Ayrıca sentezlenmiş devrede bulunması gereken gecikme ve alan özellikleri programa girilir. Verilen özelliklere sahip devre, sentezleme programı ile, tasarımcı tarafından belirtilen teknoloji kütüphanesindeki elemanlar kullanılarak elde edilir. Sentezleme sonuçları, optimal devre şeması, sentezleme sonucunda elde edilen devreden beklenen performans ve sentezlenmiş devrenin kapladığı alandır.

VHDL, karmaşık sistemlerin tasarlanmasında kullanılabilen, çok sayıda dil yapısına sahip güçlü bir donanım tanımlama dilidir. Dolayısıyla burada VHDL'in tüm özelliklerini açıklamak mümkün değildir. Bu nedenle bu bölümde VHDL'de kullanılan temel yapılar ve özellikleri açıklanacaktır.

4.3. VHDL'de Kullanılan Temel Yapılar

VHDL'de kullanılan 4 çeşit temel yapı mevcuttur. Bunlar entity, mimari (architecture), paket (package) ve konfigürasyon (configuration)'dur. Sadece entity ve mimari yapılarının kullanımı zorunludur. Paket ve konfigürasyon yapıları opsiyoneldir.

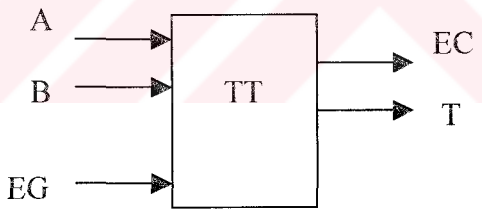
Entity yapısı tasarımın dış bağlantılarının, mimari bölümü ise içeride yapılacak işlemlerin gösterilmesinde kullanılır. Paket'te pek çok tanımlamada ortak olarak kullanılacak genel bilgiler verilir. Konfigürasyon ile yapısal tanımlamada kullanılacak alt sistemlere ait ifadeler belirtilir.

4.3.1. Entity

Entity ifadesi, tasarlanacak olan devrenin giriş/çıkış bağlantılarını tanımlar. Ancak bu bağlantılar arasındaki ilişki konusunda bilgi verilmez. Aşağıda entity ifadesinin yazım şekli görülmektedir.

```
entity entity_ismi is
        Port (giriş/çıkış sinyal isimleri : mod sinyal_türü;
              giriş/çıkış sinyal isimleri : mod sinyal_türü;
               . . .
              giriş/çıkış sinyal isimleri : mod sinyal_türü);
end entity_ismi;
```

Şekil 4.1'de bir bitlik bir tam toplayıcının (TT) bağlantıları görülmektedir. Şeklin altında da bir bitlik tam toplayıcının entity ifadesi belirtilmiştir.



Şekil 4.1. Bir bitlik tam toplayıcı

```
entity TAM_TOPLAYICI is
        Port (A, B, EG : in STD_LOGIC;
              T, EC    : out STD_LOGIC);
end TAM_TOPLAYICI;
```

Port ifadesi, devrenin çevreyle haberleşmesini sağlayan giriş/çıkışları belirtir. Her giriş/çıkış bir mod (in, out, inout, buffer) ve bir sinyal türü (signal type) ile gösterilir. Giriş/çıkışların dört modu şu şekildedir:

- **in:** Bu mod, entity'de giriş uçlarını belirtirken kullanılır.

- **out:** Bu mod, entity'de çıkış uçlarını belirtirken kullanılır. Bu uçlardaki sinyaller, entity'nin bağlı bulunduğu mimari tarafından okunamaz. Ancak başka bir entity tarafından okunabilir.
- **buffer:** Entity'de bu modda tanımlanan bir sinyal çıkış olarak kullanılır ve entity'nin bağlı bulunduğu mimari tarafından okunabilir.
- **inout:** Entity'de bu modda tanımlanan bir sinyal hem giriş hem de çıkış olarak kullanılır. Entity'nin bağlı bulunduğu mimari veya başka bir entity tarafından okunabilir.

4.3.2. Mimariler

VHDL'de, bir mimari her zaman bir entity ile ilişkilidir ve aynı isme sahiptir. Mimaride devrenin girişleri ile çıkışları arasındaki ilişkiler tanımlanır. Tasarlanacak devrenin yapacağı fonksiyon ve/veya yapısı hakkında bilgi verilir.

Bir mimari aşağıdaki yazım şekli kullanılarak tanımlanır:

architecture mimari_ismi **of** entity_ismi **is**
    type tanımlamaları
    signal tanımlamaları
    constant tanımlamaları
    function tanımlamaları
    procedure tanımlamaları
    component tanımlamaları
begin
    eşzamanlı_ifadeler
    . . .
    eşzamanlı_ifadeler
end mimari_ismi;

Her devre farklı mimarilerle tanımlanabilir. Tanımlama davranışsal (behavioral), veri akışı (dataflow) veya yapısal (structural) olmak üzere üç farklı şekilde yapılabilir.

4.3.2.1. Davranışsal mimari

Davranışsal mimaride sistemin yaptığı işlem process'ler kullanılarak program benzeri bir şekilde yazılır, fakat tasarımın nasıl gerçekleneceği konusunda ayrıntılı bilgi verilmez.

Tablo 4.1. Bir bitlik tam toplayıcı için doğruluk tablosu

A	B	EG	T	EC
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Davranışsal mimariye örnek olarak bir tam toplayıcı devresini inceleyelim. Tablo 4.1'de TAM_TOPLAYICI'ya ait doğruluk tablosu görülmektedir. Bu doğruluk tablosundan yararlanarak davranışsal mimari tasarlanır. Aşağıdaki örnekte TAM_TOPLAYICI için yazılmış davranışsal mimarinin VHDL kodu görülmektedir. Bu mimaride, duyarlılık listesinde (sensitivity list) A, B ve EG girişleri bulunan bir process vardır. Process'in duyarlılık listesindeki girişlerden herhangi birisinde değişiklik meydana geldiğinde process aktif hale gelir ve içindeki satırlar ardışıl (sequential) olarak işletilir. Bu girişlerde değişiklik olmadığında ise process çalıştırılmaz.

```
architecture BEHAVIORAL of TAM_TOPLAYICI is
begin
        process (A, B, EG)
        begin
                if (A='0' and B='0' and EG='0') then
                        T <= '0';
                        EC <= '0';
                elsif   (A='0' and B='0' and EG='1') or
                                (A='0' and B='1' and EG='0') or
                                (A='1' and B='0' and EG='0') then
                        T <= '1';
                        EC <= '0';
                elsif   (A='0' and B='1' and EG='1') or
                                (A='1' and B='0' and EG='1') or
                                (A='1' and B='1' and EG='0') then
                        T <= '0';
                        EC <= '1';
                else
                        T <= '1';
                        EC <= '1';
                endif;
        end process;
end BEHAVIORAL;
```

VHDL'de " <= " sembolü atama ifadesi olarak kullanılmaktadır. Bu sembolün sağındaki değer, solunda bulunan sinyale atanır. VHDL'de " -- " sembolünü izleyen ifadeler programcı tarafından yazılan açıklayıcı ifadelerdir. Bu ifadelerin yazılması isteğe bağlıdır.

Bir mimaride birden fazla process varsa hepsi eş zamanlı çalıştırılır, fakat process'ler içindeki satırlar yukarıdan aşağıya doğru sırayla çalıştırılır.

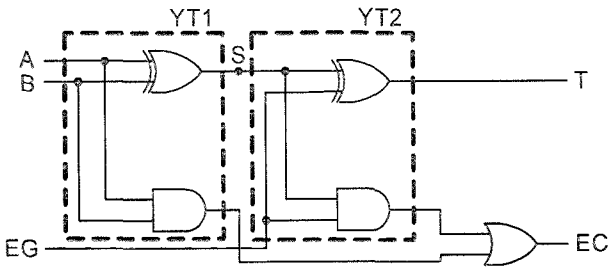
Bir bitlik tam toplayıcı, XC2S200E FPGA'sında davranışsal mimari kullanılarak sentezlendiğinde, kullanılan sistem kaynakları aşağıdaki gibi elde edilmiştir.

Number of Slices:	2	out of 2352	0%
Number of Slice Flip Flops:	1	out of 4704	0%
Number of 4 input LUTs:	3	out of 4704	0%
Number of bonded IOBs:	5	out of 146	3%

Total equivalent gate count for design: 23
Additional JTAG gate count for IOBs: 240

4.3.2.2. Veri akışı mimarisi

Veri akışı mimarisinde, kontrol işaretleri ve verilerin, toplayıcı, karşılaştırıcı, kod çözücü ve basit lojik kapılar gibi kombinasyonel devre elemanları üzerindeki eş zamanlı hareketleri tanımlanır.



Şekil 4.2. Bir bitlik tam toplayıcının lojik kapılarla gösterimi

Veri akışı mimarisine örnek olarak tekrar bir tam toplayıcı devresini inceleyelim. Şekil 4.2'de bir bitlik tam toplayıcının iç yapısı görülmektedir. Bu yapıdan yararlanılarak TAM_TOPLAYICI'nın veri akışı mimarisi elde edilir. Aşağıda TAM_TOPLAYICI için yazılmış veri akışı mimarisinin VHDL kodu görülmektedir. Bu

mimaride üç adet eş zamanlı işlenen işaret ataması vardır. Her satır, duyarlılık listesi sağ taraftaki işaretler olan bir process olarak düşünülebilir.

```
architecture DATAFLOW of TAM_TOPLAYICI is
        signal S : STD_LOGIC;
begin
        S <= A xor B;
        T <= S xor EG;
        EC <= (A and B) or (S and EG);
end DATAFLOW;
```

Burada S sinyali A ve B girişlerinin EX-OR işlemine tabi tutulduktan sonraki değerini saklamak için kullanılmıştır.

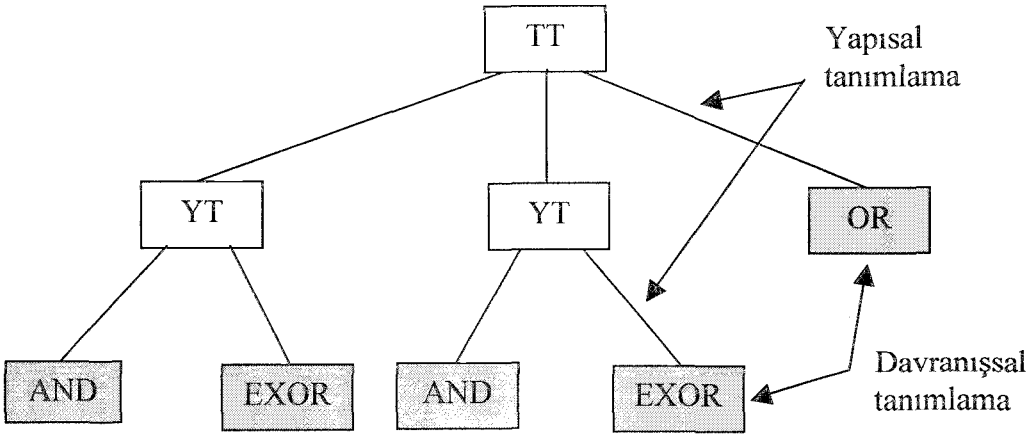
Bir bitlik tam toplayıcı, XC2S200E FPGA'sında veri akışı mimarisi kullanılarak sentezlendiğinde, kullanılan sistem kaynakları aşağıdaki gibi elde edilmiştir.

Number of Slices:	1	out of	2352	0%
Number of 4 input LUTs:	2	out of	4704	0%
Number of bonded IOBs:	5	out of	146	3%

Total equivalent gate count for design: 12
Additional JTAG gate count for IOBs: 240

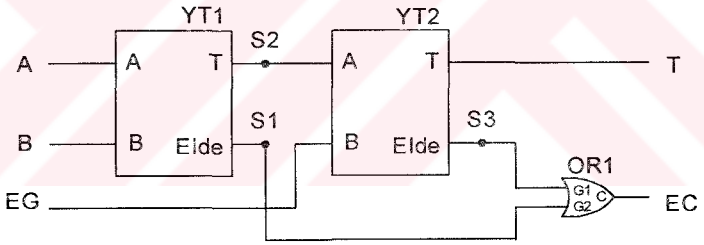
4.3.2.3. Yapısal mimari

Yapısal mimaride eş zamanlı çalışan alt modüllerin listesi ve birbirlerine bağlantıları tanımlanır. Tasarımın karmaşıklığı arttıkça bütün tasarımı bir defada tanımlamak oldukça zorlaşır. Bu sebeple, sistem daha az karmaşık alt sistemlere ayrılır. Her alt sistem tek başına gerçekleştirilebilen veya alt sistemlere ayrılabilen birimlerdir. VHDL'de yapısal mimari kullanılan yüksek seviyedeki tasarım, sadece daha önce tanımlanmış alt sistemlerin listesi ve onların birbiriyle olan bağlantılarından oluşur.



Şekil 4.3. Tam toplayıcının alt sistemlere ayrılışı

Yapısal mimariye örnek olarak yine bir tam toplayıcı devresini inceleyelim. Şekil 4.3'de tam toplayıcının alt sistemlere ayrılışı görülmektedir. Tam toplayıcı, iki yarı toplayıcı ve bir OR kapısından oluşmaktadır. Şekil 4.4'de tam toplayıcının tasarımında kullanılan iki yarı toplayıcı ve bir OR kapısı görülmektedir.



Şekil 4.4. Bir bitlik tam toplayıcının iki yarı toplayıcı ve bir OR kapısı ile gösterimi

Aşağıda TAM_TOPLAYICI için yazılmış yapısal mimarinin VHDL kodu görülmektedir. Yapı isimleri YARI_TOPLAYICI ve OR_KAPISI olan iki alt modüle ayrılmıştır.

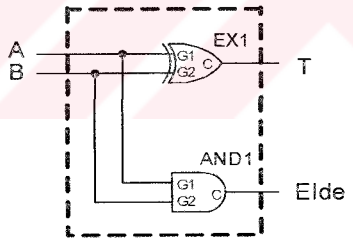
```
architecture STRUCTURAL of TAM_TOPLAYICI is
        component YARI_TOPLAYICI
                port (A, B : in STD_LOGIC;
                        ELDE, T : out STD_LOGIC);
        end component;
        component OR_KAPISI
                port (G1, G2 : in STD_LOGIC;
                        C : out STD_LOGIC);
        end component;
```

```
        signal S1, S2, S3 : STD_LOGIC;
begin
        YT1 : YARI_TOPLAYICI port map (A, B, S1, S2);
        YT2 : YARI_TOPLAYICI port map (S2, EG, S3, T);
        OR1 : OR_KAPISI port map (S1, S3, EC);
end STRUCTURAL;
```

Tam toplayıcının alt sistemlerinden OR kapısının VHDL tanımlaması aşağıda görülmektedir.

```
entity OR_KAPISI is
        port (G1, G2 : in STD_LOGIC;
              C : out STD_LOGIC);
end OR_KAPISI;

architecture BEHAVIORAL of OR_KAPISI is
begin
        C <= G1 or G2 ;
end BEHAVIORAL;
```

Yarı toplayıcılar da alt sistemlere ayrılarak AND ve EXOR kapılarının uygun şekilde bağlanmasıyla elde edilmiştir. Şekil 4.5'de bir yarı toplayıcının iç yapısı görülmektedir.



Şekil 4.5. Bir yarı toplayıcının lojik kapılarla gösterimi

```
entity YARI_TOPLAYICI is
        port (A, B : in STD_LOGIC;
              ELDE, T : out STD_LOGIC);
end YARI_TOPLAYICI;

architecture STRUCTURAL of YARI_TOPLAYICI is
        component EXOR_KAPISI
                port (G1, G2 : in STD_LOGIC;
                      C : out STD_LOGIC);
        end component;
        component AND_KAPISI
                port (G1, G2 : in STD_LOGIC;
                      C : out STD_LOGIC);
        end component;
```

```
begin
        EX1 : EXOR_KAPISI port map (A, B, T);
        AND1 : AND_KAPISI port map (A, B, ELDE);
end STRUCTURAL;
```

Yukarıda kullanılan her birimin daha önce tanımlanması ve çalışılan kütüphanenin içinde saklanması gerekir. EXOR kapısı ve AND kapısı aşağıdaki şekilde tanımlanmış olmalıdır.

```
entity EXOR_KAPISI is
        port (G1, G2 : in STD_LOGIC;
              C : out STD_LOGIC);
end EXOR_KAPISI;
architecture BEHAVIORAL of EXOR_KAPISI is
begin
        process(G1, G2)
        begin
                if      (G1='0' and G2='0') or
                        (G1='1' and G2='1') then
                        C <= '0';
                else
                        C <= '1';
                endif;
        end process;
end BEHAVIORAL;
entity AND_KAPISI is
        port (G1, G2 : in STD_LOGIC;
              C : out STD_LOGIC);
end AND_KAPISI;

architecture BEHAVIORAL of AND_KAPISI is
begin
        C <= G1 and G2 ;
end BEHAVIORAL;
```

Bir bitlik tam toplayıcı, XC2S200E FPGA'sında yapısal mimari kullanılarak sentezlendiğinde, kullanılan sistem kaynakları aşağıdaki gibi elde edilmiştir.

Number of Slices:	1	out of 2352	0%
Number of 4 input LUTs:	2	out of 4704	0%
Number of bonded IOBs:	5	out of 146	3%

Total equivalent gate count for design: 12
Additional JTAG gate count for IOBs: 240

4.3.2.4. Farklı mimarilerin karşılaştırılması

Davranışsal, veri akışı ve yapısal mimariye göre yazılan, bir bitlik tam toplayıcıya ait VHDL modülleri ayrı ayrı XC2S200E FPGA'sı ile sentezlenmiştir. Yazılım olarak Xilinx ISE WebPack 6.3i kullanılmıştır. Bu sentezlenen VHDL modüllerinin, XC2S200E FPGA'sında kullandığı sistem kaynakları Tablo 4.2'de belirtilmiştir.

Tablo 4.2. Bir bitlik tam toplayıcının, farklı mimarilerde XC2S200E FPGA'sında kullandığı sistem kaynakları

XC2S200E	**Davranışsal**	**Veri Akışı**	**Yapısal**
Number of Slices	2 / 2352 0%	1 / 2352 0%	1 / 2352 0%
Number of Slice Flip Flops	1 / 4704 0%	---	---
Number of 4 input LUTs	3 / 4704 0%	2 / 4704 0%	2 / 4704 0%
Number of bonded IOBs	5 / 146 3%	5 / 146 3%	5 / 146 3%
Toplam Eşdeğer Kapı Sayısı	23	12	12

Tablo 4.2'den görüldüğü gibi, davranışsal mimaride yazılan program sentezlendiğinde, diğer mimarilerde yazılan programlara göre FPGA'nın daha fazla sistem kaynağı kullanılmaktadır. Bunun nedeni, davranışsal mimaride yazılan programda, gerçekleştirilen bir bitlik tam toplayıcı hakkında ayrıntılı bilgi verilmemesidir. Veri akışı mimarisinde yazılan program sentezlendiğinde, bir bitlik tam toplayıcının lojik fonksiyonu tanımlandığından dolayı, davranışsal mimariye göre FPGA'nın daha az sistem kaynağı kullanılmaktadır. Benzer olarak yapısal mimaride yazılan program sentezlendiğinde, davranışsal mimariye göre FPGA'nın daha az sistem kaynağı kullanılmaktadır. Bunun nedeni, yapısal mimaride yazılan programda, bir bitlik tam toplayıcıyı oluşturan komponentlerin tanımlanmış olmasıdır.

4.3.3. Paketler

Paketlerin kullanılmasındaki amaç, iki veya daha fazla tasarım birimi tarafından ortak olarak kullanılan nesneleri bir arada toplamaktır. Pakette tanımlanan nesneler, VHDL tasarımındaki tüm birimler tarafından kullanılabilir.

Bir paket iki bölümden oluşur: Paket ifadesi (package declaration) ve paket bloğu (package body). Paket ifadesinde bazı veri tipleri, sabitler ve alt program isimleri tanımlanır. Paket bloğu, paket ifadesinde tanımlanan alt programların (fonksiyonlar veya prosedürler) çalışmasının yazıldığı bölümdür. Eğer paket ifadesinde alt program yoksa paket bloğu yazılmaz. Bir paket bloğunun ismi, paket ifadesinde kullanılan isimle aynı olmalıdır. Ancak paket bloğunun tasarımdaki yeri önemli değildir. paket ifadesiyle aynı veya farklı kaynak dosyasında bulunabilir.

Aşağıdaki örnekte bir paket tanımlaması yapılmıştır. Paket ismi conversion'dur. Bu paket ifadesinde bir tamsayıyı ikili (binary) hale dönüştüren **to_vector** isimli bir fonksiyon tanımlanmıştır. Paket bloğunda da bu fonksiyonun çalışması açıklanmıştır.

```
package conversion is
  function to_vector (size: integer; num: integer) return std_logic_vector;
end conversion;

package body conversion is
  function to_vector(size: integer; num: integer) return std_logic_vector is
    variable ret: std_logic_vector (1 to size);
    variable a: integer;
  begin
    a := num;
    for i in size downto 1 loop
      if ((a mod 2) = 1) then
        ret(i) := '1';
      else
        ret(i) := '0';
      end if;
      a := a / 2;
    end loop;
    return ret;
  end to_vector;
end conversion;
```

4.3.4. Konfigürasyon

Konfigürasyon ile yapısal mimaride kullanılacak alt sistemlerin, hangi kütüphanede ve hangi dosya ismiyle saklandığı gösterilir. Alt sistemlere ait tanımlama, farklı mimari yapılar kullanılarak birden fazla yazılmış olabilir. Konfigürasyonda,

yüksek seviyeli sistem tanımlanırken mimari yapılardan hangisinin kullanılacağı gösterilir. Yazılış şekli aşağıdaki gibidir.

```
configuration konfigürasyon_ismi of modül_ismi is
        for yüksek seviyeli sistemin mimari yapısı
                { Alt sistemlerin tanımlandığı bölüm }
        end for;
end modül_ismi;
```

TAM_TOPLAYICI için yazılmış konfigürasyon aşağıdaki gibidir. Konfigürasyon ismi TT_KONFIG dir. TAM_TOPLAYICI'nın mimari tanımlamasında kullanılan yarı toplayıcılar için WORK kütüphanesindeki, YARI_TOPLAYICI modülüne ait yapısal tanımlama, OR kapısı için ise WORK kütüphanesindeki, OR_KAPISI modülüne ait davranışsal tanımlama kullanılacaktır.

```
configuration TT_KONFIG of TAM_TOPLAYICI is
        for STRUCTURE

        for YT1, YT2 :
        YARI_TOPLAYICI use entity WORK.YARI_TOPLAYICI (Structural);
        end for;
        for OR1 :
        OR_KAPISI use entity WORK.OR_KAPISI (Behavioral);
        end for;

        end for;
end TT_KONFIG;
```

Sentezleme araçları, konfigürasyonları desteklemeyebilirler. Bu yüzden konfigürasyonlar genellikle sentezleme işleminde kullanılmazlar.

4.3.5. Tasarım kütüphaneleri

VHDL'de yazılan tanımlamalar, yazım kurallarına göre doğruluğu kontrol edildikten sonra, sentezleme ve simülasyonda kullanılmak üzere bir kütüphanede saklanır. Kütüphanede aşağıdaki birimler saklanır [4]:

- Entity'ler - paylaşılan modüller
- Mimariler - paylaşılan tasarım gerçeklemeleri
- Paketler - paylaşılan tanımlamalar

- Konfigürasyonlar - paylaşılan farklı tasarım yapıları

Bir kütüphane tek başına analiz edilebilir. Ancak belirli bir sırayla analiz edilmelidir. Örneğin, bir "entity", gerçeklenmesinin tanımlandığı mimariden önce analiz edilmelidir. Bir paket de onu kullanan tasarımdan daha önce analiz edilmelidir.

4.4. VHDL Dilinin Şematik Gösterime Göre Üstünlükleri

VHDL dili, şematik gösterimle kıyaslandığında, bazı önemli üstünlüklere sahip olduğu görülür. Bu üstünlükler aşağıda belirtilmiştir [7].

- Genelde, şematik gösterimin hazırlandığı sayfa 200 kapı içerebilir. Gerçekleştirmek istediğimiz uygulama 10000 kapı seviyesinde ise, 50 şematik sayfası gerekmektedir. Her sayfanın çıkışları diğer sayfaya giriş olarak verilmelidir. Ancak VHDL dili kullanarak tek bir text dosyası halinde aynı uygulama gerçekleştirilebilir.
- VHDL dili kullanılarak bir uygulama daha çabuk oluşturulabilir. Ayrıca uygulamada yapılacak herhangi bir değişiklik, VHDL ile daha çabuk bir şekilde gerçekleştirilebilir.
- VHDL dilinin en önemli üstünlüğü ise, FPGA'yı üreten firmaya bağımlı olmamasıdır. Böylece geliştirilen uygulama farklı üreticilerin FPGA'larına da kolaylıkla aktarılabilir. Şematik gösterimde ise, hazırlanan uygulamanın başka bir FPGA'ya aktarılması çok güçtür. Çünkü kullanılan her komponentin FPGA üreticisinin programının kütüphanesinde bulunması gerekir.

Örnek verecek olursak; 16x16'lık bir çarpıcı uygulaması şematik gösterim ile yapıldığında, 6000 kapı kullanmalıyız ve bu da 30 şematik sayfası gerektirmektedir. Tasarımı da yaklaşık 3 gün alır. Ayrıca FPGA üreticisine bağımlıdır. VHDL kullanılarak gerçekleştirilirse, 8 satırlık kod içeren bir text dosyasının hazırlanması yeterlidir. Bu programın yazılması yaklaşık 3 dakika alır ve tamamen FPGA üreticisinden bağımsızdır.

Eğer 16x16'lık çarpıcı, 32x32'lik çarpıcıya dönüştürülmek istenirse, şematik gösterimde saatlerce uğraş verilmesi gerekir. Ancak VHDL kullanılarak hazırlanan programdaki iki satırın değiştirilmesi yeterlidir. Aşağıda VHDL dili ile hazırlanmış 16x16'lık çarpıcının ve 32x32'lik çarpıcının kodları görülmektedir.

16x16'lık çarpıcı :

```
entity MULT is
        Port ( A, B : in std_logic_vector (15 downto 0);
                Y : out std_logic_vector (31 downto 0) );
end MULT;
achitecture behave of MULT is
        Y <= A * B ;
end behave;
```

32x32'lik çarpıcı :

```
entity MULT is
        Port ( A, B : in std_logic_vector (31 downto 0);
                Y : out std_logic_vector (63 downto 0) );
end MULT;
achitecture behave of MULT is
        Y <= A * B ;
end behave;
```

BÖLÜM V

BOPN'LERİN VHDL DİLİ KULLANILARAK XILINX XC2S200E FPGA'SI İLE GERÇEKLEŞTİRİLMESİ

Bu bölümde, II. bölümde açıklanan Basit Otomasyon Petri netlerin (BOPN) VHDL dili kullanılarak gerçekleştirilmesi ele alınacaktır. Burada verilen VHDL kodları, Digilent D2-SB uygulama geliştirme kartı üzerinde bulunan Xilinx XC2S200E FPGA'sı kullanılarak sentezlenmiştir.

5.1. Temel BOPN Modellerinin Xilinx XC2S200E FPGA'sı ile Gerçekleştirilmesi

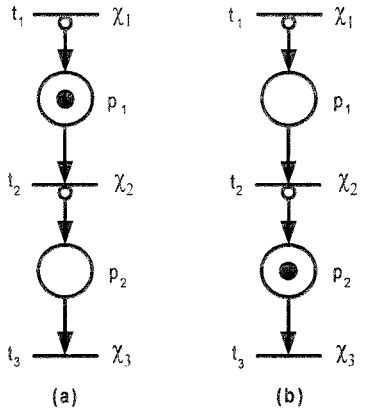
Aşağıda, II. Bölüm'de gösterilen temel BOPN modelleri ve bunların VHDL dili ile yazılmış program kodları görülmektedir. Bu modeller VHDL diline çevrilirken, geçişleri tetikleyen tetikleme şartları giriş (in) ve mevkiler hem giriş hem de çıkış (inout) modunda tanımlanmıştır. Bu BOPN modellerindeki mevkilere, aksiyon ataması yapılabilir. Ancak her mevkiye bir aksiyon atama mecburiyeti yoktur.

5.1.1. Başlangıç değeri atama (Initial marking)

Bu kısımda, bir BOPN'de başlangıçta içinde jeton bulunması gereken mevkilere jeton yerleştirilmesi işleminin nasıl gerçekleştirildiği incelenmektedir. Şekil 5.1 (a)'da bir BOPN görülmektedir. Burada başlangıçta p_1'de bir jeton depolanması gerekmektedir. Bu BOPN modelindeki geçişlere atanan tetikleme şartları ve mevkiler, entity ifadesi içinde aşağıda görüldüğü gibi tanımlanır. Bu modelde üç geçiş bulunduğundan entity ifadesinde üç tetikleme şartı tanımlanmıştır ve bunlar giriş (in) modundadır. Mevkiler hem giriş hem de çıkış (inout) modunda tanımlanmıştır. Böylece mevkilerde jeton bulunup bulunmadığı kontrol edilebilmektedir.

```
entity initial is
  Port ( x1    : in std_logic;
         x2    : in std_logic;
         x3    : in std_logic;
```

```
        p1    : inout std_logic;
        p2    : inout std_logic
      );
end initial;
```



Şekil 5.1. Bir BOPN için başlangıç değeri atama örneği

Şekil 5.1 (a)'da görülen BOPN'nin mimari yapısı aşağıda tanımlandığı gibidir. Buradaki dört atama ifadesinin hepsi de eşzamanlı (concurrent) olarak çalışmaktadır. Atama ifadelerinin ilk satırında bulunan değerler başlangıç olarak mevkilere atanır. Böylece Şekil 5.1 (a)'daki BOPN'de görülen p_1 mevkisine başlangıçta bir jeton depolanacaktır. when ifadelerinin sağındaki koşullar sağlandığında, mevkilerin jeton sayısı da ifadenin bulunduğu satırdaki değere göre değişir.

```
architecture Behavioral of initial is

begin

p1 <= '1' when ( x1='1' and p1='0' ) else
      '0' when ( x2='1' and p1='1' and p2='0' ) ;

p2 <= '0' when ( x3='1' and p2='1' ) else
      '1' when ( x2='1' and p1='1' and p2='0' );

end Behavioral;
```

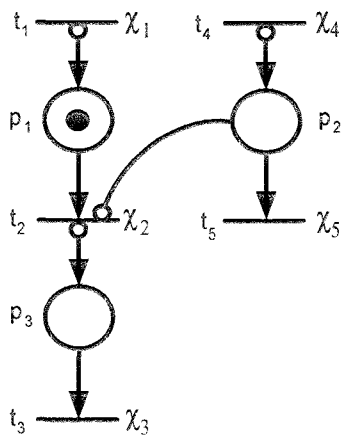
Şekil 5.1 (b)'de görülen BOPN'de ise başlangıç değeri olarak, p_1 mevkisinde hiç jeton bulunmazken p_2 mevkisinde bir jeton bulunmaktadır. Bu BOPN'deki başlangıç jeton dağılımı için bir önceki örnekten farklı olarak, mimari yapı aşağıdaki gibi yazılmalıdır.

```
architecture Behavioral of initial is

begin

p1 <= '0' when ( x2='1' and p1='1' and p2='0' ) else
      '1' when ( x1='1' and p1='0' );

p2 <= '1' when ( x2='1' and p1='1' and p2='0' ) else
      '0' when ( x3='1' and p2='1');

end Behavioral;
```

5.1.2. Yasaklama oku (Inhibitor arc)

Şekil 5.2'de içerisinde bir yasaklama oku bulunan BOPN modeli görülmektedir. Bu modelde beş geçiş bulunduğundan, aşağıda görülen entity ifadesinde giriş (in) modunda tanımlanmış beş tetikleme şartı ve hem giriş hem de çıkış (inout) modunda tanımlanmış olan üç tane mevki mevcuttur.

```
entity inhibitor is
  Port ( x1    : in std_logic;
         x2    : in std_logic;
         x3    : in std_logic;
         x4    : in std_logic;
         x5    : in std_logic;
         p1    : inout std_logic;
         p2    : inout std_logic;
         p3    : inout std_logic);
end inhibitor;
```



Şekil 5.2. Yasaklama oku modeli

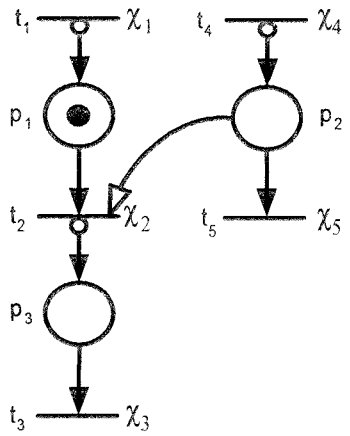
Aşağıda görülen mimari ifadesinde bu BOPN modelinin çalışma biçimi tanımlanmıştır. Başlangıçta sadece p_1 mevkisine jeton depolanmaktadır.

```
architecture Behavioral of inhibitor is

begin

p1 <= '1' when (x1='1' and p1='0') else
      '0' when (x2='1' and p1='1' and p3='0' and p2='0') ;

p2 <= '0' when (x5='1' and p2='1') else
      '1' when (x4='1' and p2='0') ;

p3 <= '0' when (x3='1' and p3='1') else
      '1' when (x2='1' and p1='1' and p3='0' and p2='0') ;

end Behavioral;
```

5.1.3. Yetkileme oku (Enabling arc)

Şekil 5.3'te içerisinde bir yetkileme oku bulunan BOPN modeli görülmektedir. Bu modelde beş geçiş bulunduğundan, entity ifadesinde beş tetikleme şartı tanımlanmıştır ve bunlar giriş (in) modundadır. Mevkiler hem giriş hem de çıkış (inout) modunda tanımlanmıştır.

```
entity enabling is
  Port ( x1    : in std_logic;
         x2    : in std_logic;
         x3    : in std_logic;
         x4    : in std_logic;
         x5    : in std_logic;
         p1    : inout std_logic;
         p2    : inout std_logic;
         p3    : inout std_logic);
end enabling;
```



Şekil 5.3. Yetkileme oku modeli

Aşağıda görülen mimari ifadesinde bu BOPN modelinin çalışma biçimi tanımlanmıştır. Başlangıçta sadece p_1 mevkisine jeton depolanmaktadır.

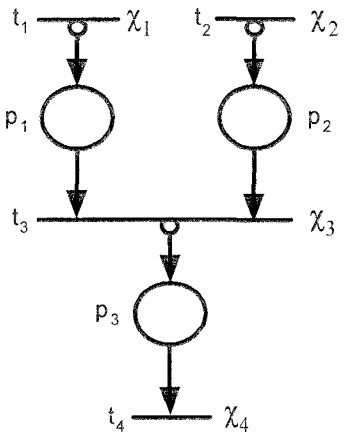
```
architecture Behavioral of enabling is

begin

p1 <= '1' when (x1='1' and p1='0') else
      '0' when (x2='1' and p1='1' and p3='0' and p2='1') ;

p2 <= '0' when (x5='1' and p2='1') else
      '1' when (x4='1' and p2='0') ;

p3 <= '0' when (x3='1' and p3='1') else
      '1' when (x2='1' and p1='1' and p3='0' and p2='1') ;

end Behavioral;
```

5.1.4. Bağlama (Join)

Şekil 5.4'te bağlama BOPN modeli görülmektedir. Bu modelde dört geçiş bulunduğundan, aşağıda görülen entity ifadesinde dört tetikleme şartı tanımlanmıştır ve bunlar giriş (in) modundadır. Mevkiler hem giriş hem de çıkış (inout) modunda tanımlanmıştır.

```
entity join is
  Port ( x1   : in std_logic;
         x2   : in std_logic;
         x3   : in std_logic;
         x4   : in std_logic;
```

```
        p1    : inout std_logic;
        p2    : inout std_logic;
        p3    : inout std_logic);
end join;
```



Şekil 5.4. Bağlama (join) modeli

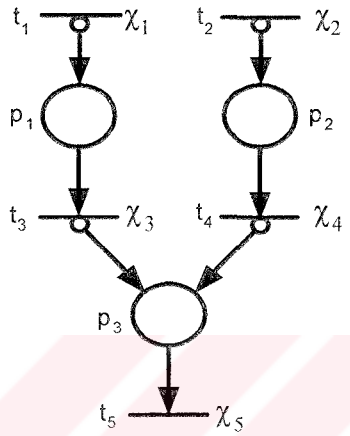
Aşağıda görülen mimari ifadesinde bu BOPN modelinin çalışma biçimi tanımlanmıştır. Buradaki üç atama ifadesinin hepsi de eşzamanlı (concurrent) olarak çalışmaktadır.

```
architecture Behavioral of join is

begin

  p1 <= '0' when (x3='1' and p1='1' and p2='1' and p3='0') else
        '1' when (x1='1' and p1='0');

  p2 <= '0' when (x3='1' and p1='1' and p2='1' and p3='0') else
        '1' when (x2='1' and p2='0');

  p3 <= '0' when (x4='1' and p3='1') else
        '1' when (x3='1' and p1='1' and p2='1' and p3='0');

end Behavioral;
```

5.1.5. Birleştirme (Merge)

Şekil 5.5'de birleştirme BOPN modeli görülmektedir. Bu modelde beş geçiş bulunduğundan, aşağıda görülen entity ifadesinde beş tetikleme şartı tanımlanmıştır ve bunlar giriş (in) modundadır. Mevkiler, hem giriş hem de çıkış (inout) modunda tanımlanmıştır.

```
entity merge is
  Port ( x1    : in std_logic;
         x2    : in std_logic;
         x3    : in std_logic;
         x4    : in std_logic;
         x5    : in std_logic;
         p1    : inout std_logic;
         p2    : inout std_logic;
         p3    : inout std_logic);
end merge;
```



Şekil 5.5. Birleştirme (merge) modeli

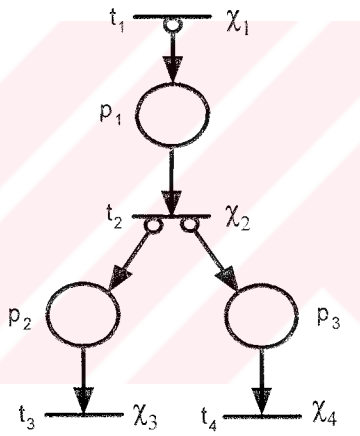
Aşağıda görülen mimari ifadesinde bu BOPN modelinin çalışma biçimi tanımlanmıştır. Buradaki üç atama ifadesinin hepsi de eşzamanlı (concurrent) olarak çalışmaktadır.

```
architecture Behavioral of merge is

begin

  p1 <= '0' when (x3='1' and p1='1' and p3='0') else
        '1' when (x1='1' and p1='0') ;

  p2 <= '0' when (x4='1' and p2='1' and p3='0') else
        '1' when (x2='1' and p2='0') ;

  p3 <= '0' when (x5='1' and p3='1') else
        '1' when (x3='1' and p1='1' and p3='0') or
                 (x4='1' and p2='1' and p3='0') ;

end Behavioral;
```

5.1.6. Çatal (Fork)

Şekil 5.6'da çatal BOPN modeli görülmektedir. Bu modelde dört geçiş bulunduğundan, aşağıda görülen entity ifadesinde dört tetikleme şartı tanımlanmıştır ve bunlar giriş (in) modundadır. Mevkiler hem giriş hem de çıkış (inout) modunda tanımlanmıştır.

```
entity fork is
  Port ( x1   : in std_logic;
         x2   : in std_logic;
         x3   : in std_logic;
         x4   : in std_logic;
         p1   : inout std_logic;
         p2   : inout std_logic;
         p3   : inout std_logic);
end fork;
```



Şekil 5.6. Çatal (fork) modeli

Aşağıda görülen mimari ifadesinde bu BOPN modelinin çalışma biçimi tanımlanmıştır. Buradaki üç atama ifadesinin hepsi de eşzamanlı (concurrent) olarak çalışmaktadır.

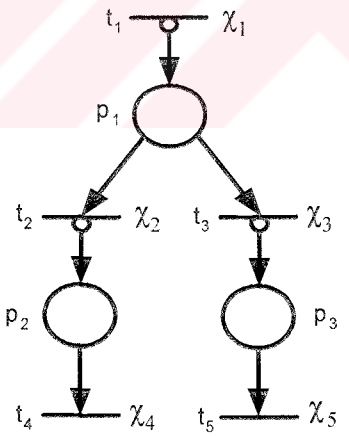
```
architecture Behavioral of fork is

begin

  p1 <= '0' when (x2='1' and p1='1' and p2='0' and p3='0') else
        '1' when (x1='1' and p1='0') ;

  p2 <= '0' when (x3='1' and p2='1') else
        '1' when (x2='1' and p1='1' and p2='0' and p3='0') ;
```

```
p3 <= '0' when (x4='1' and p3='1') else
      '1' when (x2='1' and p1='1' and p2='0' and p3='0') ;

end Behavioral;
```

5.1.7. İkilem (Conflict)

Şekil 5.7'de ikilem BOPN modeli görülmektedir. Bu modelde beş geçiş bulunduğundan, aşağıda görülen entity ifadesinde beş tetikleme şartı tanımlanmıştır ve bunlar giriş (in) modundadır. Mevkiler hem giriş hem de çıkış (inout) modunda tanımlanmıştır.

```
entity conflict is
  Port ( x1    : in std_logic;
         x2    : in std_logic;
         x3    : in std_logic;
         x4    : in std_logic;
         x5    : in std_logic;
         p1    : inout std_logic;
         p2    : inout std_logic;
         p3    : inout std_logic);
end conflict;
```



Şekil 5.7. İkilem (conflict) modeli

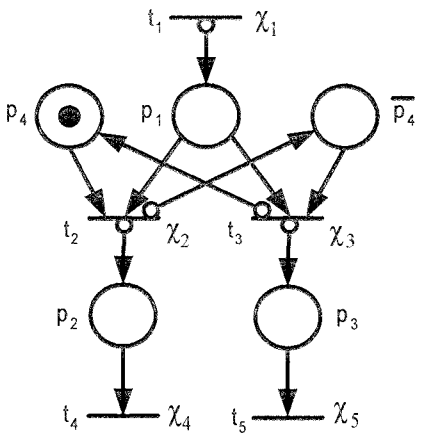
Aşağıda görülen mimari ifadesinde bu BOPN modelinin çalışma biçimi tanımlanmıştır. Buradaki üç atama ifadesinin hepsi de eşzamanlı (concurrent) olarak çalışmaktadır.

```
architecture Behavioral of conflict is

begin

        p1 <= '0' when (x2='1' and p1='1' and p2='0') or
                       (x3='1' and p1='1' and p3='0') else
              '1' when (x1='1' and p1='0') ;

        p2 <= '0' when (x4='1' and p2='1')  else
              '1' when (x2='1' and p1='1' and p2='0') ;

        p3 <= '0' when (x5='1' and p3='1') else
              '1' when (x3='1' and p1='1' and p3='0') ;

end Behavioral;
```

5.1.8. Toggle

Şekil 5.8'de toggle BOPN modeli görülmektedir. Bu modelde beş geçiş bulunduğundan, aşağıda görülen entity ifadesinde beş tetikleme şartı tanımlanmıştır ve bunlar giriş (in) modundadır. Mevkiler hem giriş hem de çıkış (inout) modunda tanımlanmıştır.

```
entity toggle is
   Port ( x1          : in std_logic;
          x2          : in std_logic;
          x3          : in std_logic;
          x4          : in std_logic;
          x5          : in std_logic;
          p1          : inout std_logic;
          p2          : inout std_logic;
          p3          : inout std_logic;
          p4          : inout std_logic;
          c_p4        : inout std_logic);
end toggle;
```



Şekil 5.8. Toggle modeli

Aşağıda görülen mimari ifadesinde bu BOPN modelinin çalışma biçimi tanımlanmıştır. Buradaki beş atama ifadesinin hepsi de eşzamanlı (concurrent) olarak çalışmaktadır.

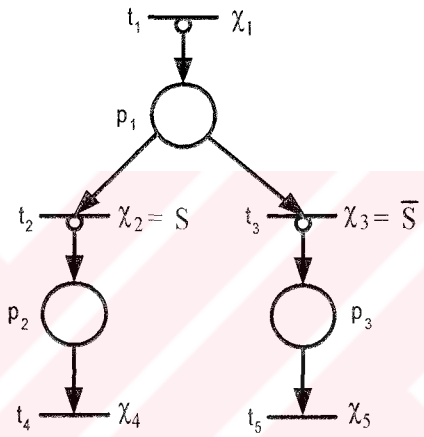
```
architecture Behavioral of toggle is

begin

    p1 <= '0' when (x2='1' and p1='1' and p4='1' and p2='0' and c_p4='0') or
                   (x3='1' and p1='1' and c_p4='1' and p3='0' and p4='0') else
          '1' when (x1='1' and p1='0');

    p2 <= '0' when (x4='1' and p2='1') else
          '1' when (x2='1' and p1='1' and p4='1' and p2='0' and c_p4='0');

    p3 <= '0' when (x5='1' and p3='1') else
          '1' when (x3='1' and p1='1' and c_p4='1' and p3='0' and p4='0');

    p4 <= '1' when (x3='1' and p1='1' and c_p4='1' and p3='0' and p4='0') else
          '0' when (x2='1' and p1='1' and p4='1' and p2='0' and c_p4='0');

    c_p4 <= '0' when (x3='1' and p1='1' and c_p4='1' and p3='0' and p4='0') else
            '1' when (x2='1' and p1='1' and p4='1' and p2='0' and c_p4='0');

end Behavioral;
```

5.1.9. Seçim (Select)

Şekil 5.9'da seçim BOPN modeli görülmektedir. Bu modelde beş geçiş bulunduğundan, aşağıda görülen entity ifadesinde beş tetikleme şartı tanımlanmıştır ve

bunlar giriş (in) modundadır. Mevkiler hem giriş hem de çıkış (inout) modunda tanımlanmıştır.

```
entity select is
   Port ( x1          : in std_logic;
          x2          : in std_logic;
          x3          : in std_logic;
          x4          : in std_logic;
          x5          : in std_logic;
          p1          : inout std_logic;
          p2          : inout std_logic;
          p3          : inout std_logic);
end select;
```



Şekil 5.9. Seçim (select) modeli

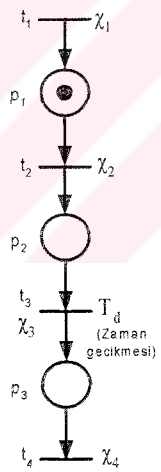
Aşağıda görülen mimari ifadesinde bu BOPN modelinin çalışma biçimi tanımlanmıştır. Buradaki üç atama ifadesinin hepsi de eşzamanlı (concurrent) olarak çalışmaktadır.

```
architecture Behavioral of select_m is

begin

        p1 <= '0' when (x2='1' and p1='1' and p2='0') or
                        (x3='1' and p1='1' and p3='0') else
              '1'  when (x1='1' and p1='0');

        p2 <= '0' when (x4='1' and p2='1') else
               '1' when (x2='1' and p1='1' and p2='0');

        p3 <= '0' when (x5='1' and p3='1') else
              '1' when (x3='1' and p1='1' and p3='0');
end Behavioral;
```

5.1.10. Zamanlı g eçişli BOPN

Şekil 5.10'da zamanlı geçişli bir BOPN modeli görülmektedir. Bu modelde dört geçiş bulunmaktadır. Bu nedenle, dört geçiş için aşağıda görülen entity ifadesinde dört tetikleme şartı tanımlanmıştır ve bunlar giriş (in) modundadır. Mevkiler hem giriş hem de çıkış (inout) modunda tanımlanmıştır. Ayrıca zamanı ayarlamak için kullanılan clk girişi (in) modunda tanımlanmıştır.

```
entity timed is
   Port ( x1 : in std_logic;
          x2 : in std_logic;
          x3 : in std_logic;
          x4 : in std_logic;
          p1 : inout std_logic;
          p2 : inout std_logic;
          p3 : inout std_logic;
          clk : in std_logic);
end timed;
```



Şekil 5.10. Zamanlı geçişli bir BOPN

Aşağıda görülen mimari ifadesinde, iki kod bloğu görülmektedir. Birisi T_d=2.68s'lik sürenin belirlenmesi için kullanılmaktadır. Diğeri de, bu BOPN modelinin çalışmasını belirtmektedir. Buradaki process ifadeleri ve process'lerin dışındaki atama ifadeleri eşzamanlı (concurrent) olarak çalışmaktadır. Process'lerin içindeki kodlar ise sıralı (sequential) olarak çalışmaktadır.

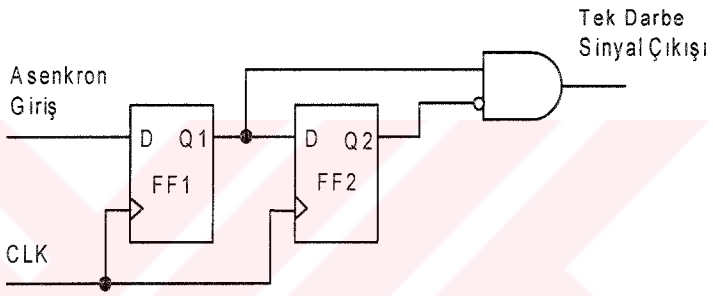
```
architecture Behavioral of timed is

signal cntClk  : std_logic_vector (27 downto 0) :="0000000000000000000000000000";
signal dvdClk, q2, Td : std_logic ;

begin

----------------------------------------------------------------------
--                2.68 s'lik Td süresinin belirlendiği VHDL kodu
----------------------------------------------------------------------

--  f(clk)= 50MHz ==> T(clk) = 20 ns
-- T(dvdclk) = 20 ns x (2^27) = 2.68 s -->yükselen kenar

process (clk, x3)
begin
        if x3='0' then
                cntClk <= "0000000000000000000000000000" ;
        elsif (clk'event and clk='1') then
                cntClk <= cntClk + 1 ;
        end if;
end process;

dvdClk <= cntClk(27)          ;

process (dvdClk, x3)
begin
        if x3='0' then
                q2 <= '0';
        elsif dvdClk'event and dvdClk='1' then
                q2 <= x3;
        end if;
end process;

        Td <= x3 and q2 ;
-------------------------------------------------------------------------------------
--                Zamanlı geçişli (timed) BOPN için VHDL kodu
-------------------------------------------------------------------------------------

p1 <= '1' when (x1='1' and p1='0') else
        '0' when (x2='1' and p1='1' and p2='0');

p2 <= '0' when (p2='1' and p3='0' and Td ='1') else
        '1' when (x2='1' and p1='1' and p2='0') ;

p3 <= '0' when (x4='1' and p3='1') else
        '1' when (p2='1' and p3='0' and Td ='1');

end Behavioral;
```

5.2. Geçişlerin Tetiklenmesi

Önceki kısımda incelenen BOPN modellerinde geçişlerin tetiklenmesi için bu geçişlere atanan tetikleme şartlarının lojik '1' ya da '0' olması gerekir. Bu tetikleme şartları FPGA'ya dışarıdan giriş olarak verilmektedir. Bu tetikleme şartlarına ilave olarak girişlerdeki sinyallerin yükselen veya düşen kenarları da tetikleme şartı olarak kullanılabilir. Bu kısımda bu işlemin nasıl yapıldığı incelenmektedir.

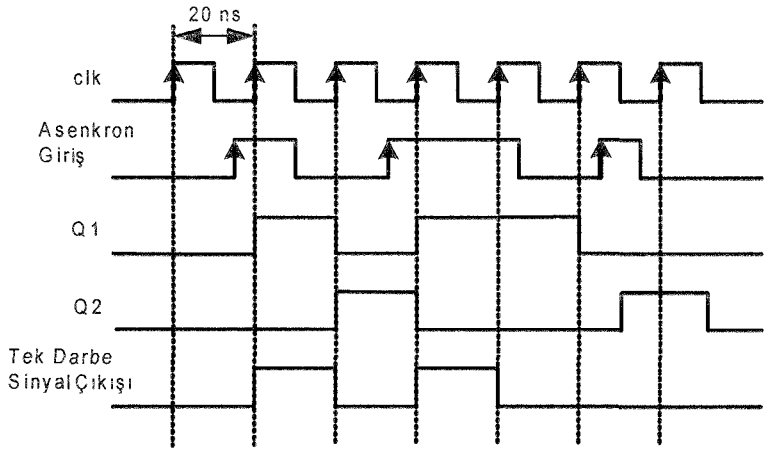
5.2.1. Yükselen kenar algılama

FPGA'nın girişlerindeki sinyallerin, yükselen kenarını algılamak için Şekil 5.11'deki devre esas alınmıştır [6].



Şekil 5.11. Yükselen kenar algılamak için kullanılan devre

Bu devrenin istenilen şekilde çalışabilmesi için, giriş darbesi en az bir clk periyodu genişliğinde olmalıdır. Aksi takdirde, giriş darbesi clk sinyalinin yükselen kenarıyla aynı zamana denk gelemeyebilir. Digilent D2-SB kartında f=50 MHz'lik clock (T=20 ns) osilatörü bulunmaktadır. Buna göre, kullanılan girişin darbe genişliğinin en az 20 ns olması gerekmektedir. Şekil 5.12'de, yükselen kenar algılamak için kullanılan devredeki, sinyallerin dalga şekilleri görülmektedir.



Şekil 5.12. Yükselen kenar algılamak için kullanılan devredeki sinyallerin dalga şekilleri

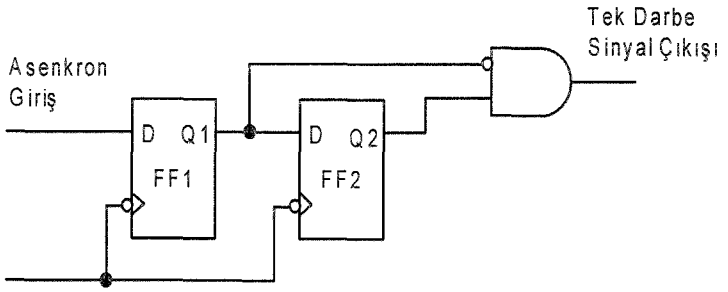
Yükselen kenar algılamak için kullanılan devrenin VHDL karşılığı aşağıda görülmektedir. Bu program bloğunda, clock (clk) sinyalinin, yükselen kenarının algılanması " clk'event and clk='1' " ifadesi ile gerçekleştirilmiştir.

```
entity r_edge_detect is
   Port ( input  : in std_logic;
          clk    : in std_logic;
          output : out std_logic);
end r_edge_detect;

architecture Behavioral of r_edge_detect is

        signal q1, q2 : std_logic;

begin

process (clk)
begin
  if clk'event and clk='1' then
        q1 <= input;
  end if;
end process;

process (clk)
begin
  if clk'event and clk='1' then
        q2 <= q1;
  end if;
end process;

output <= q1 and (not q2);
```

end Behavioral;

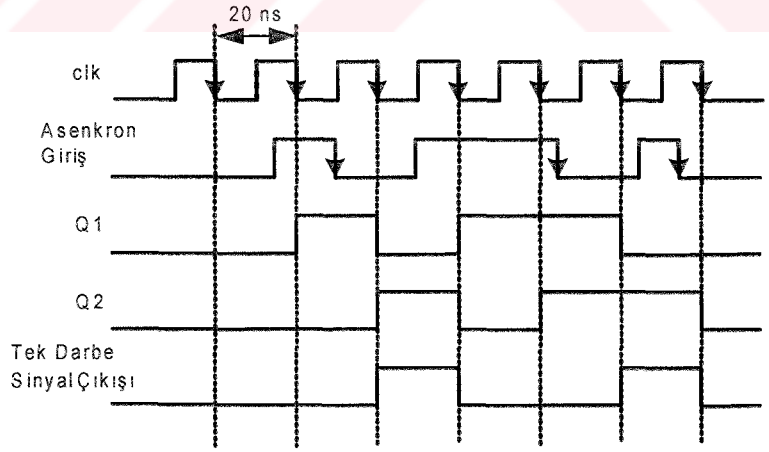
5.2.2. Düşen kenar algılama

FPGA'nın girişlerindeki sinyallerin, düşen kenarını algılamak için Şekil 5.13'teki devre esas alınmıştır.



Şekil 5.13. Düşen kenar algılamak için kullanılan devre

Bu devrenin istenilen şekilde çalışabilmesi için, giriş darbesi en az bir clk periyodu genişliğinde olmalıdır. Aksi takdirde giriş darbesi clk sinyalinin düşen kenarıyla aynı zamana denk gelemeyebilir. Digilent D2-SB kartında f=50 MHz'lik clock osilatörü (T=20 ns) bulunduğundan, kullanılan girişin darbe genişliğinin en az 20 ns olması gerekmektedir. Şekil 5.14'te, düşen kenar algılamak için kullanılan devredeki, sinyallerin dalga şekilleri görülmektedir.



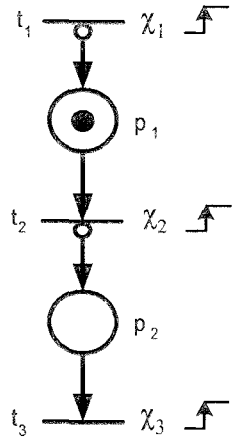
Şekil 5.14. Düşen kenar algılamak için kullanılan devredeki sinyallerin dalga şekilleri

Düşen kenar algılamak için kullanılan devrenin VHDL karşılığı aşağıda görülmektedir. Bu program bloğunda, clock (clk) sinyalinin, düşen kenarının algılanması " clk'event and clk='0' " ifadesi ile gerçekleştirilmiştir.

```
entity f_edge_detect is
   Port ( input : in std_logic;
          clk   : in std_logic;
          output : out std_logic);
end f_edge_detect;
architecture Behavioral of f_edge_detect is

        signal q1, q2 : std_logic;

begin

process (clk)
begin
  if (clk'event and clk='0') then
        q1 <= input;
  end if;
end process;

process (clk)
begin
  if (clk'event and clk='0') then
        q2 <= q1;
  end if;
end process;

output <= q1 or (not q2);

end Behavioral;
```

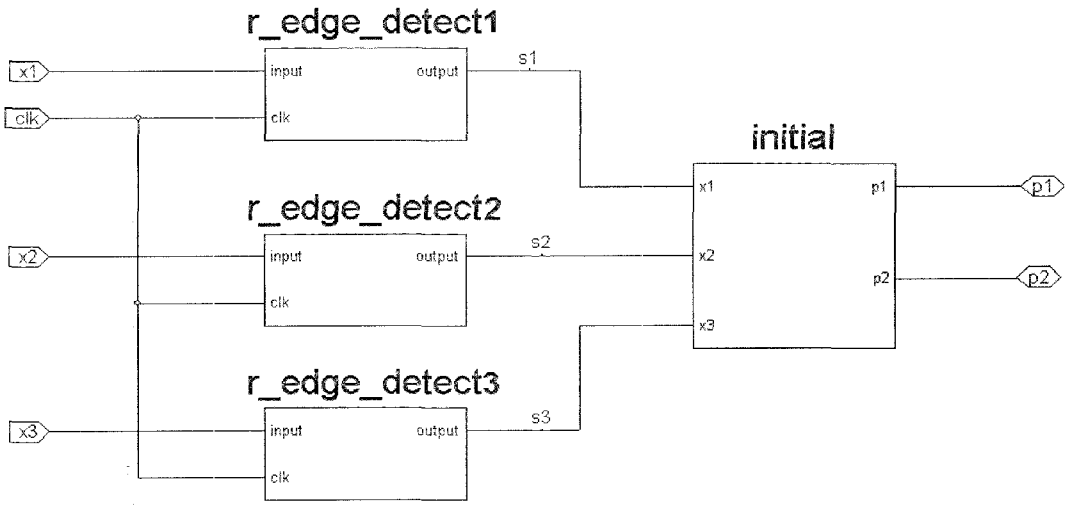
5.2.3. Geçişlerin yükselen kenar tetikleme şartı ile tetiklenmesi

Burada bir önceki kısımda anlatılan yükselen kenar algılama işleminden faydalanılarak bir BOPN'de geçişlerin yükselen kenarda nasıl tetiklendiği incelenmektedir. Benzer işlem düşen kenar sinyali ile tetikleme için de yapılabilir. Şekil 5.15'te geçişleri t_1, t_2 ve t_3 olan bir BOPN modeli görülmektedir. Bu modelde bulunan χ_1, χ_2 ve χ_3 tetikleme sinyallerinin '0' dan '1' e geçişleri esnasında t_1, t_2 ve t_3 geçişleri tetiklenebilir.



Şekil 5.15. Geçişleri yükselen kenarla tetiklenen bir BOPN modeli

Şekil 5.15'te görülen BOPN modelini FPGA'da gerçekleştirmek için, daha önceki kısımlarda tanımlanan yükselen kenar algılama (**r_edge_detect**) ve başlangıç değer atama (**initial**) VHDL kodları ayrı ayrı VHDL modüllerine yazılarak sentezlenirler. Daha sonra ayrı bir VHDL modülü oluşturularak (**top_module**), bu modül içerisinde **r_edge_detect** ve **initial** VHDL modülleri komponent olarak tanımlanırlar. Şekil 5.15'teki BOPN modelinde 3 geçiş bulunduğundan, **top_module** modülünde 3 adet yükselen kenar algılama komponenti (**r_edge_detect1**, **r_edge_detect2**, **r_edge_detect3**) kullanılmıştır. **top_module** modülünde tanımlanan 4 komponent (**r_edge_detect1**, **r_edge_detect2**, **r_edge_detect3**, **initial**) arasındaki sinyal dağılımı düzenlenerek istenilen VHDL modülü elde edilir. Şekil 5.16'da, yükselen kenarda tetiklenen başlangıç değer atama modelini oluşturan komponentlerin şematik gösterimi görülmektedir.



Şekil 5.16. Yükselen kenarda tetiklenen başlangıç değer atama modeli

Aşağıda r_edge_detect ve initial VHDL modüllerinin yapısal olarak birleştirildiği, top_module modülünün VHDL kodu görülmektedir.

```
entity top_module is
   Port ( clk   : in std_logic;
          x1    : in std_logic;
          x2    : in std_logic;
          x3    : in std_logic;
          p1    : inout std_logic;
          p2    : inout std_logic
        );
end top_module;

architecture Behavioral of top_module is

        COMPONENT r_edge_detect
        PORT(
                input : IN std_logic;
                clk : IN std_logic;
                output : OUT std_logic
                );
        END COMPONENT;

        COMPONENT initial
        PORT(
                x1 : IN std_logic;
                x2 : IN std_logic;
                x3 : IN std_logic;
                p1 : INOUT std_logic;
                p2 : INOUT std_logic
                );
        END COMPONENT;

        signal s1, s2, s3 : std_logic;

begin

        Inst_r_edge_detect1: r_edge_detect PORT MAP(
                input =>x1 ,
                clk =>clk ,
                output =>s1
        );

        Inst_r_edge_detect2: r_edge_detect PORT MAP(
                input =>x2 ,
                clk =>clk ,
                output =>s2
        );
```

```
        Inst_r_edge_detect3: r_edge_detect PORT MAP(
                input =>x3 ,
                clk =>clk ,
                output =>s3
        );

        Inst_initial: initial PORT MAP(
                x1 =>s1 ,
                x2 =>s2 ,
                x3 =>s3 ,
                p1 =>p1 ,
                p2 =>p2
        );

end Behavioral;
```

BÖLÜM VI

DÖRT KATLI MODEL ASANSÖR ve I/O TAMPON KARTI

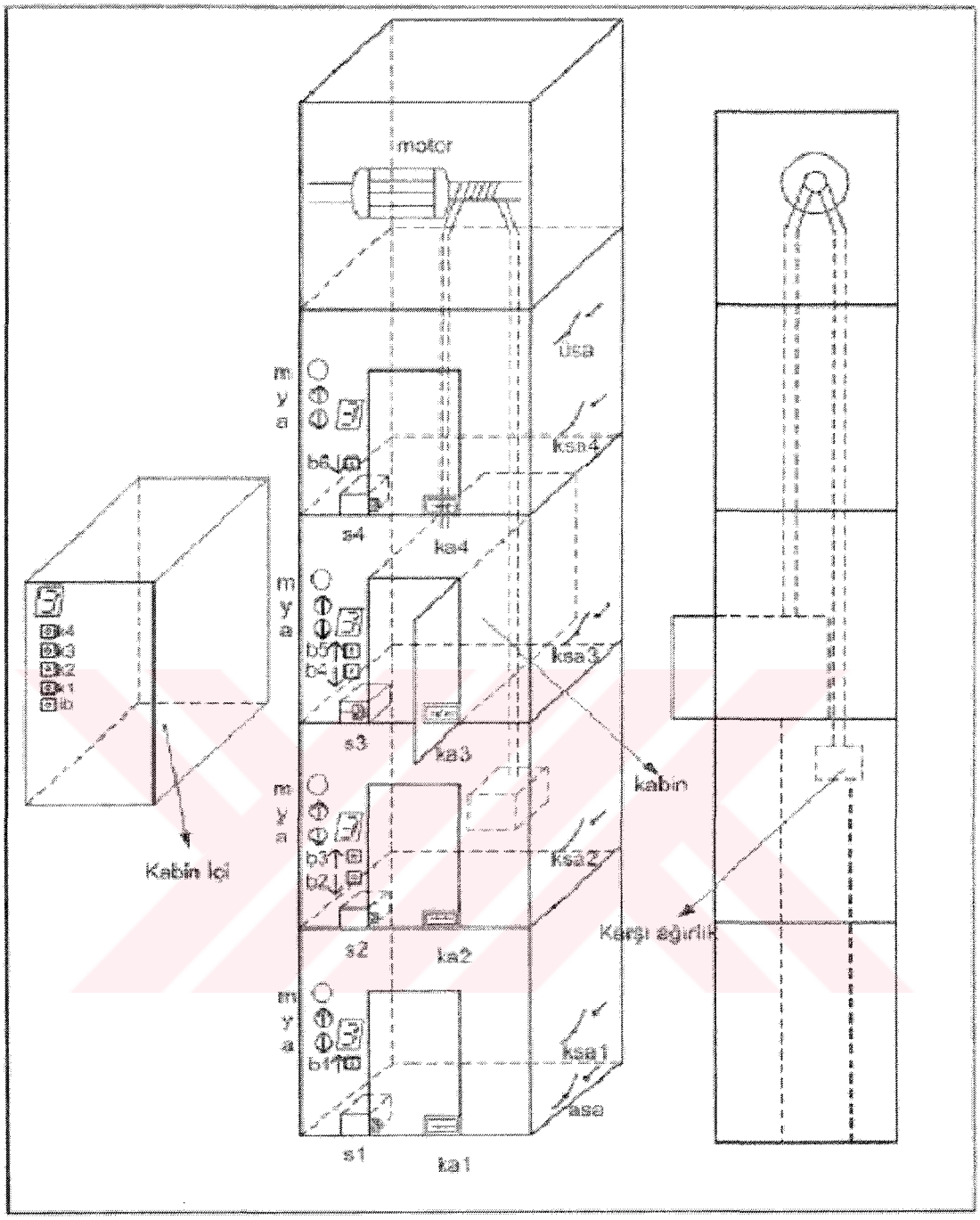
Günümüzde çok katlı apartmanların yapılması ile birlikte dairelere iniş çıkışlar çok yorucu olmaya başlamıştır. Bu nedenle apartmanlarda asansör kullanımı yaygınlaşmıştır. İlk çıkan asansörlerde sadece iniş çıkışlar önemli iken günümüzde güvenli bir şekilde iniş çıkışlar önem kazanmaya başlamıştır. Bu çalışmada kullanılan model asansör, her türlü güvenlik önlemini sağlayacak şekildedir. Eski sistem asansörler kontaktörlerle kontrol edilirken günümüzde artık mikrodenetleyicilerle veya PLC'lerle kontrol edilmektedir.

Model asansörün Digilent D2-SB kartı ile kontrol edilebilmesi için bir I/O tampon kartı tasarlanmıştır. Bu kart D2-SB kartının kullandığı LVTTL standardını TTL'ye dönüştürmek için kullanılmaktadır.

Bu bölümde dört katlı model asansör kısaca incelenecektir. Daha sonra tasarlanan I/O tampon kartı hakkında bilgi verilecektir.

6.1. Model Asansör

Kullanılan model asansör dört kattan oluşmaktadır. Şekil 6.1'de model asansörün şekli ve bu modelde kullanılan elemanlar, bu elemanlara verilen semboller görülmektedir.



Şekil 6.1. Dört katlı model asansör

Şekil 6.1'de görülen dört katlı model asansörde, kullanılan elemanlara verilen sembollerin tanımları Tablo 6.1'de verilmiştir.

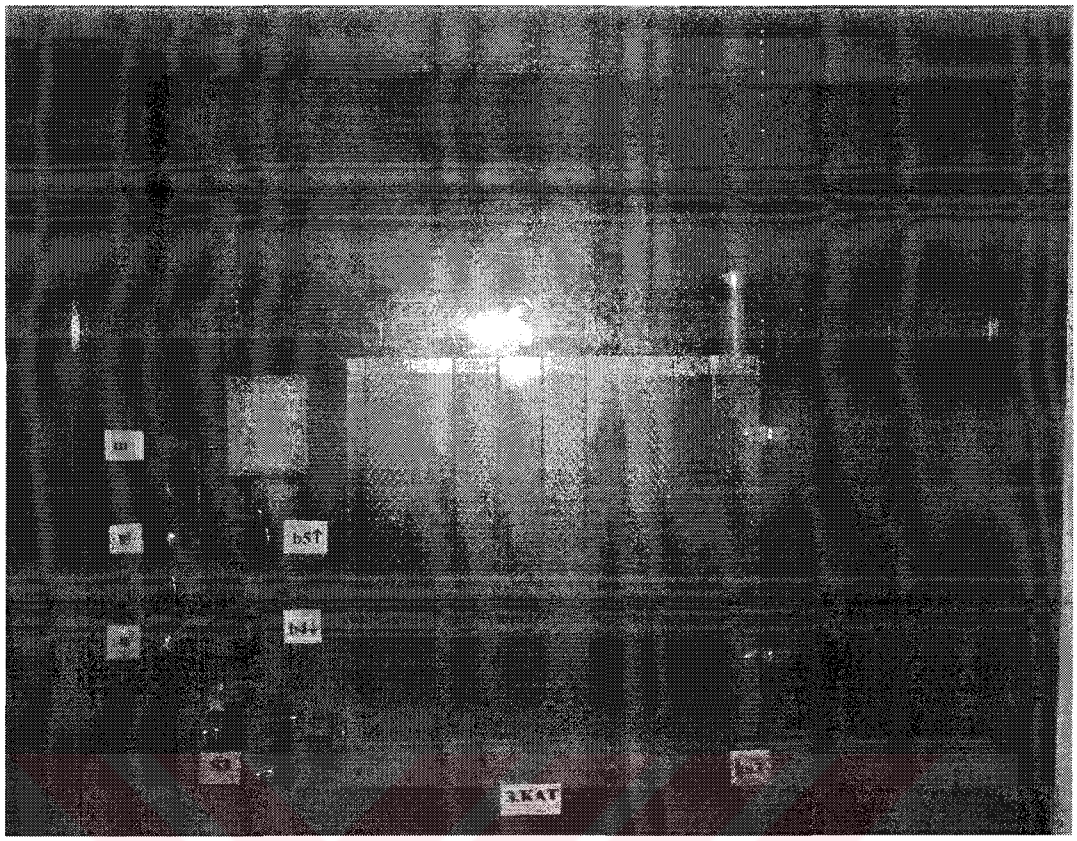
Tablo 6.1. Model asansörde kullanılan sembollerin tanımları

SEMBOLLER	TANIMLARI
ka1, ka2, ka3 ve ka4	kapı anahtarları
s1, s2, s3 ve s4	selenoid valfler
b2, b4 ve b6	katlardaki aşağı yön butonları
b1, b3 ve b5	katlardaki yukarı yön butonları
ksa1, ksa2, ksa3 ve ksa4	kat sınır anahtarları
k1, k2, k3, k4	kabin içindeki butonlar
asa	alt sınır anahtarı
üsa	üst sınır anahtarı
m	meşgul lambası
a	kabin aşağı gidiyor lambası
y	kabin yukarı gidiyor lambası
ib	imdat butonu

Bu model asansörün bir katında kullanılan elemanlar, Fotoğraf 6.1'de görülmektedir.

Her katta kapı güvenliğini sağlamak amacıyla kullanılan **selenoid valfler** bulunmaktadır. Bu selenoid valfler enerjilenmediğinde itili konumda kalarak kapıyı kilitli tutmakta ve kabin kat hizasında değilken kapının açılmasına müsaade etmemekte, kabin kat hizasına geldiğinde ise enerjilenip konum değiştirerek (yani çekili konuma geçerek) kapının açılmasına müsaade edip kabine girilmesine ya da kabinden çıkılmasına imkan tanımaktadır. Katlarda bulunan bu selenoidler 12 V DC ile çalışmaktadır.

Her katta bir **kat sınır anahtarı** bulunmaktadır. Bu anahtarlar kabinin kat hizasına gelip gelmediğini anlamak için kullanılmıştır. Bu anahtarlar normalde açık kontağa sahip olup kabin kat hizasına geldiğinde açık kontak kapanarak kabinin katta olduğu sinyalini verir. Kabin kata geldiğinde kat sınır anahtarından +5 V DC katta değilken 0 V alınmaktadır.



Fotoğraf 6.1. Model asansörün 3. katında kullanılan elemanlar

Her katın kapısında **kapı anahtarları** bulunur. Kapılarda bulunan kapı anahtarları sayesinde kapının açık mı kapalı mı olduğu anlaşılır. Kapı açık iken 0 V kapalı iken +5 V DC alınmaktadır.

En alt katta **alt sınır anahtarı** kullanılmıştır. Bu anahtar; kabinin bu seviyenin altına inmesini engellemek için kullanılmıştır. Alt sınır anahtarı da normalde açık bir kontağa sahip olup kabin bu sınıra ulaştığında açık kontak kapanacaktır. Kabin bu seviyeye geldiğinde alt sınır anahtarından +5 V DC diğer durumlarda 0 V alınmaktadır.

En üst katta **üst sınır anahtarı** kullanılmıştır. Bu anahtar; alt sınır anahtarı ile aynı mantıkta çalışmaktadır. Yani kabinin bu seviyenin üstüne çıkmasına engel olmak maksadıyla kullanılmıştır. Kabin bu seviyeye geldiğinde üst sınır anahtarından +5 V DC diğer durumlarda 0 V alınmaktadır.

Her katta **meşgul lambası** kullanılmıştır. Bu lamba (LED) kabin hareket halindeyken, ya da kapılardan birisi açıkken yani kabine biniliyor ya da iniliyorken katlara kabinin kullanılmakta olduğu sinyalini iletmektedir.

Her katta **kabin aşağı yönde gidiyor** ya da **kabin yukarı yönde gidiyor lambaları** (LED'leri) kullanılmıştır. Bu lambalar kabinin hangi yönde hareket ettiği sinyalini katlara iletmektedir.

Her katta **aşağı yön** ve **yukarı yön çağrı butonları** bulunmaktadır. Katlardan kabini çağırmak ve hangi yönde gidileceği sinyalini iletmek maksadıyla kullanılmışlardır. Bunun sebebi, model asansörün toplamalı ve toplamasız olarak iki ayrı şekilde çalışmaya uygun tasarlanmış olmasıdır. Örneğin kabin dördüncü kattan aşağı yönde giderken ikinci kattan aşağı yön çağrı butonuna basıldığına ait sinyal geldiğinde kabin ikinci katta duracak oradaki kişiyi alacak ve tekrar aşağı yönde hareket edecektir. Aynı olay asansör kabini yukarı yönde hareket ederken de gerçekleşecektir. Çağrı butonlarına basılı iken +5 V DC basılmadığı durumda ise 0 V alınmaktadır.

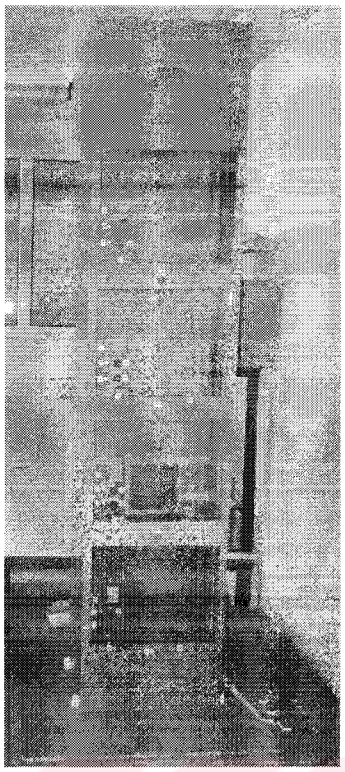
Kabin içerisinde **kat butonları** bulunmaktadır. Bunlar kabin içerisindeki kişilerin kaçıncı kata gitmek istediklerini belirtmeleri maksadıyla kullanılmışlardır. Kat butonlarına basılı iken +5 V DC basılmadığı durumda ise 0 V alınmaktadır.

Her katta ve kabin içerisinde 7 **parçalı gösterge** (seven segment display) kullanılmıştır. Bunlar kabinin kaçıncı katta olduğunu katlara ve kabin içindeki kişilere iletmek maksadıyla kullanılmışlardır.

Yine kabin içerisinde **imdat butonu** bulunmaktadır. Bu buton herhangi bir nedenle asansör kabininin katlar arasında kalması durumunda, bunu iletmek maksadıyla kullanılmıştır.

Model asansörün üstündeki ayrı bir bölümde ise kabinin aşağı veya yukarı yönde hareketini sağlayan bir 12 V'luk DC **motor** bulunmaktadır.

Kullanılan dört katlı model asansörün resmi, Fotoğraf 6.2'de görülmektedir.



Fotoğraf 6.2. Dört katlı model asansörün görünümü

6.1.1. Ara bağlantı kartı

Model asansör farklı DC gerilim seviyelerinde çalışacak şekildedir. Model asansörde bulunan ara bağlantı kartı sayesinde kumanda işlemi 2.5 V, 5 V veya 24 V DC gerilimlerinden herhangi birisi ile gerçekleştirilebilir. Böylece bir PLC, bir mikrodenetleyici veya bir FPGA ile model asansörün kumanda edilmesi mümkün olabilmektedir. Ayrıca ara bağlantı kartında, motor, selenoidler ve displayler fazla akım çektikleri için gerekli olan sürücü devreleri de bulunmaktadır.

6.2. I/O Tampon Kartı

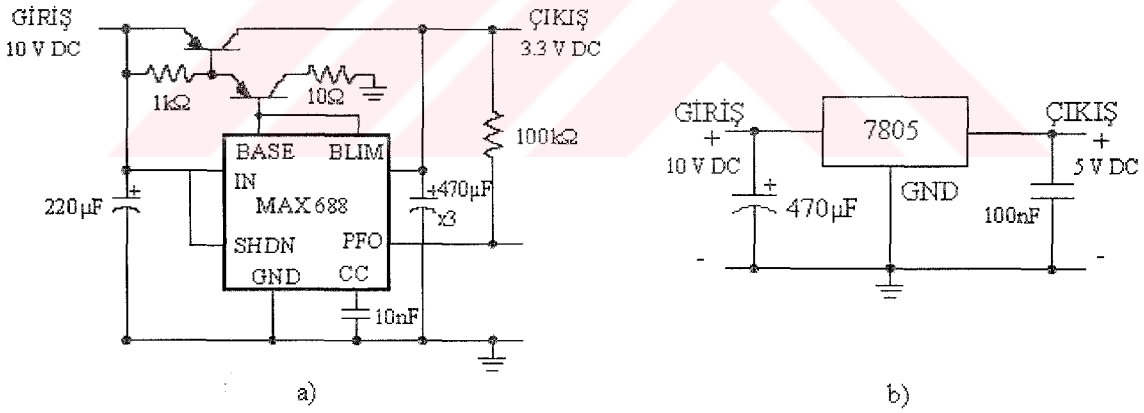
Programlanabilir ürünler çevre elemanlar ile bağlantılarını I/O pinleri yardımıyla kurarlar. Bu pinlerden dış ortama aktarılan veya dış ortamdan alınan bilgi anlamlandırılırken belirli standartlar kullanılmaktadır. Bu standartlar genellikle üretici firmalarca belirlenmektedir. Xilinx firmasının üretmiş olduğu XC2S200E FPGA'sı için kullanılan bazı I/O standartları Tablo 3.2'de verilmiştir.

Bu uygulamada FPGA'nın LVTTL standardı kullanılmıştır. Digilent D2-SB uygulama geliştirme kartı ile TTL uyumlu sistemler arasındaki LVTTL-TTL

uyumsuzluğunu gidermek ve sistemin akım kapasitesini artırmak amacıyla, I/O tampon kartı tasarlanmıştır. Bu kart tasarlanırken, kartın deneme kartı olarak da kullanımını sağlamak amacı ile temel giriş/çıkış devreleri de kart üzerine eklenmiştir. Ayrıca 24 V'luk harici sensör bilgilerini de FPGA ile algılayabilmek için 24 V'luk gerilim de 3.3 V'a dönüştürülmüştür.

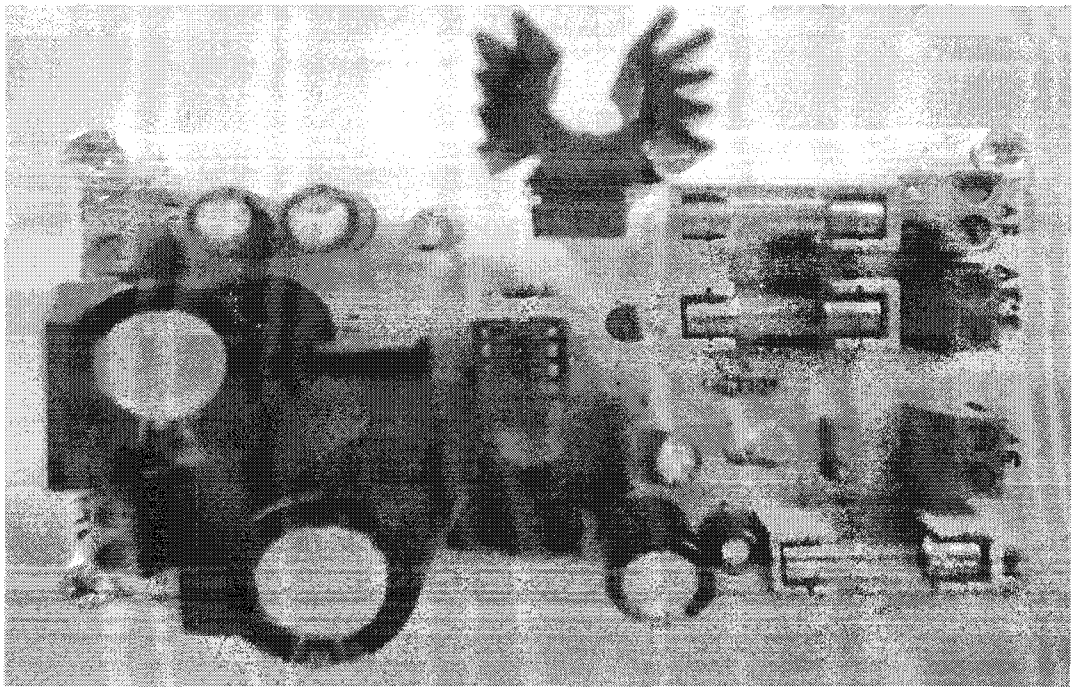
6.2.1. I/O tampon kartı besleme devresi

I/O tampon kartı üzerine eklenilen özelliklerin ve FPGA'nın besleme gerilimleri göz önüne alınarak, hazırlanacak olan deneme kartının enerjisini sağlamak üzere toprakları birbirinden ayrılmış ve regüle edilmiş 3.3 V ve 5 V'luk iki gerilime ihtiyaç duyulmaktadır. Bu gerilimleri elde etmek için iki farklı transformatör kullanılarak 220 V'luk şebeke gerilimi; 12 V'luk ve 9 V'luk AC gerilimlere çevrilmiştir. Bu giriş gerilimleri yardımıyla 7805 ve MAX688 entegre gerilim regülatörleri kullanılarak 5 V'luk ve 3.3 V'luk regüleli gerilimler elde edilmiştir. Şekil 6.2'de bu regüle devreleri görülmektedir.



Şekil 6.2. I/O tampon kartı besleme devresindeki a) 3.3 V'luk regüle devresi
b) 5 V'luk regüle devresi

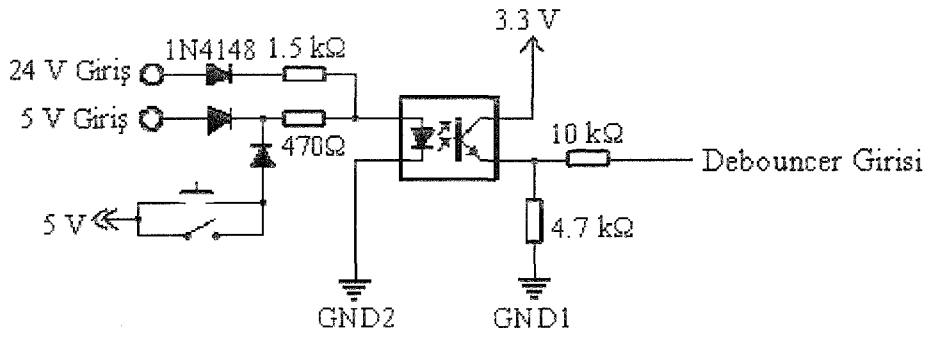
Tasarlanan I/O tampon kartı besleme devresi, Fotoğraf 6.3'te görülmektedir.



Fotoğraf 6.3. Tasarlanan I/O tampon kartı besleme devresi

6.2.2. I/O tampon kartı temel giriş devreleri

D2-SB uygulama geliştirme kartındaki bazı pinler yalnızca giriş olacak şekilde düzenlenmiştir. Giriş işlemini gerçekleştirmek üzere tasarlanılan deneme kartının giriş kısmı için push-buton ve DIP switchten oluşan devre tasarlanmıştır. Ayrıca 24 V ve 5 V değerinde çıkış verebilen sensör bilgilerini de FPGA'nın algılayabilmesini sağlamak için gerekli olan düzenlemeler yapılmıştır. Yukarıda anlatıldığı gibi FPGA LVTTL yani 3.3 V'u lojik "1" kabul eden standardı kullanmaktadır. Giriş olarak uygulanılan bu farklı değerlere sahip gerilimler bir optokuplör yardımı ile FPGA'nın algılayabileceği 3.3 V'luk gerilime çevrilmektedir. Şekil 6.3'te bir giriş için gerekli olan devre elemanlarını içeren devre şeması görülmektedir.



Şekil 6.3. Bir pine ait giriş devresi

Şekil 6.3'ten de görüldüğü gibi sisteme ait olan iki farklı gerilim mevcuttur. Bunlar 5 V'luk ve 3.3 V'luk gerilimlerdir. 3.3 V'luk gerilimin toprak bağlantısı GND1 ve 5 V'luk gerilimin toprak bağlantısı ise GND2 ile gösterilmiştir. Bu iki gerilimin toprak bağlantıları birbirinden ayrılmıştır. Harici girişlerden oluşan 24 V'luk ve 5 V'luk bir gerilim veya buton ile dip switchlere basılması durumunda optokuplörün optik verici kısmından akım geçmesiyle foto transistör iletime geçer ve çıkışta 3.3 V'luk bir gerilim görülür. Böylelikle TTL-LVTTL gerilim dönüşümü sağlanmış olur.

6.2.2.1. Debouncer devresi

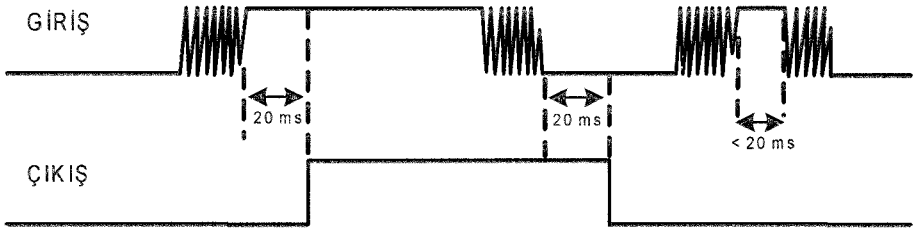
Mikroişlemci ve mikrodenetleyici temelli sistemlerde sıkça karşılaşılan sorunların başında giriş sinyalindeki gürültü gelmektedir. Bu gürültünün bir çok kaynağı mevcuttur. Giriş sinyali bir mekanik buton ise kontak arklarının neden olduğu gürültü istenmeyen sonuçlar doğuracaktır. Benzer şekilde sensör bilgilerinde de çeşitli sebeplerden kaynaklanan gürültüler bulunmaktadır. Bu tip anahtarlama gürültülerinin kaynağı ne olursa olsun hemen hemen aynı karakteristiği gösterirler.

Bir anahtarın açma ve kapama işlemleri sırasında gürültüler meydana gelmektedir. Bu gürültülerin yaklaşık şekli Şekil 6.4'te verilmiştir. Bu gürültüler nedeniyle bir defa basılan buton, sistem tarafından birkaç defa basılmış gibi işleme konulmaktadır. Örneğin buton ileri sayıcının tetikleme girişine bağlanmış ve butona her basıldığında sayıcının bir ileri sayması isteniyorsa gürültü nedeniyle sayıcının sayma değeri 1'den fazla olacaktır.

Şekil 6.4. Bir anahtarlama gürültüsüne ait dalga şekli

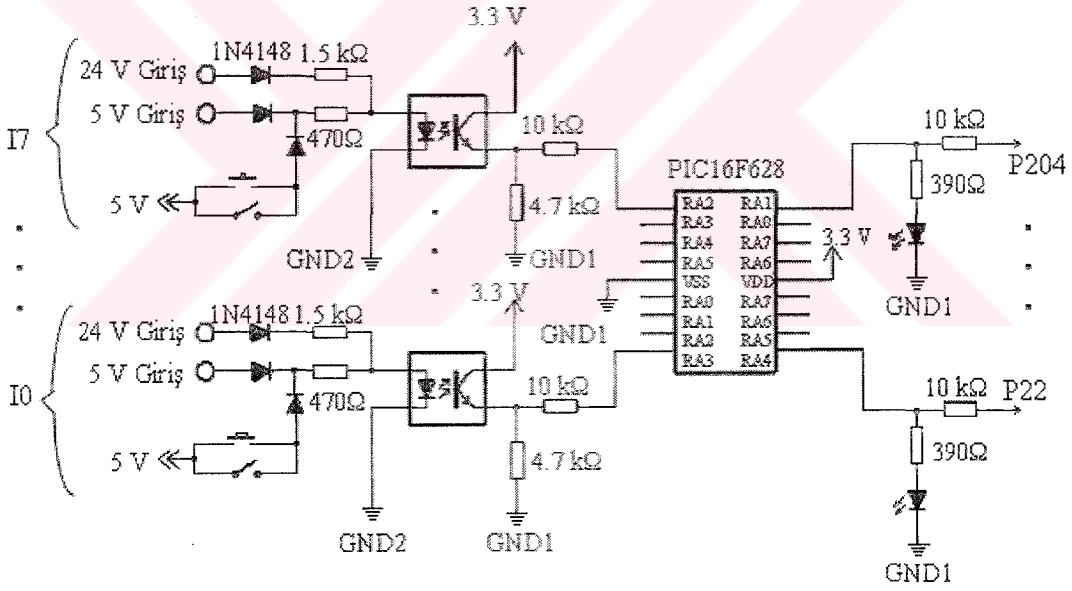
Gürültüyü etkin bir biçimde giderebilmek için programlanabilir bir elaman olan PIC ailesine mensup PIC16F628 kullanılmıştır. Yazılan programla, tuşa basıldığında ilk önce dalgalanmaların sona ermesi beklenmektedir. Dalgalanmalar bittikten sonra bir 20 ms'lik süre daha beklenerek çıkış sinyali lojik "1" yapılır. Benzer şekilde tuşun bırakılması sırasında da dalgalanmaların sona ermesi beklenir. Dalgalanmalar bittikten sonra bir 20 ms'lik süre daha beklenerek çıkış sinyali lojik "0" yapılır. Eğer tuş

sinyalinin gürültüsüz kısmının süresi 20 ms'den az ise bu durumda çıkışta hiç bir değişiklik yapılmaz. Örnek bir giriş sinyali için tasarlanan debouncer çıkış sinyali Şekil 6.5'de verilmiştir.



Şekil 6.5. Örnek bir giriş sinyali için çıkış sinyali

FPGA deneme kartı için tasarlanılan, TTL-LVTTL seviye dönüştürücü devre ve debouncer devresi eklenmiş 8 giriş içeren temel giriş devresi Şekil 6.6'da verilmiştir. Tasarlanan I/O tampon devresi 16 giriş içermektedir. Şekil 6.6'da verilen devreden iki adet kullanılmıştır.

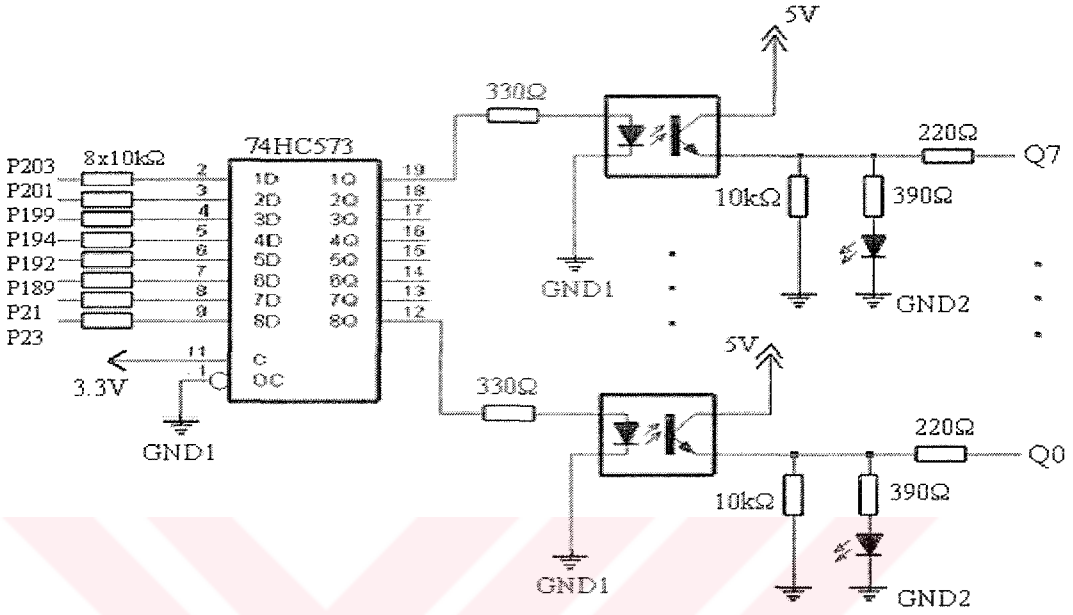


Şekil 6.6. Temel giriş devresi

6.2.3. I/O tampon kartı temel çıkış devreleri

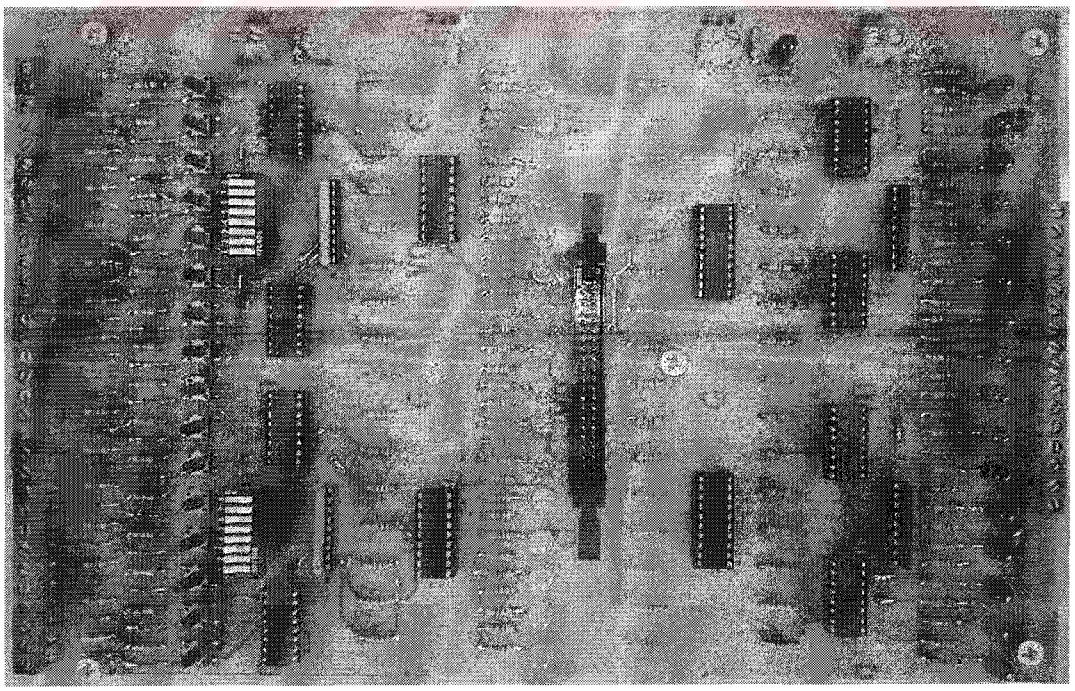
FPGA deneme kartı için çıkış katı tasarımı yapılırken iki önemli özellik göz önünde bulunduruldu. Bunlar giriş kısmında olduğu gibi, LVTTL-TTL gerilim dönüşümü ve FPGA'nın çıkış akım kapasitesinin yükseltilmesidir. LVTTL-TTL gerilim dönüşümünü sağlamak için giriş katında olduğu gibi optokuplörlü devre kullanıldı. Akım kapasitesini yükseltebilmek için ise, 74 serisi 74HC573 D-tipi latch çıkışı sürekli

aktif olacak şekilde kullanılarak tampon (buffer) olarak kullanıldı. Bu kart için toplam 16 çıkış gerçekleştirildi. Sekiz çıkış için tasarlanan devre şeması Şekil 6.7'de verilmiştir. Şekildeki devreden iki adet kullanılmıştır. Ayrıca çıkış bilgisini görebilmek amacı ile her çıkışa LED eklenmiştir.



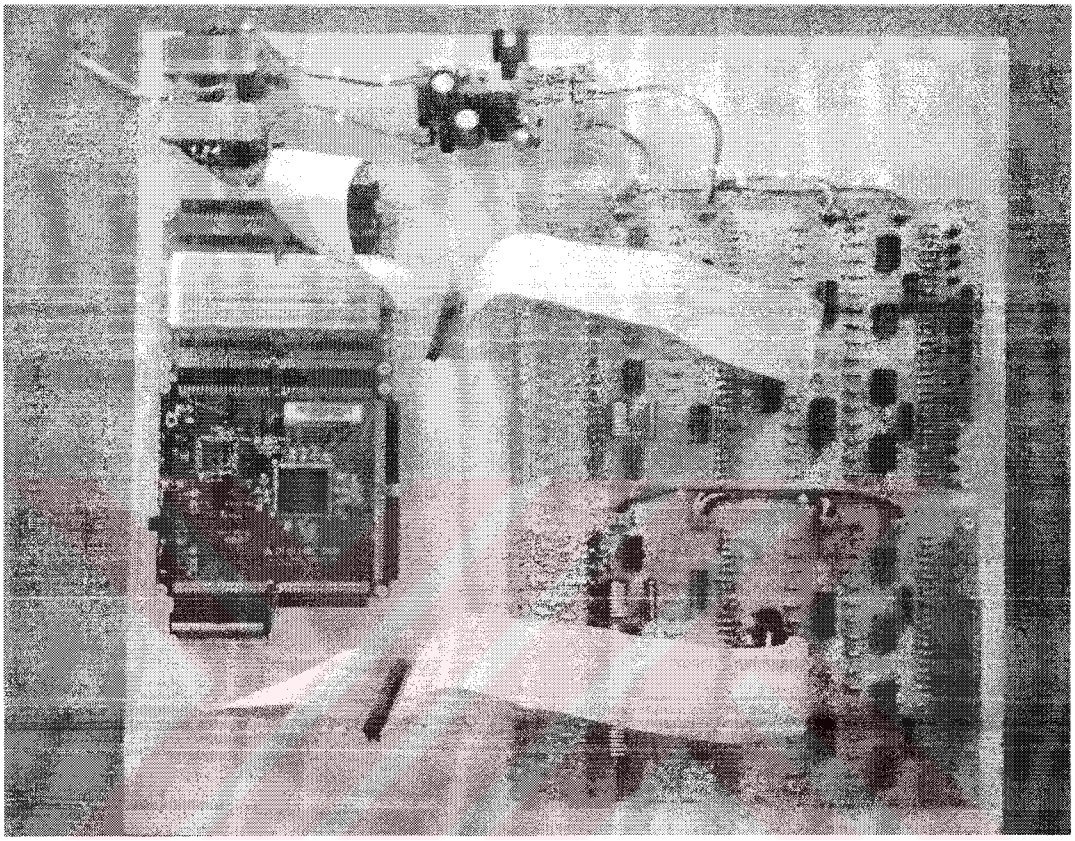
Şekil 6.7. Temel çıkış devresi

Tasarlanan I/O tampon kartı, Fotoğraf 6.4'te görülmektedir.



Fotoğraf 6.4. Tasarlanan I/O tampon kartının görünümü

Model asansörün kontrolünde kullanılan butonlar, kat sınır anahtarları ve kapı anahtarlarının sayısı 16'dan fazladır. Bu nedenle tasarlanan I/O tampon kartından iki adet yapılmıştır. I/O tampon kartlarının, D2-SB kartı ile olan bağlantıları Fotoğraf 6.5'de görülmektedir.



Fotoğraf 6.5. Tasarlanan I/O tampon kartlarının, D2-SB kartı ile olan bağlantıları

I/O tampon kartlarının, D2-SB kartındaki A2 ve C2 konnektörleri ve XC2S200E FPGA'sının pinleri ile olan bağlantıları Tablo 6.2'de görülmektedir.

Tablo 6.2. I/O tampon kartlarının, D2-SB kartındaki A2 ve C2 konnektörleri ve XC2S200E FPGA'sının pinleri ile olan bağlantıları

	C2		
Pin	Sinyal	FPGA Pin	I/O
1	GND		
2	VU		
3	VCC33		
4	PCI01	p23	Q0
5	PCI02	p22	I0
6	PCI03	p21	Q1
7	PCI04	p20	I15
8	PCI05	p18	Q15
9	PCI06	p17	I14
10	PCI07	p16	Q14
11	PCI08	p15	I13
12	PCI09	p11	Q13
13	PCI010	p10	I12
14	PCI011	p9	Q12
15	PCI012	p8	I11
16	PCI013	p7	Q11
17	PCI014	p6	I10
18	PCI015	p5	Q10
19	PCI016	p4	I9
20	PCI017	p3	Q9
21	PCI018	p206	I8
22	MC2DB0	p205	Q8
23	MC2DB1	p204	I7
24	MC2DB2	p203	Q7
25	MC2DB3	p202	I6
26	MC2DB4	p201	Q6
27	MC2DB5	p200	I5
28	MC2DB6	p199	Q5
29	MC2DB7	p198	I4
30	MC2ASTB	p194	Q4
31	MC2DSTB	p193	I3
32	MC2WRT	p192	Q3
33	MC2WAIT	p191	I2
34	MC2RST	p189	Q2
35	MC2INT	p188	I1
36			
37			
38			
39	GCLK1	p77	
40	GND		

	A2		
Pin	Sinyal	FPGA Pin	I/O
1	GND		
2	VU		
3	VCC33		
4	PAI01	p162	Q16
5	PAI02	p161	I16
6	PAI03	p160	Q17
7	PAI04	p152	I31
8	PAI05	p151	Q31
9	PAI06	p150	I30
10	PAI07	p149	Q30
11	PAI08	p148	I29
12	PAI09	p147	Q29
13	PAI010	p146	I28
14	PAI011	p145	Q28
15	PAI012	p141	I27
16	PAI013	p140	Q27
17	PAI014	p139	I26
18	PAI015	p138	Q26
19	PAI016	p136	I25
20	PAI017	p135	Q25
21	PAI018	p134	I24
22	MA2DB0	p133	Q24
23	MA2DB1	p132	I23
24	MA2DB2	p129	Q23
25	MA2DB3	p127	I22
26	MA2DB4	p126	Q22
27	MA2DB5	p125	I21
28	MA2DB6	p123	Q21
29	MA2DB7	p122	I20
30	MA2ASTB	p121	Q20
31	MA2DSTB	p120	I19
32	MA2WRT	p116	Q19
33	MA2WAIT	p115	I18
34	MA2RST	p114	Q18
35	MA2INT	p113	I17
36			
37			
38			
39	GCLK0	p80	
40	GND		

BÖLÜM VII

ASANSÖR KONTROLÜ İÇİN ELDE EDİLEN BOPN MODELLERİ

Günümüzde konutlarda ve işyerlerinde kullanılan asansörler toplamasız ve toplamalı olarak kontrol edilmektedir. Genellikle toplamasız kontrol sistemleri konutlarda, toplamalı kontrol sistemleri ise işyerlerinde tercih edilmektedir. Toplamasız kontrol sistemlerinde, kabin hareket halindeyken, katlardan yapılan çağrılar veya kabin içinden yapılan komutlar dikkate alınmaz. Toplamalı kontrol sistemlerinde ise, kabin hareket halindeyken yapılan çağrılardan, sadece kabinin hareket yönünde olan ve aynı yönü ifade edenler dikkate alınmaktadır. Toplamalı kontrol sistemlerinde, kabin hareket halindeyken, kabin içindeki butonlar aktif değildir.

Bu bölümde, bir önceki bölümde anlatılan dört katlı model asansörün, toplamasız ve toplamalı olarak kontrolü için tasarlanan Basit Otomasyon Petri net (BOPN) modelleri incelenecektir. Tasarlanan BOPN modellerinde; mevkiler kabinin konumunu belirtirken, geçişlerin tetikleme şartları da kabin içinde veya katlarda bulunan butonlar veya kat sınır anahtarlarının durumlarını ifade etmektedir.

Toplamasız ve toplamalı BOPN modelleri, ayrı ayrı VHDL modüllerine dönüştürülmüştür (toplamasız.vhd ve toplamalı.vhd). Ayrıca kabinin katta bekleme süresini (t_d=2.68s) belirleyen zaman gecikmesi modülü (t_delay.vhd) oluşturulmuştur. Toplamasız ve toplamalı kontrol sistemi için hazırlanan VHDL modülleri, zaman gecikmesi modülü ile ayrı ayrı olacak şekilde bir başka VHDL modülünde yapısal olarak birleştirilmiştir (elevator1.vhd ve elevator2.vhd). Hazırlanan bu modüllerdeki VHDL kodları ekler bölümünde verilmiştir.

Toplamasız ve toplamalı asansör kontrolü için elde edilen VHDL modülleri (elevator1.vhd ve elevator2.vhd), XC2S200E FPGA'sında ayrı ayrı sentezlenmiş ve Bölüm VI'da anlatılan model asansörde uygulanarak gerçekleştirilmiştir.

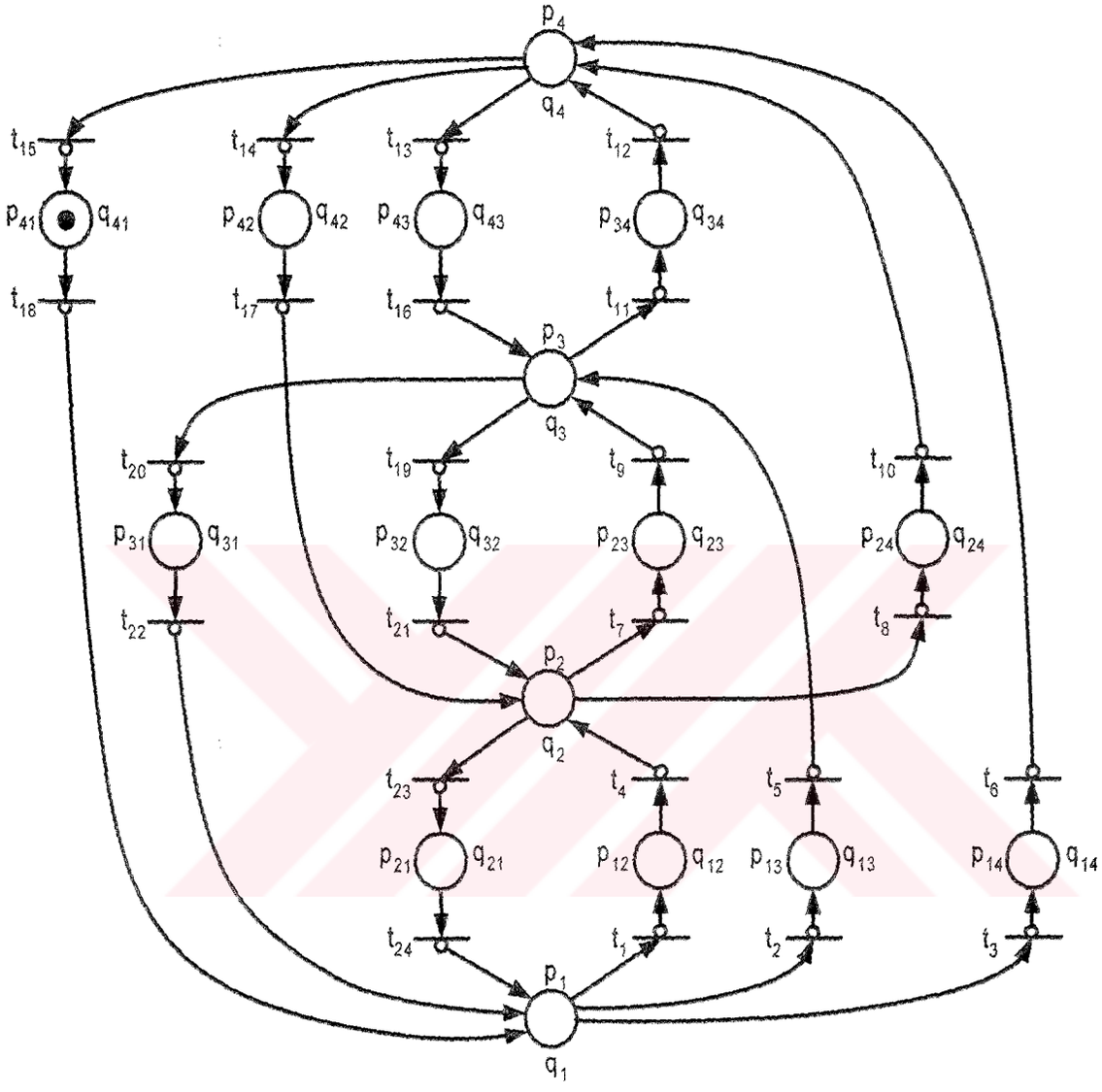
7.1. Toplamasız BOPN Modeli

Toplamasız BOPN modeli tasarlanırken, öncelikle 4 katlı bir asansör kontrol sisteminin toplamasız çalışma biçimi göz önüne alınmıştır. Bu sistemin başlıca özellikleri, maddeler halinde aşağıda belirtilmiştir.

- Kabin katta dururken, kabin içindeki butonlar sürekli aktif durumdadırlar. Böylece kabine binen bir kişi, kapı kapanmadan bir butona bastığında; kabin, kapı kapandıktan sonra, basılan butonun belirttiği kata hareket edecektir.
- Kabin katta dururken, katlarda bulunan butonlar, kabinin katta bekleme süresi dolduğunda ve tüm katların kapısı kapalı olduğunda aktiftirler.
- Kabin katta dururken, aktif olan butonlardan, ilk önce basılan dikkate alınır.
- Kabin hareket halindeyken, kabin içindeki ve katlardaki butonlar aktif değildirler.
- Tüm katların kapısı kapalı olduğunda, kabin hareket etmeye başlar. Kabin hareket halindeyken, kapılardan herhangi birisi açılırsa, kabin olduğu yerde durur.
- Kabin alt veya üst sınır anahtarlarından herhangi birisinin konumunu değiştirirse, kabin olduğu yerde durur.

Yukarıdaki özelliklerden yararlanılarak, 4 katlı model asansör için, Şekil 7.1'de görülen toplamasız BOPN modeli tasarlanmıştır. Bu BOPN modelinde, 16 mevki $P=\{p_1, p_2, p_3, p_4, p_{12}, p_{13}, p_{14}, p_{23}, p_{24}, p_{34}, p_{21}, p_{31}, p_{32}, p_{41}, p_{42}, p_{43}\}$ ve 24 geçiş $T=\{t_1, t_2, t_3, \ldots, t_{24}\}$ bulunmaktadır. Toplamasız BOPN modeli tasarlanırken, kabinin konumunu belirtmek amacıyla çeşitli mevkiler kullanılmıştır. Kabinin katlarda bulunduğu, p_1, p_2, p_3, p_4 mevkileri ile temsil edilmektedir. p_1 mevkisinde bir jeton olması, kabinin 1. katta bulunduğu anlamına gelir. Diğer mevkilerin ilk numaraları hareket edilecek/edilen katı, son numaraları da gidilecek/gidilen katı belirtmektedir. Kabinin yukarı çıkarken, hangi katlar arasında bulunduğunu belirtmek için p_{12}, p_{13}, p_{14}, p_{23}, p_{24}, p_{34} mevkileri kullanılmıştır. Örneğin p_{12} mevkisinde bir jeton bulunması, kabinin 1. kattan 2. kata doğru hareket edeceğini veya etmekte olduğunu ifade etmektedir. Kabinin aşağı inerken, hangi katlar arasında bulunduğunu belirtmek için p_{21}, p_{31}, p_{32}, p_{41}, p_{42}, p_{43} mevkileri kullanılmıştır. Örneğin p_{21} mevkisinde bir jeton

bulunması, kabinin 2. kattan 1. kata gideceğini veya gitmekte olduğunu ifade etmektedir.



Şekil 7.1. Gerçekleştirilen toplamasız BOPN modeli

Kabinin katlar arasında olduğunu belirten mevkilerin, girişlerinde bulunan geçişlere (t_1, t_2, t_3, t_7, t_8, t_{11}, t_{13}, t_{14}, t_{15}, t_{19}, t_{20}, t_{23}) tetikleme şartları olarak, kabin içindeki butonlar (k1, k2, k3, k4) veya katlardaki butonların aşağı yönlü olanları (b1, b2, b4, b6) atanmıştır. Kabin katta iken, kabin içindeki butonlar sürekli aktif durumdadırlar. Örneğin kabine binen bir kişi, kabin kapısı açık iken bir butona basarsa; kabin, kapı kapandıktan sonra basılan butonun belirttiği kata hareket edecektir. Katlardaki butonların aktif olabilmesi için ise, kabinin katta bekleme süresinin (t_d=2.68s) dolması ve tüm katların kapısının (ka1, ka2, ka3, ka4) kapalı olması gerekmektedir. Kabinin katta bekleme süresi dolduğunda, tüm katların kapısı kapalı ise,

kabin ilk basılan butonun belirttiği kata hareket etmeye başlayacaktır. Kabin hareket etmeye başladığında, kabinin hareket ettiği katın selenoidi itili konuma geçerek kat kapısının açılması engellenir. Örneğin kabin 1. katta iken t_d süresi kadar bekledikten sonra, tüm katların kapısı kapalı olduğunda, hangi butona daha önce basılırsa kabin o kata hareket edecektir. Bu esnada 1. katın selenoidi (s1) itili konuma geçerek kat kapısının açılması engellenir.

Tablo 7.1. Toplamasız BOPN modelindeki geçişler ve bu geçişlerin tetikleme şartları

Geçiş	Tetikleme şartı	Tetikleme şartını meydana getiren sinyaller
t_1	χ_1	k2 or (t_d and ka and b2)
t_2	χ_2	k3 or (t_d and ka and b4)
t_3	χ_3	k4 or (t_d and ka and b6)
t_4	χ_4	ksa2
t_5	χ_5	ksa3
t_6	χ_6	ksa4
t_7	χ_7	k3 or (t_d and ka and b4)
t_8	χ_8	k4 or (t_d and ka and b6)
t_9	χ_9	ksa3
t_{10}	χ_{10}	ksa4
t_{11}	χ_{11}	k4 or (t_d and ka and b6)
t_{12}	χ_{12}	ksa4
t_{13}	χ_{13}	k3 or (t_d and ka and b4)
t_{14}	χ_{14}	k2 or (t_d and ka and b2)
t_{15}	χ_{15}	k1 or (t_d and ka and b1)
t_{16}	χ_{16}	ksa3
t_{17}	χ_{17}	ksa2
t_{18}	χ_{18}	ksa1
t_{19}	χ_{19}	k2 or (t_d and ka and b2)
t_{20}	χ_{20}	k1 or (t_d and ka and b1)
t_{21}	χ_{21}	ksa2
t_{22}	χ_{22}	ksa1
t_{23}	χ_{23}	k1 or (t_d and ka and b1)
t_{24}	χ_{24}	ksa1
t_d : Kabinin katta bekleme süresi		
ka = ka1 and ka2 and ka3 and ka4		

Kabinin katlar arasında olduğunu belirten mevkilerin, çıkışlarında bulunan geçişlere (t_4, t_5, t_6, t_9, t_{10}, t_{12}, t_{16}, t_{17}, t_{18}, t_{21}, t_{22}, t_{24}) tetikleme şartları olarak kat sınır anahtarları atanmıştır. Örneğin kabin 1. kattan 3. kata giderken, 3. katın sınır anahtarının

(ksa3) konumu değiştiğinde, t_5 geçişi tetiklenerek p_{13}'te bulunan jeton p_3 mevkisine depolanır ve kabin durur. Bu esnada 3. katın selenoidi (s3) çekili konuma geçerek 3. katın kapısının açılmasına müsaade eder. Tablo 7.1'de toplamasız BOPN modelindeki geçişler ve bu geçişlerin tetikleme şartları görülmektedir.

Gerçekleştirilen toplamasız BOPN modelindeki mevkilere çeşitli aksiyonlar atanmıştır. Kabinin katlarda beklediği durumu belirten mevkilere (p_1, p_2, p_3, p_4), bulunulan katın selenoidi aksiyon olarak atanmıştır. Bu mevkilerden herhangi birisine jeton depolandığında, o katın selenoidi çekili konuma geçerek kabine binilmesine müsaade edecektir. Kabinin yukarı çıktığını belirten mevkilere (p_{12}, p_{13}, p_{14}, p_{23}, p_{24}, p_{34}), motorun yukarı (my) doğru çalışması aksiyon olarak atanmıştır. Bu mevkilerden herhangi birisine jeton depolandığında, tüm katların kapısı kapalı ise, kabin yukarı doğru hareket etmeye başlar. Kabinin aşağı indiğini belirten mevkilere (p_{21}, p_{31}, p_{32}, p_{41}, p_{42}, p_{43}), motorun aşağı (ma) doğru çalışması aksiyon olarak atanmıştır. Bu mevkilerden herhangi birisine jeton depolandığında, tüm katların kapısı kapalı ise, kabin aşağı doğru hareket etmeye başlar. Tablo 7.2'de toplamasız BOPN modelindeki mevkilere atanan aksiyonlar ve açıklamaları görülmektedir.

Tablo 7.2. Toplamasız BOPN modelindeki mevkilere atanan aksiyonlar

Mevki	Aksiyon	Açıklama
p1	q1=s1	Kabin 1. katta ve s1 selenoidi çekili durumdadır
p2	q2=s2	Kabin 2. katta ve s2 selenoidi çekili durumdadır
p3	q3=s3	Kabin 3. katta ve s3 selenoidi çekili durumdadır
p4	q4=s4	Kabin 4. katta ve s4 selenoidi çekili durumdadır
p12	q12=my	Kabin 1. kattan 2. kata hareket etmektedir
p13	q13=my	Kabin 1. kattan 3. kata hareket etmektedir
p14	q14=my	Kabin 1. kattan 4. kata hareket etmektedir
p23	q23=my	Kabin 2. kattan 3. kata hareket etmektedir
p24	q24=my	Kabin 2. kattan 4. kata hareket etmektedir
p34	q34=my	Kabin 3. kattan 4. kata hareket etmektedir
p43	q43=ma	Kabin 4. kattan 3. kata hareket etmektedir
p42	q42=ma	Kabin 4. kattan 2. kata hareket etmektedir
p41	q41=ma	Kabin 4. kattan 1. kata hareket etmektedir
p32	q32=ma	Kabin 3. kattan 2. kata hareket etmektedir
p31	q31=ma	Kabin 3. kattan 1. kata hareket etmektedir
p21	q21=ma	Kabin 2. kattan 1. kata hareket etmektedir

Kabin güvenliğini sağlamak amacıyla iki önlem alınmıştır. Bunlardan birincisi, kabin eğer alt sınır veya üst sınır anahtarlarından herhangi birisinin konumunu değiştirirse motorun durması ve UYARI sinyali vermesidir. Bu durumda model

asansörün enerjisi kesildikten sonra, kabin konumuna göre mekanik olarak yukarı veya aşağı çekilir. Daha sonra model asansörün enerjisi verilerek sistem çalışmasına devam eder. İkinci olarak da, kabin hareket halindeyken kapılardan herhangi birisi açılırsa motorun durmasıdır. Tüm kapılar kapalı olduğunda motor tekrar çalışmaya başlar ve kabin hareketine devam eder.

Şekil 7.1'deki BOPN modelinin çalışmasını inceleyecek olursak, ilk başta p_{41} mevkisinde jeton olduğu görülmektedir. Bunun nedeni, kabinin başlangıçta nerede olursa olsun 1. kata gelmesini sağlamak içindir. Kabin 1. katın sınır anahtarının konumunu değiştirdikten sonra durur ve s1 selenoidi çekili konuma geçerek kabine binilmesine müsaade eder. Bu durumda p_{41} mevkisinde bulunan jeton p_1 mevkisine depolanır. 1. katın kapısı açık iken, sadece kabin içindeki butonlar aktiftir. Kabinin katta bekleme süresi (t_d) dolduğunda tüm katların kapısı kapalı ise, hem kabin içindeki hem de katlardaki butonlar aktiftir. Bu butonlardan hangisine daha önce basılırsa kabin o kata hareket eder. Kabin harekete başladığında, çekili durumda bulunan (s1) selenoidi itili konuma geçerek kat kapısının açılması engellenir. Kabin hareket halinde iken, tüm butonlar etkisiz durumdadır. Kabin, gittiği katın sınır anahtarının konumunu değiştirdiğinde durur ve o katın selenoidi çekili konuma geçerek kabine binilmesine müsaade eder. Aşağıda, gerçekleştirilen toplamasız BOPN modelinin, yukarı çıkışlar ve aşağı inişler için çalışma biçimi anlatılmaktadır.

Yukarı çıkışlar için

- Kabin 1. katta iken $\{M(p_1)=1\}$, k2 butonuna basılırsa veya kabin katta t_d süresi kadar bekledikten sonra, tüm katların kapısı kapalı olduğunda, b2 butonuna basılırsa, t_1 geçişi tetiklenir ve p_1'de bulunan jeton p_{12}'ye depolanır. Bu durumda, tüm katların kapısı kapalı ise, q_{12} aksiyonu gerçekleşir ve kabin 2. kata hareket eder. Kabin 2. katın sınır anahtarının (ksa2) konumunu değiştirdiğinde, t_4 geçişi tetiklenir ve p_{12}'de bulunan jeton p_2'ye depolanır. Bu durumda, q_2 aksiyonu gerçekleşir, kabin 2. katta durur ve s2 selenoidi çekili konuma geçer. $(\mathbf{p_1 \rightarrow p_{12} \rightarrow p_2})$

- Kabin 1. katta iken $\{M(p_1)=1\}$, k3 butonuna basılırsa veya kabin katta t_d süresi kadar bekledikten sonra, tüm katların kapısı kapalı olduğunda, b4 butonuna

basılırsa, t_2 geçişi tetiklenir ve p_1'de bulunan jeton p_{13}'e depolanır. Bu durumda, tüm katların kapısı kapalı ise, q_{13} aksiyonu gerçekleşir ve kabin 3. kata hareket eder. Kabin 3. katın sınır anahtarının (ksa3) konumunu değiştirdiğinde, t_5 geçişi tetiklenir ve p_{13}'de bulunan jeton p_3'e depolanır. Bu durumda, q_3 aksiyonu gerçekleşir, kabin 3. katta durur ve s3 selenoidi çekili konuma geçer. ($\mathbf{p_1 \rightarrow p_{13} \rightarrow p_3}$)

- Kabin 1. katta iken {M(p_1)=1}, k4 butonuna basılırsa veya kabin katta t_d süresi kadar bekledikten sonra, tüm katların kapısı kapalı olduğunda, b6 butonuna basılırsa, t_3 geçişi tetiklenir ve p_1'de bulunan jeton p_{14}'e depolanır. Bu durumda, tüm katların kapısı kapalı ise, q_{14} aksiyonu gerçekleşir ve kabin 4. kata hareket eder. Kabin 4. katın sınır anahtarının (ksa4) konumunu değiştirdiğinde, t_6 geçişi tetiklenir ve p_{14}'te bulunan jeton p_4'e depolanır. Bu durumda, q_4 aksiyonu gerçekleşir, kabin 4. katta durur ve s4 selenoidi çekili konuma geçer. ($\mathbf{p_1 \rightarrow p_{14} \rightarrow p_4}$)

- Kabin 2. katta iken {M(p_2)=1}, k3 butonuna basılırsa veya kabin katta t_d süresi kadar bekledikten sonra, tüm katların kapısı kapalı olduğunda, b4 butonuna basılırsa, t_7 geçişi tetiklenir ve p_2'de bulunan jeton p_{23}'e depolanır. Bu durumda, tüm katların kapısı kapalı ise, q_{23} aksiyonu gerçekleşir ve kabin 3. kata hareket eder. Kabin 3. katın sınır anahtarının (ksa3) konumunu değiştirdiğinde, t_9 geçişi tetiklenir ve p_{23}'te bulunan jeton p_3'e depolanır. Bu durumda, q_3 aksiyonu gerçekleşir, kabin 3. katta durur ve s3 selenoidi çekili konuma geçer. ($\mathbf{p_2 \rightarrow p_{23} \rightarrow p_3}$)

- Kabin 2. katta iken {M(p_2)=1}, k4 butonuna basılırsa veya kabin katta t_d süresi kadar bekledikten sonra, tüm katların kapısı kapalı olduğunda, b6 butonuna basılırsa, t_8 geçişi tetiklenir ve p_2'de bulunan jeton p_{24}'e depolanır. Bu durumda, tüm katların kapısı kapalı ise, q_{24} aksiyonu gerçekleşir ve kabin 4. kata hareket eder. Kabin 4. katın sınır anahtarının (ksa4) konumunu değiştirdiğinde, t_{10} geçişi tetiklenir ve p_{24}'te bulunan jeton p_4'e depolanır. Bu durumda, q_4 aksiyonu gerçekleşir, kabin 4. katta durur ve s4 selenoidi çekili konuma geçer. ($\mathbf{p_2 \rightarrow p_{24} \rightarrow p_4}$)

- Kabin 3. katta iken {M(p_3)=1}, k4 butonuna basılırsa veya kabin katta t_d süresi kadar bekledikten sonra, tüm katların kapısı kapalı olduğunda, b6 butonuna basılırsa, t_{11} geçişi tetiklenir ve p_3'te bulunan jeton p_{34}'e depolanır. Bu durumda, tüm katların kapısı kapalı ise, q_{34} aksiyonu gerçekleşir ve kabin 4. kata hareket eder. Kabin 4. katın sınır anahtarının (ksa4) konumunu değiştirdiğinde, t_{12} geçişi tetiklenir ve p_{34}'te bulunan jeton p_4'e depolanır. Bu durumda, q_4 aksiyonu gerçekleşir, kabin 4. katta durur ve s4 selenoidi çekili konuma geçer. (**$p_3 \rightarrow p_{34} \rightarrow p_4$**)

<u>Aşağı inişler için:</u>

- Kabin 4. katta iken {M(p_4)=1}, k3 butonuna basılırsa veya kabin katta t_d süresi kadar bekledikten sonra, tüm katların kapısı kapalı olduğunda, b4 butonuna basılırsa, t_{13} geçişi tetiklenir ve p_4'te bulunan jeton p_{43}'e depolanır. Bu durumda, tüm katların kapısı kapalı ise, q_{43} aksiyonu gerçekleşir ve kabin 3. kata hareket eder. Kabin 3. katın sınır anahtarının (ksa3) konumunu değiştirdiğinde, t_{16} geçişi tetiklenir ve p_{43}'te bulunan jeton p_3'e depolanır. Bu durumda, q_3 aksiyonu gerçekleşir, kabin 3. katta durur ve s3 selenoidi çekili konuma geçer. (**$p_4 \rightarrow p_{43} \rightarrow p_3$**)

- Kabin 4. katta iken {M(p_4)=1}, k2 butonuna basılırsa veya kabin katta t_d süresi kadar bekledikten sonra, tüm katların kapısı kapalı olduğunda, b2 butonuna basılırsa, t_{14} geçişi tetiklenir ve p_4'te bulunan jeton p_{42}'ye depolanır. Bu durumda, tüm katların kapısı kapalı ise, q_{42} aksiyonu gerçekleşir ve kabin 2. kata hareket eder. Kabin 2. katın sınır anahtarının (ksa2) konumunu değiştirdiğinde, t_{17} geçişi tetiklenir ve p_{42}'de bulunan jeton p_2'ye depolanır. Bu durumda, q_2 aksiyonu gerçekleşir, kabin 2. katta durur ve s2 selenoidi çekili konuma geçer. (**$p_4 \rightarrow p_{42} \rightarrow p_2$**)

- Kabin 4. katta iken {M(p_4)=1}, k1 butonuna basılırsa veya kabin katta t_d süresi kadar bekledikten sonra, tüm katların kapısı kapalı olduğunda, b1 butonuna basılırsa, t_{15} geçişi tetiklenir ve p_4'te bulunan jeton p_{41}'e depolanır. Bu durumda, tüm katların kapısı kapalı ise, q_{41} aksiyonu gerçekleşir ve kabin 1. kata hareket eder. Kabin 1. katın sınır anahtarının (ksa1) konumunu değiştirdiğinde, t_{18} geçişi

tetiklenir ve p_{41}'de bulunan jeton p_1'e depolanır. Bu durumda, q_1 aksiyonu gerçekleşir, kabin 1. katta durur ve s1 selenoidi çekili konuma geçer. ($\mathbf{p_4 \rightarrow p_{41} \rightarrow p_1}$)

- Kabin 3. katta iken {M(p_3)=1}, k2 butonuna basılırsa veya kabin katta t_d süresi kadar bekledikten sonra, tüm katların kapısı kapalı olduğunda, b2 butonuna basılırsa, t_{19} geçişi tetiklenir ve p_3'te bulunan jeton p_{32}'ye depolanır. Bu durumda, tüm katların kapısı kapalı ise, q_{32} aksiyonu gerçekleşir ve kabin 2. kata hareket eder. Kabin 2. katın sınır anahtarının (ksa2) konumunu değiştirdiğinde, t_{21} geçişi tetiklenir ve p_{32}'de bulunan jeton p_2'ye depolanır. Bu durumda, q_2 aksiyonu gerçekleşir, kabin 2. katta durur ve s2 selenoidi çekili konuma geçer. ($\mathbf{p_3 \rightarrow p_{32} \rightarrow p_2}$)

- Kabin 3. katta iken {M(p_3)=1}, k1 butonuna basılırsa veya kabin katta t_d süresi kadar bekledikten sonra, tüm katların kapısı kapalı olduğunda, b1 butonuna basılırsa, t_{20} geçişi tetiklenir ve p_3'te bulunan jeton p_{31}'e depolanır. Bu durumda, tüm katların kapısı kapalı ise, q_{31} aksiyonu gerçekleşir ve kabin 1. kata hareket eder. Kabin 1. katın sınır anahtarının (ksa1) konumunu değiştirdiğinde, t_{22} geçişi tetiklenir ve p_{31}'de bulunan jeton p_1'e depolanır. Bu durumda, q_1 aksiyonu gerçekleşir, kabin 1. katta durur ve s1 selenoidi çekili konuma geçer. ($\mathbf{p_3 \rightarrow p_{31} \rightarrow p_1}$)

- Kabin 2. katta iken {M(p_2)=1}, k1 butonuna basılırsa veya kabin katta t_d süresi kadar bekledikten sonra, tüm katların kapısı kapalı olduğunda, b1 butonuna basılırsa, t_{23} geçişi tetiklenir ve p_2'de bulunan jeton p_{21}'e depolanır. Bu durumda, tüm katların kapısı kapalı ise, q_{21} aksiyonu gerçekleşir ve kabin 1. kata hareket eder. Kabin 1. katın sınır anahtarının (ksa1) konumunu değiştirdiğinde, t_{24} geçişi tetiklenir ve p_{21}'de bulunan jeton p_1'e depolanır. Bu durumda, q_1 aksiyonu gerçekleşir, kabin 1. katta durur ve s1 selenoidi çekili konuma geçer. ($\mathbf{p_2 \rightarrow p_{21} \rightarrow p_1}$)

Toplamasız BOPN modeli için yazılan VHDL modülü (topiamasız.vhd), zaman gecikmesi modülü (t_delay.vhd) ile ayrı bir VHDL modülünde (elevator1.vhd) yapısal olarak birleştirilmiştir. Bu VHDL modülü, Xilinx firmasının ISE WebPACK 6.3i

yazılımı yardımı ile XC2S200E FPGA'sında sentezlendiğinde, kullanılan sistem kaynakları aşağıdaki gibi elde edilmiştir.

Number of Slices:	54	out of	2352	2%
Number of Slice Flip Flops:	45	out of	4704	0%
Number of 4 input LUTs:	67	out of	4704	1%
Number of bonded IOBs:	25	out of	146	17%
Number of GCLKs:	1	out of	4	25%

Total equivalent gate count for design: 714
Additional JTAG gate count for IOBs: 1,248

Toplamasız asansör kontrolü için elde edilen VHDL modülü (elevator1.vhd), ekler bölümünde verilmiştir. Bu VHDL modülü, XC2S200E FPGA'sında sentezlenerek Bölüm VI'da anlatılan model asansörde uygulanarak gerçekleştirilmiştir.

7.2. Toplamalı BOPN Modeli

Toplamalı BOPN modeli tasarlanırken, öncelikle 4 katlı bir asansör kontrol sisteminin toplamalı çalışma biçimi göz önüne alınmıştır. Bu sistemin başlıca özellikleri, maddeler halinde aşağıda belirtilmiştir.

- Kabin katta dururken, kabin içindeki butonlar sürekli aktif durumdadırlar. Böylece kabine binen bir kişi, kapı kapanmadan bir butona bastığında; kabin, kapı kapandıktan sonra, basılan butonun belirttiği kata hareket edecektir.
- Kabin katta dururken, katlarda bulunan butonlar, kabinin katta bekleme süresi dolduğunda ve tüm katların kapısı kapalı olduğunda aktiftirler.
- Kabin katta dururken, aktif olan butonlardan, ilk önce basılan dikkate alınır.
- Kabin hareket ederken, kabin içindeki butonlar aktif değildirler.
- Kabin hareket ederken katlardaki butonlardan, sadece kabinin hareket yönünde olan ve aynı yönü ifade eden butonlar aktiftirler. Örneğin kabin 1. kattan 3. kata hareket ederken, kabin 2. katı geçmeden önce, sadece 2↑ butonu aktiftir. Kabin 2. katı geçtikten sonra, butonların hiçbirisi aktif değildir.
- Kabin hareket ederken herhangi bir katta durakladığında, kabin içindeki butonlardan, sadece kabinin hareket yönünde olan butonlar aktiftirler. Örneğin kabin 1. kattan 3. kata hareket ederken, kabin 2. katı geçmeden önce 2↑

butonuna basılırsa kabin 2. katta duraklayacaktır. 2. kattan binen kişi sadece 3. ve 4. kat butonlarından birisine basarsa istediği kata gidebilecektir.

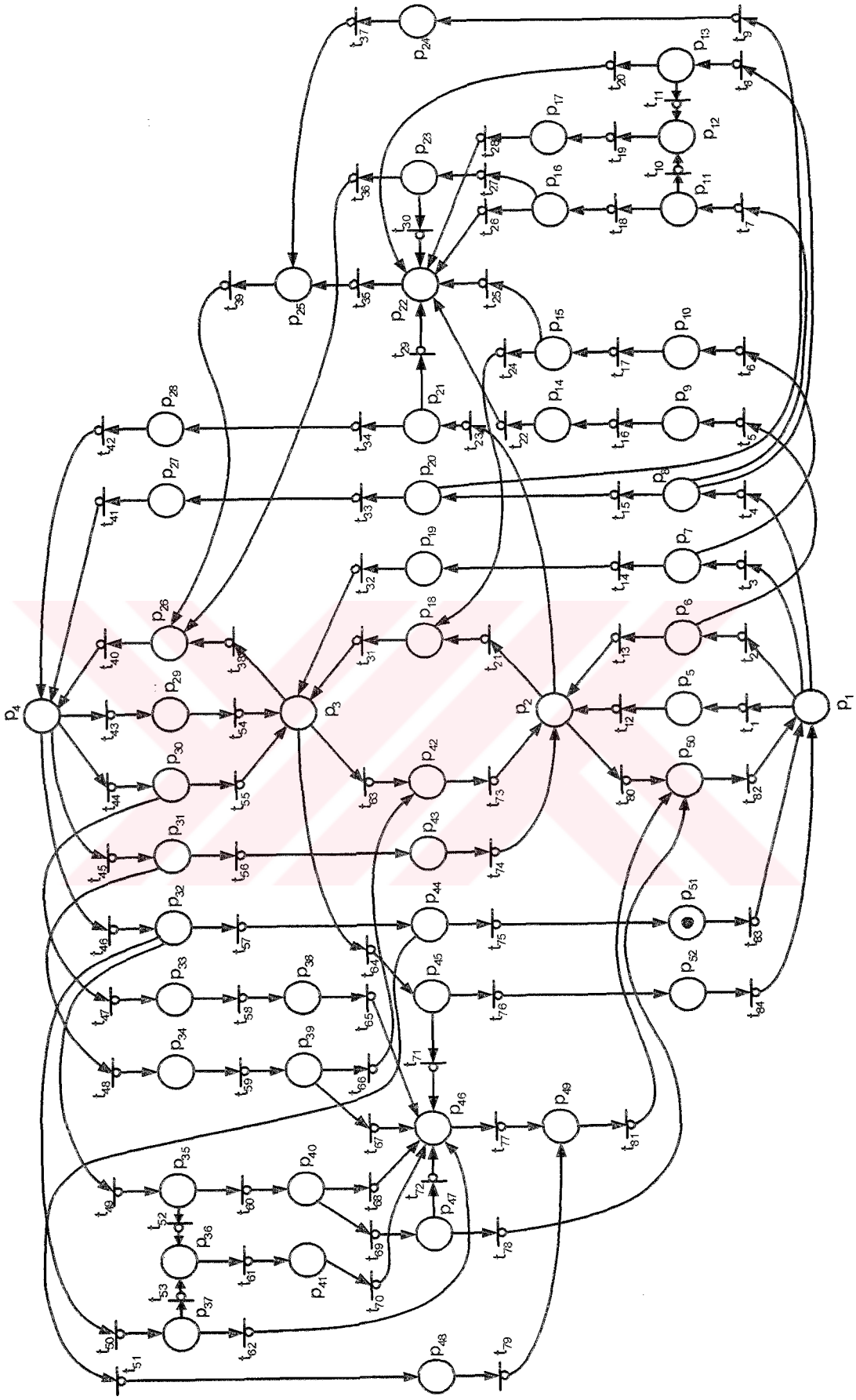
- Tüm katların kapısı kapalı olduğunda, kabin hareket etmeye başlar. Kabin hareket halindeyken, kapılardan herhangi birisi açılırsa, kabin olduğu yerde durur.
- Kabin alt veya üst sınır anahtarlarından herhangi birisinin konumunu değiştirirse, kabin olduğu yerde durur.

Tablo 7.3'te, kabinin hangi hareket yönünde, hangi butonların aktif olacağı belirtilmiştir. Ayrıca bu butonlara basıldığında, kabinin hangi katlarda duracağı da açıklanmıştır. Tablo 7.3'ün birinci satırında belirtilen durum, kabin 2. kattan 1. kata inmek için çağrıldığında geçerli değildir. Aynı şekilde, yedinci satırda belirtilen durum da, kabin 3. kattan 4. kata çıkmak için çağrıldığında geçerli değildir. Bundan sonraki kısımlarda, katlarda bulunan b1, b2, b3, b4, b5 ve b6 butonları, sırasıyla 1↑, 2↓, 2↑, 3↓, 3↑ ve 4↓ olarak ifade edilmiştir.

Tablo 7.3. Toplamalı kontrol sisteminde kabinin hareketi

Kabinin Hareket Yönü	Kabinin Konumu	Aktif Olan Butonlar	Kabinin Duracağı Katlar
1 → 2	1-2	3↑	2-3-4
1 → 3	1-2	2↑	2-3-(4)
1 → 4	1-2	2↑	2-(3)-4
1 → 4	1-2	3↑	3-4
1 → 4	2-3	3↑	3-4
2 → 4	2-3	3↑	3-4
4 → 3	4-3	2↓	3-2-1
4 → 2	4-3	3↓	3-2-(1)
4 → 1	4-3	3↓	3-(2)-1
4 → 1	4-3	2↓	2-1
4 → 1	3-2	2↓	2-1
3 → 1	3-2	2↓	2-1

Tablo 7.3'ten yararlanılarak, 4 katlı model asansör için, Şekil 7.2'de görülen toplamalı BOPN modeli tasarlanmıştır. Bu BOPN modelinde, 52 mevki $P=\{p_1, p_2, \ldots, p_{52}\}$ ve 84 geçiş $T=\{t_1, t_2, \ldots, t_{84}\}$ bulunmaktadır.



Şekil 7.2. Gerçekleştirilen toplamalı BOPN modeli

Toplamalı BOPN modeli tasarlanırken, kabinin konumlarını belirtmek amacıyla çeşitli mevkiler kullanılmıştır. Kabinin 1., 2., 3. ve 4. katlarda bulunduğunu, sırasıyla p_1, p_2, p_3 ve p_4 mevkileri temsil etmektedir. Bu mevkilerden herhangi birisinde jeton olması, kabinin o mevkinin belirttiği katta bulunduğu anlamına gelir. Kabinin hareket ederken, 2. katta durakladığını belirtmek için p_{14}, p_{15}, p_{16}, p_{17} ve p_{49} mevkileri kullanılmıştır. Aynı şekilde, kabinin hareket ederken, 3. katta durakladığını belirtmek için de p_{25}, p_{38}, p_{39}, p_{40} ve p_{41} mevkileri kullanılmıştır. Kabinin hareket ederken durakladığını belirten mevkilerden, herhangi birisinde jeton olması, kabinin belirli bir süre (t_d=2.68s) sonunda (hiçbir butona basılmamış olsa bile) hareketine devam edeceğini belirtir. Örneğin p_{14} mevkisinde bir jeton bulunması, kabinin 2. katta durakladığını ve t_d süresi sonunda 3. kata hareket edeceğini belirtir. Eğer t_d süresi dolmadan, k3 veya k4 butonlarından herhangi birisine basılırsa, kabin hemen hareket etmeye başlayacaktır. Kabinin katlar arasında bulunduğunu p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{18}, p_{19}, p_{20}, p_{21}, p_{22}, p_{23}, p_{24}, p_{26}, p_{27}, p_{28}, p_{29}, p_{30}, p_{31}, p_{32}, p_{33}, p_{34}, p_{35}, p_{36}, p_{37}, p_{42}, p_{43}, p_{44}, p_{45}, p_{46}, p_{47}, p_{48}, p_{50}, p_{51}, p_{52} mevkileri temsil etmektedir. Örneğin p_7 mevkisinde bir jeton bulunması, kabinin 1. kattan 3. kata giderken 1. ve 2. katlar arasında bulunduğunu ifade eder. Tablo 7.4'te toplamalı BOPN modelinde, kullanılan mevkiler, kabinin hareket yönü ve konumu verilmiştir.

Ayrıca bu BOPN modelinde bulunan her mevkiye çeşitli aksiyonlar atanmıştır. Kabinin katlarda bulunduğunu veya durakladığını belirten mevkilere (p_1, p_2, p_{14}, p_{15}, p_{16}, p_{17}, p_{49}, p_3, p_{25}, p_{38}, p_{39}, p_{40}, p_{41}, p_4), bulunulan katın selenoidi (s1, s2, s3, s4) aksiyon olarak atanmıştır. Böylece bu mevkilerden herhangi birisine jeton depolandığında, o katın selenoidi çekili konuma geçerek kabine binilmesine müsaade edecektir. Kabinin yukarı çıktığını belirten mevkilere (p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{18}, p_{19}, p_{20}, p_{21}, p_{22}, p_{23}, p_{24}, p_{26}, p_{27}, p_{28}), motorun yukarı (my) doğru çalışması aksiyon olarak atanmıştır. Bu durumda bu mevkilerden herhangi birisine jeton depolandığında, tüm katların kapısı kapalı ise, kabin yukarı doğru hareket etmeye başlar. Kabinin aşağı indiğini belirten mevkilere (p_{29}, p_{30}, p_{31}, p_{32}, p_{33}, p_{34}, p_{35}, p_{36}, p_{37}, p_{42}, p_{43}, p_{44}, p_{45}, p_{46}, p_{47}, p_{48}, p_{50}, p_{51}, p_{52}), motorun aşağı (ma) doğru çalışması aksiyon olarak atanmıştır. Bu durumda bu mevkilerden herhangi birisine jeton depolandığında, tüm katların kapısı kapalı ise, kabin aşağı doğru hareket etmeye başlar. Tablo 7.4'te toplamalı BOPN modelindeki mevkilere atanan aksiyonlar görülmektedir.

Tablo 7.4. Toplamalı BOPN modelindeki mevkiler ve bu mevkilere atanan aksiyonlar

Mevki	Kabinin Hareket Yönü	Kabinin Konumu	Atanan Aksiyonlar
p1	-	1	s1
p2	-	2	s2
p3	-	3	s3
p4	-	4	s4
p5	1-2	1-2	my
p6	1-2	1-2	my
p7	1-3	1-2	my
p8	1-4	1-2	my
p9	1-4	1-2	my
p10	1-3	1-2	my
p11	1-4	1-2	my
p12	1-4	1-2	my
p13	1-4	1-2	my
p14	-	2	s2
p15	-	2	s2
p16	-	2	s2
p17	-	2	s2
p18	2-3	2-3	my
p19	1-3	2-3	my
p20	1-4	2-3	my
p21	2-4	2-3	my
p22	2-4	2-3	my
p23	2-4	2-3	my
p24	1-4	2-3	my
p25	-	3	s3
p26	3-4	3-4	my

Mevki	Kabinin Hareket Yönü	Kabinin Konumu	Atanan Aksiyonlar
p27	1-4	3-4	my
p28	2-4	3-4	my
p29	4-3	4-3	ma
p30	4-3	4-3	ma
p31	4-2	4-3	ma
p32	4-1	4-3	ma
p33	4-1	4-3	ma
p34	4-2	4-3	ma
p35	4-1	4-3	ma
p36	4-1	4-3	ma
p37	4-1	4-3	ma
p38	-	3	s3
p39	-	3	s3
p40	-	3	s3
p41	-	3	s3
p42	3-2	3-2	ma
p43	4-2	3-2	ma
p44	4-1	3-2	ma
p45	3-1	3-2	ma
p46	3-2	3-2	ma
p47	3-1	3-2	ma
p48	3-2	3-2	ma
p49	-	2	s2
p50	2-1	2-1	ma
p51	4-1	2-1	ma
p52	3-1	2-1	ma

Kabinin katlar arasında olduğunu belirten mevkilerin, girişlerinde bulunan geçişlere (t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{21}, t_{22}, t_{23}, t_{24}, t_{25}, t_{26}, t_{27}, t_{28}, t_{29}, t_{30}, t_{38}, t_{39}, t_{43}, t_{44}, t_{45}, t_{46}, t_{47}, t_{48}, t_{49}, t_{50}, t_{51}, t_{52}, t_{53}, t_{63}, t_{64}, t_{65}, t_{66}, t_{67}, t_{68}, t_{69}, t_{70}, t_{71}, t_{72}, t_{80}, t_{81}) tetikleme şartları olarak, kabin içindeki butonlar (k1, k2, k3, k4) veya katlardaki butonlar duruma göre (b1, b2, b3, b4, b5, b6) atanmıştır. Bu mevkilerin çıkışlarında bulunan geçişlere (t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{18}, t_{19}, t_{20}, t_{31}, t_{32}, t_{33}, t_{34}, t_{35}, t_{36}, t_{37}, t_{40}, t_{41}, t_{42}, t_{54}, t_{55}, t_{56}, t_{57}, t_{58}, t_{59}, t_{60}, t_{61}, t_{62}, t_{73}, t_{74}, t_{75}, t_{76}, t_{77}, t_{78}, t_{79}, t_{82}, t_{83}, t_{84}) ise tetikleme şartları olarak, kat sınır anahtarları (ksa1, ksa2, ksa3, ksa4) atanmıştır. Tablo 7.5'te toplamalı BOPN modelindeki geçişler ve bu geçişlerin tetikleme şartları görülmektedir.

Tablo 7.5. Toplamalı BOPN modelindeki geçişler ve bu geçişlerin tetikleme şartları

Geçiş	Tetikleme Şartı
t_1	χ_1 = (t_d and ka and 2↓)
t_2	χ_2 = k2 or (t_d and ka and 2↑)
t_3	χ_3 = k3 or (t_d and ka and (3↓ or 3↑))
t_4	χ_4 = k4 or (t_d and ka and 4↓)
t_5	χ_5 = (ka and 3↑)
t_6	χ_6 = (ka and 2↑)
t_7	χ_7 = (ka and 2↑)
t_8	χ_8 = (ka and 3↑)
t_9	χ_9 = (ka and 3↑)
t_{10}	χ_{10} = (ka and 3↑)
t_{11}	χ_{11} = (ka and 2↑)
t_{12}	χ_{12} = ksa2
t_{13}	χ_{13} = ksa2
t_{14}	χ_{14} = ksa2
t_{15}	χ_{15} = ksa2
t_{16}	χ_{16} = ksa2
t_{17}	χ_{17} = ksa2
t_{18}	χ_{18} = ksa2
t_{19}	χ_{19} = ksa2
t_{20}	χ_{20} = ksa2
t_{21}	χ_{21} = k3 or (t_d and ka and (3↓ or 3↑))
t_{22}	χ_{22} = k3 or k4 or (t_d and ka)
t_{23}	χ_{23} = k4 or (t_d and ka and 4↓)
t_{24}	χ_{24} = k3 or (t_d and ka)
t_{25}	χ_{25} = k4 or (t_d and ka and 4↓)
t_{26}	χ_{26} = k3 or (t_d and ka and 3↑)
t_{27}	χ_{27} = k4 or (t_d and ka)
t_{28}	χ_{28} = k3 or k4 or (t_d and ka)
t_{29}	χ_{29} = (ka and 3↑)
t_{30}	χ_{30} = (ka and 3↑)
t_{31}	χ_{31} = ksa3
t_{32}	χ_{32} = ksa3
t_{33}	χ_{33} = ksa3
t_{34}	χ_{34} = ksa3
t_{35}	χ_{35} = ksa3
t_{36}	χ_{36} = ksa3
t_{37}	χ_{37} = ksa3
t_{38}	χ_{38} = k4 or (t_d and ka and 4↓)
t_{39}	χ_{39} = k4 or (t_d and ka)
t_{40}	χ_{40} = ksa4
t_{41}	χ_{41} = ksa4
t_{42}	χ_{42} = ksa4
t_d : Kabinin katta bekleme süresi	

Geçiş	Tetikleme Şartı
t_{43}	χ_{43} = (t_d and ka and 3↑)
t_{44}	χ_{44} = k3 or (t_d and ka and 3↓)
t_{45}	χ_{45} = k2 or (t_d and ka and (2↓ or 2↑))
t_{46}	χ_{46} = k1 or (t_d and ka and 1↑)
t_{47}	χ_{47} = (ka and 2↓)
t_{48}	χ_{48} = (ka and 3↓)
t_{49}	χ_{49} = (ka and 3↓)
t_{50}	χ_{50} = (ka and 2↓)
t_{51}	χ_{51} = (ka and 2↓)
t_{52}	χ_{52} = (ka and 2↓)
t_{53}	χ_{53} = (ka and 3↓)
t_{54}	χ_{54} = ksa3
t_{55}	χ_{55} = ksa3
t_{56}	χ_{56} = ksa3
t_{57}	χ_{57} = ksa3
t_{58}	χ_{58} = ksa3
t_{59}	χ_{59} = ksa3
t_{60}	χ_{60} = ksa3
t_{61}	χ_{61} = ksa3
t_{62}	χ_{62} = ksa3
t_{63}	χ_{63} = k2 or (t_d and ka and (2↓ or 2↑))
t_{64}	χ_{64} = k1 or (t_d and ka and 1↑)
t_{65}	χ_{65} = k1 or (t_d and ka and 1↑)
t_{66}	χ_{66} = k2 or (t_d and ka)
t_{67}	χ_{67} = k1 or (t_d and ka and 1↑)
t_{68}	χ_{68} = k2 or (t_d and ka and 2↓)
t_{69}	χ_{69} = k1 or (t_d and ka)
t_{70}	χ_{70} = k1 or k2 or (t_d and ka)
t_{71}	χ_{71} = (ka and 2↓)
t_{72}	χ_{72} = (ka and 2↓)
t_{73}	χ_{73} = ksa2
t_{74}	χ_{74} = ksa2
t_{75}	χ_{75} = ksa2
t_{76}	χ_{76} = ksa2
t_{77}	χ_{77} = ksa2
t_{78}	χ_{78} = ksa2
t_{79}	χ_{79} = ksa2
t_{80}	χ_{80} = k1 or (t_d and ka and 1↑)
t_{81}	χ_{81} = k1 or (t_d and ka)
t_{82}	χ_{82} = ksa1
t_{83}	χ_{83} = ksa1
t_{84}	χ_{84} = ksa1
ka = ka1 and ka2 and ka3 and ka4	

Kabin katta dururken, kabin içindeki butonlar sürekli aktif durumdadırlar. Örneğin kabine binen bir kişi, kabin kapısı açık iken bir butona basarsa; kabin, kapı kapandıktan sonra basılan butonun belirttiği kata hareket edecektir. Katlardaki butonların aktif olabilmesi için ise, kabinin katta bekleme süresinin (t_d) dolması ve tüm katların kapısının (ka1, ka2, ka3, ka4) kapalı olması gerekmektedir. Kabinin katta bekleme süresi dolduğunda, tüm katların kapısı kapalı ise, kabin ilk basılan butonun belirttiği kata hareket etmeye başlayacaktır. Kabin hareket etmeye başladığında, kabinin hareket ettiği katın selenoidi itili konuma geçerek kat kapısının açılması engellenir. Örneğin kabin 1. katta iken t_d süresi kadar bekledikten sonra, tüm katların kapısı kapalı olduğunda, hangi butona daha önce basılırsa kabin o kata hareket edecektir. Bu esnada 1. katın selenoidi (s1) itili konuma geçerek kat kapısının açılması engellenir.

Kabin hareket ederken katlardaki butonlardan, sadece kabinin hareket yönünde olan ve aynı yönü ifade eden butonlar aktiftirler. Örneğin kabin 1. kattan 3. kata hareket ederken, kabin 2. katı geçmeden önce, sadece 2↑ butonu aktiftir. Bu esnada 2↑ butonuna basılırsa, t_6 geçişi tetiklenerek p_7 mevkisinde bulunan jeton p_{10} mevkisine depolanır ve kabin 2. katta duraklar. Kabin 2. katı geçtikten sonra, butonların hiçbirisi aktif değildir.

Kabin hareket ederken herhangi bir katta durakladığında, kabin içindeki butonlardan, sadece kabinin hareket yönünde olan butonlar aktiftirler. Örneğin kabin 1. kattan 3. kata hareket ederken, kabin 2. katı geçmeden önce 2↑ butonuna basılırsa kabin 2. katta duraklayacaktır. 2. kattan binen kişi sadece 3. ve 4. kat butonlarından birisine basarsa istediği kata gidebilecektir. Bu durumda t_{24} veya t_{25} geçişlerinden birisi tetiklenerek p_{15} mevkisinde bulunan jeton, p_{18} veya p_{22} mevkilerine depolanır ve kabin istenilen kata hareket etmeye başlar.

Kabin güvenliği için iki önlem alınmıştır. Bunlardan birincisi, kabin eğer alt sınır veya üst sınır anahtarlarına değerse motorun durması ve UYARI sinyali vermesidir. Bu durumda model asansörün enerjisi kesildikten sonra kabin konumuna göre mekanik olarak yukarı veya aşağı çekilir. Daha sonra model asansörün enerjisi verilerek sistem çalışmasına devam eder. İkinci olarak da, kabin hareket halindeyken kapılardan herhangi birisi açılırsa motorun durmasıdır. Tüm kapılar kapalı olduğunda motor tekrar çalışmaya başlar ve kabin hareketine devam eder.

Şekil 7.2'deki toplamalı BOPN modelinin çalışmasını inceleyecek olursak, ilk başta p_{51} mevkisinde jeton olduğu görülmektedir. Bunun nedeni, kabinin başlangıçta nerede olursa olsun, 1. kata gelmesini sağlamak içindir. Kabin 1. katın sınır anahtarının konumunu değiştirdiğinde, t_{83} geçişi tetiklenerek p_{51} mevkisinde bulunan jeton p_1 mevkisine depolanır ve kabin 1. katta durur. Bu durumda s1 selenoidi çekili konuma geçerek kabine binilmesine müsaade eder. Örneğin kabin 1. katta iken, bekleme süresi (t_d) dolduğunda, tüm katların kapısı kapalı ise, 3. kattaki bir kişi 3↑ butonuna bastığında, t_3 geçişi tetiklenerek p_1 mevkisinde bulunan jeton p_7 mevkisine depolanır. Bu esnada kabin 3. kata hareket etmeye başlar. Kabin 2. katın sınır anahtarına gelmeden önce 2↑ butonuna basılırsa, t_6 geçişi tetiklenir ve p_7 mevkisinde bulunan jeton p_{10} mevkisine depolanır. Kabin 2. katın sınır anahtarının konumunu değiştirdiğinde, t_{17} geçişi tetiklenerek p_{10} mevkisinde bulunan jeton p_{15} mevkisine depolanır ve kabin 2. katta durur. Kabine 2. kattan binen kişi k3 butonuna basarsa veya kabinin kapısı açılmadan t_d süresi kadar bir zaman geçerse, t_{24} geçişi tetiklenir ve p_{15} mevkisinde bulunan jeton p_{18} mevkisine depolanır. Bu durumda kabin 3. kata hareket eder. Kabin 3. katın sınır anahtarının konumunu değiştirdiğinde, t_{31} geçişi tetiklenerek p_{18} mevkisinde bulunan jeton p_3 mevkisine depolanır ve kabin 3. katta durur. Kabin 2. katta iken binen kişi, eğer k4 butonuna basarsa veya 4. kattaki bir kişi 4↓ butonuna basarsa, t_{25} geçişi tetiklenir ve p_{15} mevkisinde bulunan jeton p_{22} mevkisine depolanır. Bu durumda kabin 3. kata hareket eder. Kabin 3. katın sınır anahtarının konumunu değiştirdiğinde, t_{35} geçişi tetiklenerek p_{22} mevkisinde bulunan jeton p_{25} mevkisine depolanır ve kabin 3. katta durur. Kabin 3. katta iken binen kişi, k4 butonuna basarsa veya kabin kapısı açılmadan t_d süresi kadar bir zaman geçerse, t_{39} geçişi tetiklenir ve p_{25} mevkisinde bulunan jeton p_{26} mevkisine depolanır. Bu durumda kabin 4. kata hareket eder. Kabin 4. katın sınır anahtarının konumunu değiştirdiğinde, t_{40} geçişi tetiklenerek p_{26} mevkisinde bulunan jeton p_4 mevkisine depolanır ve kabin 4. katta durur.

Toplamalı BOPN modeli için yazılan VHDL modülü (toplamalı.vhd), zaman gecikmesi modülü (t_delay.vhd) ile ayrı bir VHDL modülünde (elevator2.vhd) yapısal olarak birleştirilmiştir. Bu VHDL modülü, Xilinx firmasının ISE WebPACK 6.3i yazılımı yardımı ile XC2S200E FPGA'sında sentezlendiğinde, kullanılan sistem kaynakları aşağıdaki gibi elde edilmiştir.

Number of Slices:	139	out of	2352	5%
Number of Slice Flip Flops:	81	out of	4704	1%
Number of 4 input LUTs:	151	out of	4704	3%
Number of bonded IOBs:	27	out of	146	18%
Number of GCLKs:	1	out of	4	25%

Total equivalent gate count for design: 1,398
Additional JTAG gate count for IOBs: 1,344

Toplamalı asansör kontrolü için elde edilen VHDL modülü (elevator2.vhd), ekler bölümünde verilmiştir. Bu VHDL modülü, XC2S200E FPGA'sında sentezlenerek Bölüm VI'da anlatılan model asansörde uygulanarak gerçekleştirilmiştir.

BÖLÜM VIII

SONUÇLAR

Bu tez çalışmasının amacı, bir asansör kontrol sisteminin Petri netler yardımıyla modellenmesi ve bir Xilinx XC2S200E FPGA'sı ile gerçekleştirilmesidir.

Asansör kontrol sisteminin tasarımı için, dört katlı bir model asansör kullanılmıştır. Bu model asansör mümkün olduğunca ideale yakın bir biçimdedir. Gerçek bir asansörde kullanılan elemanların minimize edilmiş hallerini içermektedir. Örneğin katlarda bekleyen kişilerin güvenliğini sağlamak amacıyla her katta birer selenoid valf, kapıların açık veya kapalı olduğunu belirtmek için kapı anahtarları vb. elemanlar bulunmaktadır.

Dört katlı model asansör için gerekli olan kontrol sisteminin modellenmesinde, otomasyon Petri netlerin bir alt kümesi olan Basit Otomasyon Petri netlerden (BOPN) faydalanılmıştır. Basit otomasyon Petri netlerin tercih edilmesinin sebebi; her mevkide en fazla bir jeton bulunması ve kullanılan okların ağırlık değerinin de bir olmasıdır. Toplamasız ve toplamalı asansör kontrol sistemleri için ayrı ayrı BOPN modelleri tasarlanmıştır. Bu BOPN modellerinde; mevkiler kabinin konumunu belirtirken, geçişlerin tetikleme şartları da kabin içinde veya katlarda bulunan butonlar veya kat sınır anahtarlarının durumlarını ifade etmektedir.

Model asansörün toplamasız olarak kontrolünde, kabin hareket halindeyken, katlardan yapılan çağrılar veya kabin içinden yapılan komutlar dikkate alınmamaktadır. Toplamalı olarak kontrolünde ise, kabin hareket halindeyken katlardan yapılan çağrılardan, sadece kabinin hareket yönünde olan ve aynı yönü ifade edenler dikkate alınmaktadır. Toplamalı kontrol sisteminde, kabin hareket halindeyken, kabin içindeki butonlar aktif değildir.

Bu yüksek lisans tez çalışmasında Basit Otomasyon Petri netlerin FPGA (Field Programmable Gate Array) ile gerçekleştirilmesi için Xilinx firmasının Spartan-IIE

serisine ait 200 bin kapı içeren XC2S200E FPGA'sı kullanılmıştır [9]. Bu amaçla, adı geçen FPGA'yı içeren Digilent firmasının D2-SB FPGA uygulama geliştirme kartı kullanılmıştır [8].

Model asansörün Digilent D2-SB kartı ile kontrol edilebilmesi için bir I/O tampon kartı tasarlanmıştır. Bu kart D2-SB kartının kullandığı LVTTL standardını TTL'ye dönüştürmek için kullanılmaktadır. Bu kart tasarlanırken, kartın deneme kartı olarak da kullanımını sağlamak amacı ile temel giriş/çıkış devreleri de kart üzerine eklenmiştir.

Bu çalışmada model asansörün, toplamasız ve toplamalı olarak kontrolü için tasarlanan BOPN modelleri, VHDL diline çevrilerek Xilinx XC2S200E FPGA'sına ayrı ayrı yüklenmiştir. Yazılan VHDL kodları Xilinx firmasının ISE WebPACK 6.3i yazılımında sentezlenmiştir.

Bu çalışmada kullanılan FPGA, SRAM tabanlı olduğu için enerjisi kesilince, üzerinde yüklü bulunan program silinmektedir. Bunu önlemek için, Digilent D2-SB FPGA uygulama geliştirme kartındaki sokete bir Flash ROM takılabilir.

Bu çalışmadan görülmektedir ki; endüstriyel sistemlerin kontrolünün tasarlanmasında, Petri netlerden yararlanılabilir. Ayrıca işlem hızı çok yüksek olan FPGA'ların bu sistemlerin kontrolünde kullanılabileceği de görülmektedir.

KAYNAKLAR

[1] Uzam M., Petri-Net-Based Supervisory Control of Discrete Event Systems and Their Ladder Logic Diagram Implementations, The University of Salford, UK, PhD. Thesis, 1998, Chapter 2.

[2] Uzam M., Avcı M., Yalçın M. K., Digital Hardware Implementation of Petri Net Based Specifications: Direct Translation From Safe Automation Petri Nets to Circuit Elements, The International Workshop on Discrete Event System Design, DESDes'01, (2001), 1-9.

[3] Bezen Ö., Programlanabilir Mantık Devre Teknolojisinin Elektro-Optik Ürünlerde Uygulaması, ASELSAN, Sayı:66, (Eylül-2002), 24-27.

[4] Örs S. B., VHDL İle Lojik Devre Tasarımı ve DSP Uygulamaları İçin Çarpma Bloklarının Modellenmesi, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, (1998), Bölüm 2.

[5] Wegrzyn M., Adamski M. A., Monteiro J. L., The Application of Reconfigurable Logic to Controller Design, Control Engineering Practice, 6, (1998), 879-887.

[6] Lloyd L., Heron K., Koelmans A. M., Yakovlev A. V., Asynchronous Microprocessors: From High Level Model to FPGA Implementation, Journal of Systems Architecture, 45, (1999), 975-1000.

[7] Wegrzyn M., Implementation of Safety Critical Logic Controller By Means of FPGA, Annual Reviews of Control, 27, (2003), 55-61.

[8] www.digilentinc.com

[9] www.xilinx.com

[10] Zurawski R., Zhou M. C., Petri Nets and Industrial Applications: a Tutorial, IEEE Trans. on Industrial Electronics, (1994).

[11] Murata T., Petri Nets: Properties, Analysis and Applications, Proceedings of IEEE, Vol: 77, No: 4, (1989), 541-580.

[12] Gu T., Bahri P. A., A Survey of Petri Net Applications in Batch Processes, Computers in Industry, 47, (2002), 99-111.

[13] Zhou M. C., DiCesare F., Petri Nets Synthesis for Discrete Event Control of Manufacturing Systems, Kluwer Academic Publishers, Dordrechet, (1993).

[14] David R., Alla H., Petri Nets and Grafcet: Tools for Modelling Discrete Event Systems, Prentice Hall, London, UK, 1992, Chapter 1.

[15] Parnell K., Mehta N., Programmable Logic Design Quick Start Hand Book, Xilinx, 2003, Chapter 1.

[16] Perry D. L., VHDL Programming by Example, McGraw-Hill, New York, USA, 2002, Chapter 1.

EK-A

TOPLAMASIZ ASANSÖR KONTROL SİSTEMİNİ OLUŞTURAN VHDL MODÜLLERİ

Toplamasız BOPN modelinin VHDL kodu (toplamasiz.vhd) :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

----------------------------------------------------------------------------------------
--                    TOPLAMASIZ BOPN MODELİNİN VHDL KODU
----------------------------------------------------------------------------------------
-- * Başlangıçta motor aşağı doğru çalışır ve 1. kata gelince durur.
-- Daha sonra yapılan çağrılara göre kabin hareket eder.
-- * Kabin katta dururken, kabin içindeki butonlar sürekli aktif durumdadırlar.
-- Böylece kabine binen bir kişi, kapı kapanmadan bir butona bastığında;
-- kabin, kapı kapandıktan sonra, basılan butonun belirttiği kata hareket edecektir.
-- * Kabin katta dururken, katlarda bulunan butonlar, kabinin katta bekleme süresi
dolduğunda
-- ve tüm katların kapısı kapalı olduğunda aktiftirler.
-- * Kabin katta dururken; aktif olan butonlardan, ilk önce basılan dikkate alınır.
-- * Kabin hareket halindeyken, kabin içindeki ve katlardaki butonlar aktif değildirler.
-- * Tüm katların kapısı kapalı olduğunda, kabin hareket etmeye başlar.
-- * Kabin hareket halindeyken, kapılardan herhangi birisi açılırsa, kabin olduğu yerde
durur.
-- * Kabin alt veya üst sınır anahtarlarından herhangi birisinin konumunu değiştirirse,
-- kabin olduğu yerde durur.
----------------------------------------------------------------------------------------

entity toplamasiz is
        port ( k1, k2, k3, k4                   : in std_logic ; -- kabin içindeki butonlar
               b1, b2, b4, b6                   : in std_logic ; -- katlardaki butonlar
               ksa1, ksa2, ksa3, ksa4           : in std_logic ; -- kat sınır anahtarları
               ka1, ka2, ka3, ka4               : in std_logic ; -- kapı anahtarları
               asa, usa                         : in std_logic ; -- alt ve üst sınır anahtarları
               s1, s2, s3, s4                   : out std_logic ; -- kapı selenoidleri
               ma, my                           : out std_logic ;-- motor aşağı ve yukarı
               FAIL                             : out std_logic ;-- asa veya usa aktif olursa
               t_d                              : in std_logic  ;-- t_delay modülünün çıkışı
               p1234                            : out std_logic -- t_delay modülünün girişi
               );
end toplamasiz;
```

```
architecture Behavioral of toplamasiz is
        signal p1,p2,p3,p4 : std_logic ;
        signal p12,p13,p14,p21,p23,p24,p31,p32,p34,p41,p42,p43 : std_logic ;
        signal ka : std_logic ;
        signal x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,
                x17,x18,x19,x20,x21,x22,x23,x24 : std_logic ; -- tetikleme şartları

begin

ka <= ka1 and ka2 and ka3 and ka4 ;
----------------------------------------

x1 <=  k2 or (t_d and ka and b2) ;
x14 <= k2 or (t_d and ka and b2) ;
x19 <= k2 or (t_d and ka and b2) ;

x2 <=  k3 or (t_d and ka and b4) ;
x7 <=  k3 or (t_d and ka and b4) ;
x13 <= k3 or (t_d and ka and b4) ;

x3 <=  k4 or (t_d and ka and b6) ;
x8 <=  k4 or (t_d and ka and b6) ;
x11 <= k4 or (t_d and ka and b6) ;

x15 <= k1 or (t_d and ka and b1) ;
x20 <= k1 or (t_d and ka and b1) ;
x23 <= k1 or (t_d and ka and b1) ;
-----------------------------------------

x18 <= ksa1 ; x22 <= ksa1 ; x24 <= ksa1 ;
x4 <= ksa2 ; x17 <= ksa2 ; x21 <= ksa2 ;
x5 <= ksa3 ; x9 <= ksa3 ; x16 <= ksa3 ;
x6 <= ksa4 ; x10 <= ksa4 ; x12 <= ksa4 ;
-----------------------------------------

p1 <= '0' when (p1 and (x1 or x2 or x3))='1' else
        '1' when (ksa1 and (p21 or p31 or p41))='1' ;

p2 <= '0' when (p2 and (x7 or x8 or x23))='1' else
        '1' when (ksa2 and (p12 or p32 or p42))='1' ;

p3 <= '0' when (p3 and (x11 or x19 or x20))='1' else
        '1' when (ksa3 and (p13 or p23 or p43))='1' ;

p4 <= '0' when (p4 and (x13 or x14 or x15))='1' else
        '1' when (ksa4 and (p14 or p24 or p34))='1' ;
-----------------------------------------------------
p12 <= '0' when (x4 and p12)='1' else
         '1' when (x1 and p1)='1';
-----------------------------------------------------
```

```
p13 <= '0' when (x5 and p13)='1' else
        '1' when (x2 and p1)='1';
-------------------------------------------------------
p14 <= '0' when (x6 and p14)='1' else
        '1' when (x3 and p1)='1';
-------------------------------------------------------
p21 <= '0' when (x24 and p21)='1' else
        '1' when (x23 and p2)='1';
-------------------------------------------------------
p23 <= '0' when (x9 and p23)='1' else
        '1' when (x7 and p2)='1';
-------------------------------------------------------
p24 <= '0' when (x10 and p24)='1' else
        '1' when (x8 and p2)='1';
-------------------------------------------------------
p31 <= '0' when (x22 and p31)='1' else
        '1' when (x20 and p3)='1';
-------------------------------------------------------
p32 <= '0' when (x21 and p32)='1' else
        '1' when (x19 and p3)='1';
-------------------------------------------------------
p34 <= '0' when (x12 and p34)='1' else
        '1' when (x11 and p3)='1' ;
-------------------------------------------------------
p41 <= '1' when (x15 and p4)='1' else          -- ***
        '0' when (x18 and p41)='1';
-------------------------------------------------------
p42 <= '0' when (x17 and p42)='1' else
        '1' when (x14 and p4)='1';
-------------------------------------------------------
p43 <= '0' when (x16 and p43)='1' else
        '1' when (x13 and p4)='1';
-------------------------------------------------------
ma <= ka and (p41 or p42 or p43 or p31 or p32 or p21) and not asa ;
my <= ka and (p12 or p13 or p14 or p23 or p24 or p34) and not usa ;

s1 <= p1 or not ka1;
s2 <= p2 or not ka2 ;
s3 <= p3 or not ka3 ;
s4 <= p4 or not ka4 ;

FAIL <= asa or usa ;

p1234 <= p1 or p2 or p3 or p4 ;

end Behavioral;
```

Kabinin katta bekleme süresinin (t_d=2.68s) belirlendiği VHDL kodu (t_delay.vhd) :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

----------------------------------------------------------------------------------
--        Kabinin katta bekleme süresinin (t_d=2.68s) belirlendiği VHDL modülü
----------------------------------------------------------------------------------

entity t_delay is
    Port ( input : in std_logic;
           output : out std_logic;
           clk : in std_logic
         );
end t_delay;

architecture Behavioral of t_delay is
signal cntClk  : std_logic_vector (27 downto 0) :="0000000000000000000000000000";
signal dvdClk, q2 : std_logic ;
begin

--  f(clk)= 50MHz ==> T(clk) = 20 ns
-- T(dvdclk) = 20 ns x (2^27) = 2.68 s -->yükselen kenar

process (clk, input)
begin
  if input='0' then
    cntClk <= "0000000000000000000000000000" ;
  elsif (clk'event and clk='1') then
    cntClk <= cntClk + 1 ;
  end if;
end process;

dvdClk <= cntClk(27)          ;

process (dvdClk, input)
begin
  if input='0' then
    q2 <= '0';
  elsif dvdClk'event and dvdClk='1' then
    q2 <= input;
  end if;
end process;

output <= input and q2 ;

end Behavioral;
```

Toplamasız asansör kontrol sisteminin VHDL kodu (elevator_1.vhd) :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following lines to use the declarations that are
-- provided for instantiating Xilinx primitive components.
--library UNISIM;
--use UNISIM.VComponents.all;

----------------------------------------------------------------------------------------------------
--                  TOPLAMASIZ ASANSÖR KONTROL SİSTEMİNİN VHDL KODU
----------------------------------------------------------------------------------------------------

entity elevator_1 is
        PORT(
                k1 : IN std_logic;
                k2 : IN std_logic;
                k3 : IN std_logic;
                k4 : IN std_logic;
                b1 : IN std_logic;
                b2 : IN std_logic;
                b4 : IN std_logic;
                b6 : IN std_logic;
                ksa1 : IN std_logic;
                ksa2 : IN std_logic;
                ksa3 : IN std_logic;
                ksa4 : IN std_logic;
                ka1 : IN std_logic;
                ka2 : IN std_logic;
                ka3 : IN std_logic;
                ka4 : IN std_logic;
                asa : IN std_logic;
                usa : IN std_logic;
                s1 : OUT std_logic;
                s2 : OUT std_logic;
                s3 : OUT std_logic;
                s4 : OUT std_logic;
                ma : OUT std_logic;
                my : OUT std_logic;
                FAIL : OUT std_logic;
                clk : IN std_logic
                );
end elevator_1;

architecture Behavioral of elevator_1 is

        COMPONENT t_delay
```

```
        PORT(
                input : IN std_logic;
                clk : IN std_logic;
                output : OUT std_logic
                );
        END COMPONENT;

        COMPONENT toplamasiz
        PORT(
                k1 : IN std_logic;
                k2 : IN std_logic;
                k3 : IN std_logic;
                k4 : IN std_logic;
                b1 : IN std_logic;
                b2 : IN std_logic;
                b4 : IN std_logic;
                b6 : IN std_logic;
                ksa1 : IN std_logic;
                ksa2 : IN std_logic;
                ksa3 : IN std_logic;
                ksa4 : IN std_logic;
                ka1 : IN std_logic;
                ka2 : IN std_logic;
                ka3 : IN std_logic;
                ka4 : IN std_logic;
                asa : IN std_logic;
                usa : IN std_logic;
                t_d : IN std_logic;
                s1 : OUT std_logic;
                s2 : OUT std_logic;
                s3 : OUT std_logic;
                s4 : OUT std_logic;
                ma : OUT std_logic;
                my : OUT std_logic;
                FAIL : OUT std_logic;
                p1234 : OUT std_logic
                );
        END COMPONENT;

signal sgn_p1234 : std_logic;
signal sgn_t_d : std_logic;

begin

        Inst_t_delay: t_delay PORT MAP(
                input => sgn_p1234,
                output => sgn_t_d,
                clk =>clk
        );
```

```
        Inst_toplamasiz: toplamasiz PORT MAP(
                k1 =>k1 ,
                k2 =>k2 ,
                k3 =>k3 ,
                k4 =>k4 ,
                b1 =>b1 ,
                b2 =>b2 ,
                b4 =>b4 ,
                b6 =>b6 ,
                ksa1 =>ksa1 ,
                ksa2 =>ksa2 ,
                ksa3 =>ksa3 ,
                ksa4 =>ksa4 ,
                ka1 =>ka1 ,
                ka2 =>ka2 ,
                ka3 =>ka3 ,
                ka4 =>ka4 ,
                asa =>asa ,
                usa =>usa ,
                s1 =>s1 ,
                s2 =>s2 ,
                s3 =>s3 ,
                s4 =>s4 ,
                ma =>ma ,
                my =>my ,
                FAIL =>FAIL ,
                t_d =>sgn_t_d ,
                p1234 =>sgn_p1234
        );

end Behavioral;
```

EK-B

TOPLAMALI ASANSÖR KONTROL SİSTEMİNİ OLUŞTURAN VHDL MODÜLLERİ

Toplamalı BOPN modelinin VHDL kodu (toplamali.vhd) :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

----------------------------------------------------------------------------------------------------
--                    TOPLAMALI BOPN MODELİNİN VHDL KODU
----------------------------------------------------------------------------------------------------
-- * Başlangıçta motor aşağı doğru çalışır ve 1. kata gelince durur.
-- Daha sonra yapılan çağrılara göre kabin hareket eder.
-- * Kabin katta dururken, kabin içindeki butonlar sürekli aktif durumdadırlar.
-- Böylece kabine binen bir kişi, kapı kapanmadan bir butona bastığında;
-- kabin, kapı kapandıktan sonra, basılan butonun belirttiği kata hareket edecektir.
-- * Kabin katta dururken, katlarda bulunan butonlar, kabinin katta bekleme süresi
-- dolduğunda ve tüm katların kapısı kapalı olduğunda aktiftirler.
-- * Kabin katta dururken, aktif olan butonlardan, ilk önce basılan dikkate alınır.
-- * Kabin hareket ederken, kabin içindeki butonlar aktif değildirler.
-- * Kabin hareket ederken katlardaki butonlardan, sadece kabinin hareket yönünde olan
-- ve aynı yönü ifade eden butonlar aktiftirler. Örneğin kabin 1. kattan 3. kata hareket
-- ederken, kabin 2. katı geçmeden önce, sadece 2^ butonu aktiftir. Kabin 2. katı
-- geçtikten sonra, butonların hiçbirisi aktif değildir.
-- * Kabin hareket ederken herhangi bir katta durakladığında, kabin içindeki
-- butonlardan, sadece kabinin hareket yönünde olan butonlar aktiftirler. Örneğin kabin
-- 1. kattan 3. kata hareket ederken, kabin 2. katı geçmeden önce 2^ butonuna basılırsa
-- kabin 2. katta duraklayacaktır. 2. kattan binen kişi sadece 3. ve 4. kat butonlarından
-- birisine basarsa istediği kata gidebilecektir.
-- * Tüm katların kapısı kapalı olduğunda, kabin hareket etmeye başlar. Kabin hareket
-- halindeyken, kapılardan herhangi birisi açılırsa, kabin olduğu yerde durur.
-- * Kabin alt veya üst sınır anahtarlarından herhangi birisinin konumunu değiştirirse,
-- kabin olduğu yerde durur.
----------------------------------------------------------------------------------------------------

entity toplamali is
        port ( k1, k2, k3, k4           : in std_logic ; -- kabin içi butonlar
               b1, b2, b3, b4, b5, b6   : in std_logic ; -- kabin dışı butonlar
               ksa1, ksa2, ksa3, ksa4   : in std_logic ; -- kat sınır anahtarları (switchleri)
               ka1, ka2, ka3, ka4       : in std_logic ; -- kapı anahtarları
               asa, usa                 : in std_logic ; -- alt ve üst sınır anahtarlar
               s1, s2, s3, s4           : out std_logic ;-- kapı selenoidleri
```

```
            ma, my          : out std_logic ;-- motor aşağı ve motor yukarı
            FAIL            : out std_logic ;-- asa veya usa aktif olursa
            t_d             : in std_logic  ;-- t_delay modülünün çıkışı
            p1234           : out std_logic -- t_delay modülünün girişi
            );
end toplamali;

architecture Behavioral of toplamali is

    signal p1,p2,p3,p4 :   std_logic ;      -- kabinin 1.,2.,3.,4. katlarda durduğunu
    belirten mevkiler
    signal p14,p15,p16,p17,p49 : std_logic ;      -- kabinin 2. katta durakladığını
    belirten mevkiler
    signal p25,p38,p39,p40,p41 : std_logic ;      -- kabinin 3. katta durakladığını
    belirten mevkiler
    signal p5,p6,p7,p8,p9,p10,p11,p12,p13,p18,p19,
           p20,p21,p22,p23,p24,p26,p27,p28 :  std_logic ;      -- kabin yukarı
    signal p29,p30,p31,p32,p33,p34,p35,p36,p37,p42,p43,
           p44,p45,p46,p47,p48,p50,p51,p52 :  std_logic ;      -- kabin aşağı
    signal x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x21,x22,x23,x24,
           x25,x26,x27,x28,x29,x30,x38,x39 : std_logic ; -- tetikleme şartları
    signal x43,x44,x45,x46,x47,x48,x49,x50,x51,x52,x53,x63,x64,x65,x66,
           x67,x68,x69,x70,x71,x72,x80,x81 : std_logic ; -- tetikleme şartları
    signal x82,x83,x84 :   std_logic ;      -- ksa1
    signal x12,x13,x14,x15,x16,x17,x18,x19,x20,x73,x74,
           x75,x76,x77,x78,x79 : std_logic ;     -- ksa2
    signal x31,x32,x33,x34,x35,x36,x37,x54,x55,x56,
           x57,x58,x59,x60,x61,x62 : std_logic ;         -- ksa3
    signal x40,x41,x42 :   std_logic ;      -- ksa4
    signal ka : std_logic ;

begin

ka <= ka1 and ka2 and ka3 and ka4 ;
----------------------------------------------------------------------

x1 <= t_d and ka and b2;
x2 <= k2 or (t_d and ka and b3);
x3 <= k3 or (t_d and ka and (b4 or b5));
x4 <= k4 or (t_d and ka and b6) ;
x5 <=  ka and b5 ;
x6 <=  ka and b3 ;
x7 <=  ka and b3 ;
x8 <=  ka and b5 ;
x9 <=  ka and b5 ;
x10 <=  ka and b5 ;
x11 <=  ka and b3 ;
x12 <= ksa2 ; x13 <= ksa2 ; x14 <= ksa2 ; x15 <= ksa2 ; x16 <= ksa2 ;
x17 <= ksa2 ; x18 <= ksa2 ; x19 <= ksa2 ; x20 <= ksa2 ;
x21 <= k3 or (t_d and ka and (b4 or b5));
```

```
x22 <= k3 or k4 or (t_d and ka);
x23 <= k4 or (t_d and ka and b6);
x24 <= k3 or (t_d and ka);
x25 <= k4 or (t_d and ka and b6);
x26 <= k3 or (t_d and ka and b5);
x27 <= k4 or (t_d and ka);
x28 <= k3 or k4 or (t_d and ka);
x29 <=  ka and b5 ;
x30 <=  ka and b5 ;
x31 <= ksa3 ; x32 <= ksa3 ; x33 <= ksa3 ; x34 <= ksa3 ;
x35 <= ksa3 ; x36 <= ksa3 ; x37 <= ksa3 ;
x38 <= k4 or (t_d and ka and b6);
x39 <= k4 or (t_d and ka) ;
x40 <= ksa4 ; x41 <= ksa4 ; x42 <= ksa4 ;
x43 <= t_d and ka and b5 ;
x44 <= k3 or (t_d and ka and b4);
x45 <= k2 or (t_d and ka and (b2 or b3));
x46 <= k1 or (t_d and ka and b1);
x47 <=  ka and b2  ;
x48 <=  ka and b4  ;
x49 <=  ka and b4  ;
x50 <=  ka and b2  ;
x51 <=  ka and b2  ;
x52 <=  ka and b2  ;
x53 <=  ka and b4  ;
x54 <= ksa3 ; x55 <= ksa3 ; x56 <= ksa3 ; x57 <= ksa3 ; x58 <= ksa3 ;
x59 <= ksa3 ;  x60 <= ksa3 ; x61 <= ksa3 ; x62 <= ksa3 ;
x63 <= k2 or (t_d and ka and (b2 or b3));
x64 <= k1 or (t_d and ka and b1);
x65 <= k1 or (t_d and ka and b1);
x66 <= k2 or (t_d and ka);
x67 <= k1 or (t_d and ka and b1);
x68 <= k2 or (t_d and ka and b2);
x69 <= k1 or (t_d and ka);
x70 <= k1 or k2 or (t_d and ka);
x71 <=  ka and b2  ;
x72 <=  ka and b2  ;
x73 <= ksa2 ; x74 <= ksa2 ; x75 <= ksa2 ; x76 <= ksa2 ;
x77 <= ksa2 ; x78 <= ksa2 ; x79 <= ksa2 ;
x80 <= k1 or (t_d and ka and b1);
x81 <= k1 or (t_d and ka);
x82 <= ksa1 ; x83 <= ksa1 ; x84 <= ksa1 ;
-------------------------------------------------------------

p1 <= '0' when (p1 and (x1 or x2 or x3 or x4))='1' else
        '1' when (ksa1 and (p50 or p51 or p52))='1' ;

p2 <= '0' when (p2 and (x21 or x23 or x80))='1' else
        '1' when (ksa2 and (p5 or p6 or p42 or p43))='1' ;
```

```
p3 <= '0' when (p3 and (x38 or x63 or x64))='1' else
        '1' when (ksa3 and (p18 or p19 or p29 or p30))='1' ;

p4 <= '0' when (p4 and (x43 or x44 or x45 or x46))='1' else
        '1' when (ksa4 and (p26 or p27 or p28))='1' ;
-------------------------------------------------------------

p14 <= '0' when (x22 and p14)='1' else
        '1' when (x16 and p9)='1' ;

p15 <= '0' when ((x24 or x25) and p15)='1' else
        '1' when (x17 and p10)='1' ;

p16 <= '0' when ((x26 or x27) and p16)='1' else
        '1' when (x18 and p11)='1' ;

p17 <= '0' when (x28 and p17)='1' else
        '1' when (x19 and p12)='1' ;

p49 <= '0' when (x81 and p49)='1' else
        '1' when ((x77 and p46) or (x79 and p48))='1' ;
-------------------------------------------------------------

p25 <= '0' when (x39 and p25)='1' else
        '1' when ((x35 and p22) or (x37 and p24))='1' ;

p38 <= '0' when (p38 and x65)='1' else
        '1' when (x58 and p33)='1' ;

p39 <= '0' when (p39 and (x66 or x67))='1' else
        '1' when (x59 and p34)='1' ;

p40 <= '0' when (p40 and (x68 or x69))='1' else
        '1' when (x60 and p35)='1' ;

p41 <= '0' when (p41 and x70)='1' else
        '1' when (x61 and p36)='1' ;
-------------------------------------------------------------

p5 <= '0' when (p5 and x12)='1' else
        '1' when (x1 and p1)='1' ;

p6 <= '0' when (p6 and (x5 or x13))='1' else
        '1' when (x2 and p1)='1' ;

p7 <= '0' when (p7 and (x6 or x14))='1' else
        '1' when (x3 and p1)='1' ;

p8 <= '0' when (p8 and (x7 or x8 or x15))='1' else
        '1' when (x4 and p1)='1' ;
```

```
p9 <= '0' when (p9 and x16)='1' else
        '1' when (x5 and p6)='1' ;

p10 <= '0' when (p10 and x17)='1' else
        '1' when (x6 and p7)='1' ;

p11 <= '0' when (p11 and (x10 or x18))='1' else
        '1' when (x7 and p8)='1' ;

p12 <= '0' when (p12 and x19)='1' else
        '1' when ((x10 and p11) or (x11 and p13))='1' ;

p13 <= '0' when (p13 and (x11 or x20))='1' else
        '1' when (x8 and p8)='1' ;
-------------------------------------------------------------

p18 <= '0' when (p18 and x31)='1' else
        '1' when ((x21 and p2) or (x24 and p15))='1' ;

p19 <= '0' when (p19 and x32)='1' else
        '1' when (x14 and p7)='1' ;

p20 <= '0' when (p20 and (x9 or x33))='1' else
        '1' when (x15 and p8)='1' ;

p21 <= '0' when (p21 and (x29 or x34))='1' else
       '1' when (x23 and p2)='1' ;

p22 <= '0' when (p22 and x35)='1' else
        '1' when ((x29 and p21) or (x22 and p14) or (x25 and p15) or (x26 and p16) or
                (x28 and p17) or (x30 and p23) or (x20 and p13))='1' ;

p23 <= '0' when (p23 and (x30 or x36))='1' else
       '1' when (x27 and p16)='1' ;

p24 <= '0' when (p24 and x37)='1' else
        '1' when (x9 and p20)='1' ;
-----------------------------------------------------------------------

p26 <= '0' when (p26 and x40)='1' else
       '1' when ((x38 and p3) or (x36 and p23) or (x39 and p25))='1' ;

p27 <= '0' when (p27 and x41)='1' else
        '1' when (x33 and p20)='1' ;

p28 <= '0' when (p28 and x42)='1' else
        '1' when (x34 and p21)='1' ;
-----------------------------------------------------------------------
-----------------------------------------------------------------------
p29 <= '0' when (p29 and x54)='1' else
```

```
        '1' when (x43 and p4)='1' ;

p30 <= '0' when (p30 and (x47 or x55))='1' else
        '1' when (x44 and p4)='1' ;

p31 <= '0' when (p31 and (x48 or x56))='1' else
        '1' when (x45 and p4)='1' ;

p32 <= '0' when (p32 and (x49 or x50 or x57))='1' else
        '1' when (x46 and p4)='1' ;

p33 <= '0' when (p33 and x58)='1' else
        '1' when (x47 and p30)='1' ;

p34 <= '0' when (p34 and x59)='1' else
        '1' when (x48 and p31)='1' ;

p35 <= '0' when (p35 and (x52 or x60))='1' else
        '1' when (x49 and p32)='1' ;

p36 <= '0' when (p36 and x61)='1' else
        '1' when ((x52 and p35) or (x53 and p37))='1' ;

p37 <= '0' when (p37 and (x53 or x62))='1' else
        '1' when (x50 and p32)='1' ;
-------------------------------------------------------------------------

p42 <= '0' when (p42 and x73)='1' else
        '1' when ((x63 and p3) or (x66 and p39))='1' ;

p43 <= '0' when (p43 and x74)='1' else
        '1' when (x56 and p31)='1' ;

p44 <= '0' when (p44 and (x51 or x75))='1' else
        '1' when (x57 and p32)='1' ;

p45 <= '0' when (p45 and (x71 or x76))='1' else
        '1' when (x64 and p3)='1' ;

p46 <= '0' when (p46 and x77)='1' else
        '1' when ((x71 and p45) or (x65 and p38) or (x67 and p39) or (x68 and p40) or
                (x70 and p41) or (x72 and p47) or (x62 and p37))='1' ;

p47 <= '0' when (p47 and (x72 or x78))='1' else
        '1' when (x69 and p40)='1' ;

p48 <= '0' when (p48 and x79)='1' else
        '1' when (x51 and p44)='1' ;
-------------------------------------------------------------------------
```

```
p50 <= '0' when (p50 and x82)='1' else
        '1' when ((x80 and p2) or (x78 and p47) or (x81 and p49))='1' ;

p51 <= '1' when (p44 and x75)='1' else        --        ***
        '0' when (x83 and p51)='1' ;

p52 <= '0' when (p52 and x84)='1' else
        '1' when (x76 and p45)='1' ;
-------------------------------------------------------------------------

ma <= ka and (p29 or p30 or p31 or p32 or p33 or p34 or p35 or p36 or p37 or
        p42 or p43 or p44 or p45 or p46 or p47 or p48 or p50 or p51 or p52) and not asa;


my <= ka and (p5 or p6 or p7 or p8 or p9 or p10 or p11 or p12 or p13 or
        p18 or p19 or p20 or p21 or p22 or p23 or p24 or p26 or p27 or p28) and not usa;

s1 <= p1 or not ka1;
s2 <= p2 or p14 or p15 or p16 or p17 or p49 or not ka2;
s3 <= p3 or p25 or p38 or p39 or p40 or p41 or not ka3;
s4 <= p4 or not ka4;

FAIL <= asa or usa ;

p1234 <= (p1 or p2 or p3 or p4) or (p14 or p15 or p16 or p17 or p49) or
            (p25 or p38 or p39 or p40 or p41) ;

end Behavioral;
```

Kabinin katta bekleme süresinin (t_d=2.68s) belirlendiği VHDL kodu (t_delay.vhd) :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

----------------------------------------------------------------------------------
--        Kabinin katta bekleme süresinin (t_d=2.68s) belirlendiği VHDL modülü
----------------------------------------------------------------------------------

entity t_delay is
    Port ( input : in std_logic;
           output : out std_logic;
           clk : in std_logic
         );
end t_delay;

architecture Behavioral of t_delay is
signal cntClk  : std_logic_vector (27 downto 0) :="0000000000000000000000000000";
signal dvdClk, q2 : std_logic ;
begin

--  f(clk)= 50MHz ==> T(clk) = 20 ns
-- T(dvdclk) = 20 ns x (2^27) = 2.68 s -->yükselen kenar

process (clk, input)
begin
  if input='0' then
    cntClk <= "0000000000000000000000000000" ;
  elsif (clk'event and clk='1') then
    cntClk <= cntClk + 1 ;
  end if;
end process;

dvdClk <= cntClk(27)          ;

process (dvdClk, input)
begin
  if input='0' then
    q2 <= '0';
  elsif dvdClk'event and dvdClk='1' then
    q2 <= input;
  end if;
end process;

output <= input and q2 ;

end Behavioral;
```

Toplamalı asansör kontrol sisteminin VHDL kodu (elevator2.vhd) :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

----------------------------------------------------------------------------------
--        TOPLAMALI ASANSÖR KONTROL SİSTEMİNİN VHDL KODU
----------------------------------------------------------------------------------

entity elevator2 is
PORT(
                clk : IN std_logic;
                k1 : IN std_logic;
                k2 : IN std_logic;
                k3 : IN std_logic;
                k4 : IN std_logic;
                b1 : IN std_logic;
                b2 : IN std_logic;
                b3 : IN std_logic;
                b4 : IN std_logic;
                b5 : IN std_logic;
                b6 : IN std_logic;
                ksa1 : IN std_logic;
                ksa2 : IN std_logic;
                ksa3 : IN std_logic;
                ksa4 : IN std_logic;
                ka1 : IN std_logic;
                ka2 : IN std_logic;
                ka3 : IN std_logic;
                ka4 : IN std_logic;
                asa : IN std_logic;
                usa : IN std_logic;
                t_d : IN std_logic;
                s1 : OUT std_logic;
                s2 : OUT std_logic;
                s3 : OUT std_logic;
                s4 : OUT std_logic;
                ma : OUT std_logic;
                my : OUT std_logic;
                FAIL : OUT std_logic;
                p1234 : OUT std_logic
                );
end elevator2;

architecture Behavioral of elevator2 is

        COMPONENT toplamali
        PORT(
```

```
            k1 : IN std_logic;
            k2 : IN std_logic;
            k3 : IN std_logic;
            k4 : IN std_logic;
            b1 : IN std_logic;
            b2 : IN std_logic;
            b3 : IN std_logic;
            b4 : IN std_logic;
            b5 : IN std_logic;
            b6 : IN std_logic;
            ksa1 : IN std_logic;
            ksa2 : IN std_logic;
            ksa3 : IN std_logic;
            ksa4 : IN std_logic;
            ka1 : IN std_logic;
            ka2 : IN std_logic;
            ka3 : IN std_logic;
            ka4 : IN std_logic;
            asa : IN std_logic;
            usa : IN std_logic;
            t_d : IN std_logic;
            s1 : OUT std_logic;
            s2 : OUT std_logic;
            s3 : OUT std_logic;
            s4 : OUT std_logic;
            ma : OUT std_logic;
            my : OUT std_logic;
            FAIL : OUT std_logic;
            p1234 : OUT std_logic
            );
    END COMPONENT;

    COMPONENT t_delay
    PORT(
            input : IN std_logic;
            clk : IN std_logic;
            output : OUT std_logic
            );
    END COMPONENT;

signal sgn_t_d : std_logic;
signal sgn_p1234 : std_logic;

begin

    Inst_toplamali: toplamali PORT MAP(
            k1 =>k1 ,
            k2 =>k2 ,
            k3 =>k3 ,
            k4 =>k4 ,
```

```
            b1 =>b1 ,
            b2 =>b2 ,
            b3 =>b3 ,
            b4 =>b4 ,
            b5 =>b5 ,
            b6 =>b6 ,
            ksa1 =>ksa1 ,
            ksa2 =>ksa2 ,
            ksa3 =>ksa3 ,
            ksa4 =>ksa4 ,
            ka1 =>ka1 ,
            ka2 =>ka2 ,
            ka3 =>ka3 ,
            ka4 =>ka4 ,
            asa =>asa ,
            usa =>usa ,
            s1 =>s1 ,
            s2 =>s2 ,
            s3 =>s3 ,
            s4 =>s4 ,
            ma =>ma ,
            my =>my ,
            FAIL =>FAIL ,
            t_d =>sgn_t_d ,
            p1234 =>sgn_p1234
    );

    Inst_t_delay: t_delay PORT MAP(
            input =>sgn_p1234 ,
            output =>sgn_t_d ,
            clk =>clk
    );

end Behavioral;
```