**Zühal Erden**
Yardımcı Doçent
Endüstri Mühendisliği Bölümü
Atılım Üniversitesi
06836 ANKARA

**Aydan M. Erkmen**
Doçent
Elektrik-Elektronik Mühendisliği Bölümü
Orta Doğu Teknik Üniversitesi
06531 ANKARA

**Abdülkadir Erden**
Profesör
Makina Mühendisliği Bölümü
Orta Doğu Teknik Üniversitesi
06531 ANKARA

# Kavramsal Tasarım Otomasyonunda Petri Net ve Melez Otomat Sentezi

*Bu çalışmada, mühendislik tasarımındaki kavramsal tasarım aşamasının otomasyonu için Petri Net ve Melez Otomat (Hybrid Automata) sentezine dayanan PNDN adlı bir tasarım ağı modeli geliştirilmiştir. PNDN, tasarımı yapılacak ürünün mantıksal davranışını ürünün alt işlevleri arasındaki kesin ve belirsizlik içeren bilgi akışı ile iki tarzda modellemektedir. Bilgi akışındaki belirsizliklerin modellenmesinde Sezgisel Bulanık Önermeler yaklaşımı kullanılmıştır. Genel amaçlı ve işlevsel düzeyde geliştirilen model, tümleşik bir tasarım felsefesi olan mekatronik tasarımda uygulanmıştır. Bu makalede tasarım ağının yapısı ve kesin bilgi akışı modeli anlatılmaktadır.*

*Anahtar Kelimeler: Kavramsal Tasarım, Tasarım Otomasyonu, Petri Net, Melez Otomatlar, Mekatronik Tasarım, Tasarım Ağı.*

## GİRİŞ

Mühendislikte ürün tasarımının temel aşaması olan kavramsal tasarım [1,2], tanımlanan gereksinimi karşılayacak bir ürünün, ayrıntılı hesaplamalara girilmeden, yapısal ve işlevsel olarak kurgulanmasıdır. Tasarım alt süreçlerinden olan ayrıntılı tasarım ve belgelendirme ile üretim ve kalite kontrol aşamalarında belli ölçüde otomasyona geçilmiştir. Ancak tasarımın temeli olan kavramsal tasarım sürecinde, böyle yaygın ve yerleşmiş bir otomasyona yönelik kuramsal modellemeye literatürde rastlanmamaktadır. Kavramsal tasarım modelleri *tasarım süreç modelleri* ve *tasarım ürün modelleri* olarak iki ana grupta incelenebilir. Süreç modelleri, tasarım sürecinin alt süreçlerini, bunların sırasını, özelliklerini ve yöntemlerini irdeleyen modellerdir [1-8]. Ürün modelleri ise tasarlanan ürünün işlevsel ve yapısal özelliklerini ortaya koyan modellerdir [9-12]. Yayınlanmış ürün modelleri daha çok tasarımı tamamlanmış bir ürünün ayrıntılı tasarım aşamasındaki analizlerin yapılmasına yöneliktir. Modelleme yapısına baktığımızda genellikle fiziksel gerçeklemeye bağımlı olduğunu görüyoruz. Dolayısıyla kavramsal tasarım otomasyonu için işleve dayalı ve problemden bağımsız bir model gereği ortaya çıkmaktadır.

Kavramsal tasarımın en büyük özelliği insan zekası ve mühendislik yaratıcılığını birleştirme nitelikleridir. Bu nitelikler, özellikle tasarım seçeneklerinin oluşturulması aşamasının tümüyle otomasyonunu çok güç, hatta literatür bilgimize göre olanaksız kılmaktadır. Kavramsal tasarımın bir diğer aşaması, oluşturulan tasarım seçeneklerinin çeşitli ölçütlere göre değerlendirilmesi ve bu değerlendirme sonucunda geliştirilecek tasarıma karar verilmesidir. Kavramsal tasarımın bu ikinci aşamasının otomasyonu, tasarım sürecindeki zaman ve insan enerjisi kaybını azaltarak, insan zekasının seçenek oluşturmaya daha fazla yönlendirilmesini sağlayacaktır. Bu aşamadaki otomasyonun bir diğer yararı, ayrıntılı tasarım ve belgelendirme sürecine tam, doğru ve sistemli bilgi aktarımı ve sonuçta otomasyonda bütünlüğün sağlanmasıdır. Sonuç olarak, kavramsal tasarımda seçeneklerin değerlendirilmesi ve karar sürecinin otomasyonu için sistemli bir yapıya gereksinim doğmuştur. Böyle bir modelin, tasarlanacak ürüne özgü niteliklerden (kullanım alanı, yapacağı iş, kullanılacağı fiziksel ortam koşulları, geometrik kısıtlamalar v.b.) bağımsızlığı, bir başka deyişle genel amaçlı olması,

modellemesine dayanır. Tasarım ağının Petri Net ile modellemesinde, tasarlanan sistem bir melez sistem olarak düşünülmüştür. Bu durumda, sistemin dinamik davranışı, alt işlevlerin kendi içlerinde sürekli davranışı ve işlevler arası geçişin kesikli davranışından oluşmaktadır. Modelin ilk aşaması, tasarlanan sistemin melez otomat modelini oluşturmaktır. Yönlü bir çizge ile gösterilen melez otomat modelinde düğümler, sistemin sürekli davranışını ifade eden işlevleri, ayrıtlar ise bir durumdan diğerine geçmek için sağlanması gereken koşulları göstermektedir. Daha sonraki aşama, bu melez model kullanılarak, Petri Net modelinin (PNDN) otomatik olarak oluşturulmasıdır. Ana hatlarıyla, melez modeldeki düğümler Petri Net modelinde geçişleri, ayrıtlarla belirlenen koşullar geçişler için girdileri göstermekte, karar geçişleri ise söz konusu koşulların geçerli olup olmadığını gösteren karar aşamalarını ifade etmektedir. Analiz ve değerlendirmede kullanılan asıl model, Petri Net modelidir.

**Değişkenlerin Tanımı ve Formülasyonu**

Tasarım ağının oluşturulmasında ilk aşama, aşağıdaki elemanların matematiksel olarak tanımlanması ve formülasyonudur.

*(a) İşlevsel Durum Kümesi (FS)*

İşlevsel durum kümesi tasarlanan sistemin işlevsel ayrışımından elde edilen birinci düzey alt işlevlerinden oluşan bir küme olup, matematiksel olarak;

$FS = \{F_1, F_2, F_3, \ldots\ldots\ldots, F_N\}$

şeklinde ifade edilir. Burada N sistemin birinci düzeydeki alt işlev sayısını göstermektedir. İşlevsel durum kümesinde yer alan her $F_i$ {i = 1, 2, ….., N} sistemin bir işlevsel durumu olarak adlandırılır.

*(b) Sürekli Değişken Kümesi (CVS)*

Tasarımcı tarafından tanımlanacak olan sürekli değişkenler, sistemin sürekli dinamiğini ifade etmek için gerekli olan değişkenlerdir. Sistemin her bir işlevsel durumda, sürekli fonksiyonlarla anlatılan bir sürekli dinamiği olduğu varsayılmıştır. Dolayısıyla, sürekli değişkenler bu fonksiyonlarda kullanılan değişkenler olmaktadır. Sürekli değişkenler aşağıda tanımlanan bir küme ile gösterilir :

$CVS = \{V_j^C \mid V_j^C \ (j = 1, 2, \ldots.., N_C)$ : sistemin sürekli değişkenleri}

Yukarıdaki ifadede $N_C$ sürekli değişken sayısını göstermektedir.

*(c) İşlevsel Durum Matrisi (FSM)*

Sistemin her işlevsel durumu ($F_i$ , i = 1, 2, 3, …., N) için tanımlanması gereken sürekli fonksiyonlar işlevsel durum matrisi (FSM) adı verilen bir matrisle gösterilir. Bir işlevsel durum için tanımlanması gereken maksimum sürekli fonksiyon sayısı L ve $F_i$ işlevsel durumu için tanımlanan sürekli fonksiyon sayısı $L_i$ ile gösterilirse, FSM bir (NXL) matris olarak aşağıdaki gibi tanımlanır:

$$FSM = \begin{bmatrix} f_{is}(CVS), & s \le L_i \\ 0, & s \rangle L_i \end{bmatrix}$$

burada,

s = 1, 2, …, L , i =1, 2, ……, N

ve $f_{is}$ (CVS), $F_i$ işlevsel durumu için tanımlanan sürekli fonksiyonları göstermektedir.

*(d) Kesikli Değişken Kümesi (DVS)*

Kesikli Değişken Kümesi (DVS) elemanlarından her biri sistemin dinamik davranışını anlık olarak etkileyen ve kesikli geçişlere neden olan kontrol amaçlı değişkenlerdir. Bu çalışmada kesikli değişken değerlerinin ''ikili'' (binary) olduğu varsayılmıştır. Kesikli değişken sayısı $N_D$ ile gösterilirse, DVS aşağıdaki gibi tanımlanır :

$DVS = \{V_k^D \mid V_k^D \ (k = 1, 2, \ldots.., N_D)$ : sistemin kesikli değişkenleri}

*(e) Kesikli Değişken Değerleri*

Kesikli değişken değerleri aşağıdaki küme ile gösterilir :

$$DVS_{ins} = \left\{V_k^D(m) : k = 1,2,\ldots..,N_D; m = 1,2\right\}$$

burada, $V_k^D(m) = \{0,1\}_k$, elemanlarından her biri $V_k^D$ kesikli değişkeninin alacağı ''ikili'' değerleri gösterir. Dolayısıyla $DVS_{ins}$ aşağıdaki gibi tanımlanabilir :

$$DVS_{ins} = \left\{\{0,1\}_1, \{0,1\}_2, \ldots\ldots\ldots\ldots, \{0,1\}_{N_D}\right\}$$

*(f) İşlevsel Durumlar İçin Değişmez Koşullar*

İşlevsel durumlar için değişmez koşullar, sistemin söz konusu işlevsel durumla ifade edilen sürekli dinamiğini koruması için geçerli olması gereken koşulları göstermektedir. Bu koşullar modelde (CVS ∪ DVS) üzerinde tanımlanan ''predicate'' lerle ifade edilir.

*(g) Sürekli Değişken Değer Aralıkları*

Sürekli değişkenler için tanımlanması gereken değer aralıkları, bu değişkenlerin yukarıda tanımlanan değişmez koşullara göre eşiklenmesiyle elde edilir ve aşağıdaki gibi ifade edilir :

$$CVS_{ins} = \left\{V_j^c(m)\right\}$$

$V_j^C(m)$ aşağıdaki koşulu sağlayan bir küme olarak tanımlanmıştır :

$\exists$ j, $\forall$m, $\exists$ $F_i \in FS$, $\forall$ $x \in V_j^C(m)$ : INCLUDED (x, Inv($F_i$))

burada, m = 1, 2, ....., $S_j$ ve $S_j$, $V_j^C$ sürekli değişkeni için tanımlanan değer aralıklarının sayısıdır. INCLUDED (x, Inv($F_i$) ) ifadesi, $F_i$ işlevsel durumu için x değişkeni ile ilgili değişmez koşul(lar) olduğunu göstermektedir.

*(h) Kesikli Durum Geçiş Matrisi (DSTM)*

DSTM sistemin dinamik davranışında görülen kesikli geçişleri ifade eden (NXN) matris olarak aşağıdaki gibi tanımlanmıştır :

$$DSTM = \left[ T_{ij} \middle| \begin{matrix} T_{ij} = 1 \ F_i \text{ den } F_j \text{ ye kesikli geçiş var ise} \\ T_{ij} = 0 \ F_i \text{ den } F_j \text{ ye kesikli geçiş yok ise} \end{matrix} \right]$$

Bu bölümde anlatılan formülasyon tamamlandıktan sonra ikinci aşama sistemin melez otomat modelinin otomatik olarak oluşturulmasıdır.

**Melez Otomat Modelinin Oluşturulması**

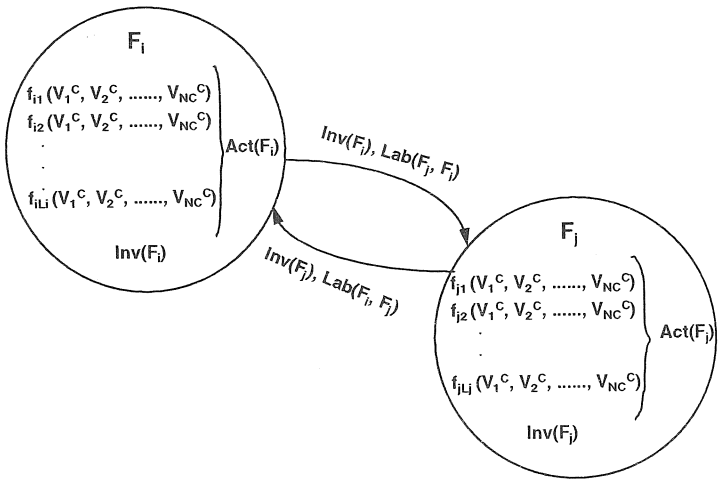
Tasarlanan sistemin ana işlevini F ile gösterirsek, sistemin melez otomat modeli ($H_F$) aşağıdaki gibi tanımlanır:

$$H_F = (Loc, Var, Lab, Act, Inv, Edg)$$

Aşağıda verilen açıklama ve tanımlar, kaynak [19]'dan bu çalışmadaki yaklaşım doğrultusunda uyarlanmış ve melez otomat modeli bu değiştirilmiş tanımlar üzerine kurulmuştur:

1) *Loc: Konumlar* (Locations) sonlu düğümler kümesi olup geliştirilen modelde işlevsel durumları ($F_i$) göstermektedir.
   $Loc \equiv FS$
2) *Var*, sonlu bir gerçek değerli değişkenler kümesidir ve modelde sürekli ve kesikli değişkenlerin birleşiminden oluşmaktadır.
   $Var \equiv CVS \cup DVS$
3) *Lab* sonlu bir *eş zaman etiketleri* kümesidir. Bu etiketler iki ve/veya daha fazla melez otomatın eş zamanlaması için kullanılmaktadır. Bu durumda ortak etiketlerle gösterilen kesikli geçişler aynı zamanda meydana gelmektedir.
4) *Act* her konum için tanımlanan sürekli fonksiyonları gösteren etkinlikleri ifade eden fonksiyonlar kümesi olup aşağıdaki gibi tanımlanır :
   $Act(F_i) = \{f_{is}(CVS), s = 1, 2, 3, ...., L_i\}$, $Act = \{ Act(F_i), i = 1, 2, 3, ....., N \}$
5) *Inv* her bir konum için değişmez koşulları ifade eden ve (CVS ∪ DVS) üzerinde tanımlanan bir ''predicate'' ile gösterilen bir fonksiyondur.
6) *Edg* sonlu bir geçişler kümesi olup, DSTM'de $T_{ij}$ = 1 ise, buna karşılık gelen geçiş ($e_{ij}$) aşağıdaki gibi tanımlanır :
   $e_{ij} : (F_i, \mu, F_j)$
   bu ifadede, $F_i$ kaynak konumu, $F_j$ hedef konumu ve μ geçiş koşulu olarak tanımlanır. Bu çalışmada, geçiş koşulunun (Inv ($F_j$)) olduğu varsayılmıştır, bu durumda;
   $\mu_{ij}$ = Inv ($F_j$) olarak gösterilir.

Yukarıdaki yapı doğrultusunda, iki alt işlevli bir tasarım için melez otomat modeli Şekil 1'de gösterilmiştir.




Şekil 1. İki alt işlevli bir tasarımın melez otomat modeli

## Petri Net Modelinin Oluşturulması

Modelin son aşaması melez model kullanılarak sistemin Petri Net modelinin (PNDN) otomatik olarak oluşturulmasıdır. PNDN aşağıdaki gibi ifade edilir:

$PNDN = (P, T, I, O)$

PNDN formülasyonu aşağıda verilmiştir :

*1)* $P = P_1 \cup P_2$

burada, $P_1 \subseteq (CVS_{ins} \cup DVS_{ins})$ ve $P_2 \subseteq (CVS \cup DVS)$ olarak tanımlanır ve aşağıdaki mantıksal ifadeleri sağlaması gerekir:

*$P_1$ için ;*

$\forall x \in P_1, \exists F_i \in Loc, \exists V_j^C \in CVS, \exists V_k^D \in DVS :$
$[ x \in V_j^C (m) \vee x \in V_k^D (m) ] \wedge INCLUDED (x, Inv(F_i))$

Yukarıdaki ifadenin açıklaması şöyle yapılır : $P_1$ kümesinin elemanları, kesikli ve/veya sürekli değişkenlerin alabileceği değerleri ifade eden ortamlar olup, bu değerler işlevsel durumlar için tanımlanan değişmez koşullarda yer alan değerlerdir.

*$P_2$ için ;*

$\exists x \in P_1, \forall y \in P_2, \exists k, \exists j, \exists m :$
$[x \in V_j^C (m) \rightarrow y = V_j^C ] \vee [ x \in V_k^D (m) \rightarrow y = V_k^D ]$

Bu ifadenin anlamı şöyle açıklanır : $P_2$ kümesinde bulunan her ortam için ($y \in P_2$); eğer $P_1$ kümesinde sürekli değişkenlerden birinin değer aralığında bulunan bir eleman ($x_s \in P_1$) varsa, bu durumda y bu sürekli değişken $x_s$ 'i gösterir veya eğer $P_1$ kümesinde kesikli değişkenlerden birinin değeri olan bir eleman ($x_k \in P_1$) varsa, bu durumda y bu kesikli değişken $x_k$'yi gösterir.

*2)* $T = Loc \cup DF$

burada,

$DF = \{df_1, df_2, df_3, ....., df_g | g = P_2$ *kümesindeki eleman sayısı}*

sonlu bir Karar İşlevleri (Decision Functions) kümesi olup, sistemin gelen bilgileri işleyerek değişken değerlerine karar verme işlevini ifade eder. Karar İşlevleri PNDN'de *anahtarlayıcı* (switch) [28] olarak gösterilir.

*3)* $I : P X T \rightarrow \{0, 1\}$

ortamlardan geçişlere tanımlanan ''ikili'' bir eşleme olup *girdi eşlemesi* olarak adlandırılır ve $I(p, t)$ ile gösterilir. Geliştirilen tasarım ağında iki çeşit girdi eşlemesi tanımlanmıştır :

a) $\forall y \in P_2, \exists z \in DF : I(y, z)$

b) $\exists x \in P_1, \exists F_i \in Loc : INCLUDED (x, Inv (F_i)) \rightarrow I (x, F_i)$

*4)* $O : T X P \rightarrow \{0, 1\}$

geçişlerden ortamlara tanımlanan ''ikili'' bir eşleme olup *çıktı eşlemesi* olarak adlandırılır ve $O(t, p) = I^{-1}(p, t)$ ile gösterilir. Tasarım ağında iki çeşit çıktı eşlemesi tanımlanmıştır :

a)$\forall z \in DF, \exists x \in P_1, \exists y \in P_2 :$
$[ I(y, z) \wedge x \in V_j^C (m) \wedge y = V_j^C ] \vee [ I(y, z) \wedge x \in V_k^D (m) \wedge y = V_k^D ] \rightarrow O(z,x)$

b)$\forall F_i \in Loc, \forall x \in P_2 : \neg ( F_i = STOP ) \rightarrow O (F_i, x 5)$

*Başlangıç Şablonu ($M_0$)* başlangıç durumunda, $P_2$ kümesinde bulunan elemanlarda, (•) işareti ile gösterilen belirteçlerden kaç tane bulunduğunu matematiksel olarak ifade eden g boyutlu bir vektördür. Bir başka deyişle başlangıç şablonu sistemin çalışmaya başladığı andaki bilgi durumunu gösterir ve aşağıdaki gibi tanımlanır :

$$M_0 = \begin{bmatrix} M_0(p_1) \\ M_0(p_2) \\ . \\ . \\ M_0(p_g) \end{bmatrix}, \quad M_o(p_r) = \begin{cases} 1 \text{ belirteç var ise} \\ 0 \text{ belirteç yok ise} \end{cases}$$

$$r = 1, 2, ....., g$$

## PNDN Tasarım Ağının Özellikleri

Bu bölümde genel Petri Net kuramındaki önemli özelliklerin kavramsal tasarımın yapısı doğrultusunda PNDN mimarisinde kullanılan yorumları ve PNDN 'de yer alan çeşitli tanımlar verilmiştir.

PNDN (P, T, I, O) şeklinde tanımlanan bir dörtlü olup burada işlevsel geçiş (FS) sayısı *N*, değişkenleri gösteren ortamların toplam sayısı *g* ve değişken değerlerini gösteren ortamların ($p_r \in P_2$, (r = 1, 2, ....., g)) sayısı da $n_r$ ile ifade edilmektedir.

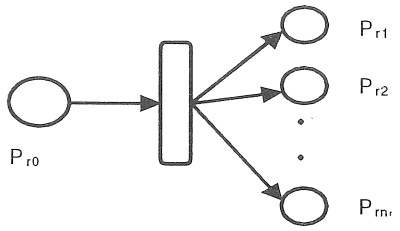
PNDN'de herhangi bir şablon g-boyutlu bir sütun vektörü olup,aşağıdaki gibi gösterilmektedir;

$$M = \begin{bmatrix} M(p_1) \\ M(p_2) \\ . \\ . \\ . \\ M(p_g) \end{bmatrix} \qquad (1)$$

burada, $M(p_r) \{ p_r \in P_2 \wedge r = 1,2,...., g \}$, $(n_r + 1)$ boyutlu bir sütun vektörüdür. $M(p_r)$, değişkenleri ($p_{r0}$) ve bunların alabileceği değerleri ($p_{rj}$, j =1, 2,....., $n_r$) bir bütün olarak yapılandıran ortamlardaki

belirteç sayısını ifade etmektedir. Burada sözü edilen gösterim Şekil 2'de grafiksel olarak gösterilmiştir. PNDN'de, kesikli değişkenlerin ($V_k^D$) 0 ve 1 şeklinde ikili değerler aldığı varsayılmıştır. Sürekli değişkenlerin ($V_j^C$) aldığı değerlerin sayısı ise $S_j$ ile gösterilmektedir. Buna göre $n_r$ aşağıdaki gibi tanımlanır:

$$n_r = \begin{cases} 2 & \text{eğer } p_r \text{ kesikli değeğişken } (V_k^D) \text{ gösteriyorsa} \\ S_j & \text{eğer } p_r \text{ sürekli değeğişken } (V_j^C) \text{ gösteriyorsa} \end{cases}$$




Şekil 2. Değişkenlerin ($p_{r0}$) ve değişken değerlerinin ($p_{rj}$) karar işlevleri ile gösterimi

Bu durumda, $M(p_r)$ aşağıdaki gibi ayrıntılandırılır:

$$M(p_r) = \begin{bmatrix} m(p_{r0}) \\ m(p_{r1}) \\ . \\ . \\ m(p_{rn_r}) \end{bmatrix} \qquad (2)$$

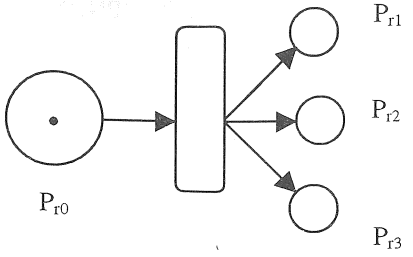
burada,

1) $m(p_{r0})$ kesikli ($V_k^D$) veya sürekli ($V_j^C$) bir değişkeni ifade eden ve ($p_{r0}$) ile gösterilen ortamlardaki belirteç sayısı,
2) $m(p_{rj}, j=1,2,3,....,n_r)$ ise ($p_{r0}$) ortamıyla gösterilen bir değişkenin alabileceği değerleri ifade eden ortamlardaki ($p_{rj}$) belirteç sayısı

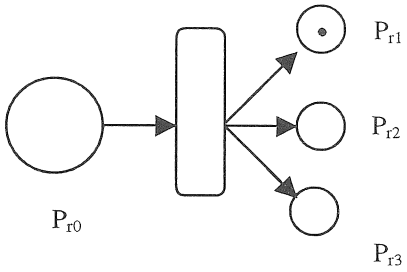
olarak tanımlanır. M şablonu en ayrıntılı şekliyle aşağıdaki biçimde gösterilir:

$$M = \left[ \begin{bmatrix} m(p_{10}) \\ m(p_{11}) \\ . \\ . \\ m(p_{1n_1}) \end{bmatrix} \begin{bmatrix} m(p_{20}) \\ m(p_{21}) \\ . \\ . \\ m(p_{2n_2}) \end{bmatrix} \cdots \begin{bmatrix} m(p_{g0}) \\ m(p_{g1}) \\ . \\ . \\ m(p_{gn}) \end{bmatrix} \right]^T \qquad (3)$$

PNDN'de, belirteçlerin bulunduğu ortamlara göre iki çeşit şablon tanımlanmıştır. *Değişken şablonunda* ($M^V$) belirteçler sadece değişkenleri gösteren ortamlara ($p_{r0}$) yerleştirilir (Şekil 3).




Şekil 3. PNDN'de değişken şablonu ($M^V$)




Şekil 4. PNDN'de değer şablonu ($M^I$)

$M^V$ aşağıdaki vektörle tanımlanmıştır :

$$M^V = \left[ \begin{bmatrix} m(p_{10}) \\ 0 \\ 0 \\ . \\ . \\ 0 \end{bmatrix} \begin{bmatrix} m(p_{20}) \\ 0 \\ 0 \\ . \\ . \\ 0 \end{bmatrix} \cdots \begin{bmatrix} m(p_{g0}) \\ 0 \\ 0 \\ . \\ . \\ 0 \end{bmatrix} \right]^T \qquad (4)$$

$M^V$ henüz sistem değişkenleriyle ilgili herhangi bir işlem yapılmadığını gösteren bir durumu ifade etmektedir. *Değişken şablonunda* ($M^I$) ise belirteçler Şekil 4'de gösterildiği gibi, yalnızca değişken değerlerini ifade eden ortamlara ($p_{rj}$, $j \neq 0$) yerleştirilir. $M^I$ matematiksel olarak aşağıdaki gibi tanımlanır :

$$M^I = \left[ \begin{bmatrix} 0 \\ m(p_{11}) \\ . \\ . \\ m(p_{1n_1}) \end{bmatrix} \begin{bmatrix} 0 \\ m(p_{21}) \\ . \\ . \\ m(p_{2n_2}) \end{bmatrix} \cdots \begin{bmatrix} 0 \\ m(p_{g1}) \\ . \\ . \\ m(p_{gn}) \end{bmatrix} \right] \qquad (5)$$

$M^I$ sistem değişkenlerine ait değerlerin karar işlevleri (DF) yardımıyla elde edildiği durumu göstermektedir. PNDN'de $M^V$ ve $M^I$ olarak iki çeşit şablon tanımı değişkenler ve değişken değerlerinin geçerliliği ayrımının yapılması gereğinden ortaya çıkmıştır.

### PNDN Tasarım Ağı'nda Ulaşılabilirlik, Eşzamanlılık, Belirsizlik ve Canlılık

PNDN tasarım ağında $M_0^V$ ile ifade edilen bir başlangıç şablonunu $M_n^{V,I}$ şeklindeki herhangi bir şablona dönüştüren bir tetikleme süreci varsa, $M_n^{V,I}$, $M_0^V$ den ***ulaşılabilir*** denir. PNDN'de yer alan değişken şablonu ve değer şablonu ayrımı, herhangi bir tetikleme sürecinde bu iki çeşit şablonun belirteç akışını ardışık olarak ifade etmesini sağlar. PNDN'de $M_0^V$ den ulaşılan olası tüm şablonların oluşturduğu kümeye ***PNDN'nin ulaşılabilirlik kümesi*** denir ve $R_{PNDN}(M_0^V)$ ile gösterilir.

PNDN tasarım ağında belirteç akışının mantıksal kurgusu aşağıdaki varsayımlara dayanmaktadır:

**Varsayım 1:** PNDN'de gerçeklenen tüm karar işlevleri ($df_r$, r=1,2,3,....,g) ulaşılabilir bir değişken şablonu $M_n^V \in R_{PNDN}(M_0^V)$ ile tetiklenebilir.

**Varsayım 2:** PNDN'de gerçeklenen tüm $F_i$ geçişlerinden yalnız biri ulaşılabilir bir değer şablonu $M_n^I \in R_{PNDN}(M_0^V)$ ile tetiklenebilir.

**Varsayım 3:** Herhangi bir $F_i$ (i=1,2,3,....,N) geçişi ulaşılabilir bir değer şablonu $M_n^I \in R_{PNDN}(M_0^V)$ ile tetiklendiğinde, $p_{rj}$ ($p_r \in P$) ortamlarındaki kullanılmayan belirteçler sıfırlanır.

PNDN'de karar işlevleri ve işlevsel durumlar olmak üzere iki çeşit geçiş tanımlanmıştır. İşlevsel geçişler "işlenmiş", karar işlevleri ise "işlenecek" bilgiyi kullanırlar. Dolayısıyla ortaya çıkan mantıksal davranış "eşzamanlı" olmaktan çok "ardışık" bir davranıştır. Bunun nedeni tasarlanan sistemin en soyut düzeydeki sürekli davranışının ardışık olmasıdır.

PNDN'de belirsizlik ağ kuramındaki belirsizlik tanımından farklı bir anlam yüklenmiştir. PNDN'de, gerçeklenen tüm karar işlevleri $M_n^V \in R_{PNDN}(M_0^V)$ şablonu ile aynı anda tetiklenebilir. Buna karşılık, gerçeklenen tüm $F_i$ geçişlerinden yalnız biri $M_n^I \in R_{PNDN}(M_0^V)$ şablonu ile tetiklenebilir.

Dolayısıyla, farklı geçişlerin tetiklenmesi için gerekli koşullar belirlidir. PNDN'de belirsizlik daha çok sistem içinde dolaşan bilgiler veya geçişlerde meydana gelebilecek belirsizliklerden kaynaklanmaktadır.

PNDN'de bir ve yalnız bir işlevsel durum geçişi ($F_i$) herhangi bir ulaşılabilir değer şablonu $M^I \in R_{PNDN}(M_0^V)$ ile tetikleniyorsa, o zaman PNDN ***canlı*** denir. PNDN'deki bu canlılık koşulu aynı zamanda ***tıkanmadan arınmış işlemi (deadlock-free operation)*** de garanti etmektedir.

### PNDN'DE KESİN BELİRTEÇ AKIŞI

PNDN için kesin belirteç akışı aşağıda tanımlanmıştır:

1. DF Karar işlevlerini gösteren herhangi bir $df_r$ geçişi için; eğer $df_r$'ın (r=1,2,3,....,g) tüm girdi ortamlarında belirteç var ve $df_r$'ın hiçbir çıktı ortamında belirteç yok ise, $M^V \in R_{PNDN}(M_0^V)$ için $df_r$ geçişi ***$M^V$-gerçeklenebilir*** denir. Gerçeklenen tüm $df_r$ geçişleri, girdi ortamlarının her birinden birer belirteç alınıp çıktı ortamlarının yalnız birine bir belirteç yerleştirilerek, aynı anda ***tetiklenir***. (1. varsayıma dayanarak). Bu durumda elde edilen yeni şablon, bir değer şablonu ($M^I$) olup $M^I$ ($df_r$) ile gösterilir.
2. İşlevsel durumları (FS) gösteren herhangi bir geçiş ($F_i$) için; eğer $F_i$'nin tüm girdi ortamlarında belirteç var ve hiçbir çıktı ortamında belirteç yok ise, $M^I \in R_{PNDN}(M_0^V)$ için $F_i$ geçişi ***$M^I$-gerçeklenebilir***. denir. Gerçeklenen bir $F_i$ geçişi, girdi ortamlarının herbirinden birer belirteç alınıp çıktı ortamlarının her birine birer belirteç yerleştirilerek ***tetiklenir***. Bu durumda elde edilen yeni şablon, bir değişken şablonu ($M^V$) olup $M^V$ ($F_i$) ile gösterilir. Her $M^I \in R_{PNDN}(M_0^V)$ için, yalnız bir FS geçişi tetiklenebilir (2. varsayıma dayanarak). Bir FS geçişi tetiklendiği anda, değişken değerlerini ifade eden ortamlardaki ($p_{rj}$), kullanılmayan tüm belirteçler geçersizdir ve bu belirteçler ***sıfırlanır*** (3. varsayıma dayanarak). Belirli PNDN için anlatılan bu belirteç akışı, matematiksel olarak aşağıdaki gibi tanımlanmıştır.

1. a) Eğer $M^V$ ($M^V \in R_{SMDM}(M_0^V)$ ) için aşağıdaki koşul sağlanıyorsa, herhangi bir karar geçişi $df_r$, $M^V$-gerçeklenebilir;

   $\forall j$ (j = 1, 2, …., $n_r$),$\exists$ $p_r \in P$: $p_r \in {}^\bullet df_r \wedge m(p_{r0}) = 1 \wedge m(p_{rj}) = 0$

   b) $M^V$ -gerçeklenebilir bir $df_r$ geçişi $M^V$ den $M^I(df_r)$ a tetiklenir ve $M^V$ [$df_r > M^I$ ($df_r$) şeklinde ifade edilir. Burada $M^I$ ($df_r$) aşağıdaki gibi tanımlanır:

$$M^I(df_r) = \begin{bmatrix} M^I(p_1) \\ M^I(p_2) \\ . \\ . \\ . \\ M^I(p_g) \end{bmatrix} \qquad (6)$$

$M^I(p_r)$, $(n_r+1)$ boyutlu vektör olup;

$$M^I(p_r) = \begin{bmatrix} m^I(p_{r0}) \\ m^I(p_{r1}) \\ . \\ . \\ . \\ m^I(p_{rm_r}) \end{bmatrix} \quad (7)$$

bu vektörün elemanları aşağıda tanımlanmıştır;

$$m^I(p_{rj}) = \begin{cases} m(p_{rj}) - 1 & j = 0 \wedge p_{rj} \in^{\bullet} df_r \\ m(p_{rj}) + 1 & j = C_j, 1 \le C_j \le n_r \wedge p_{rj} \in df_r^{\bullet} \\ m(p_{rj}) & \end{cases}$$

burada $C_j$, j için sabittir.

2. a) Eğer $M^I$ ($M^I \in R_{SMDM}(M_0^V)$) için aşağıdaki koşul sağlanıyorsa, herhangi bir işlevsel durum geçişi $F_i$, $M^I$-gerçeklenebilir;

$\forall p_{rj} \in {}^{\bullet}F_i, \forall p_r \in F_i^{\bullet} : m(p_{rj}) = 1 \wedge m(p_{r0}) = 0$

b) $M^I$-gerçeklenebilir bir $F_i$ geçişi, $M^I$ dan $M^V(F_i)$ a tetiklenir ve $M^I [F_i > M^V(F_i)$ şeklinde ifade edilir. Burada $M^V(F_i)$ aşağıdaki gibi tanımlanır:

$$M^V(F_i) = \begin{bmatrix} M^V(p_1) \\ M^V(p_2) \\ . \\ . \\ M^V(p_g) \end{bmatrix} \quad (8)$$

burada,

$$M^V(p_r) = \begin{bmatrix} m^V(p_{r0}) \\ m^V(p_{r1}) \\ . \\ . \\ . \\ m^V(p_{rm_r}) \end{bmatrix} \quad (9)$$

olup bu vektörün elemanları şöyle tanımlanmıştır;

$$m^V(p_{rj}) = \begin{cases} m(p_{rj}) + 1 & j = 0 \wedge p_{rj} \in F_i^{\bullet} \\ m(p_{rj}) - 1 & j \neq 0 \wedge p_{rj} \in^{\bullet} F_i \wedge m(p_{rj}) \neq 0 \\ 0 & j \neq 0 \wedge p_{rj} \notin^{\bullet} F_i \wedge m(p_{rj}) \neq 0 \end{cases}$$

## PNDN UYGULAMASI: AKILLI ELEKTRİK SÜPÜRGESİ (AES)-BIRINCI DÜZEY KAVRAMSAL TASARIMI

Bu bölümde, geliştirilen tasarım ağının bir mekatronik sistem tasarımı için uygulaması verilmektedir. Örneğimiz, yazılımı geliştirilmiş akıllı bir elektrik süpürgesi tasarımıdır [29]. Tasarımcıdan beklenen, herhangi bir alanı (oda, ofis v.b.) içinde bulunan eşyalara çarpmadan temizleyecek akıllı bir elektrik süpürgesi (robot) tasarımıdır. Bu tasarımda, elektrik süpürgesinin (robotun), temizlenecek alanın şekil ve boyutlarını bildiğini varsayıyoruz. Eşyalar için kısıtlamamız, bunların, kare, dikdörtgen, daire ve/veya çok kenarlı şeklinde olmasıdır. Ancak elektrik süpürgesi, eşyaların şekli, yerleri ve boyutlarıyla ilgili herhangi bir bilgiye sahip değildir. Bu bölümde, böyle bir elektrik süpürgesi için oluşturulan bir işlevsel tasarım seçeneğinin, geliştirilen tasarım ağı ile nasıl modellendiği anlatılacaktır.

PNDN'nin 2. ve 3. bölümlerde açıklanan kuramsal altyapısı AES'nin kavramsal tasarım aşamasında iki tasarım seçeneği için uygulanmıştır. Bu bölümde, birinci tasarım seçeneği (AES-DC1) için PNDN'nin nasıl oluşturulduğu ayrıntılı olarak anlatılmıştır. Daha sonra her iki tasarım seçeneği (AES-DC1 ve AES-DC2) için geliştirilen PNDN yapıları ulaşılabilirlik ve canlılık yönünden karşılaştırmalı olarak analiz edilmiştir.

*İşlevsel Durum Kümesi (FS)*

Modelde ilk aşama, tasarımın birinci düzeyde alt işlevler olarak ifade edilmesidir. Bu tasarım seçeneği için 4 alt işlev düşünülmüştür. Bunlar ;

$FS = \{F_1, F_2, F_3, F_4\}$

$F_1$ : İLERLE (MOVE) $F_2$ : SÜPÜR (SWEEP)

$F_3$ : DÖN (TURN) $F_4$ : DUR(STOP)

olarak tanımlanmıştır.

*Sürekli Değişken Kümesi (CVS)*

Bu tasarım seçeneği için tanımlanan sürekli değişkenler şunlardır :

x : elektrik süpürgesinin x-konumu

y : elektrik süpürgesinin y-konumu

θ : süpürge ile x ekseni arasındaki açı

p : süpürgenin emme basıncı (bu tasarımda süpürme işinin emme ile yapıldığı varsayılmıştır)

Bu tanımlamalara bağlı olarak Sürekli Değişkenler Kümesi (CVS) aşağıdaki gibi tanımlanmıştır ;

$CVS = \{V_1^C, V_2^C, V_3^C, V_4^C,\} = \{x, y, \theta, p\}$ $N_c = 4$

*Kesikli Değişken Kümesi (DVS)*

Kesikli Değişkenler Kümesi (DVS) iki elemandan oluşmaktadır ;

$DVS = \{V_1^D, V_2^D\} = \{osit, csit\}$ $N_D = 2$

burada,

osit: süpürgenin önünde herhangi bir engel olup olmadığını gösteren kesikli değişken,

csit: süpürgenin önündeki bölgenin temizlenmiş olup olmadığını gösteren kesikli değişken,

olarak tanımlanır.

*İşlevsel Durum Matrisi (FSM)*

FSM (NXL) boyutlu bir matris olup her bir işlevsel durum ($F_i$) bir sürekli fonksiyon kümesi ile gösterilir. AES-DC1 için FSM aşağıdaki gibi verilmiştir (. zamana gore türevi göstermektedir) :

FSM =

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $F_1$ | $\dot{x} = V_x$ | $\dot{y} = V_y$ | $\dot{\theta} = 0$ | $\dot{p} = 0$ |
| $F_2$ | $\dot{x} = V_x$ | $\dot{y} = V_y$ | $\dot{\theta} = 0$ | $\dot{p} = P_C$ |
| $F_3$ | $\dot{x} = 0$ | $\dot{y} = 0$ | $\dot{\theta} = \theta_C$ | $\dot{p} = 0$ |
| $F_4$ | $\dot{x} = 0$ | $\dot{y} = 0$ | $\dot{\theta} = 0$ | $\dot{p} = 0$ |

*Kesikli Değişken Değerleri*

IASM-DC1 için ikili kesikli değişken değerleri aşağıdaki gibi tanımlanmıştır;

$DVS_{ins} = \{\{ V_1^D(1), V_1^D(2) \}, \{V_2^D(1), V_2^D(2) \}\} = \{ \{0, 1\}_{osit}, \{0, 1\}_{csit} \}$

osit = 1 (engel var) osit = 0 (engel yok)

csit =1 (temiz), csit = 0 (kirli)

*İşlevsel Durumlar ($F_i$) için Değişmez Koşullar*

Yukarıda tanımlanan alt işlevlerin etken olması için gerekli değişmez koşullar aşağıdaki gibi tanımlanmıştır:

$Inv(F_1) : x \le x_{room} \wedge y \le y_{room} \wedge osit = 0 \wedge csit = 1$

$Inv(F_2) : x \le x_{room} \wedge y \le y_{room} \wedge osit = 0 \wedge csit = 0$

$Inv(F_3) : x \le x_{room} \wedge y \le y_{room} \wedge osit = 1$

$Inv(F_4) : x > x_{room} \wedge y > y_{room}$

Burada $x_{room}$ ve $y_{room}$ temizlenecek alanın boyutlarıdır.

*Sürekli Değişken Değerleri*

Bu tanımlar sürekli değişkenlerin alabileceği değerler için aralıklar vermektedir ve bu aralıklar aşağıdaki küme ile gösterilir;

$CVS_{ins} = \{ V_1^C(n), V_2^C(n) \} = \{\{x \le x_{room}, x > x_{room} \}, \{ y \le y_{room}, y > y_{room} \}\}$

*Kesikli Durum Geçiş Matrisi (DSTM)*

Aşağıda verilen DSTM'de, AES-DC1 geçerli kesikli durum geçişleri 1, geçersiz olanlar 0 ile gösterilmiştir:

DSTM =

| | $F_1$ | $F_2$ | $F_3$ | $F_4$ |
|---|---|---|---|---|
| $F_1$ | 0 | 1 | 1 | 1 |
| $F_2$ | 1 | 0 | 1 | 1 |
| $F_3$ | 1 | 1 | 0 | 1 |
| $F_4$ | 0 | 0 | 0 | 0 |

AES-DC1 Melez Otomat ve PNDN modelleri sırasıyla Şekil 5 ve Şekil 6'da gösterilmiştir. AES için geliştirilen PNDN tasarım ağının $M_0^V$ başlangıç şablonu aşağıda tanımlanmıştır:

$$M_0^V = \left[ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right]$$

AES-PNDN için ulaşılabilirlik kümesi $\{R_{PNDN}(M_0^V)\}$, olası tüm şablonları içerecek şekilde aşağıda gösterilmiştir:

$R_{PNDN}(M_0^V) = \{ M_1^I, M_2^I, \ldots\ldots\ldots, M_{16}^I \}$ ve $M_h^I$ (h = 1,2, ……., 16) olacak şekilde verilmiştir:

$$M_1^I = \left[ \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right]$$

$$M_2^I = \left[ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right]$$

$$M_3^I = \left[ \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right]$$

$$M_4^I = \begin{bmatrix} \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} \end{bmatrix}$$

$$M_5^I = \begin{bmatrix} \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} \end{bmatrix}$$

$$M_6^I = \begin{bmatrix} \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} \end{bmatrix}$$

$$M_7^I = \begin{bmatrix} \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} \end{bmatrix}$$

$$M_8^I = \begin{bmatrix} \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} \end{bmatrix}$$

$$M_9^I = \begin{bmatrix} \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} \end{bmatrix}$$

$$M_{10}^I = \begin{bmatrix} \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} \end{bmatrix}$$

$$M_{11}^I = \begin{bmatrix} \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} \end{bmatrix}$$

$$M_{12}^I = \begin{bmatrix} \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} \end{bmatrix}$$

$$M_{13}^I = \begin{bmatrix} \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} \end{bmatrix}$$

$$M_{14}^I = \begin{bmatrix} \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} \end{bmatrix}$$

$$M_{15}^I = \begin{bmatrix} \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} \end{bmatrix}$$

$$M_{16}^I = \begin{bmatrix} \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} \end{bmatrix}$$

$R_{PNDN}(M_0^V)$, $M_0^V$ başlangıç şablonundan ulaşılan 16 şablon içerir. Bu şablonlardan{ $M_h^I$ (h = 1,2, ....., 16) }, $M_5^I$, $M_7^I$, $M_9^I$, $M_{11}^I$, $M_{13}^I$, $M_{14}^I$, $M_{15}^I$ ve $M_{16}^I$ şablonları aynı anda birden fazla işlevsel durum geçişinin tetiklenmesine neden olduğundan, AES-DC1 için oluşturulan PNDN "canlı" değildir. Bu durum robot pozisyonunun x ve y olarak iki ayrı değişkenle ifade edilmesinden kaynaklanmaktadır. Bu aşamada AES için robot pozisyonunu en üst düzeyde "pos" adlı tek bir değişkenle ifade eden ikinci bir tasarım seçeneği (AES-DC2) oluşturulmuş ve bu seçeneğe ait PNDN Şekil 7'de gösterilmiştir. AES-DC2 için tüm ulaşılabilir şablonlarda bir ve yalnız bir işlevsel durum geçişi tetiklenmekte olup "çözümsüzlük" yoktur. Dolayısıyla, AES-DC2 için oluşturulan PNDN canlıdır. Bu durumda ikinci alternatif seçilerek aşağıdaki tasarım önerileri geliştirilmiştir:

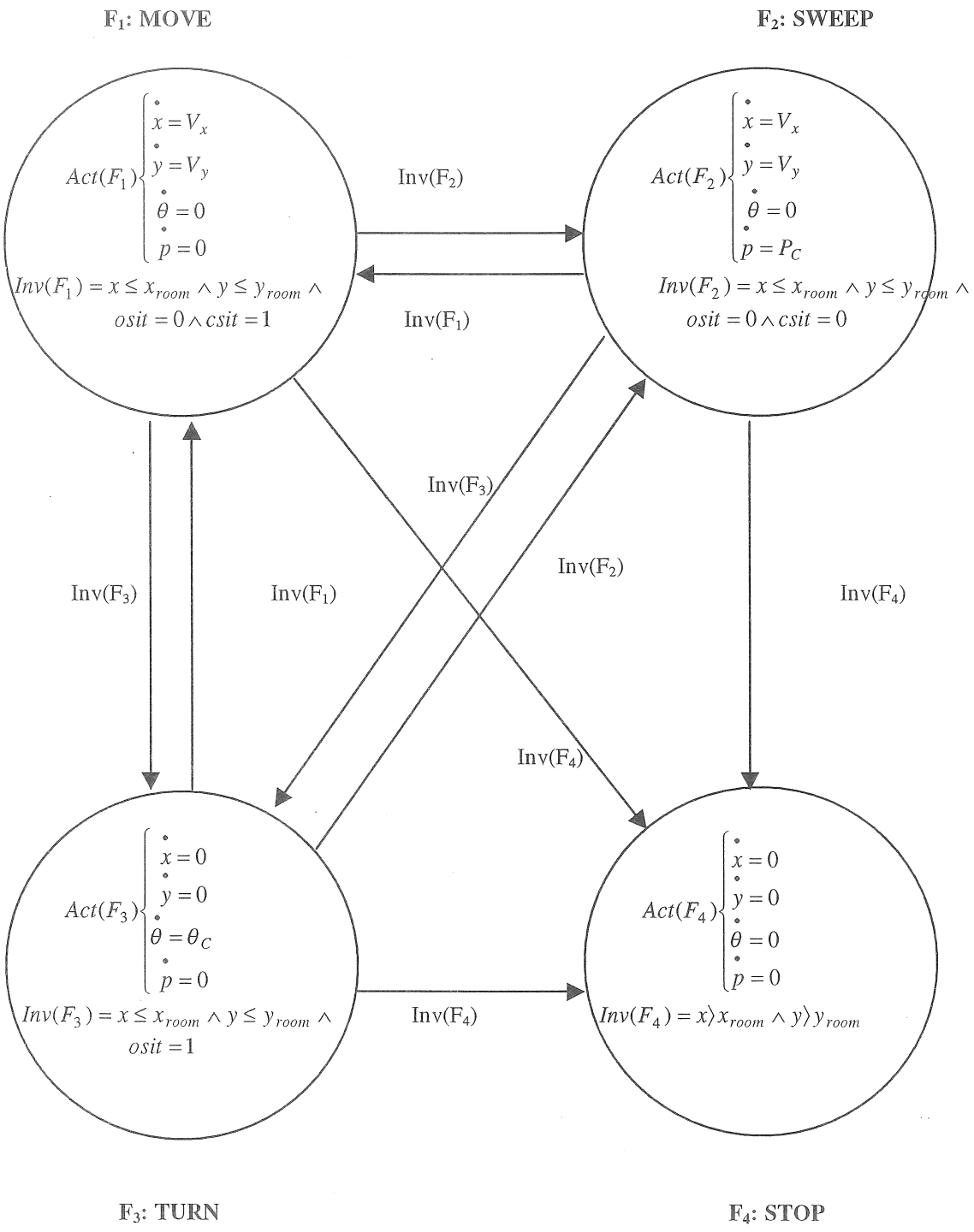
- Robot pozisyonunu belirleyen ve robot odanın içindeyse "pos = 1 ", robot odanın dışındaysa "pos = 0" çıktısı veren bir alt system/eleman kullanılmalıdır.
- Hareket sırasında robotun önünde bir engel olup olmadığını belirleyen ve engel varsa "1", engel yoksa "0" çıktısı veren bir alt system/eleman kullanılmalıdır.
- Robotun önündeki alanın temiz olup olmadığını belirleyen ve alan temiz ise "1", kirli ise "0" çıktısı veren bir alt system/eleman kullanılmalıdır.

AES DC1 ve DC2 için oluşturulan PNDN'lerde eşzamanlılık tüm karar geçişlerinin aynı anda tetiklenmesiyle ortaya çıkmaktadır.

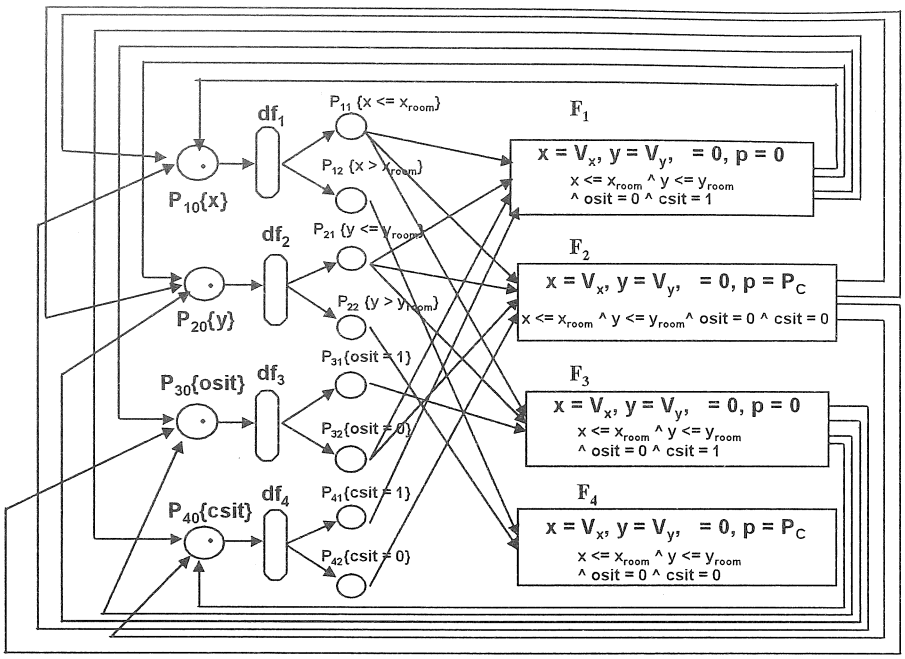
## TARTIŞMA VE SONUÇLAR

Bu araştırmada, mühendislik tasarımında kavramsal tasarımın ikinci aşamasının otomasyonu için Petri Net ve Melez Otomat sentezine dayalı

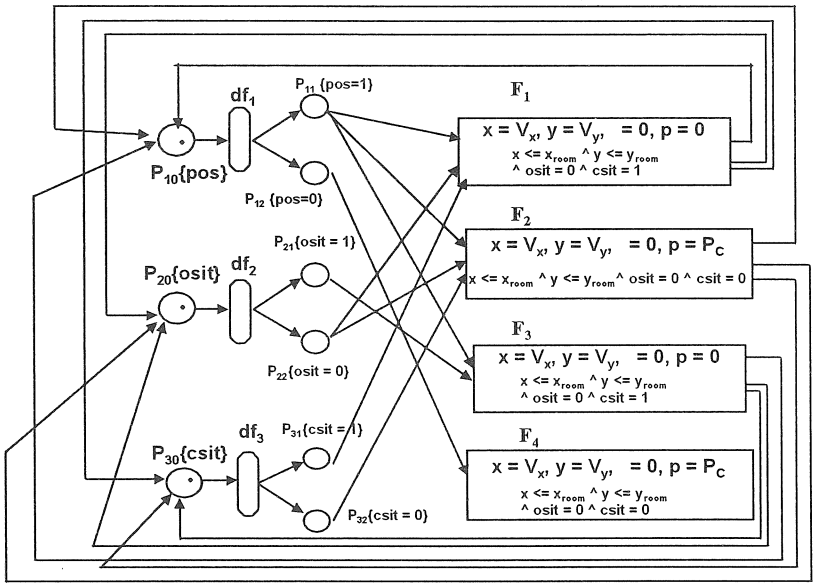
PNDN adlı bir tasarım ağı geliştirilmiştir. PNDN kuramı herhangi bir fiziksel gerçeklemeye dayanmadığından, genel amaçlı bir modeldir. Bu nedenle modelin uygulamaları, tümleşik bir tasarım felsefesi olan mekatronik tasarım örneklerinden seçilmiştir. Geliştirdiğimiz bu tasarım otomasyonu modelleme tekniği, bu makalede verilen robot süpürge dışında çok sayıda sistemin kavramsal tasarımı için uygulanmıştır [22]. Geliştirilen modelin algoritmik yapısı aşağıdaki gibi özetlenebilir :




Şekil 5. AES - DC1'in Melez Modeli




Şekil 6. AES-DC1'in PNDN modeli ve başlangıç şablonu ($M_0^V$)




Şekil 7. IASM – DC2'nin PNDN modeli ve başlanıç şablonu ($M_0^V$)

1) Tasarımcı tarafından geliştirilen işlevsel tasarım seçeneğinin matematiksel ifadesi,
2) Bu matematiksel ifadenin otomatik olarak melez otomat modeline dönüştürülmesi,
3) Melez modelden Petri Net modeline otomatik geçiş,
4) Petri Net modeli üzerinde, önerilen işlevsel tasarımın analizi ve değerlendirilmesi.

PNDN tasarım ağının literature getirdiği önemli katkı, kavramsal tasarımın modellenmesinde Petri Net ve Melez Otomatların ilk kez kullanılmasıdır. PNDN ayrıca tasarlanan sistemde kesin ve belirsizlik içeren durumlardaki bilgi akışını [27,28] modelleme özelliği ile de literatürdeki diğer kavramsal tasarım modellerinden farklıdır. PNDN için geliştirilmiş olan "DNS-Design Network Simulator" adlı yazılım [30,31] tasarımcının birçok işlevsel tasarım seçeneğini bilgisayar ortamında, çok kısa bir zamanda değerlendirmesini sağlamaktadır. PNDN yapısı, herhangi bir işlevsel durumun çeşitli düzeylerde alt işlevlere ayrıştırılmasına ve bunların analizine uygundur. Dolayısıyla, seçeneklerin en basit işlevlerine kadar değerlendirilmesi mümkündür. Matematiksel değerlendirme yanında, Petri Net'lerin modelledikleri sistem(ler)i görsel olarak ifade etme özelliği, bu tasarım ağının değişik tasarımcılar arasında kolay ve çabuk bilgi alış verişine olanak sağlayacağı düşünülmektedir. Geliştirilen modelin bir başka özelliği, genel amaçlı olmasıdır. Böylece, gerek değişik tasarım konuları için ayrı ayrı kullanıma, gerekse bu çalışmadaki mekatronik tasarım örneği gibi tümleşik sistem tasarımına uygulanabilir.

## SYNTHESIS OF PETRI NETS AND HYBRID AUTOMATA IN CONCEPTUAL DESIGN AUTOMATION

In this study, a design network model called PNDN, which is based on the synthesis of Petri Nets and Hybrid Automata is developed in order to automate the conceptual design phase of the engineering design procedure. PNDN models the logical behaviour of a design artifact through the deterministic and nondeterministic information flow between the sub functions of the system. The uncertainties that exist in a nondeterministic PNDN is modelled using Intuitionistic Fuzzy Propositions approach. Since PNDN is a general purpose model and developed on a functional basis, it is applied to mechatronic design that is an integrated design philosophy. In this paper the structure of the PNDN and deterministic information flow are presented.

Keywords: Conceptual Design, Design Automation, Petri Net, Hybrid Automata, Mechatronic Design, Design Network.

## KAYNAKÇA

1. Pahl, G. ve Beitz, W., *Engineering Design-A Systematic Approach*, The Design Council, London, UK, 1988.
2. Ullman, D.G., *The Mechanical Design Process*, McGraw-Hill, Inc., USA, 1992.
3. Tomiyama, T. et al., Metamodel : A Key to Intelligent CAD Systems, *Research in Engineering Design*, 1(1989), 19-34.
4. Pugh, S., *Total Design*, Addison-Wesley Publishing Company, UK, 1991.
5. Suh, N.P., Bell, A.C. ve Gossard, D.C., On an Axiomatic Approach to Manufacturing and Manufacturing Systems, *ASME Transactions, Journal of Engineering for Industry*, 100 (1978) 2, 127-130.
6. Serrano, D., *Constraint Management in Conceptual Design*, Ph.D. Thesis, Massachusetts Institute of Technology, USA, 1987.
7. Kusiak, A. ve Park, K., Concurrent Engineering: Decomposition and Scheduling of Design Activities, *Int. J. of Prod. Res.*, 28 (1990) 10, 1883-1900.
8. Brown, D.C. ve Chandrasekaran, B., Expert Systems for a Class of Mechanical Design Activity, *Knowledge Engineering in Computer-Aided Design*, Ed. John. S. Gero, 259-283, Elsevier Science Publishers B. V., North-Holland, 1985.
9. Kannapan, S.M. ve Marshek, K.M., An Algebraic and Predicate Logic Approach to Representation and Reasoning in Machine Design, *Mechanism and Machine Theory*, 25 (1990) 3, 335-353.
10. Kowalski, J., Modelling Knowledge-Based System for Optimum Machine Design, *Mechanism and Machine Theory*, 27 (1992) 4, 491-505.
11. Johnson, D.E. ve Johnson, J.R., *Graph Theory with Engineering Applications*, The Ronald Press Company, USA, 1972.
12. Cagan, J., A Graph-Based Representation to Support Structural Design Innovation, *Artificial Intelligence in Design '91*, Ed. John S. Gero, Butterworth-Heinemann Ltd., Oxford, UK, 1974.

13. Buur, J., *A Theoretical Approach to Mechatronic Design*, Doktora Tezi, Technical University of Denmark, IK Publication 90.74A, Lyngby, Denmark, 1990.
14. Auslander, D.M. ve Kempf, C.J., *Mechatronics-Mechanical System Interfacing*, Prentice Hall Inc., USA, 1996.
15. Murata, T., Petri Nets: Properties, Analysis and Applications, *Proceedings of the IEEE*, 77 (1989) 4, 541-580.
16. Peterson, J. L., Petri Nets, *ACM Comput. Surveys*, 9 (1977) 223-252.
17. Reisig, W., *Petri Nets : An Introduction*, Springer-Verlag, Germany, 1985.
18. Reisig, W., *A Primer in Petri Net Design*, Springer-Verlag, Germany, 1992.
19. Alur et al., The Algorithmic Analysis of Hybrid Systems, *Proceedings of the 11 th Int. Conf. On Analysis and Optimization of Discrete Event Systems, Lecture Notes in Control and Information Sciences 199* (1994), 331-351, Springer-Verlag.
20. Henziger, T.A. ve diğerleri, What's Decidable About Hybrid Automata?, *Proceedings of the 27th Annual Symposium Theory on the Theory of Computing*, 373-382, ACM Press, (1995).
21. Puri, A. ve Varaiya, P., Verification of Hybrid Systems Using Abstractions, *Hybrid Systems II, LNCS 999*, Springer-Verlag (1995).
22. Erden, Z, *A Petri Net-Based Inference Network for Design Automation at Functional Level Applied to Mechatronic Systems*, Doktora Tezi, ODTÜ, 1999.
23. Erden, Z., Erkmen, A.M. ve Erden, A., A Petri Net-Based Design Network with Applications to Mechatronic Systems, *Journal of Integrated Design and Process Science*, 2 (1998) 3, 32-48.
24. Erden Z., Erkmen, A.M. ve Erden, A., Generation of Functional Cells for a Mechatronic Design Network, *Proc. of the 5th UK Mechatronics Forum Int. Conference (Mechatronics '96) with the 3rd Int. Conf. on Mechatronics and Machine Vision in Practice (M²VIP '96)*, 1, 233-238 Guimaraes, Portugal, 1996.
25. Erden, Z., Erkmen, A.M. ve Erden, A., Modeling of the Job-Resource Interaction in a Mechatronic Design Process by Using Petri Net Theory, *Proc. 11th International Conf. on Engineering Design, ICED 97*, 2, 753-756, Tampere, Finland, 1997.
26. Erden, Z., Erkmen, A.M. ve Erden, A., A Petri Net-Based Inference Network for Design Automation under Nondeterminism Applied to Mechatronic Systems, *Proc. the 6th UK Mechatronics Forum International Conference (Mechatronics 98)*, 17-22, Skövde, Sweden, 1998.
27. Erden, Z., Erkmen, A.M. ve Erden, A., Handling Uncertainty in Design Automation Using Intuitionistic Fuzzy Propositions, *Proc. the 12th Int. Conference on Engineering Design, ICED 99*, Munich, Germany, 1999.
28. Tabak, D. and Levis, A.H. (1985) Petri Net Representation of Decision Models. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-15:812-818.
29. Erden, Z. ve Erden, A., Design Note : A Simulation Software for an Intelligent Area Sweeping Machine, *Proc. of the 1st Int. Mechatronic Design and Production Workshop*, 339-343, ODTÜ, 1993.
30. Güroğlu, S., *Implementation of an Algorithm for a Petri Net-Based Design Inference Network*, Yüksek Lisans Tezi, ODTÜ, 1999.
31. Güroğlu, S., Erden, Z., Erkmen, A.M. and Erden, A., The Design Network Simulator (DNS): An Implementation Software for a Petri Net-Based Design Network Applied to Mechatronic Design, *Proc. the 6th Int. Conf. on Mechatronics and Machine Vision in Practice, M²VIP 99*, 193-202, Ankara, Turkey, 1999.