İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

BULANIK PETRİ AĞLARIYLA BİR DEMİRYOLU TRAFİK SİSTEMİNİN MODELLENMESİ VE FPGA ÜZERİNDE GERÇEKLENMESİ

YÜKSEK LİSANS TEZİ

Tamer TAŞ

Anabilim Dalı : Elektronik Mühendisliği

Programı : Elektronik Mühendisliği

Tez Danışmanı: Doç. Dr. Mürvet KIRCI

EYLÜL 2010

ÖNSÖZ

İnsanoğlunun varoluşundan günümüze kadar toplumsal gereksinimlerin karşılanması için teknolojik sistemlere sıkça başvurulmuştur. Yüzyıllardan beri üstel şekilde artarak çığ gibi büyüyen bilimsel bilginin insanın mantığı, zekası ve hayal gücüyle birleşmesi sonucu bu teknolojik sistemler mümkün olduğunca iyileştirilerek hayata geçirilmeye çalışılmıştır. Belirli girdi kümelerine göre beklenen çıktı kümelerini veren bu analog ve dijital sistemlerin tüm zamanlarda istenen şekilde çalışmasının garanti altına alınmasının gerekliliği ayrı bir çalışma alanı doğurmuştur. Bütün istisnaların ve olasılıkların göz önüne alınmasıyla birlikte çeşitli algoritmik tabanlara dayanan modelleme yöntemleri oluşturulmuştur. İşte Petri Ağları da eşzamanlı olaylardan dağıtık olaylara kadar, stokastik yaklaşımdan belirleyici olmayan (non-deterministic) yapılara kadar birçok ihtiyaca cevap verebilecek öngörüyle hazırlanan bir modelleme düşüncesi ortaya koymuştur. Bulanık Petri Ağları ise Bulanık Mantık bilimindeki matematiksel kuralların Petri Ağlarına uygulanmasıyla elde edilmiş dönüşümsel ve etkileyici bir model önermektedir. Bu tez çalışmasında Bulanık Mantık – Petri Ağları bütünleşmesi ile uygulamaya yönelik neler yapılabileceği anlatılmış, Bulanık Petri Ağlarının demiryolu trafik ağlarına temel teşkil eden yapılara göre uyarlanması, sonuç olarak gerçek hayattan alınan bu sistemin FPGA üzerinde sayısal devre olarak gerçeklenmesi ele alınmıştır.

Tez çalışmamda bilgi birikimi ve tecrübesiyle bana yol gösteren, ilgisi ve desteğiyle beni cesaretlendiren değerli danışman hocam Doç. Dr. Mürvet KIRCI'ya yürekten teşekkürlerimi sunuyorum.

Ayrıca manevi desteklerini kesintisiz sağlayan tüm arkadaşlarıma, akılcı ve mantıklı hayat görüşleriyle ömrüm boyunca bana eşsiz bir örnek olan ağabeyim Zafer TAŞ'a, bugünlere gelmemde en büyük emeği geçen annem Sevil TAŞ ve babam Muzaffer TAŞ'a minnet duygularımı iletmek isterim. TÜBİTAK'a da yüksek lisans eğitimim boyunca sağladığı maddi desteklerinden dolayı çok teşekkür ediyorum.

Mayıs 2010

Tamer TAŞ

Elektronik Mühendisi

İÇİNDEKİLER

Sayfa

KISALTMALAR

ABS : Anti-lock Braking System
BPA : Bulanık Petri Ağı
DGD : Durum Geçiş Diyagramı
DOM : Degree of Membership
FF : Flip Flop
FPGA : Field Programmable Gate Array
FPN : Fuzzy Petri Net
PN : Petri Net
RTL : Register Transfer Level
SLR : Single-Lens Reflex
TYTSM : Tren Yolu Trafik Sistemi Modeli
UML : Unified Modeling Language
ÜD : Üyelik Derecesi
VHDL : VHSIC Hardware Description Language
VHSIC : Very High Speed Integrated Circuit

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

BULANIK PETRİ AĞLARIYLA BİR DEMİRYOLU TRAFİK SİSTEMİNİN MODELLENMESİ VE FPGA ÜZERİNDE GERÇEKLENMESİ

ÖZET

Bu tez çalışmasında ayrık sistemlerin tanımlanmasında kullanılan etkin bir matematiksel ve grafiksel modelleme yöntemi olan Petri ağlarının bulanık mantık kurallarına dayanarak gerçeklenmesi ele alınmıştır. Klasik Petri ağlarının sayısal devreli sistemlere uyarlanabilmesi, bulanık mantık kurallarına göre yönetilen bir Bulanık Petri ağının da benzer şekilde elde edilebilmesine ön ayak olmuştur. BPA ile modellenmiş demiryolu trafik sisteminin sayısal devre halinde tasarlanması ve gerçek hayatta çalışırlığının gösterilmesi de FPGA ile mümkün olmuştur.

İkinci bölümde, Petri ağlarının bileşenleri tanıtılmış, dinamizmi sağlayan jetonların hangi koşullar altında ateşlenebildiği örnek diyagramlarla açıklanmıştır. Petri ağlarının matematiksel altyapısı grafiksel gösterimiyle birlikte desteklenmiştir. Petri ağlarının analizinde öne çıkan temel karakteristik özellikleri anlatılmıştır.

Üçüncü bölümde, bulanık mantığın (fuzzy logic) dayandığı kurallar, klasik mantıktan ayrılan yönleri, uygulamada kullanım alanları ve Petri ağlarıyla ilişkisi ele alınmıştır. Ayrıca tezin üçüncü bölümden sonraki kısmında BPA ile algoritmik modelleme örneklerle birlikte gösterilmiştir.

Dördüncü yani son bölümde, bulanık mantık temelli Petri ağlarının sayısal devrelerle ilişkisi belirlenmiştir. Temel amaç kapsamlı bir ayrık sistemin BPA ile tasarlanması olup uygulama çalışması olarak bir demiryolu trafik sisteminin BPA ile modellenmesi, ilgili sayısal devrenin elde edilmesi, FPGA üzerinde sentezlenmesine yer verilmiştir.

MODELLING OF A RAILWAY TRAFFIC SYSTEM WITH FUZZY PETRI NETS AND IMPLEMENTATION ON FPGA

SUMMARY

Throughout the thesis study, design and implementation of Petri nets, which is an effective mathematical and graphical method used for describing discrete systems, based on the principles of fuzzy logic are analyzed. Adaption of classical Petri nets to digital circuit systems led to obtain Fuzzy Petri nets that are managed according to the principles of fuzzy logic. Design of FPN modeled railway traffic system in the form of digital circuit and the executability in real life was possible via FPGA.

In the second chapter, the elements of Petri nets are demonstrated and then the firing properties of the tokens that provide dynamic characteristics are explained by the help of sample diagrams. The mathematical infrastructure of Petri nets are supported with graphical notation. The featured fundamental characteristics of Petri nets are examined.

In the third chapter, the basic rules for fuzzy logic, distinguishing features from classical logic, usage areas in application and the relationship between fuzzy logic and Petri nets are discussed. Furthermore, in the following part after 3rd, algorithmic modelling with FPN is indicated with examples.

In the fourth namely last chapter, the relationship between Petri nets that based on fuzzy logic and digital circuits are determined. The essential purpose is to design a comprehensive discrete system with Fuzzy Petri nets. As an application study, the Fuzzy Petri net modelling of a railway traffic system, generation of relevant digital circuit and synthesis on FPGA is included.

1. GİRİŞ

Doğada var olan bir çok dinamik sistem birbirlerine bağlı olmayan değişkenlerden etkilenmektedir. Bir insan vücudu, farklı işlevlere sahip organların bir araya gelmesiyle oluşmuş karmaşık ve ayrık bir sisteme benzetilebilir. Örneğin kalp kendi çapında vücuda kan pompalama görevi yaparken diyafram kası solunum sisteminin çalışmasında görev alır, karaciğer kandaki şeker miktarını ayarlar, böbrekler ise su ve elektrolit dengesini düzenler. Bu örnek genişletilerek kullanılacak olursa çevremizdeki olayların belirli koşullar altında gerçekleştiği ve ayrık olayların dolaylı da olsa birbirlerinden etkilendiği sonucuna varılabilir. Sistemlerin modellenmesinde lineer ya da lineer olmayan analog sinyaller ile bilgisayar ortamında işlenecek sayısal sinyallerin kullanılması mümkündür. Günlük hayatta rastlanan hemen hemen tüm olaylar belirli bir zaman aralığında sürekli olarak değişkenlik gösteren analog sinyallerdir. Çevremizdeki bu tür olayları kolaylıkla anlayabilmek için işaretlerin bilgisayar ortamında işlenebilmesi, dolayısıyla sayısal işaretlere çevrilmesi gerekir. Bilgisayar ortamına aktarılan veriler üzerinde yazılımsal algoritmalar geliştirilerek sistemler modellenebilmektedir.

İşte 1962'de Darmstadt Teknik Üniversitesinden Carl Adam Petri "Communication with Automata" adlı doktora teziyle Petri ağları adı verilen modelleme tekniğini ortaya çıkarmıştır (Braue ve ReisigCarl, 2006). Ayrık sistemlerin modellenmesinde sıkça kullanılmakta olan Petri ağları klasik lojikte kullanılan durum geçiş diyagramlarına oldukça benzemekle birlikte karakteristiği gereği DGD'larına göre daha kapsamlı sistemlerde de kullanılabilmektedir. Petri ağları belirli koşullar yerine geldiğinde tetiklenen olayları, sonraki durumları ve sistemin dinamik yapısını analiz etmekte oldukça güçlü özellikler gösterir. Hem matematiksel hem görsel şekilde kolaylıkla gösterilebilmesi bir avantajdır.

Günümüzde standart olarak kullanılagelen UML aktivite diyagramları gibi Petri ağları da seçimli, eş zamanlı ve ardışıl olayların ifade edilmesinde görsel yetenekler sunmaktadır. UML ve benzeri kavramlardan farklı olarak üstün olduğu alan ise

sistemlerin yeterli olarak tanımlanabilmesine olanak verecek bir dile sahip olması ve matematiksel olarak açıklanmasının etkili olmasıdır.

Petri ağları güçlü modelleme yetenekleri nedeniyle hem yazılım sektöründe hem de endüstriyel kontrol sistemlerinde geniş bir kullanım alanı doğmuştur. Üretim sistemlerinde performans analizi ve hata kontrolünde, robotik ve yapay zeka uygulamalarında, yapay sinir ağları ve iletişim sistemlerinde, programlama algoritmalarında, donanım tasarımında ve daha sayılabilecek bir çok alanda uygulama olanağı bulmuştur. Üzerinde durulması gereken bir başka önemli konu ise Bulanık Mantık adı verilen sistemdir. İlk defa 1965 yılında Kaliforniya Üniversitesinden (Berkeley) bilgisayar bilimleri alanında çalışmalar yapan Prof. Lotfi A. Zadeh tarafından ortaya konmuştur (Hellmann, 2010). Özellikle 1980'lerden sonra özellikle Japonların bulanık mantık üzerine yaptıkları çalışmalarıyla önemli gelişmeler katedilmiştir. Klasik Boole Cebri lojiğinde bir değişkenin alabileceği değer iki olasılıklı iken (yüksek-alçak, 0 veya 1, doğru-yanlış…gibi) bulanık mantık sınırlı ve belirli bir aralıkta bulunan herhangi bir değeri alabilmektedir. Bu açıdan çok-değerli lojik olarak da anılmaktadır. Böyle bir sistemin oluşturulmasının nedeni insan beyninin doğayı nasıl algıladığı ile yakından ilişkilidir. Dikkat edilirse dünyada meydana gelen bir çok olay kesin olmayan yargılarla, ikili olasılık dışında başka ihtimallerle de tasvir edilebilir. Örneğin genel olarak bir suyun sıcaklığının $5\,C^o$ olması "biraz soğuk" diye nitelendirilirken, $20\,C^o$ olması "ılık" diye, aynı suyun sıcaklığının $50\,C^o$ olması "biraz sıcak" şeklinde, $90\,C^o$ olması "çok sıcak" olarak belirtilebilir. Tamamen insanın duyu organlarıyla algılanan bu veriler bilgisayar ortamına aktarılırken bulanık mantık kurallarıyla birlikte etkin bir şekilde gösterilebilir. Böylece bilgisayarın "insan gibi düşünmesi" yani yapay zekanın temel kavramı da işletilmiş olur. Bilgisayar programlarında belirsiz kavramların bulanık mantığın belirli matematiksel kurallarıyla işlenmesi mümkündür. Bulanık mantık kesin olarak tanımlanmamış belirsiz değişkenlerle ilgilendiğinden olasılık teorisiyle sıkça karıştırılmaktadır fakat daha farklı bir alternatif yöntemdir. Bulanık mantığın bilgisayar sistemlerinde uygulanması, geleneksel yöntemlere göre kurulum çalışmalarında verimlilik yani karmaşıklığın ve sistemi yüklemek için harcanan çabanın azaltılması, maliyet kazancı gibi avantajlar sağlamaktadır. Bulanık mantık uygulamaları karmaşık endüstriyel süreçlerde, ev elektroniği, akıllı sistemlerde ve

daha bir çok alanda kullanılmaktadır. Gün geçtikçe literatürde de bulanık mantık üzerine yapılan yayınlarda artış olmaktadır.

Bulanık mantık kurallarının Petri ağlarıyla birleştirilmesi de BPA'ların oluşturulmasını sağlamıştır. Bu durum geleneksel Petri ağı tanım kümesine yeni değişkenler ve fonksiyonların eklenmesini gerekli kılmıştır. Belirli uygulamalarda BPA'lardan yararlanılmakta ve karar verme süreçleri bu sayede iyileştirilmektedir. Çok bileşenli karmaşık sistemlerin BPA'lar ile modellenmesinde aslında temel olarak Petri ağlarının çalışma ilkelerinden yararlanılır. Yine benzer şekilde BPA'ların donanım olarak elde edilebilmesi de Petri ağlarının gerçekleştirilmesinde kullanılan yöntemlerle yakından ilişkilidir.

Sayısal devre tasarımında önemli bir yeri olan donanım sistemlerine örnek olarak da FPGA denilen sayısal tümleşik devreler verilebilir. Türkçe olarak "Alanda Programlanabilir Kapı Dizileri" anlamına gelmektedir. Burada "alanda" ifadesiyle anlatılmak istenen de FPGA üretiminden sonra da değişik coğrafyalarda tekrar tekrar programlanabiliyor olmasıdır. Kullanıcı özgürlüğü sunan geniş çaptaki programlanabilir lojik blok sayısı ve yüksek hızı nedeniyle günümüzde oldukça sık tercih edilen lojik devre donanımı sentezleme platformudur. Bir bilgisayar üzerinde çalışan sürücü yazılımıyla kolaylıkla kontrol edilebilen, üzerindeki mantıksal kapı bağlantılarının da ayarlanabildiği esnek bir yapıdır. Karmaşık devrelerin de sentezlenmesindeki üstünlüğü ve işlevsel yeteneği bu sistemlerin gün geçtikçe daha da geliştirilmesini sağlamaktadır. BPA'lar da çok sayıda lojik kapı içeren farklı tipteki flip-floplar, karşılaştırıcılar ve veri seçici (multiplexer) elemanlarının uygun şekilde konfigürasyonu ile FPGA üzerinde gerçeklenebilmektedir. Tasarlanan devrelerin en az maliyetle ve en yüksek performansla elde edilebilmesi simülasyon ve optimizasyon çalışmalarını gerektirmektedir.

Tezin ikinci bölümünde, Petri ağlarının getirdiği dinamik yaklaşım dayandığı matematiksel ve grafiksel öğeleriyle birlikte açıklanmıştır.

Üçüncü bölümde bulanık mantığın karakteristik özellikleri, kural ve prensipleri, bulanık mantık metodolojisinin uygulamadaki kullanım alanları örneklerle birlikte anlatılmıştır. BPA'larına giriş yapılmış, farklı yazılımlarda nasıl tanımlanabileceği gösterilmiştir.

Tezin son bölümü olan dördüncü bölümde bir demiryolu trafik sistemi modelinin BPA'lar ile tasarımı ve FPGA üzerinde gerçeklenmesi konusu üzerinde durulmuştur.

2. PETRİ AĞLARI

2.1 Petri Ağları Tanım ve Kavramlar

Petri ağlarında olaylar "geçiş"ler ile temsil edilir. Bir geçişin (olay, transition, event) gerçekleşebilmesi için belirli koşulların yerine gelmesi gereklidir. Bu koşullara ilişkin bilgiler ise "yer"ler (koşullar, places, conditions) içerisinde yer alır. Bir geçişe "giriş" (input) olarak ilişkilendirilmiş yerler ilgili olduğu geçişin gerçekleşebilmesi için gereken koşullar ile de ilişkilendirilmiş demektir. Geri kalan yerler ise geçişe "çıkış" olarak bağlanmış olup ilgili geçişin gerçekleşmesiyle etkilenen koşullarla ilişkilendirilmiş demektir.

Geçişler, yerler ve aralarındaki belirli ilişkiler Petri ağı grafiklerinin temel bileşenlerini oluşturur. Bir Petri ağında iki çeşit bağlantı düğümü (yerler ve geçişler) ve bu düğümleri (nodes) birbirine bağlayan yönlü bağlantı çizgileri bulunur.

Petri ağları iki parçalı grafiklerdir, yani bir yönlü ok aynı cins bağlantı düğümlerini birbirine bağlayamaz, bir geçiş düğümünü bir yer düğümüne ya da bir yer düğümünü bir geçiş düğümüne bağlayabilir. Petri ağları grafiklerinin daha kapsamlı bir tanımı şöyle yapılabilir (Cassandras ve Lafortune, 2008):

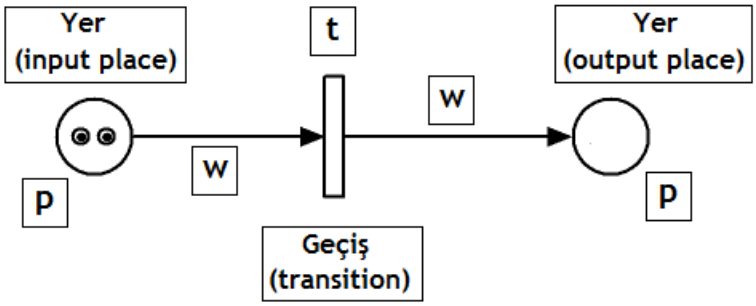
Petri ağı grafiği:

Bir Petri ağı grafiği (veya Petri ağı yapısı) (P, T, A, w) parametrelerini içeren ağırlıklı ve iki parçalı bir grafiktir (Şekil 2.1) ve aşağıdaki koşulları sağlar:

- P, *yerlerin* sonlu kümesidir (grafikteki bir bağlantı düğümü cinsi)
- T, *geçişlerin* sonlu kümesidir (grafikteki diğer bağlantı düğümü cinsi)
- $A \subseteq (P \times T) \cup (T \times P)$, yerlerden geçişlere veya geçişlerden yerlere doğru yönlendirilmiş okların kümesidir.
- $w : A \rightarrow \{1, 2, 3, \ldots\}$, oklar üzerindeki *ağırlık fonksiyonu*dur.

Burada (P, T, A, w)'nin izole edilmiş yer ve geçişleri bulunmadığı varsayılmaktadır.

$(P \cap T = \emptyset$ ve $P \cup T \neq \emptyset)$



Şekil 2.1: Petri Ağı Grafiğinin Temel Bileşenleri.

Yerlerin kümesi $P = \{p1, p2, \ldots, pn\}$ ile,

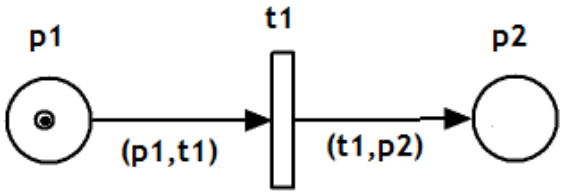
Geçişlerin kümesi $T = \{t1, t2, \ldots, tm\}$ ile gösterilir ve

Yerlerin sayısı $|P| = n$,

Geçişlerin sayısı $|T| = m$ ile temsil edilir.

Tipik bir ok ise eğer yer düğümünden geçiş düğümüne doğru yönleniyorsa (pi, tj),

Geçiş düğümünden yer düğümüne yönleniyorsa (tj, pi) şeklinde gösterilir (Şekil 2.2).



Şekil 2.2: Petri Ağı Grafiğinin Temel Yapısı.

Petri ağları durum geçiş diyagramlarına benzer ama farklıdır. Bir Petri ağı grafiğinde düğümler ya P kümesinden seçilebilen yerler ya da T kümesinden seçilebilen geçişlerdir. DGD'de ise bir durum geçişine sebep olan her bir olay için tek bir ok kullanılır. Bir Petri ağı grafiğinde iki düğümü birleştirmek için çok sayıda ok kullanılabilir veya aynı şekilde okların sayısını temsil eden her bir oka bir ağırlık atanabilir.

Bir Petri ağının tanıtımında;

tj geçişine giren giriş yerlerinin kümesini $I(tj)$ ile,

tj geçişinden çıkan çıkış yerlerinin kümesini $O(tj)$ ile gösterebiliriz.

Böylece şu bağıntı kümelerini tanımlayabiliriz:

$$I(tj) = \{pi \in P : (pi, tj) \in A\}, \quad O(tj) = \{pi \in P : (tj, pi) \in A\} \qquad \textbf{(2.1)}$$

(2.1)'e benzer bir gösterim verilen pi: $I(pi)$ ve $O(pi)$ yerleri için giriş ve çıkış geçişlerini tanımlamak için kullanılabilir.

Petri ağı grafiklerini tanımlarken iki çeşit düğüm arasındaki farkı şekilsel olarak da ortaya koymak gerekir. Bu nedenle yerler yuvarlak çember şeklinde, geçişler de çubuk şeklinde çizilmelidir. Yerleri ve geçişleri birbirine bağlayan oklar bağlantı okları kümesi olan A kümesinin elemanlarını oluşturur. Bunun için *pi* yerinden *tj* geçişine yönlendirilmiş bir ok $pi \in I(tj)$ anlamına gelir.

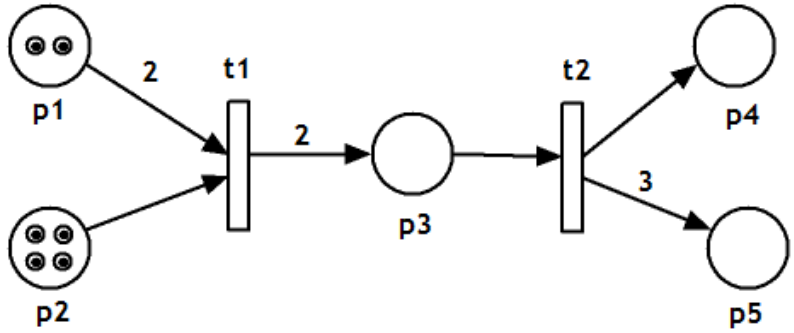
Ayrıca eğer $w(pi, tj) = k$ ise *pi* den *tj'ye k* adet ok bulunur, ya da başka bir deyişle tek bir ok *k* ağırlığı ile gösterilir. Benzer şekilde, eğer *tj* geçişinden *pi* yerine yönlendirilmiş *k* tane ok var ise bu $pi \in O(tj)$ ve $w(tj, pi) = k$ anlamına gelir. Genellikle Petri ağı grafiği üzerinde ağırlıklar çoklu oklar ile temsil edilir. Buna rağmen Petri ağı üzerinde çok sayıda ağırlık bulunduğu zaman daha sade ve güzel bir gösterim için tek bir okun çizgisi yanına ağırlığı yazılır. Bir okun ağırlığı (w) ise pozitif bir tamsayı ile gösterilir. Eğer okun üzerine ağırlık yazılmamış ise otomatik olarak $w(pi, tj) = 1$ ve $w(tj, pi) = 1$ kabul edilir (Şekil 2.2.). Ağırlık fonksiyonu ile ilgili şu bağıntılar geçerlidir:

$pi \notin I(tj)$ olduğu zaman $w(pi, tj) = 0$, $pi \notin O(tj)$ olduğu zaman da $w(tj, pi) = 0$ eşitliği gerçeklenir. Ayrık sistemlerin, fuzzy kontrolörlerinin, bilgisayar programlarının ve benzeri sistemlerin Petri ağları ile modellenmesinde Çizelge 2.1'deki dönüşümler dikkate alınır (Murata, 1989, Shim ve Lee 2010).

Çizelge 2.1 : Geçiş ve Yerlerin Farklı Anlamları

Giriş Yerleri	**Geçişler**	**Çıkış Yerleri**
Ön koşullar	Olaylar	Sonraki koşullar
Giriş verisi	Hesaplama basamağı	Çıkış verisi
Giriş sinyalleri	Sinyal işlemcisi	Çıkış sinyalleri
Gerekli kaynaklar	Görev veya iş	Serbest kalan kaynaklar
Koşullar	Mantık önermesi	Sonuçlar
Sürücü (Buffer)	İşlemci (Processor)	Sürücü (Buffer)

Petri ağlarının tanımını yaparken dinamik davranışı sağlayan nesnelerden (jeton, obje, token, marking) de bahsetmek gereklidir. Petri ağlarının 5'inci parametresi de nesnelerdir. $Mo: P \rightarrow N$ (0,1,2,3,…) gösterimi *Mo* başlangıç durumunda jetonların Petri ağı üzerinde dağılımını anlatır. Şekil 2.3'teki Petri ağı grafiğini dikkate alırsak, Mo: {p1→ 2, p2→ 4, p3→ 0, p4→ 0, p5→ 0} Genellikle yerlerin indislerine göre ardışık olarak daha sade gösterim de yapılmaktadır (Muscholl, 2010): Mo: {2, 5, 0, 0, 0}



Şekil 2.3: Petri Ağlarında Jetonların Durumu ve Geçişlerin Ateşlenmesi

Marking (imleme, işaretleme) M harfi ile gösterilir ve jetonların Petri ağı içerisinde hangi yerlerde bulunduğuna ilişkin bir fonksiyonu ifade eder.

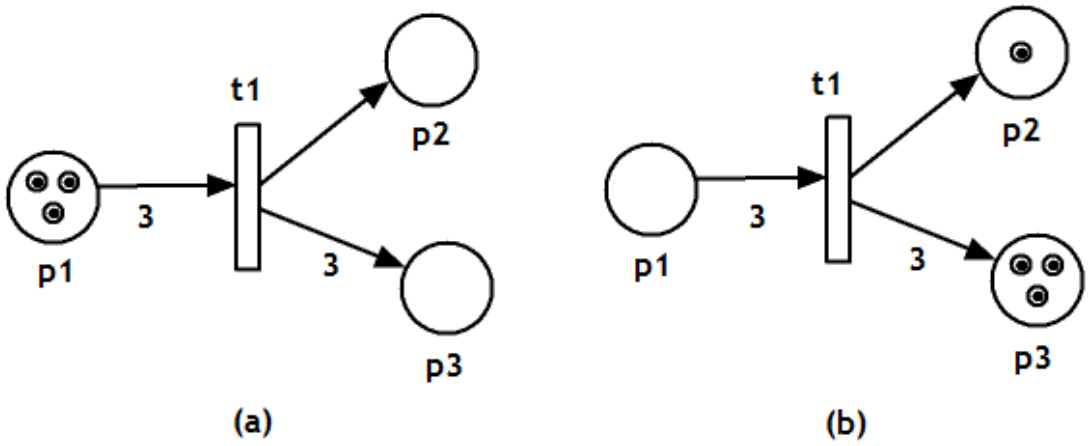
Eğer hiçbir p yerinde birden fazla jeton bulunmuyor ise imleme fonksiyonu M(p) farklı şekillerde de gösterilebilir: Mo = {p5, p7, p8} veya Mo = {p1, p1, p2, p2, p2, p2, p2} şeklinde de gösterimler mümkündür. M(pi) ifadesi bir M imleme fonksiyonunda, bir "pi" yerinde bulunan jeton sayısını belirtir.

5 parametreli bir Petri ağının genel tanımı ise PN = (P, T, A, W, Mo) şeklindedir.

Eğer özel bir başlangıç imlemesi (Mo, initial marking) tanımlanmadıysa Petri ağı yapısı N = (P, T, A, W) şeklinde, bir başlangıç imlemesi verildiyse PN = (N, Mo) şeklinde gösterilir ($P \cap T = \varnothing$ ve $P \cup T \neq \varnothing$).

Birçok sistemin davranışı sistem durumları ve değişimleri cinsinden tanımlanabilir. Bir sistemin dinamik davranışının benzetimini yapabilmek için, Petri ağındaki bir durum ya da işaretleme, aşağıdaki ateşleme kuralına (firing rule) göre değişir (Murata, 1989):

1) Bir t geçişinin etkin olması için; kendisine giriş olarak bağlanmış her bir p yerinde w(p,t) jeton bulunmalıdır. Burada w(p,t) p'den t'ye bağlanan okun ağırlığını belirtir.
2) Etkin olan bir geçiş ateşlenebilir de ateşlenmeyebilir de (herhangi bir olayın gerçekleşmesine bağlı olarak).
3) Etkin bir t geçişinin ateşlenmesi, t'ye giriş yapan p giriş yerlerinden bağlı oldukları okun ağırlığı w(p,t) kadar jeton eksilmesine ve t'nin çıkışında bulunan her p çıkış yerine bağlı oldukları okun ağırlığı w(t,p) kadar jeton eklenmesine neden olur (Şekil 2.4) (Zurawski ve Zhou, 1994).



Şekil 2.4 : (a) t1 geçişi etkin, (b) t1'in ateşlenmesinden sonraki durum

Durum geçiş fonksiyonu için M(p) imleme fonksiyonu yerine x gösterimini kullanırsak, $f : N^n \times T \rightarrow N^n$, ve Petri ağı (*P, T,A, w, x*) aşağıdaki bağıntıları sağlar:

Bir *tj* geçişi için ancak ve ancak;

tüm $pi \in I(tj)$ için $x(pi) \geq w(pi, tj)$ **(2.2)**

olduğunda $tj \in T$ 'dir denebilir.

Eğer *f*(**x**, *tj*) tanımlandıysa $x' = f(x, tj)$ fonksiyonu aşağıdaki gibi kurulabilir:

$x'(pi) = x(pi) - w(pi, tj) + w(tj, pi), \quad i = 1, \ldots, n$ **(2.3)**

(2.2) koşulu durum geçiş fonksiyonunun sadece etkin geçişler için tanımlandığını gösteriyor, bu açıdan Petri ağlarındaki "etkin geçiş", otomata'da (automaton, otomatik robot sistemi) "uygulanabilir olay" anlamına gelir. Oysa otomata'da durum geçiş fonksiyonları rastgele olurken Petri ağı, yapısında tanımlanan Petri ağı durum geçiş fonksiyonuna uyar. Bunun içindir ki **(2.3)** ile tanımlanan sonraki durum, bariz bir şekilde bir t geçişinin I(t) giriş ve O(t) çıkış yerlerine ve bu yerleri geçişlere bağlayan okların ağırlıklarına bağlıdır.

(2.3)'e göre, eğer *pi, tj'*nin bir giriş yeri ise *pi* den *tj*'ye yönlendirilmiş okun ağırlığı kadar jeton kaybeder, eğer *pi, tj'*nin bir çıkış yeri ise *tj* den *pi*'ye yönlendirilmiş okun ağırlığı kadar jeton kazanır. Açıkça söylemek gerekirse, *pi'*nin, *tj'*nin hem çıkış hem de giriş yeri olması; **(2.3)**'deki durumda *pi'*den *w*(*pi, tj*) adet jeton azalması ve aynı zamanda hızla yeni *w*(*tj, pi*) adet jetonu geri koymasıyla mümkün olabilir. Bir Petri ağında ateşlenme olduğu zaman sistemdeki toplam jeton sayısının korunması da gerekmez (Şekil 2.4). Bu durum **(2.3)**'te de açıkça görülebilir. Böylece aşağıdaki **(2.4)** bağıntıları geçerlidir:

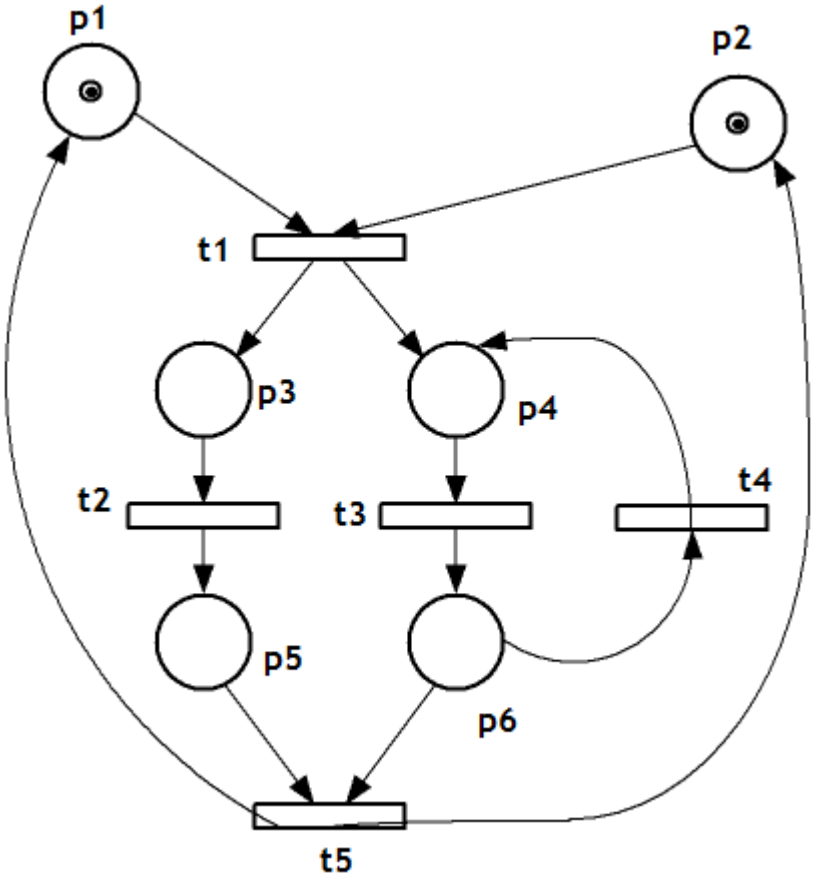
$$\sum_{pi \in P} w(tj,\ pi) > \sum_{pi \in P} w(pi,\ tj) \qquad \text{veya} \qquad \sum_{pi \in P} w(tj, pi) < \sum_{pi \in P} w(pi, tj)$$

(2.4)

Bu durumda $x' = f(x, tj)$ herhangi bir şekilde x'ten daha az ya da daha fazla jetona sahip olabilir. Genellikle belirli sayıda geçiş ateşlenmesi ile birlikte son durumda **x** = [0, . . . , 0] olması veya Petri ağındaki yerlere ilişkin jeton sayılarının bir dizi ateşlenme sonunda rastgele olarak çoğalması mümkündür. Jeton sayılarının dengesiz ve rastgele artması Petri ağlarının otomata'dan ayrılan önemli bir farkıdır ki tanım olarak sonlu durumlu makinelerde (automata) sonlu sayıda durum bulunur (Cassandras ve Lafortune, 2008). Buna karşın sonlu bir Petri ağı grafiğinde sınırsız sayıda durumla karşılaşılabilir.Aşağıda Şekil 2.5'te Petri ağındaki değişkenler örnek üzerinde gösterilmektedir (Shim, ve Lee, 2010):

Çizelge 2.2: Şekil 2.5'teki grafiksel ilişkinin matematiksel ilişkiler dizisi

Yer ve Geçişler, İmleme	**Girişler**	**Çıkışlar**
P: {p1, p2, p2, p4, p5,p6}	I(t1) = {p1,p2}	O(t1) = {p3,p4}
T: {t1, t2, t3, t4, t5}	I(t2) = {p3}	O(t2) = {p5}
M1: {1, 1, 0, 0, 0, 0}	I(t3) = {p4}	O(t3) = {p6}
	I(t4) = {p6}	O(t4) = {p4}
	I(t5) = {p5,p6}	O(t5) = {p1,p2}



Şekil 2.5: Petri ağı bileşenlerinin grafiksel model ile gösterimi

Petri ağlarının vektörel gösteriminde matrislerden de yararlanılabilmektedir: Şekil 2.5'teki Petri ağı grafiğindeki geçişler, yerlerin durumu dikkate alınırsa aşağıdaki bağıntılar geçerlidir:

P={p1, p2, p3, p4, p5}

T={t1, t2, t3, t4, t5} şeklinde olup 5x5'lik matris ile gösterimi şöyledir:

$$P_i = \begin{array}{c} \begin{matrix} t1 & t2 & t3 & t4 & t5 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{matrix}$$

$$P_o = \begin{array}{c} \begin{matrix} t1 & t2 & t3 & t4 & t5 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{matrix}$$

şeklinde gösterilirse, sadece bu iki matris ile Petri ağı grafiği çizilebilir. Yerlere göre giriş-çıkış ilişkileri verilen matrislerde 1 rakamı kendisine karşı gelen yer ile geçiş arasında bağlantı olduğunu, 0 ise herhangi bir doğrudan bağlantı oku bulunmadığını gösteriyor. Bu matrisleri çıkardıktan sonra bir de M1 işaretleme-imleme kümesini yazarsak jetonların başlangıç durumunu da grafiğe ekleme şansımız olur: M1 = {1,0,0,0,0,}. Örnek bir F1 ateşlenmesi de yine matrislerle gösterilebilmektedir:

$$F1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{matrix} \qquad F1' = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{matrix}$$

Önceki durumda F1 iken sonraki durum F1' olmuştur.

2.2 Petri Ağlarının Karakteristik Özellikleri ve Analizi

Sınırlılık **:** Bir Petri ağı eğer her bir p yerindeki jetonların sayısı sabit ve belirli bir k sayısını geçmiyor ise sınırlıdır (Mo'dan erişilebilen herhangi bir imleme için) ve *k-bounded* şeklinde gösterilir. Bir Petri ağı 1-bounded ise güvenli Petri ağı adını alır.

Kapsanabilirlik: Bir PN(N, Mo) ağında bir M imlemesi eğer M'(p) >= M(p) koşulunu sağlayan bir M imlemesi R(Mo) içerisinde yer alıyorsa kapsanabilirlik özelliğini gösterir. Kapsanabilirlik L1-canlılık özelliği (potansiyel ateşlenebilirlik) ile yakından ilişkilidir.

Erişilebilirlik : Bir Mn'nin başlangıç imlemesinin Mo'dan erişilebilir olması için Mo'ı Mn'ye dönüştüren bir dizi ateşlemenin var olması gerekmektedir. Erişilebilirlik, karar verilebilir olmakla birlikte üstel özellik gösterir.

Tersinirlik: Bir Petri ağının tersinir olması için, M'nin başlangıç imlemesi Mo'dan erişilebilen her bir Mn imlemesi için, Mo da Mn 'den erişilebilir olmalıdır. Tersinir bir Petri ağı daima başlangıç imlemesine ya da durumuna geri döndürülebilir.

Süreklilik: Süreklilik, bir Petri ağındaki bir etkin geçişin ateşlenmesi nedeniyle herhangi bir başka geçişin pasif hale getirilmemesi özelliğidir. Sürekli bir Petri ağında bir geçiş bir kere etkin olduktan sonra ateşlenene kadar etkin olmaya devam eder. Bazı durumlarda iki farklı geçiş aynı koşul kümesi tarafından etkin hale getirilebilir. Sürekli olmayan bir Petri ağına örnek olarak Şekil 2.6 verilebilir.

Korunma : Korunma özelliği, Petri ağında herhangi bir yol üzerinde erişilen tüm durumlar için belirli sayıda jetonun muhafaza edilmesidir. Aslında bu özellik sınırlayıcı olabilir, çünkü jeton sayıları Petri ağı grafiğindeki yer ve geçişlerin dağılımına ve bağlantısına göre değişken özellik gösterebilmektedir. Jeton sayısı her zaman korunmaz. Bu yüzden bir ağırlık vektörü tanımlanmıştır (Cassandras ve Lafortune,2008), şöyle ki; $p1, p2, \ldots, pn$ yerlerine ilişkin tüm $i = 1, \ldots, n$ için $\gamma i \geq 0$ olmak üzere $\boldsymbol{\gamma} = [\gamma 1, \gamma 2, \ldots, \gamma n]$ ağırlık vektörüne bağlı olarak korunma özelliği tanımlanır (Burada Petri ağlarındaki oklardaki ağırlıklarla karıştırılmamalıdır). (basitlik için γi tamsayı seçilmiştir). Bir x0 başlangıç durumu olan N Petri ağı $\boldsymbol{\gamma} = [\gamma 1, \gamma 2, \ldots, \gamma n]$ vektörüne göre eğer tüm $\mathbf{x} \in R(N)$ durumları için $\sum_{i=1}^{n} \gamma i x(pi)$ sağlıyorsa korunmalıdır denir. Genelde, jetonların eksilmesi ya da artması modellenmek istenen ayrık olay sisteminin fiziksel "korunma" özelliklerini etkilemelidir.

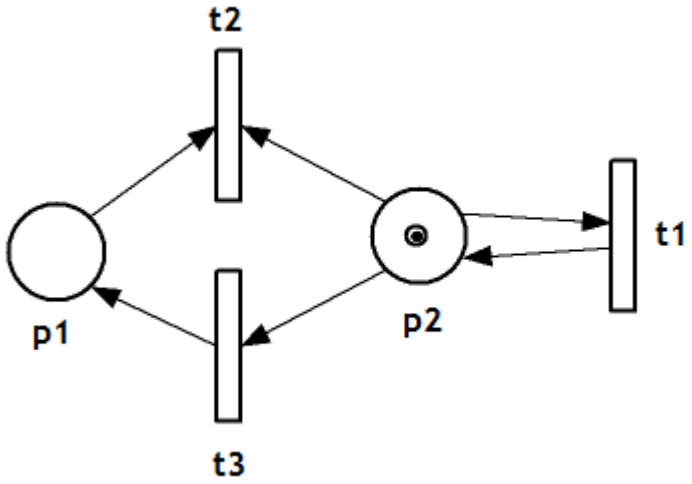
Canlılık: Bir N Petri ağının başlangıç durumu Mo'dan erişilebilen herhangi bir başka durumda herhangi bir geçişin eninde sonunda ateşlenebiliyor olmasına yol açacak en az bir örnek yol var ise N Petri ağı canlıdır.

Bir Petri ağının,

- L1-canlı (potansiyel olarak ateşlenebilir) olması için bir t geçişinin L(Mo) içerisinde herhangi bir ateşleme dizisinde en az bir defa ateşlenebiliyor olması gereklidir.
- L2-canlı olması için, verilen bir k pozitif tamsayısı için bir t geçişinin herhangi bir L(Mo) ateşleme sırasında en az k defa ateşlenebiliyor olması gereklidir.
- L3-canlı olması için, L(Mo) içerisindeki bir ateşleme dizisinde sonsuz olarak var olabiliyor olması gereklidir.
- L-4 canlı veya "canlı" olması için, R(Mo) içerisindeki her bir M imlemesi için L1-canlı olması gereklidir. Yani Mo başlangıç durumundan erişilebilmesi mümkün olan her durum için L1-canlı olması gereklidir.

Bir Petri ağında bir t geçişi hiçbir ateşleme işleminden sonra hiç ateşlenecek duruma gelmiyor ise ilgili Petri ağı "ölü" (dead) ya da L0-canlı olarak nitelendirilir.

Örnek olarak Şekil 2.6'da t2 geçişi hiçbir zaman ateşlenemeyeceği için ölüdür. t3 geçişi sadece bir defa ateşlenebileceği için L1-canlıdır. t1 geçişi sonsuz defa ateşlenebileceği için L3-canlıdır fakat L4-canlı değildir çünkü eğer t3 ateşlenirse ölü hale gelir.



Şekil 2.6: Petri ağı grafiğinde canlılık özelliklerinin testi için bir örnek

3. BULANIK PETRİ AĞLARI (BPA)

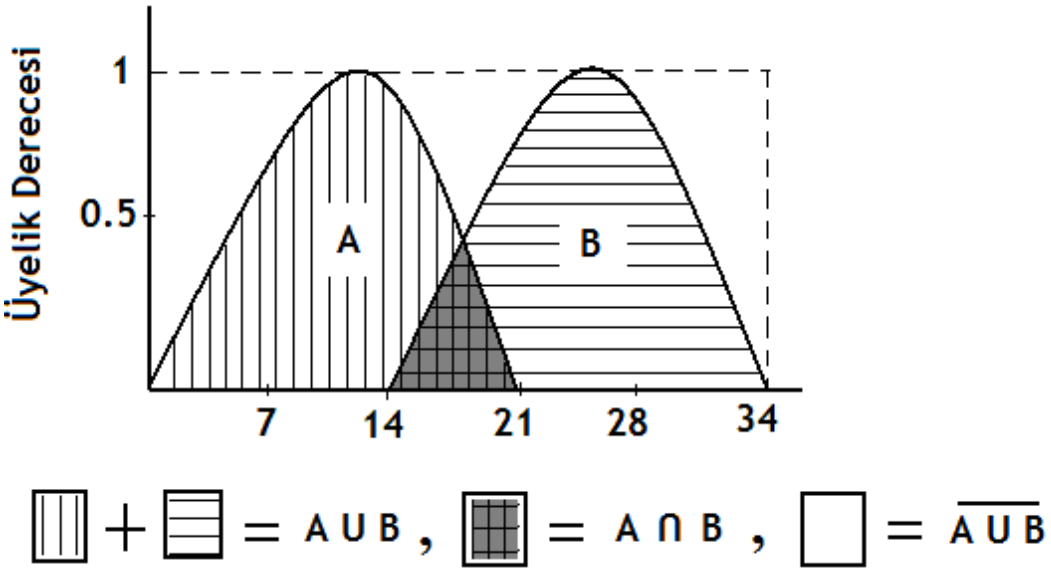
3.1 Bulanık Mantık (Fuzzy Logic) Kavramı

Klasik mantıkta doğruluk tablosunda "Doğru" ya da "Yanlış" şeklinde iki farklı olasılığın olması insanın karar verme tarzını ve insan mantığını tanımlarken bazen yetersiz kalır. Bulanık mantık insan mantığını tanımlayabilmek amacıyla 0 (Yanlış) ve 1 (Doğru) arasındaki tüm aralığı kullanır. Bulanık mantık bulanık kümelerdeki işlemlere dayanır (Jantzen, 2010). A ve B'nin aynı evrensel kümeye dahil olduğu bir durumda; (a) A ve B'nin kesişimi, $A \cap B \equiv a \min b$ şeklinde tanımlanır **min** operatörü a ve b'deki birbirine karşı gelen elemanların her birinin birbiriyle karşılaştırılmasıdır.

(b) A ve B'nin birleşimi, $A \cup B \equiv a \max b$ şeklinde gösterilir ve aynı şekilde a ve b'de birbirine karşı gelen elemanlara terim terim maksimum işlemi uygulanmasıyla olur.

(c) A'nın tümleyeni, $\overline{A} \equiv 1 - a$ şeklinde olup a'daki her üyelik değeri 1'den çıkarılır.

Yine, Y'nin bir X bulanık alt kümesi, $X \subseteq Y$ şeklinde yazılır ve X'in üyelik fonksiyonu Y'nin üyelik fonksiyonuna eşit ya da daha düşüktür (Şekil 3.1).



Şekil 3.1: 3 Temel operatör (Birleşim, Kesişim, Tümleyen)

(Yaralıoğlu, 2004)'de bulanık mantık kavramı matematiksel formüller ve görselleriyle birlikte aşağıdaki gibi irdelenmiştir. Karar vericiler hangi şartlarda ve boyutlarda karar verirlerse versinler, bir belirsizlik ortamı içinde bu işlevlerini yerine getirmek zorundadırlar. Verilen kararların doğruluğu ise, söz konusu belirsizliğin riske dönüştürülebildiği ölçüde sağlanacaktır. Ancak karar vericiler karar sürecinde klasik bilimsel yaklaşım ve bu yaklaşımın içerdiği yöntemleri kullanıyorlarsa, sonuçta verilen kararlar, iyi – kötü, güzel – çirkin, doğru – yanlış, evet – hayır, siyah – beyaz ya da 0 – 1 gibi tek taraflı olmayan yönlü kararlar olacaktır. Aslında dikkat edilecek olursa gerçek dünyada tam ve kesin bir ayrım söz konusu değildir. Doğadaki olayların işleyişi çoğu zaman belirsiz ya da kesin olmayan olgulara dayanır. Başka bir şekilde ifade edilecek olunursa, güzel ve çirkin arasında orta-vasat değerler de bulunur, iyi ve kötü arasında nötr ve benzeri tasvirler de yapılabilir, 0 ve 1 arasında $1/3, \sqrt{0.9}, \sqrt[3]{0.5}$ gibi ara değerler olabilir. Belirli sınırlar arasında değişik değerler alabilen ve karar verme basamaklarını belirleyen, karar verme süreçlerini etkileyen yöntemlerin arayışı sonucunda sistematik bir düşünce ortaya koyulmuştur (Banks ve Hayward, 2002). Loutfi Zadeh' in Bulanık Mantık Teorisi bu konudaki bilimsel bilgilerin analiziyle ortaya çıkmış etkili bir yöntem önermiştir. Bununla birlikte, klasik mantık ile bulanık mantık arasındaki temel farklılıklar Çizelge 3.1' de gösterilmiştir.

Çizelge 3.1: Klasik Mantık ve Bulanık Mantık Arasındaki Temel Farklılıklar

Klasik Mantık	Bulanık Mantık
A veya A Değil	A ve A Değil
Kesin	Kısmi
Hepsi veya Hiçbiri	Belirli Derecelerde
0 veya 1	0 ve 1 Arasında Süreklilik
İkili Birimler (Boolean Algebra)	Bulanık Birimler (Fuzzy Terms)

Zadeh' e göre bulanık mantık çoklu değerler alabilen sistemlerden oluşur (Liu, 1998). Klasik mantıkta, Boole cebrinde bir değişken ya 0 ya da 1 değeri alabilirken bulanık mantık önermeleri 3 ya da daha fazla sayıda önerme içermektedir.

Bulanık mantığın başlıca özellikleri aşağıdaki gibi sıralanabilir:

- "doğru" , "çok doğru" , "az çok doğru" v.b. gibi sözel olarak ifade edilen (linguistik-dilsel-değişkenli) doğruluk derecelerine sahip olması,

- Geçerliliği kesin değil fakat yaklaşık olan çıkarım kurallarına sahip olması,
- Her kavramın bir derecesinin var olması,
- Her mantıksal sistemin bulanıklaştırılabilmesi,
- Bulanık mantıkta bilginin, bulanık kısıtlara ait değişkenlerin esnekliği veya denkliğiyle yorumlanması (Venkateswaran, P.R., Bhat, J., 2006).

3.1.1 Bulanık kümeler ve üyelik fonksiyonları

Bulanık mantık, sayıların komşuluğu kavramına dayanır (Yaralıoğlu, 2004). Karar sürecinde bir durum bir sayıyla ifade ediliyorsa, söz konusu durumun kabul edilirliği o sayının gerçekleşmesinde sağlanacaktır. Ancak söz konusu sayıya yakın sayılar karar sürecinin bir parçası olarak algılanmayacaktır. Oysa belirli bir güven katsayısında bu sayıların farklı popülasyonların üyeleri olduğunu öne sürmek de istatistiksel açıdan yanlış olacaktır. Örnek vermek gerekirse, bir otoyol üzerinde taşıtlarda hız sınırının 120 km\saat ile aşıldığı bir durumda aslında trafik cezası gerektiren değerin 100 km\saat civarında veya üzerinde olması da bir ön şart olarak kabul edilebilir. Bu durumda aynı amaca hizmet eden sayıların komşuluğundan söz etmek mümkündür.

Eğer A $R \in (-\infty, +\infty)$' da, söz konusu kümenin bir elemanı ise $\mu_A(x)$ üyelik fonksiyonu $R \rightarrow [0,1]$ aralığında oluşur. Diğer bir deyişle A kümesi $A = [a_1, a_3]$ aralığında ise genel olarak $\mu_A(x)$ üyelik fonksiyonu **(3.1)** bağıntısıyla gösterilebilir.

$$\mu_A(x) = \begin{cases} 0, & x < a_1 \\ 1, & a_1 \le x \le a_3 \\ 0 & x > a_3 \end{cases}$$

(3.1)

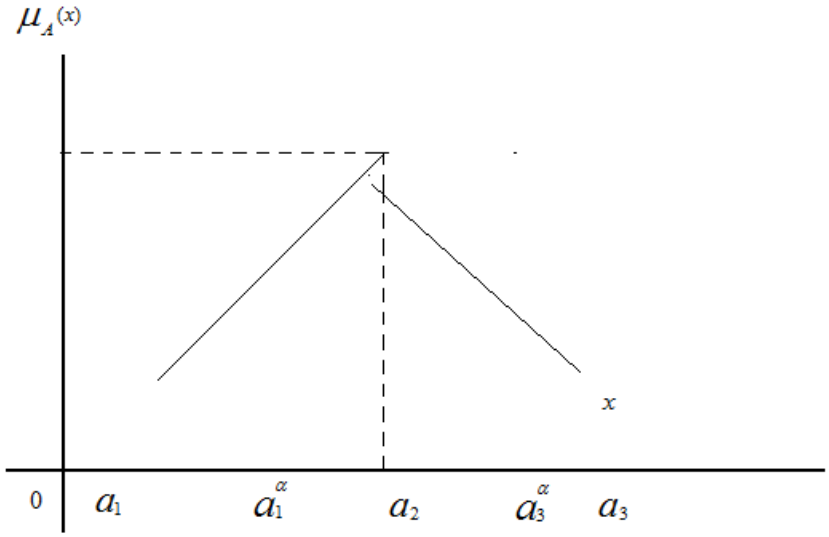
Üyelik fonksiyonları genellikle, üçgensel üyelik fonksiyonları ve yamuk üyelik fonksiyonları olmak üzere 2 farklı şekilde ele alınmaktadır.

$\mu_A(x)$ üçgensel üyelik fonksiyonu, **(3.2)**'deki bağıntılarda görülmektedir.

$$\mu_A(x) = \begin{cases} 0, & x < a_1 \\ \dfrac{x - a_1}{a_2 - a_1}, & a_1 \le x \le a_2 \\ \dfrac{a_3 - x}{a_3 - a_2}, & a_2 \le x \le a_3 \\ 0, & x > a_3 \end{cases}$$

(3.2)

(3.2) formülüne göre küme, $A = (a_1, a_2, a_3)$ olmalıdır. Burada a_2 normal değerli üyelik olarak tanımlanabilir. Bulanık Mantık bu noktada bir α katsayısına bağlı olarak a_2' ye yakın değerlerin, bu değere yüklenen anlam ile temsil edileceğini varsaymaktadır. Diğer bir deyişle a_2' deki belirsizlik, varsayılacak ya da dağılıma göre bulunabilecek bir α katsayısı ile tolere edilebilir. Söz konusu komşuluk Şekil 3.2' de grafikte gözlenebilir (Liu, 1998, Yaralıoğlu, 2004).



Şekil 3.2: Sayıların komşuluğuna ilişkin grafik

α değeri bulanık mantıkta kesim katsayısı olarak adlandırılır. a_1^{∞} ve a_3^{∞} sayıları ise a_2 normal değerinin komşuluğunu oluşturan aralığın alt ve üst sınır değerleridir. Diğer bir deyişle a_1^{∞} ve a_3^{∞} aralığındaki tüm sayılar a_2 normal değeri ile aynı anlama sahiptir. a_1^{∞} ve a_3^{∞} değerleri **(3.3)** ve **(3.4)** formülleri yardımıyla bulunabilir.

$$\frac{a_1^{\alpha} - a_1}{a_2 - a_1} = \alpha$$

(3.3)

$$\frac{a_3 - a_3^{\alpha}}{a_3 - a_2} = \alpha$$

(3.4)

(3.3) ve **(3.4)** formüllerinden $\forall \alpha \in [0,1]$ için $A_{\infty} = [a_1^{\infty}, a_3^{\infty}]$ aralığı oluşturulabilir. a_1^{∞} ve a_3^{∞}değerleri **(3.5)** ve **(3.6)** formüllerinde gösterilmiştir. a_1^{∞} **(3.3)**'ten, a_3^{∞} **(3.4)**'ten çekilirse,

$$a_1^{\alpha} = \alpha(a_2 - a_1) + a_1$$

(3.5)

$$a_3^{\alpha} = a_3 - (a_3 - a_2)\alpha$$

(3.6)

Örneğin üçgensel bulanık mantık sayılarına ilişkin küme $A = (a_1, a_2, a_3)$,

A= (-7, -2, 2) ise bu durumda **(3.2)**'deki formülünden üyelik fonksiyonu aşağıdaki gibi bulunur.

$$\mu_A(x) = \begin{cases} 0, & x < -7 \\ \frac{x+7}{6}, & -7 \le x \le -2 \\ \frac{2-x}{4}, & -2 \le x \le 2 \\ 0, & x > 2 \end{cases}$$

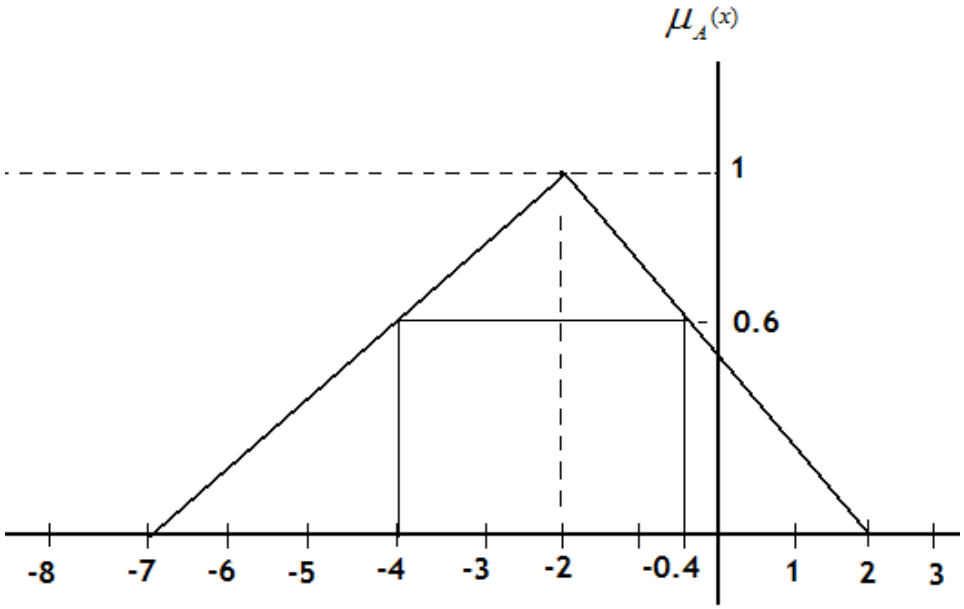
Eğer bir karar verici α kesim katsayısını 0,6 olarak belirlediyse -2 normal değerinin komşuları **(3.5)** ve **(3.6)** formüllerinden

$a_1^{\infty} = \alpha(\alpha_2 - \alpha_1) + a_1 = 0.6\ (5) + (-7) = -4,$

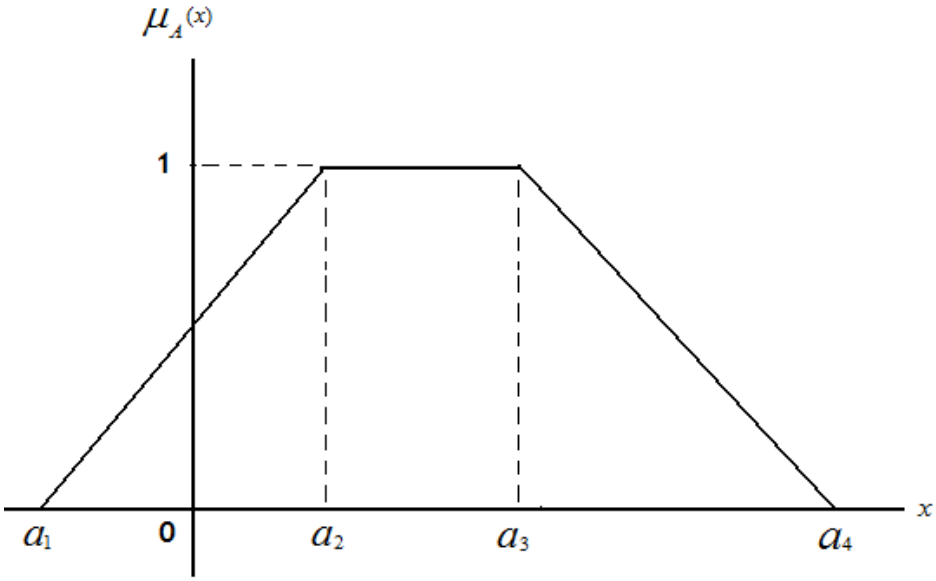
$a_3^{\infty} = a_3 - (a_3 - a_2)a = 2 - (2+2)\ 0.6 = -0.4$ olur.

Bu sonuca göre, -2 normal değeriyle aynı anlama gelen sayı değerlerinin kümesi [-4, -0.4] aralığından oluşur. Bu örneğe ilişkin bir grafik çizilirse Şekil 3.3 gibi bir sonuç elde edilir. Eğer bulanık mantık sayılarına ilişkin kümede normal kabul edilen iki değer varsa diğer bir deyişle küme, $A = (a_1, a_2, a_3, a_4)$ şeklinde 4 belirleyici

değerden oluşuyorsa bu durumda üyelik fonksiyonu yamuk üyelik fonksiyonu tipinde olacaktır (Şekil 3.4).



Şekil 3.3: A= (-7, -2, 2) kümesinin komşuluğu



$$\mu_A(x) = \begin{cases} 0, & x < a_1 \\ \dfrac{x - a_1}{a_2 - a_1}, & a_1 \le x \le a_2 \\ 1, & a_2 \le x \le a_3 \\ \dfrac{a_4 - x}{a_4 - a_3}, & a_3 \le x \le a_4 \\ 0, & x > a_4 \end{cases}$$

Şekil 3.4: Yamuk sayı komşuluğu ve yamuk üyelik fonksiyonu $\mu_A(x)$

Çizelge 3.2: Bulanık Mantığın Kullanım Alanları (Yaralıoğlu, 2004)

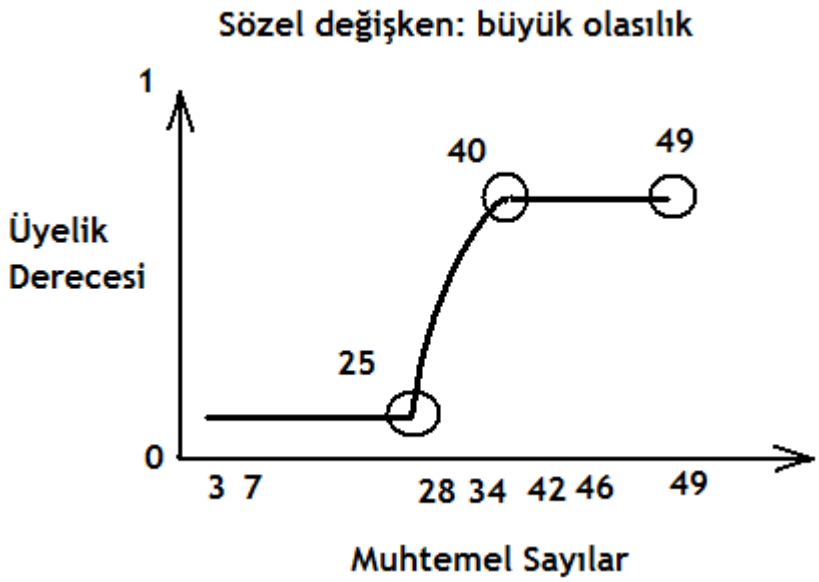
ÜRÜN	FİRMA	BULANIK MANTIĞIN İŞLEVİ
Asansör Denetimi	Fujitec –Toshiba Mitsubishi Hitachi	Yolcu trafiğini değerlendirir. Böylece bekleme zamanı azalır.
SLR Fotoğraf Makinesi	Sanyo –Fisher Canon Minolta	Ekranda birkaç obje olması durumunda en iyi odak ve aydınlatmayı belirler
Video Kayıt Cihazı	Panasonic	Cihazın elle tutulması nedeniyle çekim sırasında oluşan sarsıntıları ortadan kaldırır.
Çamaşır Makinesi	Matsushita*	Çamaşırın kirliliğini, ağırlığını, kumaş cinsini sezer, ona göre yıkama programını seçer.
Elektrik Süpürgesi	Matsushita	Yerin durumun ve kirliliğini sezer ve motor gücünü uygun ayarlar.
Su Isıtıcısı	Matsushita	Isıtmayı kullanılan suyun miktar ve sıcaklığına göre ayarlar.
Klima	Mitsubishi	Ortam koşullarını değerlendirerek en iyi çalışma durumunu algılar, odaya birisi girerse

		soğutmayı arttırır.
ABS Fren Sistemi	Nissan	Tekerleklerin kilitlenmeden frenlenmesini sağlar.
Çelik Endüstrisi	Nippon Steel	Geleneksel denetleyicilerin yerini alır.
Sendai Metro Sistemi	Hitachi	Hızlanma ve yavaşlamayı ayarlayarak rahat bir yolculuk sağlanmasının yanı sıra durma konumunu iyi ayarlar, güçten tasarruf sağlar.
Çimento Sanayi	Mitsubishi Chem	Değirmende ısı ve oksijen oranı denetimi yapar.
Televizyon	Sony	Ekran kontrastını, parlaklığını ve rengini ayarlar.
El Bilgisayarı	Sony	El yazısı ile veri ve komut girişine olanak tanır.

*Matsushita firmasının yeni ismi 2008'de "Panasonic Corporation" olmuştur.

3.1.2 Bulanık mantığın bilgisayar uygulamaları

Sözel değişkenler (linguistic variable) bulanık mantığın temelini oluşturmakla birlikte, bilgisayarlar ile günlük hayattaki problemleri tanımlama ve çözümleme gibi işlemlerde büyük rol oynamaktadır. Aslında sözel değişkenlerin bulanık mantıkta yer alması klasik Boole cebrindeki 0 ve 1 değerlerinin arasında dağıtık bir şekilde değer tutulabilmesini sağlar. Örneğin; bir odanın sıcaklığı veya bir kaloriferin sıcaklığı hakkında "Sıcak" yorumunu yapmak mümkündür. Fakat buradaki "Sıcak" şeklinde tarif edilen sözel değişken; odanın veya kaloriferin içinin kastediliyor olmasına gore farklı anlamlar kazanacaktır. Bir sözel değişkenin "0" değerine atanması o sözel terimin doğru olmadığı, "1" değerine atanması ise ilgili terimin doğru olduğu anlamına gelir. Günlük hayattaki konuşmalarımızda "sözel değişkenler" (sıcak, biraz sıcak, çok sıcak…gibi) sıkça kullanılmakta olup çevresel koşullara göre veya insan duygularına, gözlemlerine göre çok önemli bilgiler taşıyor olabilir. Örneğin Şekil 3.5'deki "büyük olasılık" adlı sözel değişken üyelik derecesi olarak 0 ile 1 arasında dağılan değerler alır ve gerçek dünyada 25-40 aralığında değişir (0 en küçük sayı, 1 en büyük sayı anlamına gelir). Aynı uzay içerisindeki (örneğin bir sayısal loto kolonu) her bir sayı değerine bir sözel değişken karşı gelir. Bu sözel değişkenlerin bilgisayar ortamında işlenebilmesi için uygun şekilde tanımlanması gereklidir. Şekil 3.5'deki grafik seçilen sayısal loto rakamları ile sözel bir terim olan "muhtemel sayılar" arasındaki ilişkiyi göstermektedir. Küçük ve Büyük alanları dışında, ortanca sayılar her iki alanla da ilişkili olup herhangi bir alanla daha çok alakalı da olabilmektedir (25-40 arası). Grafikteki yatay eğri ölçülen bir sayısal loto kolonunda oynanan muhtemel sayıları göstermektedir. Dikey eksen ise ölçülen verinin hangi sözel değişkenle nasıl bir yakınlığı olduğunu gösterir.



Şekil 3.5: Sözel değişkenlerin grafik gösterimi – Sayısal loto rakamları

Çoğu bulanık mantık yazılımında genel olarak bir sözel değişkenin tanımlanması aşağıdaki kod bölümünde olduğu gibidir (Banks ve Hayward, 2002). Aynı algoritma mantığı değişik yazılım dillerinde tanımlanabilir. Örneğin bir sayısal loto kolonunda 1 - 49 arasında bulunan 49 adet tamsayı bulunmaktadır. Bunlardan bazıları büyük, bazıları küçük işaretsiz sayılardır. Bu durumda, bir "muhtemel_sayilar" değişkeni sözel değişken olan "buyuk_olasilik" ve "kucuk_olasilik" ile Şekil 3.5'deki grafikteki 4 adet kırılma noktası dikkate alınarak ilişkilendirilmiştir.

```
LINGUISTIC muhtemel_sayilar TYPE unsigned int min 0 max 49
{   MEMBER buyuk_olasilik { 25, 40, 49}

    MEMBER kucuk_olasilik { 1, 25, 40}  }
```

Şekil 3.5'teki grafikte yatay eksende yer alan sayıların üyelik dereceleri bulunabileceği gibi bağlı bulunduğu dilsel değişken kategorisi de (buyuk_olasilik, kucuk_olasilik) belirlenebilir. Örneğin sayısal loto kuponundaki "7" sayısı kucuk_olasilik alanına girer, "34" sayısı hem kucuk_olasilik hem de buyuk_olasilik alanına girer, "46" sayısı da sadece buyuk_olasilik alanına girer. Böylece gerçek sayısal değer ile dilsel değişkenler birbiriyle ilişkilendirilmiş olur. Şekil 3.5 ve benzeri grafiklerdeki sınır değerleri ve grafiğin şekli stokastik olarak ya da analiz yapan kişinin tanımladığı herhangi bir kurala göre ayarlanabilir.

İnsanın doğasından alınan bir başka örnek de yemek tarifleridir (Banks ve Hayward, 2002). Bir yemek tarifi aslında basamaklı ve sırasıyla uygulanması gereken bir yapıda olduğundan dolayı bir algoritmaya benzetilebilir:

1. Gerekli malzemeleri bir tencere içerisine boşaltınız.
2. 1 litre (4 bardak) soğuk su ekleyiniz.
3. Ocağı hafif kısarak kaynayıncaya kadar karıştırınız.

Yukarıdaki yemek tarifi örneğinde hem sayısal değerler hem de bulanık değerler bulunmaktadır. Bir sözel terim olan "tencere " nin katılacak malzemeye göre uygun seçildiği varsayılmıştır. 1 litre soğuk su aslında 4 bardak suya karşı gelmeyebilir fakat %5 hata oranıyla da olsa yapılacak aktivite için ihtiyacı karşılayacak derecede yeterlidir. "soğuk su" da bir sözel değişkendir ve suyun sıcaklığını donma noktası ile (herkesin kabul ettiği bir soğukluk) daha yüksek bir sıcaklık (soğuk kabul edilebilecek bir seviyeye kadar) arasını tanımlar. Herhangi bir bilgisayar dilinin gücü de bir problemi ilgili problemle alakalı bir şekilde tanımlamasından gelir. Sözel değişkenler insan arayüzü de dahil olmak üzere birçok uygulama ile ilişkilidir. Bulanık mantık ile geliştirilmiş olan başarılı çalışmalar insanlar tarafından tasarlanmış olan ve sayısal ve reel terimlerle kolaylıkla tanımlanamayacak uygulamaları kapsar. Tost makineleri, çamaşır makineleri, çevresel kontrol aygıtları, metro ve tren mekanizmaları, asansörler, kamera odaklaması gibi örnekler bunlardan sadece bir kaçıdır.

Sözel değişkenler mevcut uygulamaları veya bu uygulamaların kurulumunu basitleştirmekten ziyade bir problemi tanımlayabilmek için etkin bir araç olarak kullanılırlar. Uygulamaların hem reel sayısal terimler hem de sözel değişkenler domeninde hesaplanabilmesi mümkündür. Herhangi bir ortamdaki süreç kontrolü gibi lineer olmayan problemler bulanık mantık kullanıldığında etkileyici bir şekilde sistemlerin hızlı çalışmasına da imkan vermektedir. Bulanık mantık bu şekilde lineer olmayan kontrol probleminin çözümüne temel oluşturmaktansa mümkün olabilecek bazı çözümlerin tanımlanmasına yardımcı olmasıyla da dikkat çekmektedir. Bulanık mantık kavramını ortaya çıkaran Dr. Lotfi Zadeh'e göre (Li ve Rosano, 2000) günlük hayatta kullanılan sıradan dil bize aslında bir sistemin içeriğinde özel olarak bulunan birçok tanımlayıcı nitelik sunar. Klasik bir kullanıma örnek verilecek olursak, hava durumu olarak, "hava çok sıcak" dediğimiz zaman "hava 40 derecedir" tarzındaki bir

söyleyişten daha geçerli bir söylem olarak düşünülebilir. Çünkü 40 derecelik sıcaklık Türkiye'nin Marmara Bölgesi insanı için çok sıcak, Akdeniz Bölgesi insanı için orta seviye, Kuzey Afrika-Arabistan bölge insanın için ılıman olduğuna dair açıklayıcı bir nitelik taşır.

Havanın rutubetli olması ise anlamsal olarak içeriğinde iki farklı bilgi barındırır: hava sıcaktır ve bağıl nem yüksektir. Aynı gün hava hem sıcak hem nemli veya hem soğuk hem nemli olabilir. Genelde sözel değişkenler birbiriyle kesişirler yani aynı anda birden fazla sözel değişken geçerli olabilir.

Rutubetli hava yüksek nem ve yüksek sıcaklığın var olduğunu ifade eder. Herhangi bir gün; bulanık mantık domeninde atanacak değerler için çok sayıda sözel değişken içerebilir (Sıcak ve Rutubetli, Nemli, Sıcak, Soğuk, Soğuk ve Rutubetli).

Günlük hayatta da sıkça karşılaştığımız üzere, insan duyu organlarıyla algılanabilen bir nicelik olan C^o cinsinden sıcaklık için bir termometre kullanmadan kesin yargılarda bulunamayız. Onun yerine, reel sayılarla ifadeden çok bulanık mantık ifadelerinden yararlanırız. Bir bilgisayar programı üzerinde bulanık mantık nitelikleri işleneceği zaman yeni değişken tipi olarak sözel değişken kullanılmalıdır.

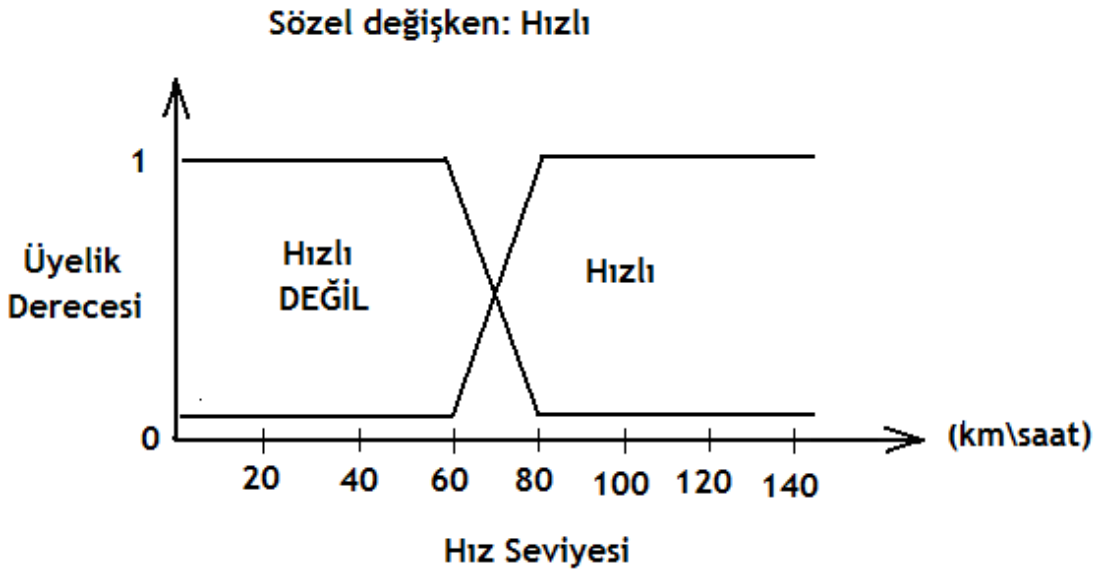
Örnek olarak bir otoyol hız kontrol sistemi düşünülecek olursa, hız sınırı aşıldığı zaman hız kontrol ve uyarı sistemi bir alarm işareti vermelidir. Aşağıdaki bulanık mantık ifadesi hız sınırının kaç olduğu ile ilgilenmez iken, uyarı verip vermeme derecesini bulanık mantık ifadesiyle belirliyor (Banks ve Hayward, 2002).

```
IF hiz IS fazla THEN trafik_hiz_uyari_cihazi IS on;
```

Benzer şekilde bir trafikte taşıtların hızını otoyolda kontrol eden bir sistem belirli bir hızdan daha yavaş giden araçları da takibe alabilir. Belirli bir hız değerinden daha yavaş gidilmemesi gereken durumlarda da uyarı vermek üzere programlanabilir. Bu örnek daha da genişletilerek, önceden belirlenmiş hız aralıklarında (çok yavaş, yavaş, orta, hızlı, çok hızlı) taşıt hızı uyarı sistemi farklı renklerde ve ses frekanslarında kullanıcıyı uyarabilir.

Sözel değişken tiplerinin character, string, real ve float gibi veri tiplerinin yanında önemli bir yeri vardır. Aslında birçok yüksek seviye dillerdeki sınıflandırılmamış veri tiplerinin (enumerated data types) bir versiyonu olarak da düşünülebilir. Bulanık mantık operatörleri sözel değişkenleri işleyebilmek için etkili bir yöntem de

sunmaktadır. Daha önce de belirtildiği üzere bulanık mantığın işlenmesinde ele alınması gereken temel kavramlar sözel değişkenlerdir. Sözel değişkenler aracılığıyla kesin olmayan ama bir sistemi başarıyla tasvir eden yapısal senaryolar üretilebilir. Öncelikle, bir bilgisayarda sözel değişkenlerin işlenebilmesi için belirli terimlerle bilgisayarın anlayacağı dilde tanımlanması gereklidir. Şekil 3.6'daki grafik ölçülen hız seviyesi ile sözel değişken "Hızlı" arasındaki ilişkilendirmeyi gösteriyor (Banks ve Hayward, 2002). Burada "Hızlı" kelimesi insani duyguların ölçüsüne göre kişisel duyu organlarıyla belirlenmesine rağmen burada daha tutarlı bir form oluşturulmuştur. Belirli noktalarda "Hızlı" diye bir yorum yapabiliyorken yine belirli noktalarda kesinlikle "Hızlı değil" de denilebilir. "Hızlı" ile "Hızlı DEĞİL" bölgeleri arasındaki alan iki değişkenden de belli derecelerde değer alır. Yatay eksen hız seviyesinin sayısal ve reel değerini göstermekle birlikte dikey eksen sözel değişkenin ilgili hız seviyesi değeriyle hangi seviyede ilişkili olduğunu gösterir.



Şekil 3.6: Sözel değişkenlerin ilişkileri ve temel kavramlar

Hız seviyesini grafiksel olmayan bir yöntem ile aşağıdaki gibi ifade etmek mümkündür. Bu betimlemede sayısal ve reel sıcaklık seviyeleri *unsigned int* ve sözel değişken *HIZLI* ise bir yamuk şeklini oluşturan belirli parametrelerle tanımlanır.

```
LINGUISTIC Hiz_seviyesi TYPE unsigned int MIN 0 MAX 140

{MEMBER HIZLI { 60, 80, 100, 120, 140 }       }
```

Bir bilgisayar programına HIZLI adlı bir sözel değişken ekleyebilmek için grafiksel gösterimi anlamlı bir koda dönüştürmek gerekir. Aşağıdaki C kod bölümü bu işlemin nasıl yapılacağını özetlemektedir. Burada, Hiz_Seviyesi_HIZLI fonksiyonu 0 ile 255 arasında ölçeklendirilmiş bir üyelik fonksiyonunu geri çağırıyor, böylece bir trafik hız kontrolörü cihazından algılanan bir hızın hangi seviyede "Hızlı" olduğu ortaya çıkarılabilecektir. Bu tarz basit bir hesaplama bulanık mantık operatörleri için en önemli araçlardan biri olarak sayılabilir. ("Crisp value" terimi bulanık mantıkta üyelik derecesi belirlenecek olan değer olarak anılmaktadır.)

```
unsigned int Hiz_Seviyesi; /* Hiz seviyesinin algilanan degeri
(crisp value) */

unsigned char Hiz_Seviyesi_HIZLI (unsigned int Crisp_Value)

{  if (Crisp_Value < 60) return(0);

       else{

             if (Crisp_Value > 80) return(255);

            else

            {  return(((Crisp_Value - 60) * 3) + 7); }

           }

}
```

Aynı kodlar mikrodenetleyiciler için de yazılabilir (Banks ve Hayward, 2002). Bulanık değişkenlerle yapılan işlemler bulanık operatörlerin Boole Cebri'nde karşılığı olan operatörlerle benzer özellik gösterir (f_ve, f_veya ve f_degil). Bu operatörler de gömülü C derleyicilerinde aşağıdaki gibi 3 makro ile tanımlanabilir:

```
#define f_bir 0xff

#define f_sifir 0x00

#define f_veya(x,y) ((x) > (y) ? (x) : (y))

#define f_ve(x,y) ((x) < (y) ? (x) : (y))

#define f_degil(x) (f_bir+f_sifir-x)
```

“Hızlı” adlı sözel değişken doğrudan bir anlama sahiptir, yani hız seviyesi artar ise “Hız” teriminin de bulanık mantıktaki seviyesi yükselir. Fakat örneğin hem hızlı giden hem de ivmeli giden bir araba için sözel değişken artık bu iki parametrenin karışımı, kombinasyonu gibi düşünülebilir. Bir taşıt hız kontrol sisteminin parametreleriyle ifade edilmesi şöyle mümkün olabilir:

```
IF Hız_Seviyesi IS HIZLI AND ivme IS ivmeli THEN
trafik_hiz_uyari_cihazi IS red alarm;
```

Farklı değişkenlerin aynı sözel üye adlarına sahip olması mümkündür. Çoğu programlama dilindeki *enumerated type* veya *structure* elemanlarında olduğu gibi tek ve eşsiz olma gibi bir zorunluluk da yoktur. Bu aşamada vurgulanması gereken önemli bir nokta da çoğu sözel değişkenlere ilişkin yorumların yukarıdaki eşitliğin genel bir formu olduğudur.

Örnek olarak “tasit_profili” adlı bir değişkenin sözel tanımlarını düşünmek mümkündür. “tasit_profili” değişkeni birkaç şekilde sözel terimlerle ifade edilebilir: {hizli ve ivmeli, ivmeli, hizli, yavaş, yavaş ve ivmeli}. Bu küme elemanlarına baş hece ve harfleriyle hitap edersek sistem sırasıyla, HIZIV, IVMELI, HIZLI, YAVAS, YAVIV şeklinde tasvir edilebilir. Dikkat edilmesi gereken nokta ise “tasit_profili” değişkeninin bir sayısal ve reel değere sahip olmadığıdır. Örnek olarak, “tasit_profili ivmeli” veya “tasit_profili yavas” derken herhangi bir rakam atamasi yapilmayacaktir. “tasit_profili” o açıdan *void* değişkeni olarak düşünülebilir.

Tüm “tasit_profili” elemanları bulanık mantık eşitliklerine dayanmaktadır. Aşağıdaki kod bölümünde “tasit_profili” tanımlanmaktadır (Banks ve Hayward, 2002).

```
LINGUISTIC tasit_profili TYPE void
{
MEMBER HIZIV { FUZZY (Hiz_Seviyesi IS HIZLI AND ivme IS IVMELI) }
MEMBER IVMELI { FUZZY ivme IS IVMELI }
MEMBER HIZLI{ FUZZY Hiz_Seviyesi IS HIZLI }
MEMBER YAVAS{ FUZZY Hiz_Seviyesi IS YAVAS }
MEMBER YAVIV { FUZZY (Hiz_Seviyesi IS YAVAS AND ivme IS IVMELI) }
}
```

"tasit_profili" içerisindeki HIZIV'ın üyelik derecesini (Degree of Membership-DOM) hesaplayabilmek için Hiz_Seviyesi içindeki "HIZLI"nın ve IVME içindeki "IVMELI"nin DOM değerini hesaplamak gereklidir. Sonrasında ise bulanık AND operatörüyle birleştirmek gerekir. Aşağıdaki kod parçası ise "tasit_profili is HIZIV" deyiminin gerçeklenmiş halini göstermektedir (Banks ve Hayward, 2002). "tasit_profili"nin her bir sözel değişken terimleri için benzer eşitliğin oluşturulması gerekmektedir.

```
unsigned int tasit_profili_HIZIV( unsigned int Crisp_Value)
{return (f_ve (Hiz_Seviyesi_HIZLI (Crisp_Value),
IVME_IVMELI(Crisp_Value)));
}
```

Genellikle uygulamada iki sözel değişkenin birleştirilerek yeni bir sözel değişken tanımlandığı veya iki sözel değişkenin basit bir kombinasyonunu oluşturan bulanık bir kural bulmak gibi durumlarla karşılaşılır. İki sözel değişkeni birleştirerek hesaplamalar yapmak daha önce açıklanmış denklemlerden göreceli olarak daha az karmaşık olmaktadır.

Bulanık mantık kümelerinin elemanları $\frac{Üyelik\_Dereces}{Gerçek\_Deger}$ şeklinde ifade edilirse burada çizginin üzerine yazılan *Üyelik_Derecesi*'nin, çizginin altında yer alan *Gerçek_Deger*'in üyelik derecesini işaret ettiği söylenebilir. Bu bulanık mantık küme elemanları bir kümenin elemanlarını oluşturuyor da olabilir. Örnek olarak; A= $\left\{ \frac{0.2}{5}, \frac{0.6}{4}, \frac{0.7}{3}, \frac{0.8}{2} \right\}$ ve B= $\left\{ \frac{0.3}{5}, \frac{0.8}{4}, \frac{0.9}{3}, \frac{0.1}{2} \right\}$ bulanık kümelerinde $\overline{A}$, $\overline{B}$, $A \cap B$, $A \cup B$ işlemleri yapılabilir. Bu işlemleri matematiksel olarak MATLAB bilgisayar programı ile aşağıdaki gibi hesaplamak mümkündür.

```
% Bulanık islem yapilacak olan iki matris girilmelidir.

      A=input ('ilk matrisi giriniz A: ');
      B=input ('ikinci matrisi giriniz B: ');

secenek =input('İslem yapilmasi icin ilgili secenegi giriniz (1,2
ya da 3): ');
   %secenek 1 Tumleyen
   %secenek 2 Kesisim
   %secenek 3 Birlesim
```

```
      if (secenek == 1)

           secenek1=input ('birinci matrisin tumleyeni için 1,
ikinci matrisin tumleyeni için 2 degerini giriniz: ');

           if (secenek1 ==1)
             [m,n]=size(A);
              Z = ones(m)-A;

           else
              Z = ones(m)-B;
           end

      end

      if (secenek == 2)
          Y = min(A,B)
      end

      if (secenek == 3)
         X = max (A,B)
      end
```

Program klasik küme işlemleri olan birleşim, kesişim, tümleyen operasyonları için bulanık mantık versiyonunu A ve B kümesi elemanları için hesaplayacaktır. Programın farklı seçenekler için çıktıları ise şöyle olacaktır:

1. ($\overline{A}$):

```
ilk matrisi giriniz A: [0.2 0.6 0.7 0.8]
ikinci matrisi giriniz B: [0.3 0.8 0.9 0.1]
İslem yapilmasi icin ilgili secenegi giriniz (1,2 ya da 3): 1
birinci matrisin tumleyeni için 1, ikinci matrisin tumleyeni için 2
degerini giriniz: 1
Z =
    0.8000    0.4000    0.3000    0.2000
```

2. ($\overline{B}$):

```
ilk matrisi giriniz A: [0.2 0.6 0.7 0.8]
ikinci matrisi giriniz B: [0.3 0.8 0.9 0.1]
İslem yapilmasi icin ilgili secenegi giriniz (1,2 ya da 3): 1
birinci matrisin tumleyeni için 1, ikinci matrisin tumleyeni için 2
degerini giriniz: 2
Z =
    0.7000    0.2000    0.1000    0.9000
```

3. ($A \cap B$):

```
ilk matrisi giriniz A:  [0.2 0.6 0.7 0.8]
ikinci matrisi giriniz B: [0.3 0.8 0.9 0.1]
İslem yapilmasi icin ilgili secenegi giriniz (1,2 ya da 3): 2
Y =
```

```
    0.2000    0.6000    0.7000    0.1000
```

4. ($A \cup B$):

```
ilk matrisi giriniz A: [0.2 0.6 0.7 0.8]
ikinci matrisi giriniz B: [0.3 0.8 0.9 0.1]
İslem yapilmasi icin ilgili secenegi giriniz (1,2 ya da 3): 3
X =
    0.3000    0.8000    0.9000    0.8000
```

Misal olarak, VHDL kodları olarak üyelik fonksiyonları oluşturulurken 8-bit değerli olması önerilebilir. Bunun da nedeni tasarlanacak olan bulanık sistemdeki gerçek değerlerin (crsip values) 0-255 arasındaki sayıları gösterebiliyor olmasıdır. Ondalık sayılarda 0 sayısı, 8-bitlik ikili kodlamada "0000_0000" şeklinde, ondalık 255 sayısı da "1111_1111" şeklinde gösterilebilir. Buna örnek teşkil edecek bir üyelik derecesi fonksiyonu parçası aşağıdaki gibi tanımlanabilir:

```
type membership is (aktif, pasif);  -- kullanıcı yeni bir veri tipi tanımlar
type membership_func is ok;
dilsel_degisken: membership;
1_inci_deger: std_logic_vector(7 downto 0); -- grafikteki bir noktaya değer atanır
Grafik_1_inci_bolge: std_logic_vector(7 downto 0); -- üyelik bölgesi 1. bölge
2_inci_deger: std_logic_vector(7 downto 0); -- grafikteki bir başka noktaya değer
                                            -- atanır
Grafik_2_inci_bolge: std_logic_vector(7 downto 0);  -- üyelik bölgesi 2. bölge
type uyelik_fonksiyonu is array(natural range <>) membership_func;
constant dilsel_degisken_adi :üyelik fonksiyonu := ((dilsel_degisken => aktif,
1_inci_deger => x"08", 2_inci_deger => x"10",
Grafik_2_inci_bolge => "44"), dilsel_degisken => pasif, 1_inci_deger => x"FF",
Grafik_1_inci_bolge => x"FF", 2_inci_deger => x"FF",
Grafik_2_inci_bolge => x"FF"));
```

Buradaki kodlarda herhangi bir üyelik derecesi fonksiyonuna yönelik gerçek değerlerin grafiğin hangi bölgesine düştüğünü belirleyen, dilsel değişkenler olarak da çıktı sağlayan bir bulanık yapı oluşturulmuştur (Vuong, P.T., Madni, A.M., and Vuong, J.B.,, 2006).

Bir IF-THEN formunda kural yaratma yapısının kodlarını da oluşturmak mümkündür (Vuong, P.T., Madni, A.M., and Vuong, J.B.,, 2006). Örneğin, "IF a is X AND b is

Y THEN c is Z" ifadesinde, "AND" bulanık operatörü ile iki parametre arasındaki minimum değer alınır. VHDL'de ise iki değişken arasındaki her bir kural değerlendirme aşamasında aşağıdaki "minimum" fonksiyonu kullanılır.

```
function minimum (a, b: std_logic_vector) return std_logic_vector is
variable min: std_logic_vector(7 downto 0) := (others => '0');
        begin
                if a < b then min := a;
                else min := b;
                end if;
        return min;
end minimum;
```

Ayrıca "OR" operatörü ile aynı çıkışa ait birden fazla kural birleştirilebilmektedir. IF-THEN yapısındaki dilsel kuralın aşağıdaki gibi tanımlandığı varsayılırsa,
IF (a1 is X1 AND b1 is Y1) OR (a2 is X2 AND b2 is Y2) THEN c is Z, işlemi parantez içlerindeki işlemlerin sonuçlarının maksimum değerleri alınarak hesaplanır (Vuong, P.T., Madni, A.M., and Vuong, J.B.,, 2006).. VHDL kodlarında, aynı çıkışa ait tüm kuralların sonucunu belirlemek için iki girişin maksimum değerinin alındığı bir fonksiyon gereklidir.

```
function maximum(a, b: std_logic_vector) return std_logic_vector is
variable max: std_logic_vector(7 downto 0) := (others => '0');
        begin
                if a > b then max := a;
                else max := b;
                end if;
        return max;
end maximum;
```

Minimum ve maksimum fonksyionları da birleştirilerek genel bir doğruluk değerini gösteren bir kural elde edilebilir: Z <= maximum(minimum(X1, Y1), minimum(X2, Y2));

3.2 Bulanık Petri Ağlarının Mimarisi

Petri ağları birçok sisteme uygulanabilir grafiksel ve matematiksel bir araç olmasına rağmen yetersiz kaldığı durumlar da mevcuttur. Buna neden olan sınırlayıcı etkenler ise şöyle sıralanabilir:

- Yer düğümlerinin nesneleri (jeton) negatif olmayan tamsayılardan oluşması,
- Geçiş düğümlerinin ateşleme özelliklerinin bir eşik değerinin bulunmayışı,
- Giriş ve çıkışlar için ağırlık fonksiyonu pozitif bir sayı ile sınırlanmış olması.

Bu yüzden, bulanık davranışı olan bir sistemin tanımlanmasında klasik mantık ile işleyen Petri Ağları yetersiz kalmaktadır. Tüm bunların sonucunda çeşitli Bulanık Petri Ağları (BPA) önerilmiştir (Xiao-zhong, 1998, Bhat, 2006).

Bulanık Petri Ağları aslında geleneksel lojik mantık ile çalışan Petri Ağlarının daha geniş bir kümesini oluşturur. Başka bir deyişle Petri Ağları; Bulanık Petri Ağlarının bir alt kümesidir. Bulanık Petri Ağları (BPA) bulanık mantık ile çalışan bir sistemdeki veriyi temsil etmek ve sistemin davranışını modellemek amacıyla kullanılmaktadır (Korpeoglu ve Yazici, 2006). Ayrıca sistemin özelliklerini kontrol etmek, statik analizler uygulamak için de önemli bir araçtır. Petri ağlarının eşzamanlı, senkron, asenkron, paralel, belirleyici olmayan (non-deterministic) ve dağıtık yapıdaki bilgi işleme sistemlerini modelleme konusundaki başarısı da önemli bir avantajdır .

Bulanık Petri ağları da bulanık ifadelerin işlenmesinde etkilidir (Şekil 3.7). Bulanık kurallarla işleyen bir sistemin ya da uygulamanın Bulanık Petri ağlarıyla modellenmesinin önemli avantajları vardır (Korpeoglu ve Yazici, 2006):

- BPA'larının grafiksel gösterimi geliştiricilerin bulanık kurallara dayanan sistemleri kurmalarını ve bu sistemlerde ayar yapmalarını kolaylaştırır.
- Bulanık mantık kurallarına dayanan dinamik sistemlerin davranışı da modellenebilir. Petri ağlarındaki imlemelerin (marking) değerlendirilmesi ile dinamik davranışlı sistemin simülasyonu için de kullanılabilmektedir. BPA'lardaki jetonların hareketiyle birlikte hangi sonuçlara ulaşılacağının analizi yapılabilir (Pascal, J.D., Cardoso, J., Andreu, D., Valette, R., 1996).
- BPA'lar tüm kuralların taranması ihtiyacını ortadan kaldırır. Kural tabanlı çalışan sistemlerin verimliliğini BPA'lardaki geçişler ve okları bulanık mantık tabanlı yapısıyla kullanarak geliştirir.

- BPA'lar analitik yetenekleri modellenmiş bir sistemin özelliklerinin irdelenmesine yardımcı olur, böylelikle ayrıntılar ortaya çıkarılabilir.

Bulanık mantık ağları Petri ağlarının modifiye edilmesiyle birlikte kullanılmaktadır (Pedrycz, W., 1999). Böylelikle kural-tabanlı karar sistemleri temsil edilebilir, çalıştırılabilir. Fuzzy simülasyon teorisi de "MIN" operatörünü Boole cebrindeki "AND" problemleriyle, "MAX" operatörü de "OR" problemleriyle ilgilenmesi için uyarlanmıştır.

Genelleştirilmiş bir BPA yapısı aşağıdaki gibi 8 farklı parametre ile tanımlanabilir (Korpeoglu ve Yazici, 2006):

BPA = (P, T, D, I, O, f, α, β) → Bulanık petri ağı kümesinin elemanları

P = {P1,P2, . . . ,Pn} → yerlerin sonlu kümesi

T = {t1, t2, . . . , tm} → geçişlerin sonlu kümesi

D = {d1,d2, . . . ,dn} → önermelerin sonlu kümesi

$P \cap T \cap D = \Psi$, |P| = |D|

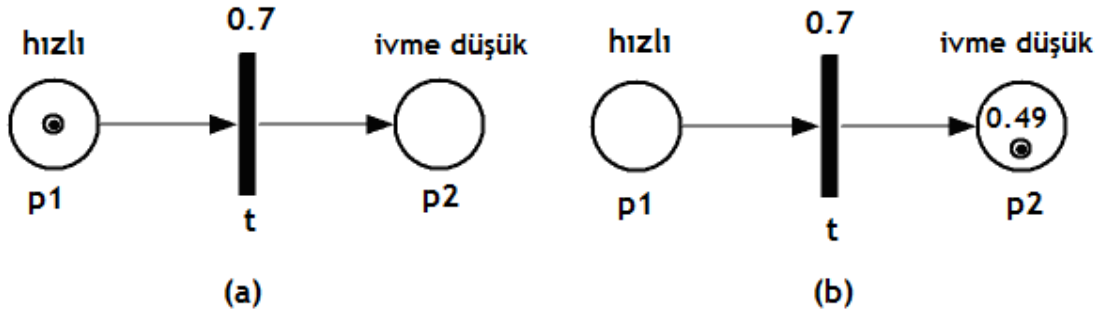
I: T→P$^{\infty}$T → giriş fonksiyonu, geçişlerden yerlere yollar çizer

O: T → P$^{\infty}$ → çıkış fonksiyonu, geçişlerden yerlere yollar çizer

f: T → [0,1] → atama fonksiyonu, geçişlerden 0 ve 1 arasındaki reel değerlere yollar çizer

α: P → [0,1] → atama fonksiyonu, yerlerden 0 ve 1 arasındaki reel değerlere yollar çizer

β: P → D → atama fonksiyonu, yerlerden önermelere birebir ve kapsayan yollar çizer



Şekil 3.7: İşaretli BPA örneği (a) Ateşleme öncesi (b) Ateşleme sonrası

P = {p1,p2} T = {t1} D = {hız fazla, ivme düşük}

I(t1) = {p1}, O(t1) = {p2}

F(t1) = 0.7

α(p1) = 0.9, α(p2) = 0

β(p1) = hız fazla

β(p2) = ivme düşük

3.2.1 Petri ağları ile bulanık petri ağları arasındaki ilişki

Petri ağları ile BPA'lar arasında yakın bir ilişki bulunmaktadır. Şöyle ki Petri ağları için önerilen tanımın daha genişletilmiş hali BPA'ları oluşturur. Alternatif bir BPA tanımı vermek gerekirse, (Xiao-Zhong, 1998) daha sade bir BPA tanımlaması 6 farklı parametre ile belirlenebilir:

BPA = {*P, T, W,* $\alpha(t)$, $\tau(t)$, $M_o(P)$ }

(3.7)

Bir BPA'yı tanımlayabilen **(3.7)** küme içeriğindeki parametreler ise şöyledir:

P: Bulanık yerlerin (places) sonlu kümesidir, yerlerdeki jetonlar negatif olmayan herhangi bir reel sayı olabilir.

T: Bulanık geçişlerin (transitions) sonlu kümesini oluşturur.

W: $P \times T$ veya $T \times P$ üzerindeki bulanık ilişkiyi tanımlar ve ağırlık fonksiyonu (weight function) olarak tanımlanır. Ağırlık fonksiyonu, yer düğümünden geçiş düğümüne bağlanan yol üzerindeki giriş miktarını veya geçiş düğümünden yer düğümüne bağlanan yol üzerindeki çıkış miktarını ifade eder.

$\alpha(t)$: Geçiş düğümlerinin bağlantı gücünü gösterir.

$\tau(t)$: Geçiş düğümlerinin ateşleme eşik değerini gösterir.

$\alpha(t)$ ve $\tau(t)$ *T* geçişleri üzerinde tanımlanan negatif olmayan reel fonksiyonlardır.

$M_o(P)$: Bağlangıç durumundaki imlemeyi (marking), başlangıç kaynağını gösteren *P* üzerindeki negatif olmayan reel fonksiyondur.

Klasik Petri ağları aslında Bulanık Petri ağlarının özel bir durumudur. Bir BPA'da $M_o(P)$ ve *W* tamsayı fonksiyonlar, $\alpha(t)$ = 1, $\tau(t)$ =0 ise bu klasik Petri ağını ifade eder. BPA'nın diğer elemanları (*P* ve *T*) ise klasik Petri ağlarıyla aynı anlamda kullanılır (Chezalviel, B.P., Cardoso, J., 1996).

3.2.2 BPA'larda ateşlenme

Bir BPA'da bulunan bir M üzerindeki t geçişi eğer $\forall p \in {}^\bullet t$ ise izinlidir, aktiftir. Bu durumda aynı zamanda $M(p) \geq W(p,t) \geq \tau(t)$ bağıntısı geçerli haldedir. ${}^\bullet t$ ifadesi bir t geçişine giriş olarak bağlanan tüm giriş yerleri kümesini, $t^\bullet$ ise ilgili t geçişinin çıkışındaki tüm çıkış yerleri kümesini gösterir.

Eğer M üzerinde bir geçiş aktif ise, t geçişinin ateşlenmesi yeni bir M' meydana getirir. Bu durum M[t > M' ile gösterilir ki böylece M', M durumundan erişilebilen bir başka durum demektir. $\forall p \in P$ için,

$p \in {}^{\bullet}t$ durumunda, M(p) - W(p,t) = M'(p)

$p \in t^{\bullet}$ durumunda, M(p) + W(p,t) = M'(p) olur ve diğer durumlarda ise bir değişiklik olmadığı anlamına gelir, ve bağıntı M(p) = M'(p) olur. Bu kural BPA'nın dinamik davranışını gösterir.

3.2.3 BPA'larda IF-THEN yapısı

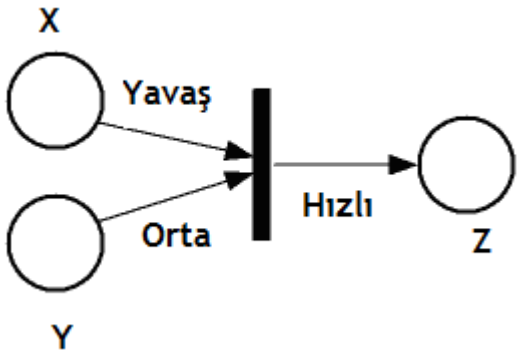
IF-THEN uygulaması bir mantık kuralı göstergesi olup, "IF" özel mantık kurallarına dayanan unsurları belirlemek amacıyla kullanılan bir önermedir. "THEN" ilgili sonuçları göstermek için kullanılan bir sonuçtur. "IF-THEN" yapısı şöyle ifade edilerek tanımlanmıştır:

IF β1 THEN β2 (CF) Burada β1 önceki nitelikleri gösteriyor iken, β2 izleyen sonuçlar ifade eder. CF (certainty factor) bir kesinlik faktörü olup CF'nin büyük olması "IF-THEN" yapısının kesinliğinin daha yüksek olduğunu gösterir.

Karmaşık bilgi kuralları için yukarıdaki nitelikler aşağıdaki gibi oluşturulabilir:

IF β1 AND β2 THEN β3 (CF) (Barro, S., Bugarin, A.J., 1994).

Burada örnek olarak β1 ve β2'nin birleştirilmesi için bir "Min" operatörü de bulunarak $\beta 3 = \frac{\beta 1 + \beta 2}{2}$ şeklinde ortalaması alınabilir. Değişik varyasyonlar mümkündür. Ayrıca birden fazla kuralın tek bir kural altında birleştirilmesi için de "Max" operatörü önerilebilir. BPA'lar için temel bir örnek de verilebilir (Şekil 3.8) (Kluska, 2009, Knybel, J., Pavliska, V., 2005,).



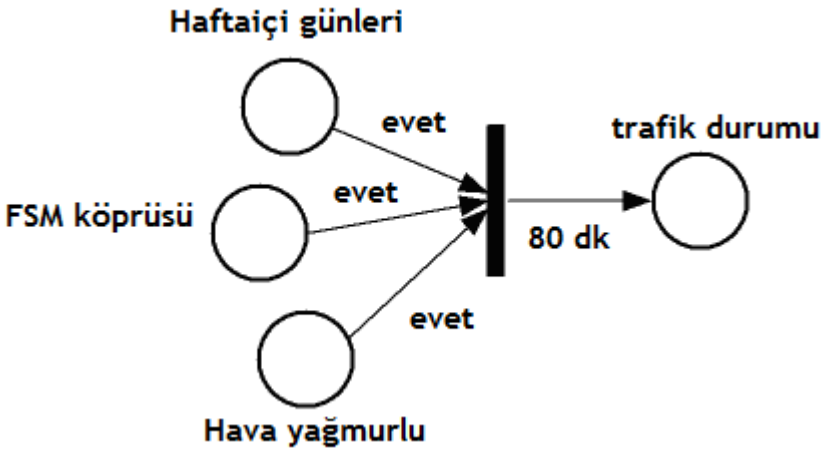
Şekil 3.8: IF X is YAVAS AND Y is ORTA THEN Z is HIZLI IF

İstanbul'daki boğaz geçiş hatları yani, 1. köprü olan Boğaziçi köprüsü ile, 2. köprü olan Fatih Sultan Mehmet (FSM) köprüsündeki trafiğin zamana göre ve hava durumuna göre ulaşılmak istenen hedefe erişim süresiyle ilgili faktörlere bağlı tümleşik bir model tanımlanabilir. Değişkenler hava durumu, zaman, güzergah olarak 3 farklı parametre ile belirlenmiştir (FSM kullanılmadığı zaman Boğaziçi köprüsü tercih ediliyor).

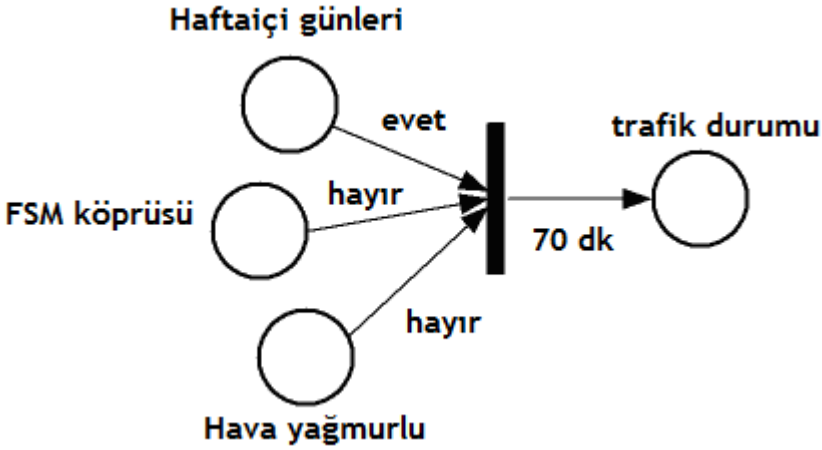
Bu değerlere göre, Kozyatağı-Mecidiyeköy hattında hedefe varış süresinin değişimi farklı koşullarda analiz edilecektir. Aşağıdaki kriterlere göre bir model oluşturulacaktır:

- Haftaiçi günleri olduğu zaman (Evet), FSM köprüsü kullanılıyor ise (Evet), hava yağmurlu ise (Evet), Kozyatağı-Mecidiyeköy arası 80 dakika sürer.
- Haftaiçi günleri olduğu zaman (Evet), FSM köprüsü kullanılmıyor ise (Hayır), hava yağmurlu değil ise (Hayır), Kozyatağı-Mecidiyeköy arası 70 dakika sürer.
- Haftaiçi günleri olmadığı zaman (Hayır), FSM köprüsü kullanılıyor ise (Evet), Kozyatağı-Mecidiyeköy arası 60 dakika sürer.

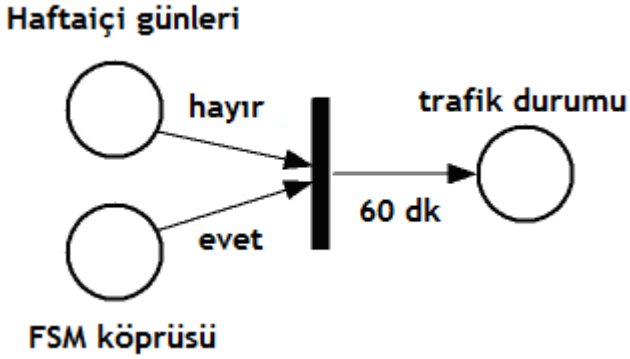
Tasarlanan bu model üzerine görsel olarak kurallar üretilerek Petri ağı ile bağımlılıklar görülebilecektir (Şekil 3.9, Şekil 3.10, Şekil 3.11).



Şekil 3.9: Kural 1

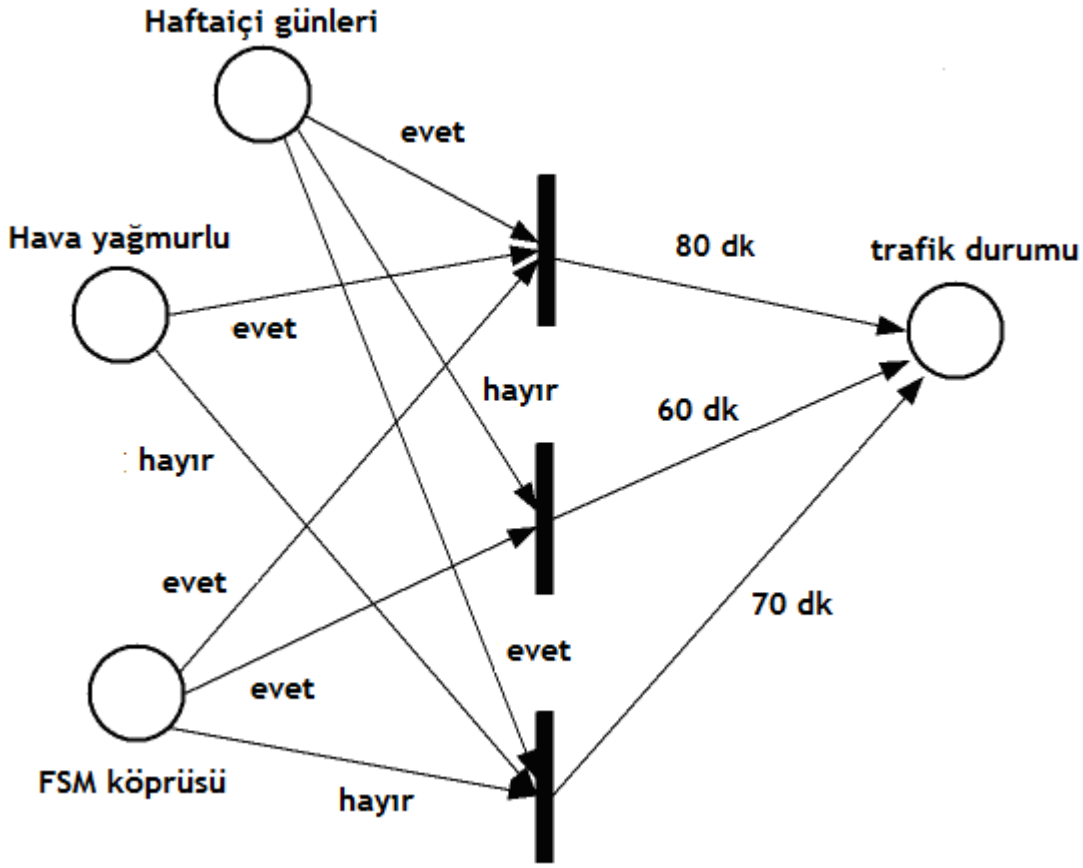


Şekil 3.10: Kural 2



Şekil 3.11: Kural 3

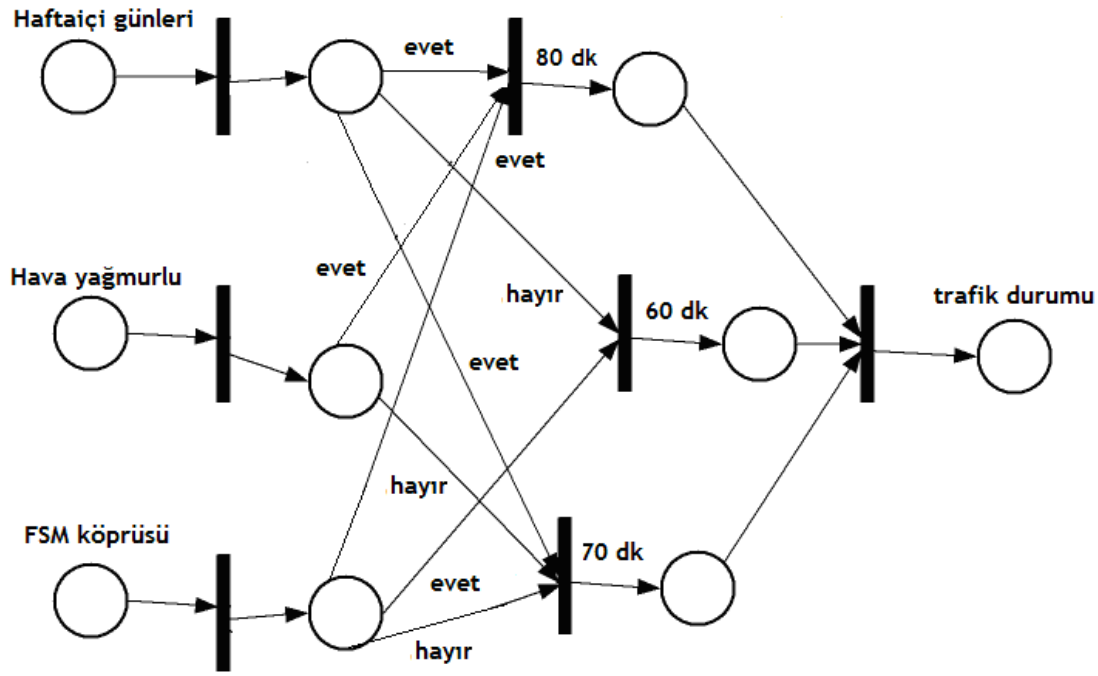
3 kural da tanımlandıktan sonra bu bağımsız kurallar tek bir petri ağında birleştirilebilir (Şekil 3.9). Bağlantılar aynı şekilde özet niteliğinde gibi görünüyor olsa da sonuçta ortaya çıkan bulanık kümedeki her bir elemanın üyelik seviyesine bakılacak olursa tüm geçişlerin aktif olduğu durumdaki ihtimallere bakılarak özel terimlerin maksimum seviyesi bulunabilir.



Şekil 3.12: Kuralların basit bir şekilde birleştirilerek bağlantılarının gösterilmesi

Petri ağlarında bir düzenleme yaparak optimum ağ yapısı elde edilir (Şekil 3.6).

Son bir dönüşüm ile çıkışa yönelik bulanık küme IF-THEN kuralları ile birleştirilerek bulunabilecektir. Jetonların da Petri ağı sistemine eklendiği düşünülürse "trafik durumu" ile belirtilen yer son durumdaki bulanık kümeyi içerecektir. Üyelik seviyesine göre de trafik durumunun nasıl olacağı konusunda güvenilir bir sonuç elde edilebilecektir.



Şekil 3.13: Geliştirilen modelin son versiyonu

3.2.4 Bulanık petri ağlarının erişilebilirlik analizi

Bulanık Petri ağlarında erişilebilirlik analizi 4 adımda anlatılabilir. $\forall M_1, M_2 \in [M_0>$. ilişkisi için M_1'den M_2'ye erişilebilirlik ilişkisi oluşturulabilir, erişilebilirlik ağacı düzenlenebilir (Zhong, L.X., 1998).

Adım 1: Ana düğüm olarak M_1 alınır.

Adım 2: M_1 başlangıç düğümü olarak kabul edilir. Eğer;

F(M_1) = Φ veya $\forall M \in F(M_1)$ için $M_2 \notin$ [M> ise M_1'den M_2'ye erişilebilirlik yoktur.

Eğer $M_2 \in F(M_1)$ ve $M_1(p) \geq W(p,t) \geq \tau(t)$ ise ağaç üzerinde M_2 adlı yeni bir düğüm tanımlanır. Ateşleme kuralı ile M_2'nin imleme yapısı belirlenir, böylece "M_1'den M_2'ye erişilebilir" denilebilir.

Diğer durumlarda, tüm $M' \in F(M_1)$ için, eğer $M_2 \notin$ [M> ve $M_1(p) \geq W(p,t) \geq \tau(t)$ ise ağaçtaki bir başka düğüme M' verilir, M'nin imlemesi

de ateşleme kuralıyla belirlenir. M', M_1'in doğrudan arkasından gelen düğüm olarak öngörülür ve bitmeyen (non-end) düğüm olarak adlandırılır.

Adım 3: Eğer bitmeyen düğüm var ise Adım 4'e geçilir, yok ise Adım 2'ye gidilir.

Adım 4: Eğer $M_2 \notin$ [M> ise M_1'den M_2'ye erişilebilirlik yoktur. Diğer durumlarda M_1 [t_1, t_2,....., t_n > M_2 sağlayacak şekilde t_1, t_2,....., t_n biçiminde bir geçiş sıralaması oluşur. Bu da M_1'den M_2'ye erişilebilirlik olduğu anlamına gelir.

BPA'lardaki erişilebilirliği daha ayrıntılı incelemek açısından 6 önermeli bir sistem örneği verilebilir. Bu sistem de BPA'ların erişilebilirliği ile bilgilerin nasıl temsil edileceğini gösterir. 6 adet önerme için $\ddot{o}_1, \ddot{o}_2$,... $\ddot{o}_6$ şeklinde gösterim yapılırsa, bulanık kurallar ile bilgi veritabanı kuralları dizisi aşağıdaki gibi oluşturulabilir:

Kural 1: If $\ddot{o}_2$ then $\ddot{o}_1$ (CF=0.8)

Kural 2: If $\ddot{o}_1$ then $\ddot{o}_6$ (CF=0.8)

Kural 3: If $\ddot{o}_6$ then $\ddot{o}_5$ (CF=0.8)

Kural 4: If $\ddot{o}_5$ then $\ddot{o}_4$ (CF=0.8)

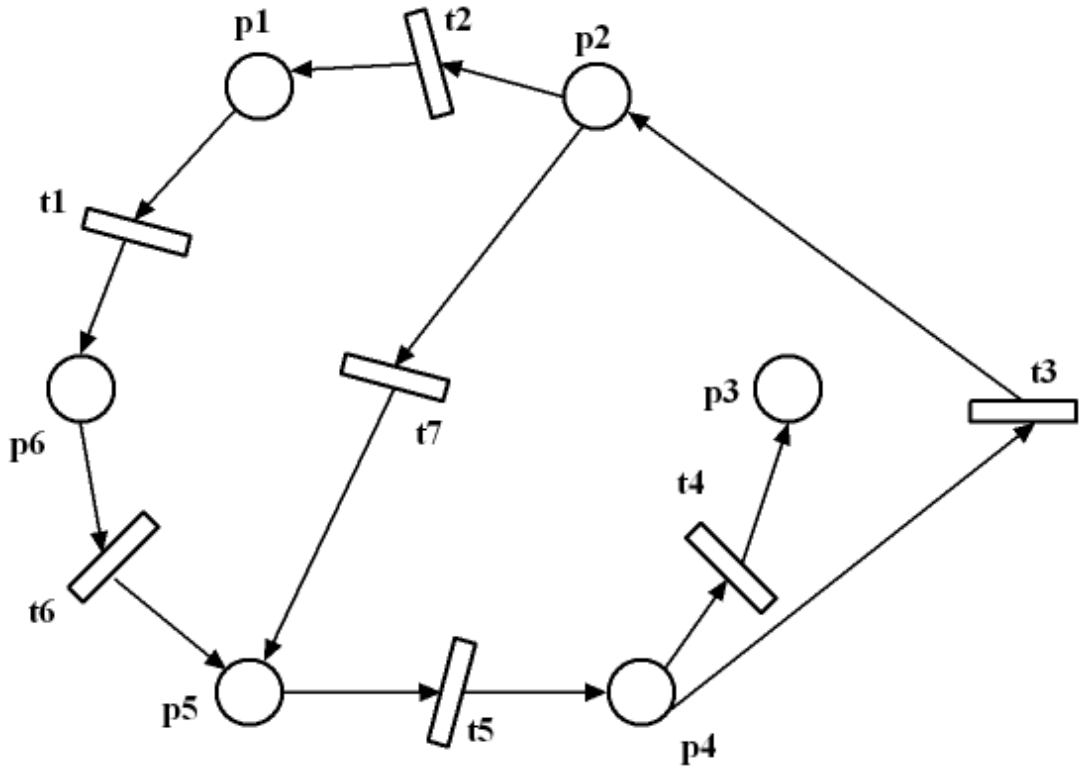
Kural 5: If $\ddot{o}_4$ then $\ddot{o}_3$ (CF=0.7)

Kural 6: If $\ddot{o}_4$ then $\ddot{o}_2$ (CF=0.6)

Kural 7: If $\ddot{o}_2$ then $\ddot{o}_5$ (CF=0.7)

Burada, τ eşik değeri 0.2, $M_0(\ddot{o}_2) = 0{,}8$ başlangıç imlemesi verildiğini düşünebiliriz.

(CF=Certainty Factor)



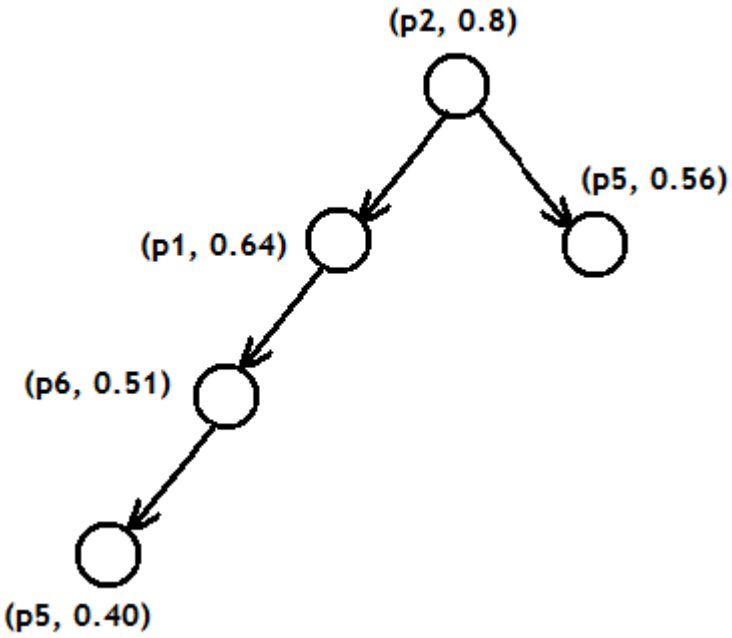
Şekil 3.14: Bir BPA Sistemi Örneği

Önermeler BPA'lardaki yerlere karşı geliyor, meydana gelen olaylar ise önceden belirlenmiş kesinlik katsayılarına göre geçişler ile temsil ediliyor. Buna göre, Önermeler $\rightarrow$ Yerler ve $\ddot{o}_i \rightarrow p_i$ dönüşümü uygulanabilir (Bkz. **Şekil 3.14**). Burada $M_o(p_2) = 0.8$ verildiğine göre diğer yerler 0 olarak kabul edilebilir: Mi(0.8, 0, 0, 0, 0, 0). Kesinlik faktörleri aynı zamanda geçişlerin sinyal güçlerini temsil etmektedir. Buna göre,

$\alpha(t_2) = 0.8$, $\alpha(t_1) = 0.8$, $\alpha(t_6) = 0.8$, $\alpha(t_5) = 0.8$, $\alpha(t_4) = 0.7$, $\alpha(t_3) = 0.6$, $\alpha(t_7) = 0.7$

$\forall t$ için $\tau(t) = 0.2$ olarak verildiğine göre, ağırlık fonksiyonu da tanımlayabiliriz. W değişken ağırlık fonksiyonu ve W(p,t)=M(p), $\forall p \in t^{\bullet}$, W(t,p)=minW(p',t). $\alpha(t)$, $\forall p \in {}^{\bullet}t$ şeklinde tanımlanabilir. Buna göre W(p,t) ifadesi t geçişinin sinyal gücü ile t geçişinin giriş bağlantısının ağırlık fonksiyonunun minimum değeriyle çarpımı olarak verilir. Burada Mi(0.8, 0, 0, 0, 0, 0) ifadesinde değeri 0 olan jeton içermeyen yerler çıkarılırsa ifade Mo(p_2,0.8) şeklinde daha kısa ve sade bir şekilde de gösterilebilir. **Şekil 3.15**'te de görüldüğü üzere p5'e iki farklı yoldan erişilebilirlik mümkündür. Mo(p2, 0.8)'den M1(p5, 0.40) ve M2(p5, 0.56)'ya

erişilebilirlik vardır. p5'in imlemesi yüksek değer olan 0.56 seçilebilir, kesinlik katsayısı daha yüksektir.



Şekil 3.15: BPA'nın erişilebilirlik ilişkisi

4. BULANIK PETRİ AĞLARININ BİR DEMİRYOLU TRAFİK SİSTEMİNE GÖRE MODELLENMESİ VE FPGA ÜZERİNDE GERÇEKLENMESİ

Bu bölümde Bulanık Petri ağlarının bir tren yolu sistemine uyarlanması yapılacaktır. Modelde kullanılan sistem parçası gerçek hayattaki versiyonuyla aynı olup temel amaç tren yollarındaki ve hizmetin işleyişindeki herhangi bir aksaklıktan dolayı servis dışı kalma durumuna çözüm aramaktır. Tren sevkiyatları tasarlanan Bulanık Petri ağına göre işleyecektir. Bu sistem bir çok yönüyle ayrık bir sistem olup BPA'larla gerçeklenerek sayısal devre olarak sentezlenebilir.

Bu çalışmada ise Çizelge 4.1'deki temel değişkenlerden bir kısmı parametre olarak seçilerek sistemin sade bir yapı kazanması amaçlanmıştır. Tasarlanacak olan sistemin büyüklüğü ne olursa olsun tasarımı yapılan BPA ana mantık teşkil ederek çekirdek yapı görevi üstlenecektir.

Demiryolu trafiğini etkileyen değişkenler şöyle sıralanabilir:

Çizelge 4.1 Tren Sevkiyat Parametreleri (Cheng ve Yang, 2009)

Sevkiyat Karar Faktörleri	**Sevkiyat Seçenekleri**
Tren tipi	Ek sefer treni
Tamamlanmamış yol	Tren seferinin iptali
Trenlerin birbirini izleme düzeni	Birleştirilmiş tren yapısı
Ortalama durak bekleme süresi	Trenin geri dönmesi
Tren gecikme süresi	
Yolcu yoğunluğu	
Yolculuk tipi	

BPA'ya yönelik sistemin tasarımı kağıt üzerinde genel olarak şöyle olur:

- Bulanık Petri ağlarındaki yer, geçiş ve dalların her birine tren sevkiyat karar faktörleri, sevkiyat seçenekleri, bağlantı kurulacak ilişkiler ve bunlara ait formülasyon atanır.
- Sayısal devrenin donanım tasarımı VHDL kodları ile yapılacağından, bulanık mantık verilerini girdi olarak kabul edecek ve çeşitli kombinasyonlarla çeşitli kararlar verecek sistem IF-THEN yapısıyla algoritmik olarak belirlenir.

Sonuç olarak gerçekleştirilen tümdevre Altera FPGA Quartus-II yazılımında simülasyonu yapılır. Sentezlenebilir devre son olarak FPGA üzerinde gerçeklenir.

Bu çalışmada tasarlanan BPA yapısına girdi olarak kabul edilen bulanık verilerin içeriği rastlantısal bilgiler, gerçek dünaya ilişkin geçmiş tecrübeler, mantıksal

yaklaşımlar dikkate alınarak yaratılmıştır. Örneğin Tayvan'da tren yolu ağındaki tren sevkiyat özellikleri yaklaşık 10 yıl süren tecrübe sonucunda belirli üyelik fonksiyonlarının çıkarılmasını sağlamıştır (Cheng ve Yang, 2009). Bu bağlamda global ölçekte karşılaşılan pratik bulguların ülkemizdeki demir yolu (ekspres, banliyö trenleri, metro, tramvay hatları…vb) ağı için de geçerli olabileceği kabul edilebilir. Benzer şekilde mantıksal yaklaşımlar ile de örnek veriler elde edilebilir. Nitekim İstanbul'da öğleden sonra 14.00-15.00 arası ya da gece saatleri olan 23.00-24.00 arasında bir metro demiryolu hattının yoğun olması beklenmez. Aksine haftaiçi işyerlerinin mesai bitimi olan 18.00-19.00 arasında sefer periyodunun, dolayısıyla tren yolcu kapasitesinin artırılması gerekir. Bu da tren sevkiyat değişkenlerinden biri olan "yolcu yoğunluğu" maddesini doğrudan etkileyen bir gerçektir.

4.1 Tren Yolu Trafik Sistemi Modeli (TYTSM)

TYTSM oluşturulurken bulanık mantık kurallarıyla işleyen Petri ağları, BPA'lara ilişkin üyelik dereceleri de belirli grafiklerle hesaplanmıştır. Tren sevkiyatlarının nasıl düzenleneceği önceden belirlenmiş bir kural veritabanına göre yapılmalıdır. Buna göre kural veritabanı IF-THEN yapılarıyla oluşturulabilir (Çizelge 4.2).

Çizelge 4.2: Tren sevkiyatları için kural veritabanı

Kurallar	Tren Sevkiyatı Kural Veritabanı
Kural 1	IF: Tren tipi AND Tamamlanmamış mesafe AND Ortalama durak bekleme süresi, THEN: Alınacak Aksiyon Tren ekleme
Kural 2	IF: Tren gecikme süresi AND Sefer türü THEN: Alınacak Aksiyon Trenin geri dönmesi
Kural 3	IF: Tren gecikme süresi AND Sefer türü AND Tren tipi THEN: Alınacak Aksiyon Trenin geri dönmesi
Kural 4	IF: Tamamlanmamış mesafe AND Tren gecikme süresi THEN: Alınacak Aksiyon Tren ekleme
Kural 5	IF: Tamamlanmamış mesafe AND Tren gecikme süresi AND Sefer türü THEN: Alınacak Aksiyon Tren ekleme
Kural 6	IF: Tamamlanmamış mesafe AND Tren gecikme süresi AND Yolcu yoğunluğu, THEN: Alınacak Aksiyon Tren birleştirme

Aşağıda, BPA'daki her bir yere (p1, p2…p9) bulanık mantıkta sıkça kullanılan dilsel değişken ataması yapılmıştır. Burada "dilsel değişken" terimi ile kastedilen şey

"Erteleme süresi"'nin 5 dakika olması ile "kısa bir zaman dilimi", 25 dakika olması ile "uzun bir zaman dilimi" gibi anlamların üretilmesidir.

Tren Sevkiyat Değişkenleri:

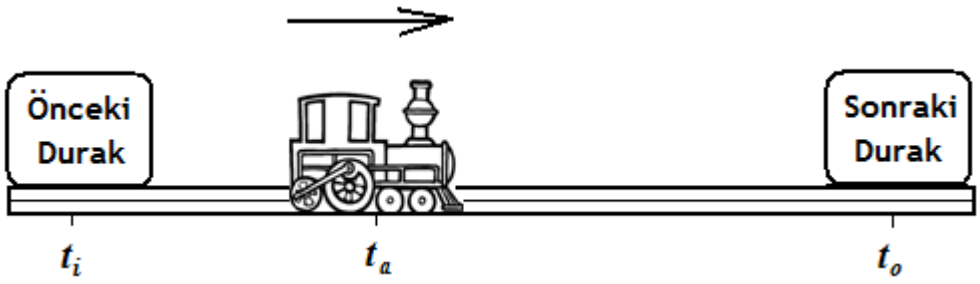
- Tren gecikme süresi (dakika): (5, 10, 15, 20, 25) → p1
- Ortalama durak bekleme süresi (dakika): (1, 2, 3, 4, 5) → p2
- Tren tipi: (ekspres ya da banliyö) → p3
- Tamamlanmamış mesafe (km): (20, 40, 60, 80, 100) → p4
- Sefer türü: (mesai saati, gün içi, gece yarısı, haftasonu) → p5
- Yolcu yoğunluğu: (çok az, az, orta, fazla, çok fazla) → p6

Alınacak Aksiyon:

- Trenin geri dönmesi → p7
- Tren ekleme → p8
- Tren birleştirme → p9

Tren sevkiyat değişkenleri tanımlanacak olursa;

a) ***Tren gecikme süresi (t_g)*** **:** Demiryolu hattında kayıtlı trenlerin duraklara varış saatleri haftaiçi ve haftasonu önceden bellidir. Buna göre bir trenin iki durak arasında seyir halinde iken gideceği durağa varacağı zaman ile varması gereken zaman arasındaki fark "Tren gecikme süresi"ni verir. Şekil 4.1'de t_i, önceki duraktan hareket saatini t_a, trenin anlık saatini t_o, sonraki durağa varması gereken saati gösteriyor. Buna göre misal olarak, iki durak arası yolculuğun 20 dakika sürdüğü bir senaryoda, $t_o = 15.00$ ise, $t_i = 14.50$ olduğu zaman "Tren gecikme süresi" $t_g = 10$ dakika olacaktır. Bu da demiryolu hattında bir düzensizlik anlamına gelir ve p7, p8 ve p9 aksiyonlarının birinin alınmasını gerektirir.



Şekil 4.1: Tren gecikme süresi

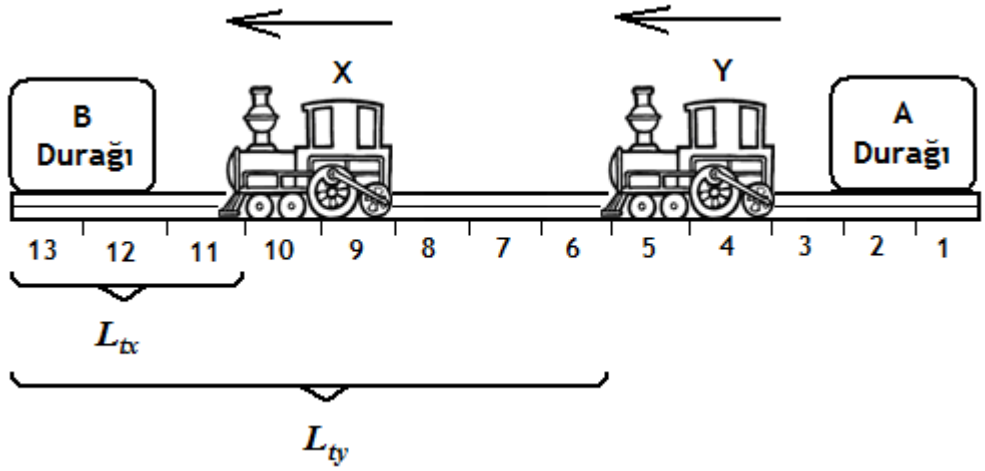
Benzer bir senaryoda, trenin önceki duraktan hareket saatinin t_i =14.40, yani istenen durum olduğu göz önüne alınırsa, demiryolu güzergahında meydana gelebilecek bir sorun nedeniyle herhangi bir t_a anında da trenin sonraki durağa geç kalacağı öngörülebilir. Yolun tam ortasında yani t_a = (t_i + *10 dk*) anında hızı normalin yarısına düşerse yine sonraki durağa varış saati 10 dk gecikebilecektir. Bu öngörü seyir halinde analiz edilerek ilgili aksiyonun alınması sağlanabilir. Bu çalışmada gecikme süreleri sadelik açısından 5'in katları halinde 5 dk, 10 dk, 15 dk, 20 dk, 25 dk olarak düzenlenmiştir.

b) ***Ortalama durak bekleme süresi*** ***(t_b):*** Tren belirli bir durağa geldiğinde yolcu yoğunluğu, demiryolu hattındaki bir yoğunluk veya başka bir olay nedeniyle normalden fazla bekleyebilir. Normalden fazla bir süre beklediğinde bu fazlalık zamana "Ortalama durak bekleme süresi" denir. Ortalama durak bekleme süresi t_b, bu çalışmada 1 dk, 2 dk, 3 dk, 4 dk, 5 dk değerlerinden birini alabilecektir.

c) ***Tren tipi (B, E):*** Bu demiryolu modeli çalışmasında genel olarak iki tipte tren olduğu düşünülüyor: Banliyö treni (B) ve Ekspres (E) tren. B sınıfına giren yani Banliyö trenlerinin şehir için ulaşımda kullanıldığını, E sınıfına giren yani Ekspres trenlerin de şehirler arası ulaşımlarda faaliyet gösterdiğini söyleyebiliriz. Trenin tipi alınacak her aksiyonu belirlemede kullanılmadığı halde bazı aksiyonların çeşitlerinin belirlenmesinde etkilidir.

d) ***Tamamlanmamış mesafe (L_{ti}):*** Bir demiryolu hattındaki trenlerin herhangi bir uygunsuz durum olduğunda gideceği durağa olan uzaklığı "Tamamlanmamış mesafe"yi gösterir (Şekil 4.2). Tamamlanmamış mesafe L_{ti}'de yer alan "i" indeks numarası, tamamlanmamış mesafenin hangi tren için ifade edildiğini bildirmek açısından kullanılabilir. L_{tx}, X treninin tamamlanmamış mesafesini, L_{ty} ise Y treninin tamamlanmamış mesafesini birim uzaklık olarak belirtmektedir. Şekil 4.2'de A durağından B durağına X ve Y trenlerinin belirtilen yönde yol alması resmedilmiştir. Kolaylık ve sadelik açısından güzergah birbirine eşit uzaklıktaki birimlere ayrılmış ve iki trenin davranışı incelenmiştir. Şekil 4.2'deki örnekte L_{tx} = 3 birim iken L_{ty} = 8 birimdir. İki tren arasındaki bu tamamlanmamış mesafe farkı alınacak aksiyonun çeşidinde büyük öneme sahiptir. Mantık olarak, bu trenlerin ikisinde de bir sorun yaşandığında sefer aksaklığının önlenmesi için $L_{tx} < L_{ty}$ ilişkisi nedeniyle bu demiryolu

hattındaki A-B durağı arasındaki X ve Y trenlerine farklı şekilde müdahale edilmesi gerekmektedir. Örnek olarak X treninin B durağına gelmesi için 3 birimlik bir yol kalmıştır. Bu nedenle X trenine "Trenin geri dönmesi" aksiyonu yerine "Tren ekleme" ya da "Tren birleştirme" aksiyonlarının uygulanması mantıklı olacaktır. Y trenine ise "Trenin geri dönmesi" işleminin yapılması yerinde olacaktır. Çünkü A durağından sadece 5 birim ötede sorunla karşılaşılmıştır.



Şekil 4.2: Tamamlanmamış mesafe

Bu çalışmada ise "Tamamlanmamış mesafe" olan L_{ti} için 20 km, 40 km, 60 km, 80 km, 100 km değerlerinden birini alabileceği öngörülmüştür.

e) ***Sefer türü*** ***(****mesai saati, gün içi, gece yarısı, haftasonu****):*** Sefer türü demiryolu hattında çalışacak olan trenlerin çalıştığı tarih ve saatlerin niteliği ile ilgilidir. Haftaiçi ve haftasonu sefer türlerinin diğer sefer türleriyle kesişimi var gibi gözükse de gerek sefer sayısı ve sıklığı gerekse sefer saati, yoğunluğu açısından ayrık olarak algılanmalıdır. Neticede haftaiçi ve haftasonu yolcuların ulaşım aracı davranışları değişkendir. Bu arada mantık olarak Haftaiçi = {mesai saati, gün içi, gece yarısı} şeklinde bir küme de tanımlanabilir. O yüzden "haftaiçi" adlı başka bir sefer türünün tanımlanmasına gerek yoktur. Haftasonu işyerlerinin çalışmadığı kabul edilmiştir. Bütün bu yaklaşımlar tüm modelin işleyişini etkileyecek bir kurgu olarak düşünülebilir. Tren yolcularının büyük çoğunluğunun çalışan kesim olduğu ve bu kişilerin de ulaşım ihtiyaçlarının demiryolu ağı ile de sağlanabildiği düşünülürse "mesai saati" sefer türü daha kolay algılanabilir. Mesai saati, işyerlerinin açılış ve kapanış saatlerini kapsayan belirli zaman dilimlerinde gerçekleştirilen sefer türüdür. Gün içi, bir gün içinde sabah 06.00 ile akşam 23.59 arasında, mesai saati dışındaki zamanlarla ilgili sefer

türüdür. Geceyarısı, gece 24.00 ile sabah 05.59 arasındaki zaman diliminde yapılan sefer türüdür. Haftasonu ise haftasonları yapılan sefer türlerini ifade etmektedir. Bu çalışmada sefer türü 4 farklı dilsel değişkenle temsil edilmektedir: mesai saati, gün içi, gece yarısı, haftasonu.

f) ***Yolcu yoğunluğu*** ***(****çok az, az, orta, fazla, çok fazla****):*** Yolcu yoğunluğu, bir demiryolu hattındaki trenleri kullanan kişi sayısıdır. Yolcu yoğunluğu yılın belirli zamanlarında çeşitli nedenlerle artmakta veya azalmaktadır. Dolayısıyla BPA modelinin oluşturulması öncesinde giriş verilerinin belirlenmesinde geçmiş dönemde insanların davranışlarıyla ilgili bilgiler de derlenerek bir veri havuzu oluşturulmalı ve bulanık mantık verileri buna göre ortaya çıkarılmalıdır. Yolcu yoğunluğu diğer tren sevkiyat değişkenlerinden sefer türü ile de ilgili gibi gözükse de bağımsız bir veri olarak algılanmalıdır. Yolcu yoğunluğunun önemi "Tren birleştirme" aksiyonunun alınmasında ortaya çıkmaktadır.

Demiryolu hattındaki bir trenle ilgili herhangi bir istenmeyen durum kombinasyonuyla karşılaşıldığında çeşitli aksiyonların alınması gerekir. Alınacak Aksiyonlar tanımlanacak olursa;

i) ***Trenin geri dönmesi:*** Bir trenin çalışmasını tren sevkiyat değişkenlerinin bazılarının bir arada gerçekleşmesi etkilemektedir. "Trenin geri dönmesi" sefer halindeki bir trenin çeşitli nedenler karşısında ana istasyona ya da bir önceki durağına çağrılması sonucu geri dönmesi eylemidir. Mevcut sorunun giderilmesi için "Trenin geri dönmesi" aksiyonunu alan trenin geri dönme işlemini tamamlamasıyla sorunun geçici ya da kalıcı olarak çözülebileceği beklenir.

ii) ***Tren ekleme:*** Demiryolunda seyahat eden yolcuları taşıyan tren elektriksel, mekanik, statik ya da davranışsal nedenler dolayısıyla artık işleyemez durumda ise "Tren ekleme" aksiyonu alınır. Bir bakıma bozuk trenle sağlam tren arasındaki takas işlemidir. Bu durumda yeni bir tren sorunlu trendeki yolcuları almak için gönderilir ve sefer iptal olmadan yolcu taşımacılığına devam edilir. "Tren ekleme" ile hatlardaki sorun geçici olarak çözüme kavuşturulmuş olur. Farklı mantıksal durumların (tren sevkiyat değişkenleri) kombinasyonlarıyla bu aktivite tetiklenebilir. Bu seçeneğin aktif olması için "Yolcu yoğunluğu" dışındaki diğer tren sevkiyat değişkenlerinin etkili olması gerekir.

iii) ***Tren birleştirme:*** Lokomotifi sorunlu olup da vagonları kullanılabilecek bir trene başka bir lokomotifli-vagonlu trenin gönderilmesi ve ilgili vagonların yeni gönderilen trene bağlanmasına "Tren birleştirme" denir. Burada iptal olabilecek

vagon sayısına göre yeni vagon gönderileceğinden trendeki yolcu yoğunluğu önemli bir kriter olacaktır. Bu yöntem ile demiryolu hattındaki verimlilik zarar görmez, çünkü kapasite azalması "Trenin geri dönmesi" ve "Tren ekleme" aksiyonlarındaki gibi fazla değildir.

Tüm Tren Sevkiyat Değişkenleri ile Alınacak Aksiyon elemanları, anlamsal ve mantıksal olarak ayarlanabilir özelliktedir, uygulamaya ve amaca yönelik olarak değiştirilebilir. Bu çalışmada ise her bir bileşene keyfi olarak bir amaç atanmış ve önerilen koşullara göre sistemin işleyişi gösterilmiştir. Bileşen sayısı, içeriği, dayandığı prensipler, sınır değerleri gibi bir çok karakteristik esnektir ve ihtiyaca göre belirlenebilir.

Şekil 4.3'teki BPA modelinde nesneler (jetonlar) gösterilmemiştir, çünkü ilgili giriş yerlerindeki koşullar sağlandığı anda jetonlar oluşacaktır. Bu nesneler dinamik aktiviteyi tetikleyecektir.

Tasarlanacak olan BPA'nın bileşenleri giriş yerleri, geçişler ve çıkış yerleri olacaktır (Şekil 4.3). Bu BPA elemanlarından;

Giriş yerleri, demiryolu hattındaki tren seferlerinin düzenlenmesinde tren sevkiyat değişkenlerinin bulunduğu 6 elemanlı kümeyi temsil eder (Bkz. **(4.1)**).

Çıkış yerleri de geçişler ile sağlanan giriş yerlerinin kombinasyonları sonucunda alınacak aksiyonları temsil eder ve bu sonuçlar da 3 elemanlı bir küme oluşturur (Bkz. **(4.2)**).

Geçişler, giriş yerlerine karşı gelen bulanık değerlerle hangi işlemin yapılacağını tanımlayan olayı temsil eder. Örneğin IF-THEN kural oluşumunda IF ile THEN arasındaki kuralları oluşturan formüller ya da kuralın yapısı geçişleri oluşturur.

BPA için burada geçişlerin giriş yerlerine ilişkin kümelerin birleşimi alınırsa,

$$I(t1) \cup I(t2) \cup I(t3) \cup I(t4) \cup I(t5) \cup I(t6) = \{p1, p2, p3, p4, p5, p6\}$$

(4.1)

olduğu görülür.

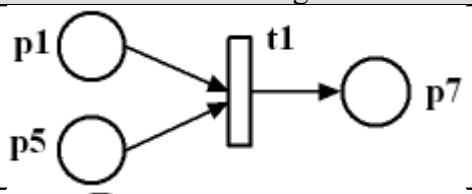
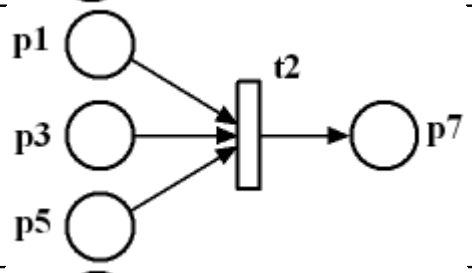
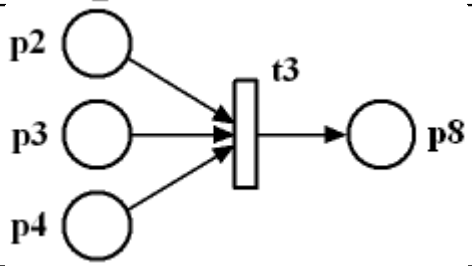
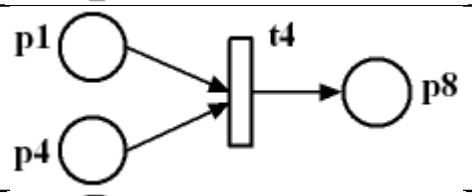
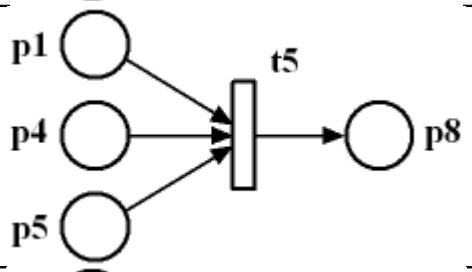
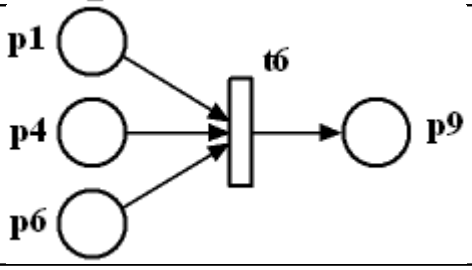
Aynı şekilde geçişlerin çıkış yerlerine ilişkin kümelerin birleşimi alınırsa da,

$$O(t1) \cup O(t2) \cup O(t3) \cup O(t4) \cup O(t5) \cup O(t6) = \{p7, p8, p9\}$$

(4.2)

eşitliği sağlanır.

Çizelge 4.3: Giriş ve Çıkış Fonksiyonlarının BPA Gösterimi

Giriş Fonksiyonu	Çıkış Fonksiyonu	BPA Grafiği
$I(t1) = \{p1, p5\}$	$O(t1) = \{p7\}$	
$I(t2) = \{p1, p3, p5\}$	$O(t2) = \{p7\}$	
$I(t3) = \{p2, p3, p4\}$	$O(t3) = \{p8\}$	
$I(t4) = \{p1, p4\}$	$O(t4) = \{p8\}$	
$I(t5) = \{p1, p4, p5\}$	$O(t5) = \{p8\}$	
$I(t6) = \{p1, p4, p6\}$	$O(t6) = \{p9\}$	

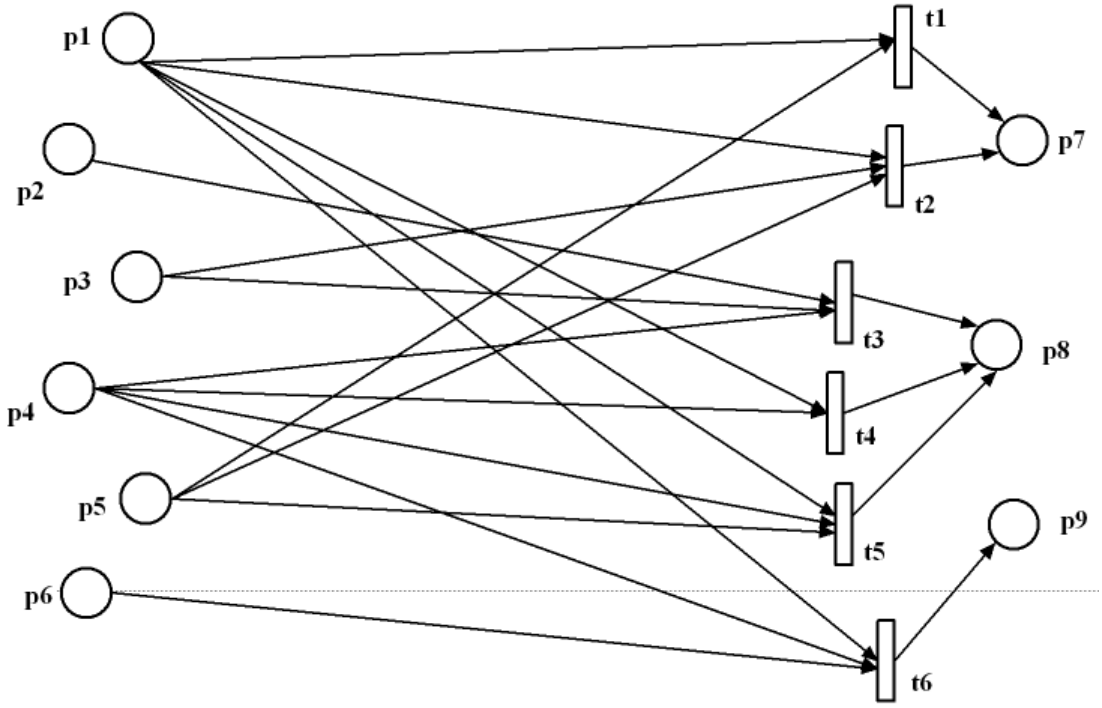
Klasik Petri ağlarının hem grafiksel hem de matematiksel olarak tanımlanabilmesi BPA'lar için de geçerlidir (Çizelge 4.3). Çizelge 4.3'teki tüm BPA grafiklerinin birleştirilmesiyle Şekil 4.3'deki trenyolu BPA modeli elde edilmiş olur.

Çizelge 4.4: BPA Elemanlarının Dilsel Değişken Olarak Anlamları

BPA Elemanı Türü	Dilsel Değişken Olarak Anlamı
Giriş Yerleri	Tren sevkiyat değişkenleri

Geçişler	Giriş yerlerinin önceden belirlenmiş bir kurala göre öngörülen kombinasyonları
Çıkış Yerleri	Alınacak Aksiyon

Burada tasarlanacak olan BPA'da giriş yerleri Tren Sevkiyat Değişkenleri (p1, p2….p6) olacak, çıkış yerleri de Alınacak Aksiyon (p7, p8, p9) olacaktır.



Şekil 4.3: Trenyolu BPA Modeli

Oluşturulan BPA modeli ayrık bilgileri işlemektedir, yani farklı durumlar birbirlerinden bağımsızdır. Şöyle ki tren tipi ile erteleme süresi bambaşka ayrık koşulları temsil etmektedir.

BPA'daki geçişlerin duyarlılığı (üyelik derecesi, eşik değeri) sistemde alınacak aksiyonu belirleyen en temel değişkenlerden biridir. Tasarımcı tarafından belirlenecek bu değer dikkatle test edilmeli ve optimum aralığa çekilmelidir. Nitekim geçişlerin eşik değerinin çok düşük tutulması gereksiz yere aksiyon alınmasına yol

açabilir. Aynı şekilde geçişlerin duyarlılığının yüksek tutulması (1'e yakın olması) da gerekli aksiyonun alınamamasına neden olacaktır.

Hazırlanan modelin kendi içinde mantıklı ve tutarlı olması gerekir. Çünkü günlük hayata uyarlandığı zaman herhangi bir eksiklik ya da yanlışlık demiryolu seferlerinin aksamasına neden olur bu da maddi kayıplara anlamına gelir.

Tasarlanan BPA modeli demiryolları dışında benzer başka sistemler için de alternatif olarak kullanılabilir. Havayolu taşımacılığı, araç filo yönetimi, yer altı metro istasyonları, liman ve denizyolu yönetimi ve daha bir çok alana uyarlanabilir. Uygulamaya ve sistemin kapsamına göre parametre ve aksiyon sayısı, çeşidi, sıklığı, üyelik derecesi grafiği değişebilir. Şüphesiz gerçek hayatta trenlerin çalışma tarzı yolcuların talep yoğunluğuna, periyoduna ve zamanına göre değişecektir. Bu durumda bu çalışmada kullanılan model değişkenleri olan "erteleme süresi", "ortalama durak bekleme süresi", "yolcu yoğunluğu" gibi maddelerin tekrar düzenlenmesi gerekecektir.

Bu çalışmada tren modelleme sisteminde meydana gelen istenmeyen durumlar demiryolu işleyişiyle örneklenerek incelenmiştir. Tren seferlerinin kalkış periyotları, sefer saati ve sayısı, hızı ve kapasitesi, durak sayıları ve yerleri gibi değişkenler ayrı bir inceleme alanı oluşturur. Demiryolu hatlarının düzenli işleyişinde sistemin mantıklı çalışmasıyla birlikte insan faktörünün de önemi unutulmamalıdır. Demiryolu seferi yöneticilerinin herhangi bir alışılmadık durumda BPA çıktılarına göre iletim hatlarını düzenlemesi, operasyonel prosedürleri buna göre uygulaması sistemin çalışırlığını etkileyecektir.

BPA sistemindeki giriş değişkenleri pratik yaşamdaki tecrübelere dayanılarak oluşturulabilir. Sisteme özgü parametreler tanımlanarak yine istenilen aksiyonların alınması sağlanabilir. Hatta BPA'nın giriş yerlerine atanan girdilerin doğruluğunun güncel tutulması için sistematik ve anlık olarak kaynak verileri işlenebilir. Bu yaklaşım, tasarlanan sistemin başarısını ve güvenilirliğini artıracaktır.

4.1.1 Bulanık mantık değerlerinin belirlenmesi

Petri ağları uygulamada üretim hatları sistemlerinde modelleme ve analiz amaçlı olarak sıklıkla kullanılmakla birlikte davranışsal seçim döngülerinde de yararlanılabilmektedir. Petri ağının bulanık verilerle çalışması gerçek hayatta karşılaşılan verilerin analog olmasıyla da ilişkilidir. Böylece dilsel değişkenlerle 0-1

arasındaki değerler eşleştirilecek ve sistemin karar verme yeteneği geliştirilmiş, genişletilmiş olacaktır.

Bu çalışmada hazırlanan BPA'ya girdi olacak olan "giriş yerleri" için atanan bulanık veriler belirsizliğin göstergesi olarak algılanabilir. BPA modelini oluşturan parametrelerin belirsizliklerinin belirli alanlara atanması grafiklerle tanımlandığı gibi mümkün olmuştur.

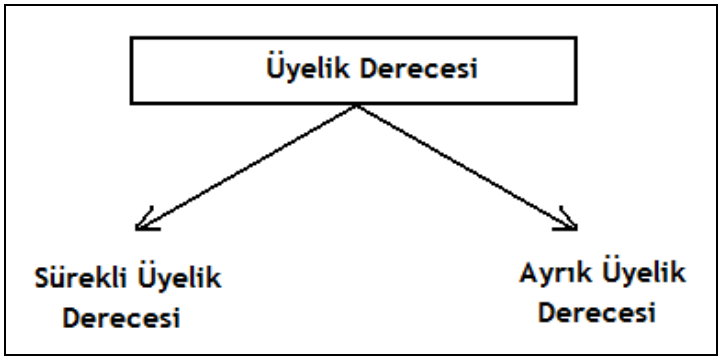
Model parametreleri (tren sevkiyat değişlenleri) geçerliliği ve doğruluğu kişisel tecrübeye dayanan, denemelerle kazanılan geçmiş tecrübeleri yansıtan verilerle temsil edilebileceği gibi belirli kriterlere göre düzenlenmiş formüllerle yorumlamak suretiyle de oluşturulabilir. Örneğin Tayvan demiryollarında trenlerin işleyişinin geçmiş tecrübelere dayandırılarak hazırlanması ile ilgili çalışmalar yapılmıştır (Cheng ve Yang, 2009). Sistem bu verileri girdi olarak kabul etmekte ve ona göre çıktılar üretmektedir. Bu çalışmada da rastlantısal olarak üretilmiş değerler bir üyelik fonksiyonunda ilgili olduğu aralığa yönlenecek ve önceden tanımlanmış bir kurallar veritabanına göre aksiyon alınması sağlanacaktır.

Giriş yerlerinde belirli koşulların kombinasyonunun gerçekleşmesi geçişler üzerinde çıkış yerlerindeki aksiyonların alınmasını sağlayacak olayların gerçekleşmesine zemin hazırlar. Geçişlerde hangi olayların sergileneceğine de tasarımcı karar verir. Bu çalışmada belirli kombinasyonların bir araya gelmesi farklı olayların meydana gelmesini sağlıyor. Tasarlanan Trenyolu BPA Modelindeki geçişlerde bulanık mantıkta kullanılan MAX ve MIN (sırasıyla AND, OR operatörlerinin karşılığı) operatörleri kullanılabileceği gibi sonuç değeri bulanık seviye aralıkları olan 0-1 arasında tutacak başka işlemler de uygulanabilir. Örnek verilecek olursa, giriş yerlerinden gelen bulanık değerler (b1, b2,…b6) 0-1 aralığında olacaktır. 0-1 aralığında olan çoklu sayılarla yapılacak olan işlemler de yine 0-1 aralığında olmalıdır. Trigonometrik sin(x), cos(x), ortalama değer ve benzeri matematiksel işlemlerin kullanılması mümkündür:

$0 < \sin(x) < 1, \qquad 0 < \cos(x) < 1, \qquad 0 < \frac{\beta 1 + \beta 2}{2} < 1, \qquad 0 < \frac{\beta 1 + \beta 2 + \beta 3}{3} < 1$

Üyelik dereceleri de ayrık ya da sürekli olup olmamasına göre ikiye ayrılır. Sürekli üyelik derecelerinde grafik belirli bir aralıkta her türlü gerçek değeri alabilir. Örneğin 0-5 aralığındaki herhangi bir reel sayı (tren gecikme süresi) 0,7 üyelik derecesine karşı gelir. Fakat Tren tipi (E ya da B) gibi değişkenler ayrıktır çünkü iki farklı

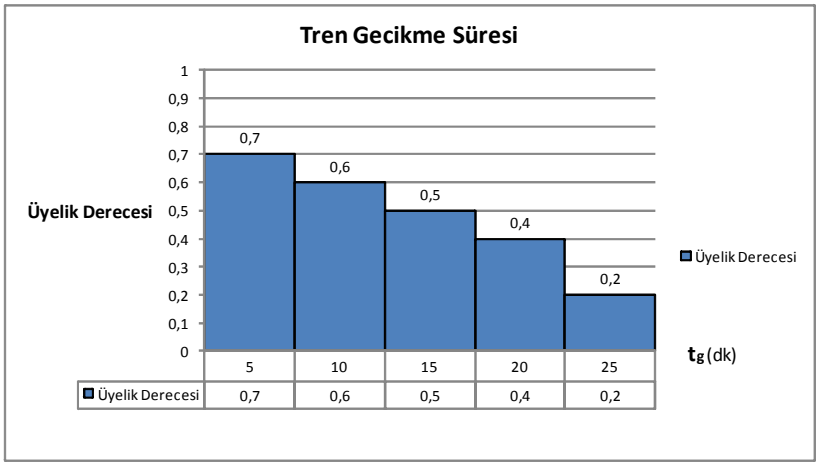
olasılıktan biri geçerli olacaktır. Buna göre E sınıfı trenlere belirli bir üyelik derecesi atanır, B sınıfı trenlere de yine bir üyelik derecesi atanır. Herhangi bir aralık söz konusu değildir (Şekil 4.4).



Şekil 4.4: Üyelik derecesi sınıflandırması

Bu çalışmada her bir BPA giriş yeri maddesine özel bir üyelik fonksiyonu atanmıştır. Üyelik fonksiyonları kurgulanırken ilgili maddenin önem derecesi, pratik yaşamda mümkün olabilecek karşılaşılma sıklığı, kullanım tarzı gibi nitelikleri dikkate alınmıştır. Bazı üyelik fonksiyonları ise rastlantısal değişkenlerle tanımlanmıştır. Üyelik fonksiyonları ile her bir model değişkeninin (tren sevkiyat değişkenleri, giriş yerleri) önemlilik derecesi 0-1 arası değerlere dönüştürülmektedir. Bulanık mantık biliminde değişik üyelik fonksiyonu belirleme yöntemleri önerilmiş, üçgensel, yamuk ve benzeri şekillerle sıkça grafikler oluşturulduğu belirtilmiştir. Fakat bu fonksiyonların daha gerçekçi olabilmeleri için de gerçek hayattaki verilerle beslenmesi, giriş verilerinin üyelik fonksiyonlarına atanması aşamasındaki doğruluk yüzdesini artıracaktır.

Tren Gecikme Süresi: Aşağıdaki Şekil 4.5'te tren gecikme sürelerinin belirli bir üyelik fonksiyonuyla hesaplanmış üyelik derecesi görülmektedir. Üyelik fonksiyonu hazırlanırken sinc-fonksiyonundan yararlanılmıştır. Bu açıdan tren gecikme süresi grafiği ile sinc fonksiyonunun grafiği şekil olarak birbirine benzemektedir.



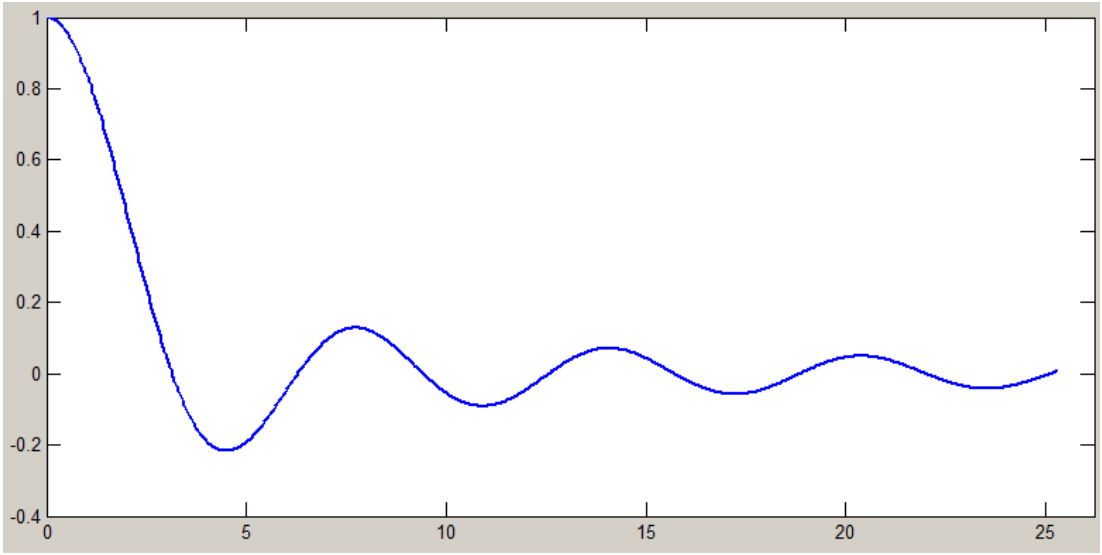
Şekil 4.5: Tren Gecikme Süresinin Üyelik Derecesi

Sinc fonksiyonu Şekil 4.5'teki grafiği oluşturmada yardımcı olabilir (Smith, S.W., 1999). Genel gösterimi, y=sinc(x) şeklinde olup 2π genişliğinde ve 1 birim uzunluğunda bir kare dalga işaretinin sürekli ve ters Fourier dönüşümüdür. Sinc fonksiyonu aşağıdaki ifadelerle tanımlanmıştır:

$$x = 0 \text{ ise} \quad \sin c(x) = 1; \qquad x \neq 0 \text{ ise} \quad \sin c(x) = \frac{\sin(\pi t)}{\pi t};$$

(4.3)

Sinc fonksiyonunun MATLAB ile çizilmiş grafiği ise şöyledir:



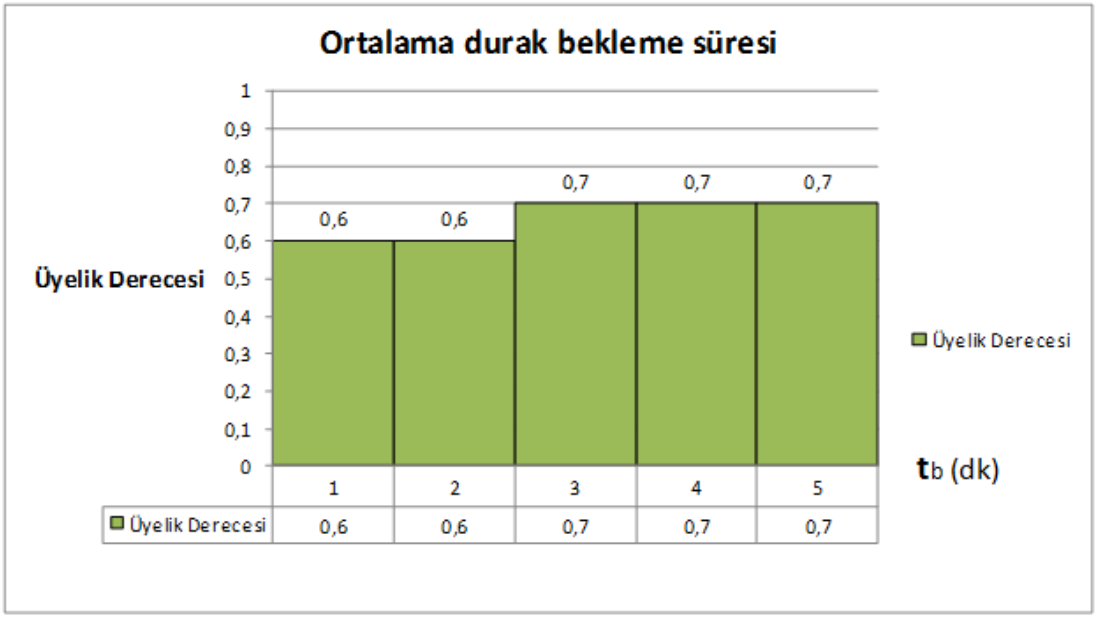
Şekil 4.6: Sinc fonksiyonu grafiği

Şekil 4.5'teki grafikte y-ekseni üyelik derecesini, x-ekseni de tren gecikme süresinin dakika cinsinden değerini gösteriyor. Burada sistemi bulanık hale getirmek için ve dolayısıyla BPA'nın giriş yerlerinin değerini bulmak için sinc fonksiyonundan yararlanılmıştır. g(x) üyelik fonksiyonu, ilgili x değerine karşı gelen y değerini gösterdiğinde, y = g(x) = $\left[\frac{\sin(\pi x)}{\pi x}. 100 - 1{,}02\right]$ ile tanımlanabilir. Bu hesaplandıktan sonra sayı yuvarlandığında üyelik derecesi bulunmuş olur.

Çizelge 4.5: Tren Gecikme Süresinin Üyelik Derecesinin Bulunması

Tren Gecikme Süresi (dk)	Sinc(x) Fonksiyonu	g(x) Fonksiyonu	Üyelik Derecesi
5	sinc(5) = 0.01724	0,704	0,7
10	sinc(10) = 0,01660	0,640	0,6
15	sinc(15) = 0,01555	0,535	0,5
20	sinc(20) = 0.01416	0,396	0,4
25	sinc(25) = 0.01248	0,228	0,2

Ortalama durak bekleme süresi: Ortalama durak bekleme süresine empirik olarak ya da fonksiyonel olarak bir üyelik fonksiyonu atanabilir. Bir $z = \varphi(t)$ şeklinde üyelik fonksiyonu tanımlansın. Bu durumda $z = \varphi(t) = \frac{\sin(10t) + \cos(10t)}{2}$ biçiminde bir fonksiyon tanımlanabilir. Şekil 4.7'de ortalama durak bekleme süresine ilişkin üyelik derecesi grafiği gösterilmiştir.



Şekil 4.7: Ortalama durak bekleme süresi grafiği

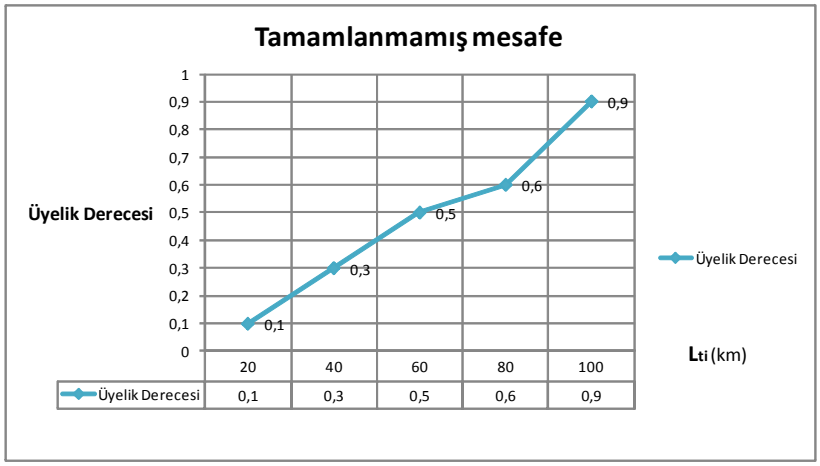
Ortalama durak bekleme süresi ve üyelik dereceleri Çizelge 4.6'da görülmektedir.

Çizelge 4.6: Ortalama durak bekleme süresi ve üyelik derecesi

Ortalama durak bekleme süresi (dk)	$\varphi(t)$ Üyelik fonksiyonu	Üyelik Derecesi
1	$\varphi(1) = 0{,}579$	0,6
2	$\varphi(2) = 0.640$	0,6
3	$\varphi(3) = 0{,}683$	0,7
4	$\varphi(4) = 0{,}704$	0,7
5	$\varphi(5) = 0{,}704$	0,7

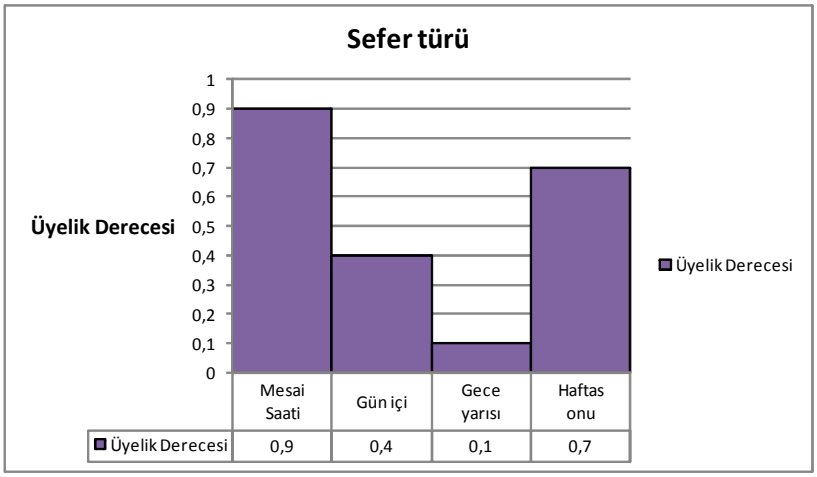
Tren tipi: Tren tipi için de iki farklı seçenek vardır: E sınıfı ve B sınıfı trenler. Bir aralık tanımlı olmadığı için ayrık bir üyelik derecesine sahiptir, keyfi olarak Ekspres trenlerin üyelik derecelerine: 0,3, Banliyo trenlerin üyelik derecelerine: 0,7 verilebilir. Bu veri seti ilgili hattın yoğunluğuna göre, geçmiş tecrübelere göre de belirlenebilir.

Tamamlanmamış mesafe: Bu veri seti rastlantısal olarak hazırlanmıştır. İhtiyaca göre ya da bulanık verilerin sonuçlarında ortaya çıkan aksiyonlara göre şekillendirilebilir. Şekil 4.8'de tamamlanmamış mesafeye ilişkin üyelik dereceleri gösterilmiştir. 20 km ve katları şeklinde gerçek veriler girilmiştir.



Şekil 4.8: Tamamlanmamış mesafenin üyelik derecesi grafiği

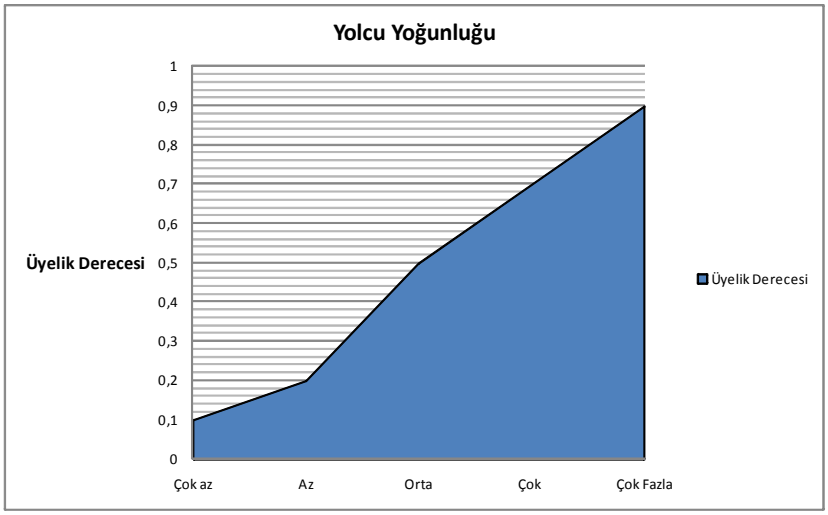
Sefer türü: Sürekli bir üyelik derecesine sahiptir. Çünkü mesai saati, gün içi, gece yarısı, haftasonu gibi dilsel değişkenler sürekli bir saat aralığı içerisinde tanımlıdırlar. Üyelik dereceleri genel olarak rastgele seçilmiş olmakla birlikte, belli bir mantığa oturması açısından sefer türünün yoğunluğu ile mantıksal orantı yapılarak bulanık değerler belirlenmiştir (Şekil 4.9).



Şekil 4.9: Sefer türü için üyelik derecesi grafiği

Yolcu yoğunluğu: Çok az, az, orta, fazla, çok fazla gibi 5 farklı dilsel değişken bulunmaktadır. Bulanık değerlerin matematiksel anlam kazanması için 0-1 arasındaki

bulanık değerlere dönüştürülmesi gerekmektedir. Buna göre Şekil 4.10'daki grafik tasarlanabilir. Çok az, az, orta, fazla, çok fazla gibi 5 farklı dilsel değişken için gerçek sayı değeri atanırsa sırasıyla 0-49, 50-99, 100-149, 150-199, 200-249 gibi aralıklar verilebilir. Grafikte sadelik açısından bu değerler gösterilmemiştir.



Şekil 4.10: Yolcu yoğunluğunun üyelik derecesi grafiği

4.1.2 BPA'nın VHDL kodlarının hazırlanması

BPA'nın VHDL kodlarının hazırlanması IF-THEN yapısına dayanmaktadır. Trenyolu BPA modelinin anlaşılmasını amaçlayan, temel olarak seçim-karar odaklı sade bir yapı hazırlanmıştır. Uygulamaya yönelik olarak sistemin karmaşıklığı artırılabilir. Bu da ilişkiler örgüsünün (giriş yerlerinin kombinasyonlarının) hem sayısı hem de içeriğinin düzenlenmesiyle gerçekleştirilebilir. FPGA üzerinde gerçekleştirilen devre bulanık mantık değerleriyle birlikte gelen

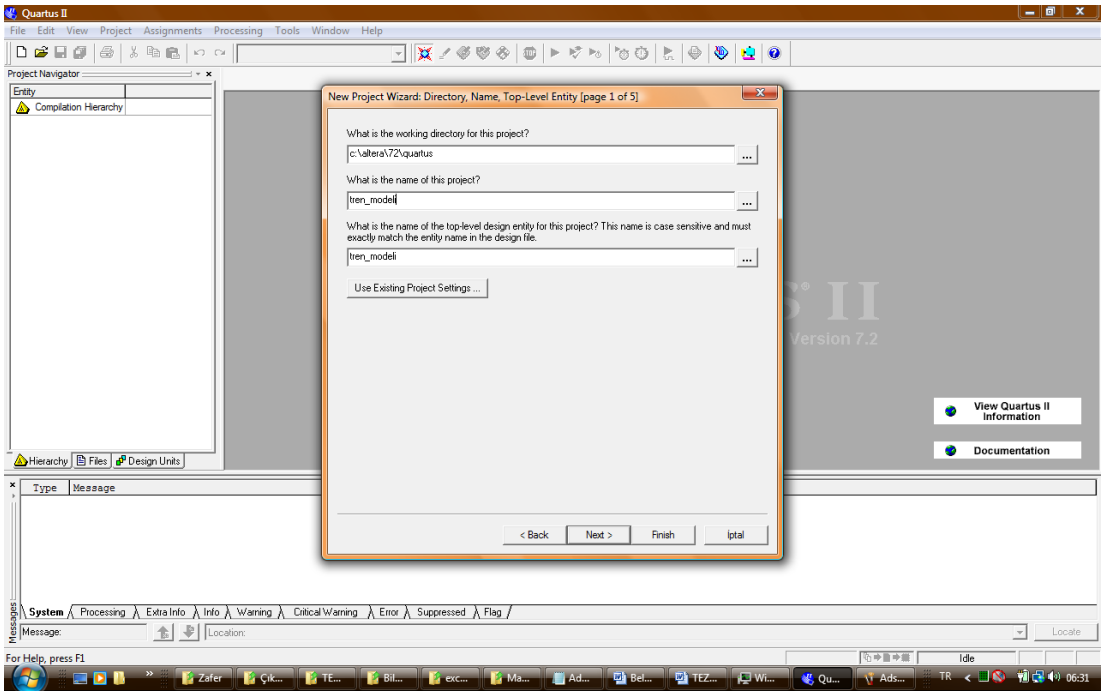
```
-- ============================================================

-- dosya adi:  tren_modeli.vhd     bpa=Bulanık Petri Ağı

-- Yazar:      Tamer TAŞ

-- Yıl:        2010

--   ============================================================--
-- Program 6 girişli 3 çıkışlı bir bulanık petri ağını ifade ediyor

-- Process() argümanları BPA'nın geçişleri olarak belirlenmiştir.
```

```
-- Böylece geçişlerdeki her bir aktif durum sistemin karar vermesini
-- tetikler
--  If-THEN yapısı bulanık mantıkta karar vermeyi sağlar.
--  std_logic_vector veri tipi 3 bitlik olarak ayarlanmıştır, çünkü
--  en az log2n
--  adet değişken bulunur. Burada 8 adet yer (place) bulunuyor.
-- IF yapısı sadece ardışıl yapı olan process içerisinde
-- çalışabilir.
-- ============================================================


LIBRARY ieee;
USE ieee.std_logic_1164.ALL;     -- standart kütüphane, çok değerli
                                 -- lojik sistem


ENTITY  tren_modeli IS
   PORT (p1              : IN  STD_LOGIC_VECTOR (2 downto 0);
         p2              : IN  STD_LOGIC_VECTOR (2 downto 0);
         p3              : IN  STD_LOGIC_VECTOR (2 downto 0);
         p4              : IN  STD_LOGIC_VECTOR (2 downto 0);
         p5              : IN  STD_LOGIC_VECTOR (2 downto 0);
         p6              : IN  STD_LOGIC_VECTOR (2 downto 0);

         p7              : OUT STD_LOGIC_VECTOR (2 downto 0);
         p8              : OUT STD_LOGIC_VECTOR (2 downto 0);
         p9              : OUT STD_LOGIC_VECTOR (2 downto 0)

        );
END tren_modeli;

-- ============================================================
-- ============================================================
```

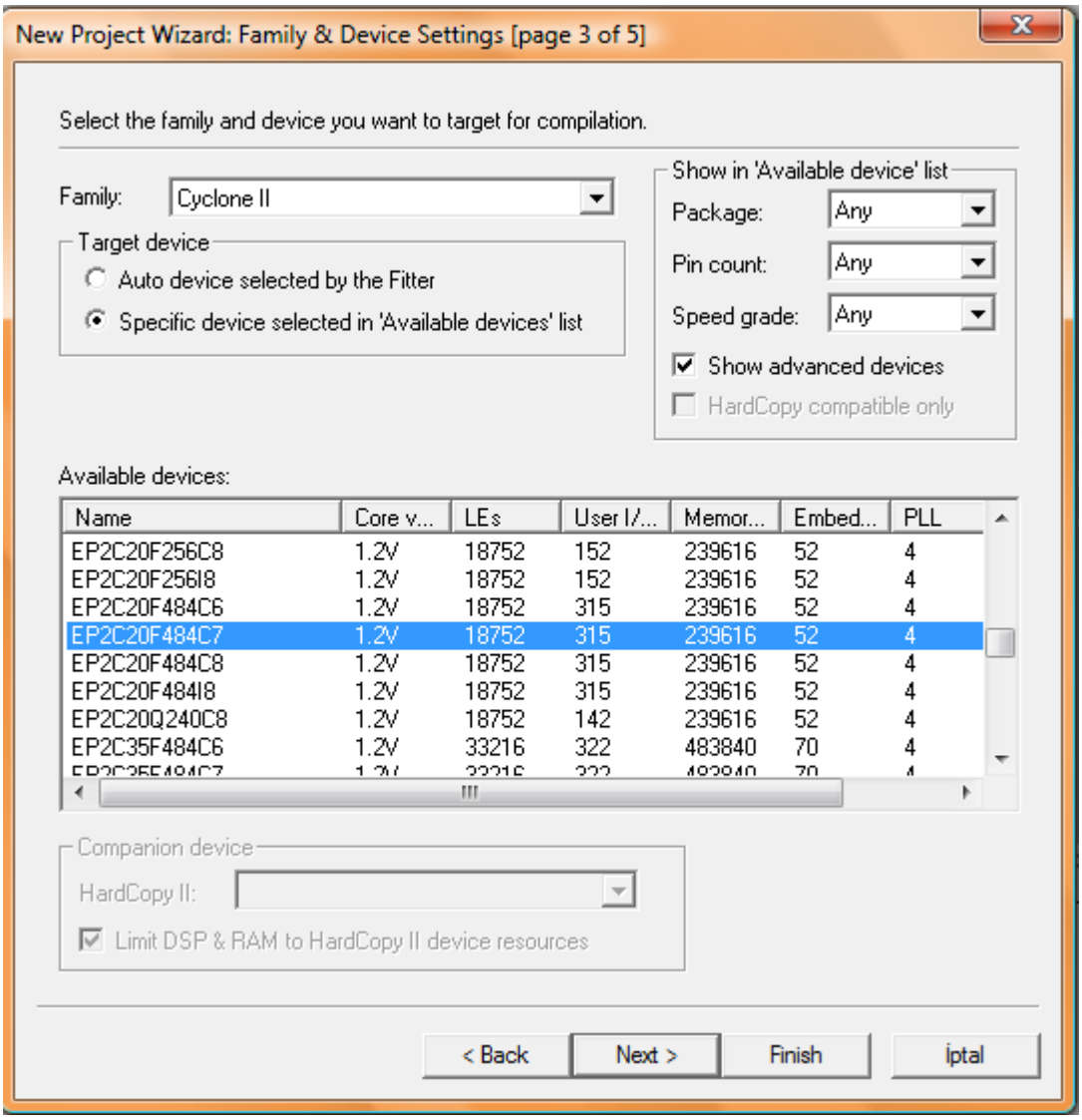
```
ARCHITECTURE davranis OF tren_modeli IS

    SIGNAL t1 : STD_LOGIC;
    SIGNAL t2 : STD_LOGIC;
    SIGNAL t3 : STD_LOGIC;
    SIGNAL t4 : STD_LOGIC;
    SIGNAL t5 : STD_LOGIC;
    SIGNAL t6 : STD_LOGIC;

-- ============================================================
BEGIN

karar_verme:PROCESS (p1, p2, p3, p4, p5)

BEGIN

  IF    (t1='Z') THEN

    p7 <= (NOT p1 OR NOT p5) NAND (p1 AND NOT p5);

  ELSIF  (t2='Z') THEN

    p7 <= (p1 AND p3 AND p5) NOR (NOT p1 AND p3 AND p5);

  ELSIF  (t3='Z') THEN

    p8 <= (p2 AND p3 AND p4) XOR (p2 OR p3 OR NOT p4);

  ELSIF  (t4='Z') THEN
```

```
    p8 <= (p1 AND p4) XNOR (NOT p1 OR p4);

  ELSIF  (t5='Z') THEN

    p8 <= (p1 AND p4 AND p5) NOR (NOT p4 AND p5 AND p1);

  ELSIF  (t6='Z') THEN

    p9 <= (p1 AND p4 AND p6) NAND (NOT p1 AND p4 AND NOT p6);



  ELSE          -- t1, t2, t3, t4, t5, t6 'Z' ye eşit değilse

    p7 <= "XXX";
    p8 <= "XXX";
    p9 <= "XXX";

  END IF;

END PROCESS karar_verme;

END davranis;
```

Altera Quartus-II programında yazılımın simülasyonu gerçeklenmiştir:

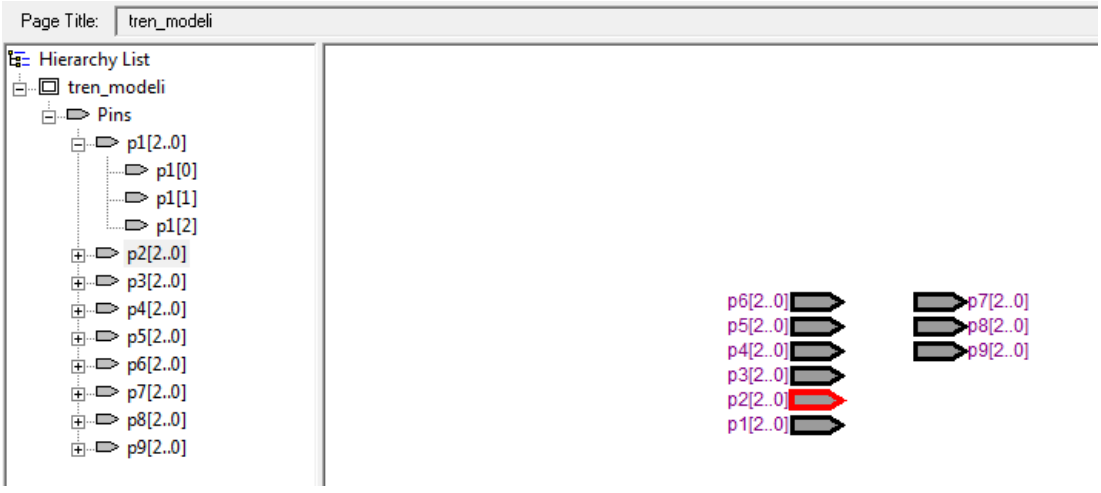


Şekil 4.11: Altera Quartus-II Programında Project Wizard Oluşturulması



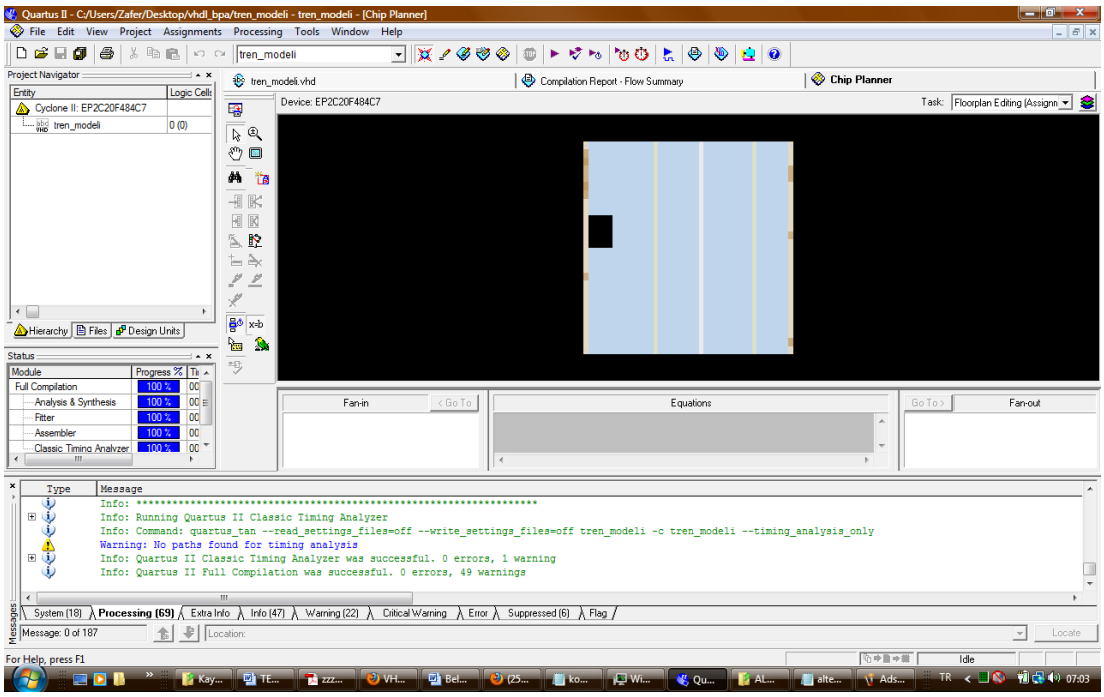
Şekil 4.12: Altera Cyclon-II Ailesinden ilgili modelin seçimi

Giriş ve çıkış yerlerinin pin dağılımı gösterimi ise şöyledir:



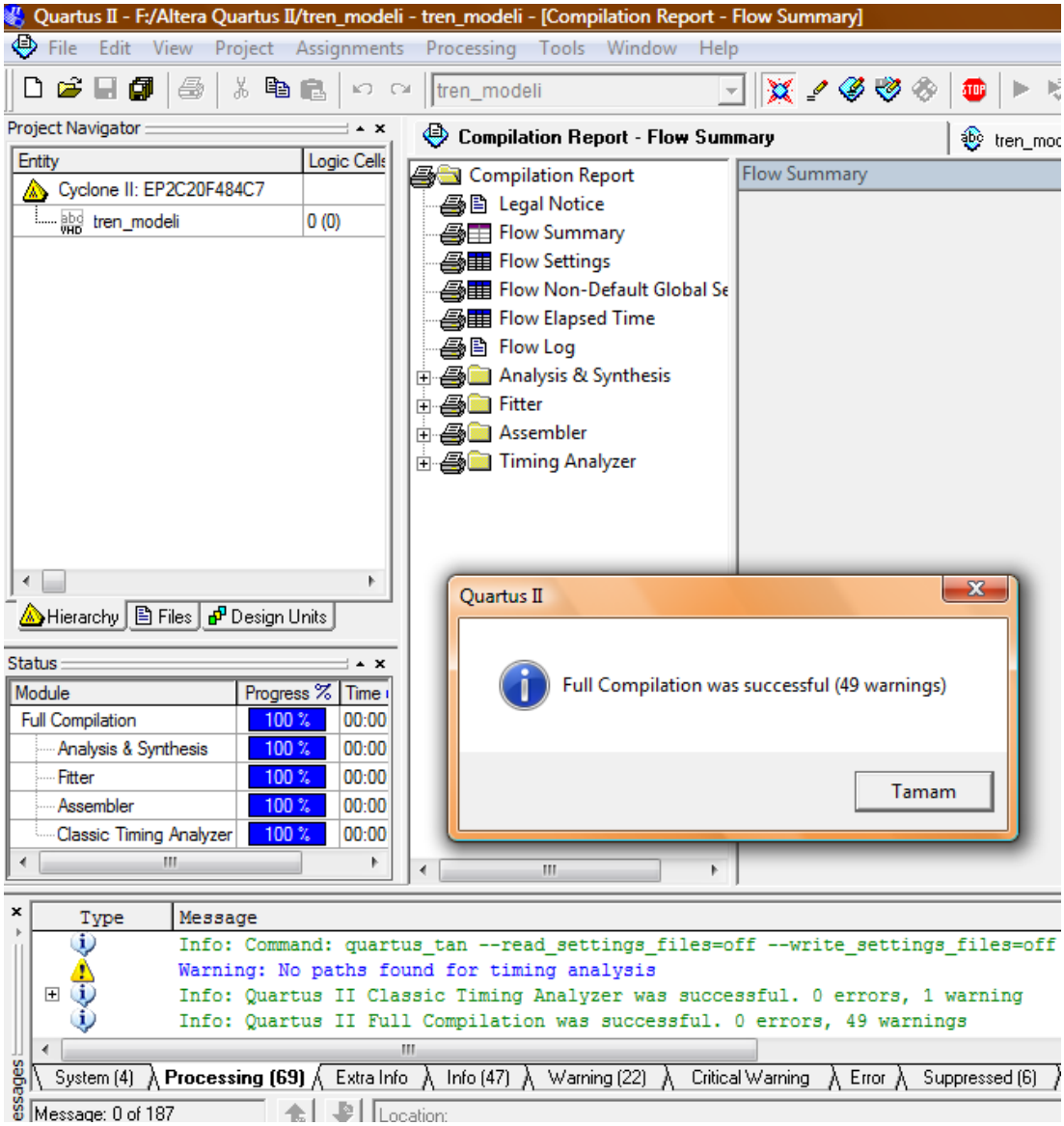
Şekil 4.13: Tren_modeli tasarımının RTL görünümü

Altera Cylon-II EP2C20F484C7 model kiti üzerindeki devrenin yerleşimi aşağıdaki gibidir:



Şekil 4.14: Quartus-II derleyicisinde tasarlanan çipin yer planı (floorplan)

Programda VHDL kodları başarıyla derlenmiştir:



Şekil 4.15: VHDL kodlarının derlenmesi sonucu ekran görüntüsü

5. SONUÇ VE YORUMLAR:

Bu tez çalışmasında Petri ağlarının bulanık mantık üzerine kurulu yapıya dönüşümü sonrasında nasıl davranacağı, bulanık mantık kurallarının bilgisayar ortamında nasıl tanımlanacağı ve bir demiryolu trafik sisteminin BPA'lar ile modellenmesi ve tasarlanan devrenin FPGA üzerine gerçeklenmesinin ayrıntıları anlatılmıştır. Genel olarak varılmak istenen hedef, Petri ağlarının dinamik yaklaşımının gerçek hayattaki sistemlere nasıl yansıdığının anlaşılmasıdır. Bu konunun anlaşılması için tez boyunca örnekler verilerek düşünce desteklenmiştir. Petri ağlarının etkili sistem modelleme avantaj ve yeteneklerinin yanı sıra matematiksel ve grafiksel anlatımının kolaylığı dikkat çekici olmuştur. Temel karakteristik özelliklerine özet olarak değinilerek dayandığı matematiksel bağıntılar açıklanmıştır. Ayrık sistemlerin ve kontrolör yapılarının tasarımı için kullanılabilecek güçlü yöntemlerden biri olduğu söylenebilir.

Bulanık mantık teorisi ise ana hatlarıyla incelenmiş ve Boole cebri ile farkları irdelenmiştir. Kullanım alanlarıyla ilgili örnekler sıralanmış ve 1960'lı yıllardan günümüze kadar bulanık mantık teorisine yönelik uygulamaların gelişimi gösterilmeye çalışılmıştır. Bulanık mantık teorisinde dile getirilen kuralların Petri ağlarına uygulanmasıyla birlikte BPA denilen yapıların ortaya çıkması ve lojik devre dönüşümünden bahsedilmiştir.

Tez sonunda, Petri ağlarının durum geçiş diyagramlarına benzeyen yapısı fakat daha dinamik ve sınırsızlık özelliği göstermesi önemli bir nitelik olarak belirlenmiştir.

BPA'lar ile bir demiryolu trafik sisteminin modeli gösterilmiş, sayısal devre yapısına nasıl dönüştürülebileceği algoritmik olarak açıklanmıştır.

Bulanık mantık ile Petri ağlarının senteziyle tasarlanan Trenyolu BPA modeli kurgusu ve kapsamı açısından benzer uygulamalara altyapı sağlayabilecek potansiyeli içeriğinde bulundurmaktadır.

Petri ağlarının VHDL kodlarıyla lojik devre elemanlarına çevrilmesi, daha üst seviye olarak RTL tasarımında üstlendiği rol FPGA üzerine gerçeklenmesiyle daha çarpıcı olarak ortaya çıkacaktır.

KAYNAKLAR

Banks, W., and Hayward, G., 2002. *Fuzzy Logic in Embedded Microcomputers and Control Systems*, 75 pages, Byte Craft Limited, Ontario, Canada

Barro, S., Bugarin, A.J., 1994, Fuzzy Reasoning Supported by Petri Nets, Department of Electronics and Computation, University of Santiago de Compostela, IEEE Transactions on Fuzzy Systems Vol.2, No.2, May 1994, Santiago de Compostela, Spain

Brauer, W., and Reisig, W., 2006. Adam Petri and "Petri Nets", translated from *Informatik-Spektrum*, Vol. 29, Nr. 5, pp 369-374, Springer-Verlag

Cassandras, C.G., and Lafortune, S., 2008. *Introduction to Discrete Event Systems,* 781 pages, Second Edition, Springer Science+Business Media, LLC, New York, USA

Cheng, Y.H., and Yang, L.A., 2009. *Expert Systems with Applications*, A Fuzzy Petri Nets approach for railway traffic control in case of abnormality: Evidence from Taiwan railway system, Department of Transportation and Communication Management Science, Published by Elsevier Ltd, National Cheng Kung University, No. 1, University Road, Tainan City, Taiwan

Chezalviel, B.P., Cardoso, J., 1996, Logic and Fuzzy Petri Nets, LCMI-UFSC, Campus University, Florianopolis, Brazil

Hellmann, M., (2010). Fuzzy Logic Introduction, Cedex, France, Retrieved March 12, 2010, http://epsilon.nought.de/tutorials/fuzzy/fuzzy.pdf

Jantzen, J., (2010). Tutorial on Fuzzy Logic, Retrieved March 12, 2010, from http://www.iau.dtu.dk/~jj/pubs/logic.pdf

Kluska, J., 2009. *Analytical Methods in Fuzzy Modeling and Control,* Studies in Fuzziness and Soft Computing Volume 241, 272 pages, Springer-Verlag Berlin Heidelberg

Kluska, J., and Hajduk, Z., 2004, Digital Implementation of Fuzzy Petri Net Based on Asynchronous Fuzzy RS Flip-Flop, Springer-Verlag Berlin Heidelberg

Knybel, J., Pavliska, V., 2005, Representation of Fuzzy IF-THEN rules by Petri Nets, University of Ostrava Institute for Research and Applications of Fuzzy Modeling, Research report No. 84, Czech Republic

Korpeoglu, B.B., and Yazici, A., 2006, A fuzzy Petri net model for intelligent databases, Available online 18 September 2006

Li, X., and Rosano, L.F, 2000. Adaptive fuzzy petri nets for dynamic knowledge representation and inference, *Expert Systems with Applications* 19 (2000) 235–241, Published by Elsevier Science Ltd., Centre for Instrumentation Research, National University of Mexico (UNAM), Mexico City, Mexico

Liu, K.F.R., 1998. A Fuzzy Petri Net-Based Expert System and Its Application to Damage Assessment of Bridges, 161 pages, *Phd Thesis***,** Department of Civil Engineering, National Central University, Chungli, Taiwan

Murata, T., 1989. Petri Nets: Properties, Analysis and Applications, Proceedings of the IEEE, Vol.77, No.4

Muscholl, A., (2010). Petri Nets, (tutorial, J. Esparza), Retrieved March 12, 2010, from http://www.labri.fr/perso/anca/

Pascal, J.D., Cardoso, J., Andreu, D., Valette, R., 1996, Fuzzy Petri nets and their application in CIME, French Research Programme for Artificial Intelligence, March 22, 1996, Toulouse Cedex, France

Pedrycz, W., 1999, Generalized fuzzy Petri nets as pattern classifiers, Department of Electrical and Computer Engineering, University of Alberta, 2 July 1999, Elsevier Science, Canada

Shim, H., and Lee, Y., (2010). Petri Nets - Properties, Analysis and Applications Synthesizing Petri Nets from State-Based Models, Retrieved March 12, 2010, from http://camars.kaist.ac.kr/~maeng/cs710/esd07/Petri%20Nets%20-%20Properties,%20Analysis%20and%20Applications.pdf

Smith, S.W., 1999, The Scientist and Engineer's Guide to Digital Signal Processing, Second Edition, pp 212-217, California Technical Publishing, San Diego, California

Venkateswaran, P.R., Bhat, J., 2006, Fuzzy Petri net Algorithm for Flexible Manufacturing Systems, Department of Instrumentation and Control Engineering, Department of Chemical Engineering, Manipal Institute of Technology, ACSE Journal, Volume (6), Issue (1), January 2006, Manipal, Karnataka

Vuong, P.T., Madni, A.M., and Vuong, J.B., 2006, VHDL Implementation For a Fuzzy Logic Controller, Automation Congress, 2006. WAC '06. World, Budapest, BEI Technol. Inc.

Yaralıoğlu, K., 2004. *"Uygulamada Karar Destek Yöntemleri":* Bulanık Mantık ve Karar Verme, İlkem Ofset, İzmir

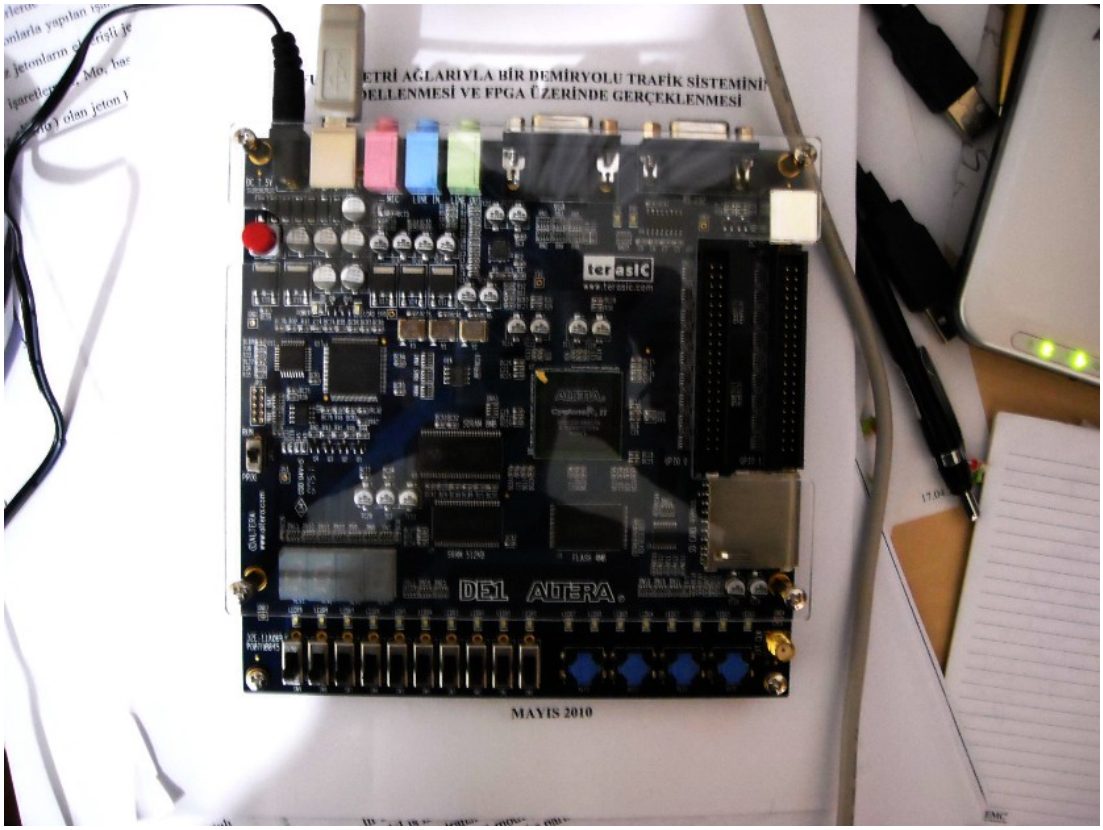
Zhong, L.X., 1998, Knowledge Representation and Reasoning Based on Fuzzy Petri Nets, Department of Computer Science and Information Engineering, Tianjin University of Science and Technology, China

Zurawski, R., and Zhou, M., 1994. Petri Nets and Industrial Applications: A Tutorial, *IEEE Transactions on Industrial Electronics,* Vol.41, No.5

EKLER

EK A.1 : Altera Cyclon II FPGA Kiti
EK B.1 : Altera Quartus-II Programının Kullanımı
EK C.1 : Tasarımla İlgili VHDL Kodları

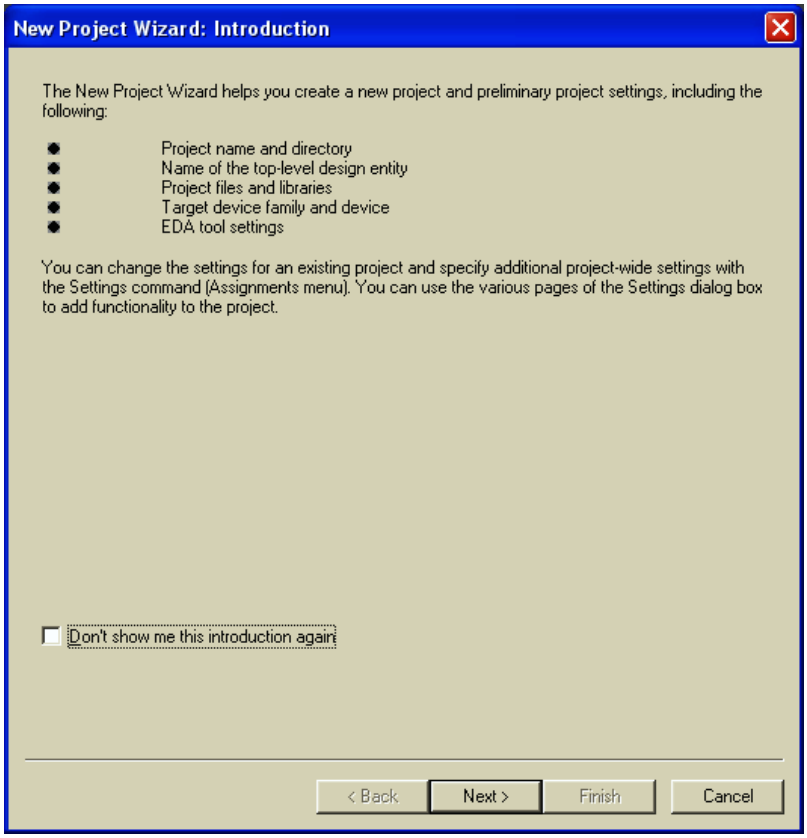
EK A.1: Altera Cyclon II FPGA Kiti



Şekil A.1: Altera Cyclon II FPGA board

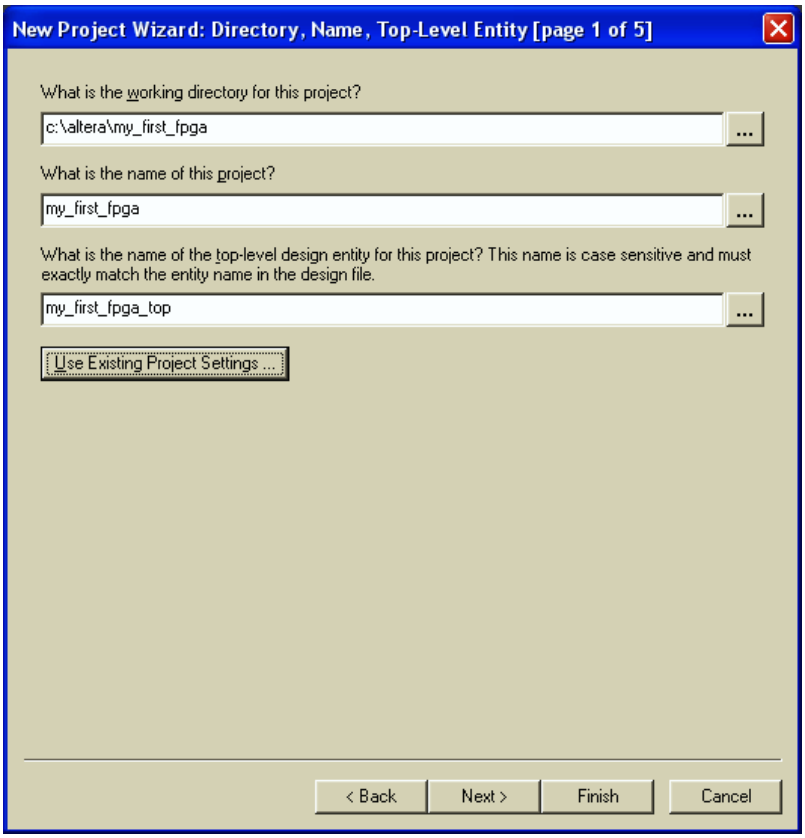
EK B.1: Altera Quartus-II Kullanımı

1. Quartus II yazılımına **File > New Project Wizard.** dan ulaşılabilir. Burada giriş penceresi açılır işlemlere başlamak için.



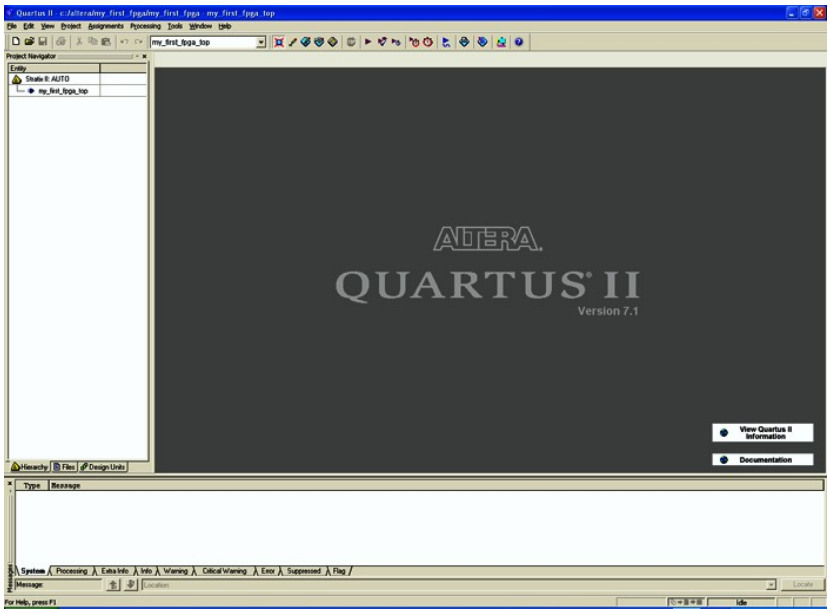
Şekil B.1.: Quartus II başlangıç penceresi

2. Sonra Next tuşuna tıklanır
3. Yapılacak proje için kayıt klasörü belirlenir
4. Dosya adı yazılır.

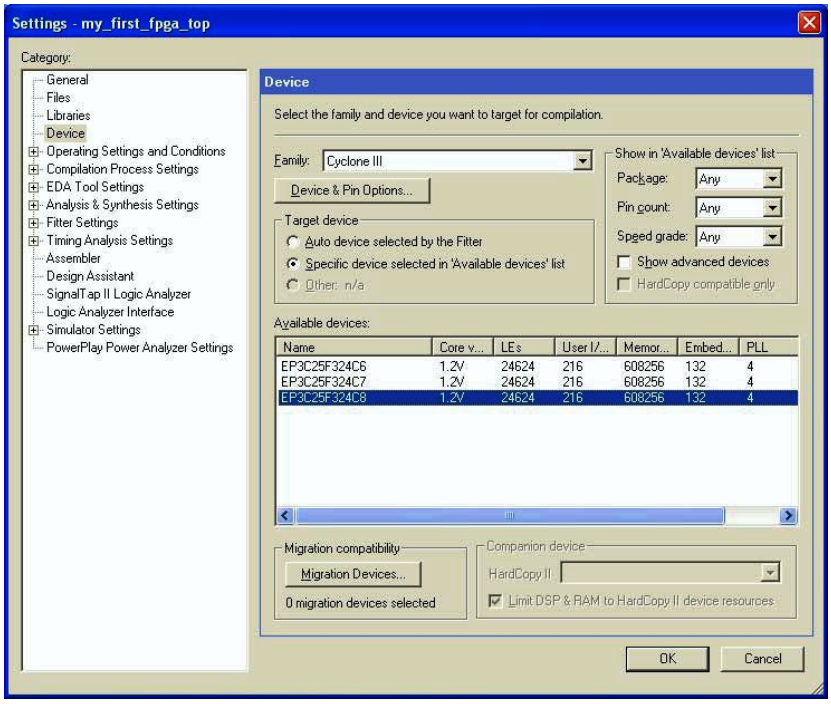


Şekil B.2: Dosya adı yazma ekranı

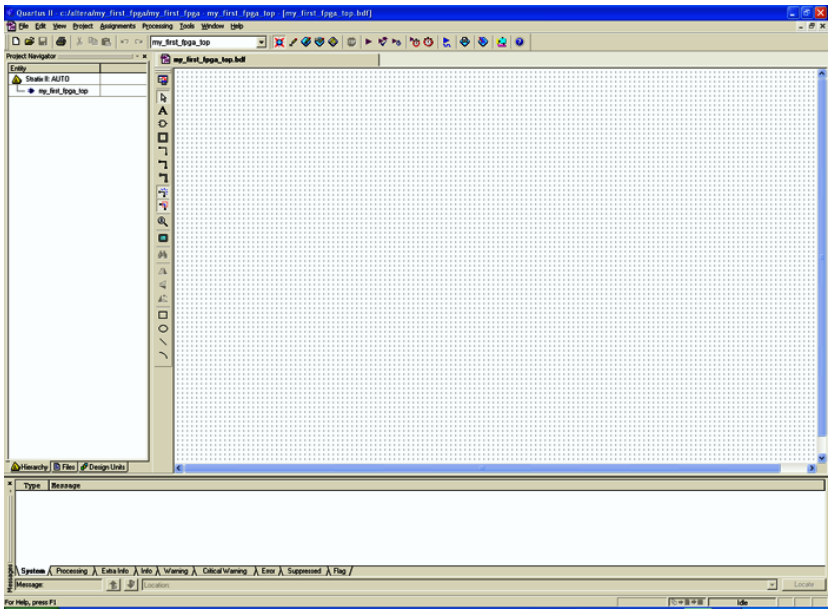
5. Sonra Finish tıklanır.



Şekil B.3: Altera Quartus II ana ekranı



Şekil B.4: FPGA seçimi yapılacak pencere



Şekil B.5: Quartus ana tasarım ekranı

6. Burada Dosya menüsünden Yeni seçilerek HDL kodu eklenir.

EK C.1: Örnek VHDL Kodları

```
----WITH/SELECT/WHEN yapısıyla veri secici, temel Petri agi donusum yapisi----

library ieee;
use ieee.std_logic_1164.all;

entity veri_secici IS
port ( a, b, c, d: IN std_logic;
sel: IN std_logic_vector (1 downto 0);
y: OUT std_logic);
end veri_secici;

architecture veri_secici2 of veri_secici is
begin
WITH sel SELECT
y <= a when "00",
 b when "01",
 c when "10",
 d when others ;
end veri_secici2;



--- D tipi flip floplar ile bir bitlik veri saklanir, Petri agi ile yerler temsil edilir---
---asenkron isaret rst ile saglanliyor,saat isaretinin yukselen kenarinda durum
degisiyor---

library ieee;
use ieee.std_logic_1164.all;

entity d_tipi_flipflop is
port  ( d, clk, rst: IN std_logic;
q: OUT std_logic);
end d_tipi_flipflop;

architecture calisma of d_tipi_flipflop is
begin
process (rst, clk)
begin
if (rst='1') then
q <= '0';
elsif (clk'EVENT and clk='1') then

q <= d;
end if;
end process;
end calisma;
```

---Burada ise Petri aglarinda baglantilarda kullanilabilen NAND girisli bir D-tipi flip flop kodlari yer aliyor, kutuphane tanimlanmayabilir, bit veri tipi destekleniyor---

```
library ieee;
use ieee.std_logic_1164.all;

entity nand_girisli_d is
port ( a, b, clk: IN bit;
q: OUT bit);
end nand_girisli_d;

architecture calisma2 of nand_girisli_d is
signal temp :bit;
begin
temp <= a nand b;
process (clk)
begin
if  (clk'EVENT and clk='1') then q<=temp;
end if;
end process;
end calisma2;
```

ÖZGEÇMİŞ

Ad Soyad: Tamer TAŞ

Doğum Yeri ve Tarihi: Balıkesir, 1984

Adres: Gülbahar Sk. Giray Apt. No:8 D:15 Kozyatağı Kadıköy / İSTANBUL

Lisans Üniversite: İstanbul Teknik Üniversitesi (İTÜ), 2007