NÖRAL VE BULANIK SİSTEM HÜCRE AKTİVASYON YAKLAŞIMLARI VE FPGA'DA DONANIMSAL GERÇEKLENMESİ

Mehmet Ali ÇAVUŞLU[1], Cihan KARAKUZU[2]

[1]Kocaeli Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Müh. Anabilim Dalı, Kocaeli, alicavuslu@gmail.com,
[2]Bilecik Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Bilecik, cihan.karakuzu@bilecik.edu.tr

ÖZET

Günümüzde nöral ve bulanık sistemler, çok geniş bir alanda kullanılan yöntemlerdir. Bu sistemlerin kendi özelliklerini sağlayan bir donanım ortamında gerçeklenmesi önemlidir. FPGA'lar paralel veri akışı ve paralel işlem yapma özellikleri ile nöral ve bulanık sistemlerin gerçeklenmesinde tercih edilen donanım olmaya başlamıştır. Bu sistemlerde kilit role sahip olan hücre aktivasyon fonksiyonunun, donanım üzerinde gerçeklenmesi önemlidir. Bu çalışmada, sıklıkla kullanılan logaritmik sigmoidal, hiperbolik tanjant sigmoidal ve Gauss tipi aktivasyon fonksiyonlarının matematiksel yaklaşımlarının tek duyarlıklı kayan noktalı sayı formatıyla FPGA'de gerçeklenmesi irdelenmiştir. Her bir fonksiyon için FPGA'de gerçeklemeye uygun yaklaşımı karşılaştırmalı olarak verilmiş, Xilinx firmasına ait Virtex 5 xc5vlx110-3ff1153 FPGA'sında gerçeklenerek elde edilen sentez sonuçları sunulmuştur. Elde edilen deneysel sonuçlar, önerilen gerçekleme yaklaşımlarının çok az donanımsal kaynak tükettiğini göstermiştir. Önerilen bu yaklaşımlar kullanılarak çeşitli yapılarda nöral ve bulanık sistemler gerçeklenebilir.

Anahtar Kelimeler: Hücre aktivasyonları, FPGA, kayan noktalı sayı, donanımsal gerçekleme, nöral sistemler, bulanık sistemler.

NEURAL AND FUZZY SYSTEM CELL ACTIVATION APPROXIMATIONS AND HARDWARE IMPLEMENTATIONS ON FPGA

ABSTRACT

Currently, neural and fuzzy systems are methods which have found wide application area. Implementation of these systems on a hardware platform providing their own features is important. With parallel data streaming and processing features, FPGAs have become a preferable hardware platform for implementing of neural and fuzzy systems. In hardware implementation of these systems, the cell activation function, which is the most important unit, is of key importance. In this study, implementation mathematical approximations of commonly used logarithmic sigmoid, hyperbolic tangent sigmoid and Gaussian activation functions on FPGA using single-precision floating-point number format is investigated. For the each function, suitable approach to implement on FPGAs is comparatively given and obtained the actual synthesis results from the Xilinx Virtex 5 xc5vlx110- 3ff1153 FPGA are presented. Obtained experimental results show that proposed implementation approaches have consumed very little hardware resources. Using these proposed approaches, neural and fuzzy systems in various structures can be implemented.

Keywords: Cell activations, FPGA, floating point number, hardware implementation, neural systems, fuzzy systems.

1. GİRİŞ

Nöral ve bulanık-nöral sistemleri, özellikle doğrusal olmayan sistemler üzerindeki modelleme, kontrol, kestirim vb. görevlerde başarılı kılan en önemli birimler aktivasyon ve üyelik fonksiyonlarıdır. Yapay sinir ağlarında aktivasyon fonksiyonu sahip olduğu doğrusal olmayan giriş-çıkış karakteristiği ile problemin doğrusal olmayan unsurlarını modellemede etkin rol alır. Bulanık, bulanık-nöral

sistemlerde ise üyelik fonksiyonları (ÜF) giriş ve çıkış uzayında bulanıklaştırma işlevi yerine getirirler. Bu işlev bulanık sistemde tanımlı kuralların aktifliğini belirlerken, dolaylı olarak giriş-çıkış uzayları arası eşleştirmenin karakterini de belirler. Doğrusal olmayan aktivasyon ve ÜFler genel olarak üstel fonksiyonla tanımlıdırlar. Bu üstel fonksiyonların donanım ortamında gerçeklenmesi nöral, bulanık ve bulanık-nöral sistemlerin donanımsal gerçeklenmesinde kilit role sahiptir.

Paralel veri akışı ve işlem yapabilme özelliklerine sahip FPGA'lerle, ağır işlem yükü gerektiren gerçek zamanlı uygulamalar gerçeklenebilmektedir[1]. FPGA kullanımı, son yıllarda birçok nöral [2-8] ve bulanık-nöral sistem gerçeklemede ön plana çıkmıştır. Donanımsal nöral sistem gerçeklemesi ve eğitimi çalışmalarında, aktivasyon fonksiyonlarının gerçeklenmesinde bakma tablosu [4, 8], parçalı doğrusal [3, 5, 7], parabolik [2] ve fonksiyonel [6] yaklaşımlar kullanılmıştır. Bulanık-nöral sistemlerin gerçeklenmesi ve eğitimi çalışmalarında üyelik fonksiyonlarının donanımsal gerçeklenmesine kısaca bakarsak: [9, 10]'da Gauss üyelik fonksiyonu (GÜF) için ikinci dereceden doğrusal olmayan fonksiyon yaklaşımı, [11]'de yine GÜF için Taylor serisi açılımına dayalı bakma tablosu yaklaşımı, [12]'de de GÜF için bakma tablosu yaklaşımı kullanılmıştır. [13]'de ANFIS ve DFNN yapılarının donanımsal gerçeklenmesinde GÜF için [17]'de verilen fonksiyonel yaklaşım kullanılmıştır. [14-16]'da üçgen üyelik fonksiyonları kullanılmıştır.

Nöral ağların bazı modellerinin türev tabanlı eğitimlerinin gerçeklenmesi için aktivasyon fonksiyonun türevinin alınabilir olması gereklidir. Bu nedenle bakma tablosu, parçalı doğrusal ve parabolik yaklaşımlar, türevlenebilir olma şartını tam olarak sağlayamamaktadırlar. Benzer durum doğrusal olmayan ÜF kullanan bulanık veya bulanık-nöral ağlar için de geçerlidir. Özellikle türev tabanlı eğitim algoritmalarıyla ÜF parametrelerinin optimizasyonunda ÜFlerin türevlenebilir olması gereklidir.

Bu çalışmada doğrusal olmayan aktivasyon fonksiyonlarının en çok kullanılan türlerinden olan logaritmik sigmoidal, tanjant hiperbolik fonksiyonlarının ve bulanık-nöral sistemlerde kullanılan Gauss tipi ÜFlerin matematiksel yaklaşımlarının ve bu yaklaşımların türevlerinin FPGA'da donanımsal gerçeklenmesi üzerine odaklanılmıştır. Uygulama Xilinx firmasına ait Virtex 5 xc5vlx110-3ff1153'da 32 bit kayan noktalı sayı formatında gerçeklenmiştir.

2. ARİTMETİK İŞLEMLER

Sayısal ortamda aritmetik işlemleri yapabilmek için kullanılan sayı formatlarından duyarlılık bakımından en göze çarpanı, IEEE'nin kayan noktalı sayılar ve sabit noktalı sayılar formatlarıdır. Bu çalışmada, kullanıcıya sağladığı hassasiyet ve dinamiklik sebebiyle IEEE 754 formatındaki tek duyarlıklı kayan noktalı sayılar ile çalışılmıştır. Bu sayı formatı ve bu format ile aritmetik işlemler aşağıda kısaca tanıtılmıştır.

2.1 Kayan Noktalı Sayılar

Kayan noktalı sayılar, sayısal ortamda gerçek sayıların bir gösterim biçimidir. Hassas işlem yapma özelliğine sahip kayan noktalı sayılar, dinamiklik özelliği ile de geniş sayı gösterim aralığı imkanı sunmaktadırlar. En çok kullanılan kayan noktalı sayı gösterim standardı IEEE 754 standardıdır. Bu çalışmada tek duyarlı (32 bit) IEEE 754 standardı kullanılmış olup, bu gösterim formatı Şekil 1'de verilmiştir.

Şekil 1. IEEE 754 Kayan-noktalı sayı gösterimi

Eşitlik 1'de gerçel bir sayının kayan-noktalı sayılarda gösterimi verilmiştir. Eşitlik 2'de bias, (3)'te ise üs ifadesi tanımlanmıştır.

$$Sayı = (-1)^s 2^{e-bias}(1+f) \quad (1)$$

$$bias = 2^{n-1} - 1 \quad (2)$$

$$e = bias + floor(\log_2 Sayı) \quad (3)$$

Eşitliklerde, ***s*** işaret bitini temsil eder. İşaret biti "0" ise sayının pozitif, "1" ise negatif olduğunu belirtir. ***e*** üs değerini ve ***f*** çarpan değerini gösterirken, ***n*** üs değerini tutan bit sayısıdır. Çarpan değeri daima sıfır ile bir arasında olmalıdır.

Yukarda verilen bilgiler doğrultusunda gerçel bir sayıyı IEEE 754 standardında kayan noktalı sayı formatına çevirelim: Gerçel sayımız 7 ve n = 8 olsun. Sayı pozitif olduğundan $s = 0$'dır. $bias = 2^{8-1} - 1 = 127$ olarak hesaplanır, 8 bitlik üs kısmı ise $e = bias + floor(\log_2 7) = 129 = (10000001)_2$'dır.
Şu halde sayımızı $7 = (-1)^0 2^2 (1.75)$ şeklinde yazabiliriz. Çarpan kısmı $f = 1.75 - 1 = 0.75$'dır. Çarpan ifadesinin ikili sayı tabanında elde edilişi aşağıda gösterilmiştir.

$$\begin{array}{ccccc} 0.75 & * & 2 & \rightarrow & 1 \\ 0.5 & * & 2 & \rightarrow & 1 \\ 0 & * & 2 & \rightarrow & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{array}$$

Ve nihayet, 7 sayısının IEEE 754 standardına göre 32 bitlik gösterimi Şekil 2'de verildiği gibi elde edilir.

0	10000001	11000000000000000000000

Şekil 2. 7 sayısının kayan noktalı sayı gösterimi

2.2 Kayan Noktalı Sayılarda Toplama/Çıkarma

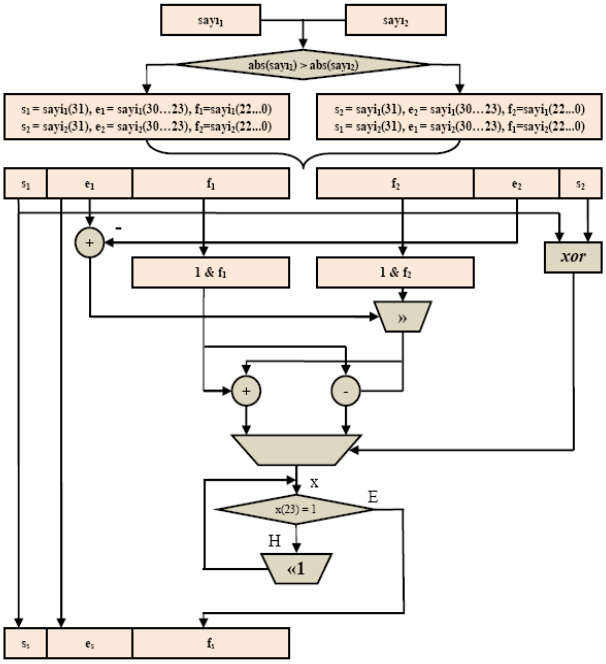
Öncelikle, işlem yapacağımız sayıların, matematiksel dönüşümlerini elde edersek;

$Sayi_1 = 7.362 = (-1)^0 2^2 (1.8405)$
$s_1 = 0 \quad e_1 = 129 \quad 1.f_1 = 1.8405$
$Sayi_2 = 2.594 = (-1)^0 2^1 (1.297)$
$s_2 = 0 \quad e_2 = 128 \quad 1.f_2 = 1.297$

Yukarıda verilen iki sayının toplama işleminde ilk olarak iki sayıdan mutlak büyük olanın işareti, sonuçta elde edilecek sayının işaret biti olarak atanır ($s_{sonuc} = 0$). Üslü sayılarda toplama işlemi yapabilmek için sayıların aynı tabanda ve aynı kuvvette olması gerekmektedir. Bu nedenle mutlak küçük olan sayının üst kuvveti mutlak büyük olan sayının kuvvetine eşitlenir. Aynı zamanda mutlak büyük sayının üst kuvveti sonucun üs ifadesi olarak atanır ($e_{sonuc} = 129$). Üs ifadeleri eşitlendikten sonra mutlak küçük sayının *1.f* ifadesi de $2^{e_1 - e_2}$'ye bölünür. Bu bilgiler doğrultusunda $Sayi_2 = 2.594 = (-1)^0 2^2 (0.6485)$ elde edilir. Sonuç ifadesinin çarpan kısmı $1.f_{sonuc} = 1.8405 + 0.6485 = 2.489$'dır. Sonuç ifadesi ise $Sonuc = (-1)^0 2^2 (2.489)$ şeklinde elde edilir. Daha öncede belirtildiği üzere çarpan ifadesi [0-1) aralığında olmalıdır. Bu nedenle sonuç ifadesini tekrar yazarsak; $Sonuc = (-1)^0 2^3 (1.2445)$ elde edilir. Bu ifadeleri gerçel sayımızı elde etmek için kullanırsak; $Sayı = (-1)^0 (1.2445) 2^3 = 9.956$ elde edilir. Gerçel sayılarımızın toplamı da 7.362 + 2.594 = 9.956'dır. Çıkarma işleminde, toplama işleminden farklı olarak çarpan kısımları çıkarma işlemine tabi tutulmaktadır. Kayan nokta formatında iki sayının toplama aritmetiğine ait işlemlerin akış şeması Şekil 3'de verilmiştir.

2.3 Kayan Noktalı Sayılarda Çarpma

Kayan noktalı sayılarda çarpma işlemini toplama örneğindeki sayılar üzerinde gösterelim: Sonucun işaret bitini elde etmek için iki sayının işaret bitleri XOR işlemine tabi tutulur ($s_{sonuc} = s_1 \; xor \; s_2 = 0 \; xor \; 0 = 0$). Üs ifadesi $e = e_1 + e_2 - bias$ ile elde edilir ($e_{sonuc} = 129 + 128 - 127 = 130$). Çarpan kısmı;
$1.f_{sonuc} = 1.f_1 * 1.f_2 = \; 1.8405 * \; 1.297 = \; 2{,}3871285$
şeklinde bulunur. Sonuç ifadesi $Sonuc = (-1)^0 2^3 (\; 2.3871285)$'dir. Daha önce de belirtildiği üzere çarpan ifadesi [0-1) aralığında olmalıdır. Bu nedenle sonuç ifadesini tekrar yazılırsa;
$Sonuc = (-1)^0 2^4 (1{,}19356425)$ elde edilir. Bu ifadeleri gerçel sayımızı elde etmek için kullanırsak; $Sonuc = (-1)^0 2^4 (1{,}19356425) = 19.097028$ elde edilir. Gerçel sayılarımızın çarpımı da 7.362 x 2.594 = 19.097028'dir. Kayan nokta formatında iki sayının çarpma aritmetiğine ait iki tabanında işlemlerin akış şeması [18]'den incelenebilir.



Şekil 3. Toplama işlemi akış diyagramı

2.4 Kayan Noktalı Sayılarda Bölme

Kayan noktalı sayılarda bölme işlemi toplama örneğindeki sayılar üzerinde şu şekilde yapılmaktadır: Sonucun işaret bitini elde etmek için iki sayının işaret bitleri XOR işlemine tabi tutulur ($s_{sonuc} = s_1 \; xor \; s_2 = 0 \; xor \; 0 = 0$). Üs ifadesi $e_{sonuc} = bias + \; e_1 - e_2$ ile elde edilir ($e_{sonuc} = 127 + 129 - 128 = 128$). Çarpan kısmı; $1.f_{sonuc} = 1.f_1 \; / \; 1.f_2 = \; 1.8405 \; / \; 1.297 = \; 1.4190439475$'dir. Şu halde, $Sonuc \cong (-1)^0 2^1 (\; 1.4190439475)$'dır. Bu ifadeleri gerçel sayımızı elde etmek için kullanırsak; $Sonuc \cong (-1)^0 2^1 (1.4190439475) \cong 2.838087895$ elde edilir. Gerçel sayılarımızın bölümü de $7.362 \; / \; 2.594 \cong 2.838087895$'dir.

3. AKTİVASYON FONKSİYONLARININ FPGA'DA DONANIMSAL GERÇEKLENMESİ

Nöral ve bulanık-nöral hücre modellerinde, hücrenin gerçekleştireceği işleve göre çeşitli tipte aktivasyon fonksiyonları kullanılabilir. Aktivasyon fonksiyonları sabit parametreli ya da uyarlanabilir parametreli seçilebilir. Bu çalışmada, yaygın olarak kullanılan tanjant hiperbolik, logaritmik sigmoidal ve Gauss aktivasyon fonksiyonlarındaki üssel ifadenin FPGA'da doğrudan gerçeklenememesi sebebiyle, bu fonksiyonların yaklaşımları FPGA'de donanımsal olarak gerçeklenmiştir. Aşağıda bu yaklaşımların donanımsal gerçeklenmeleri kısaca tanıtılacaktır.

3.1 Logaritmik Sigmoidal

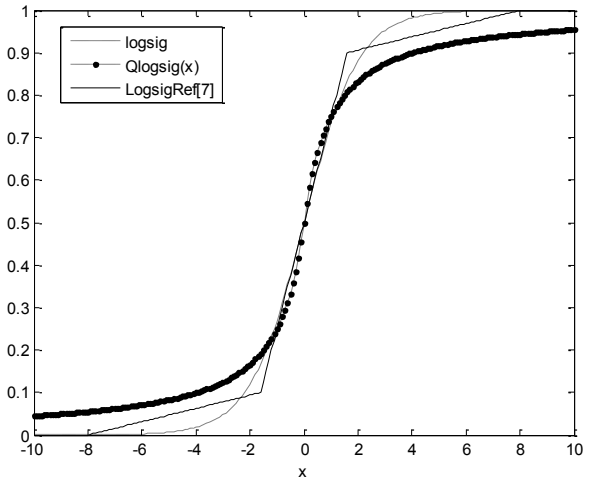
Logaritmik sigmoidal fonksiyonu [0 1] aralığında çıkış veren, yapay sinir ağlarında çok tercih edilen bir aktivasyon fonksiyonudur. Logaritmik sigmoidal fonksiyonunun matematiksel ifadesi (4)'de verilmiştir.

$$logsig(x) = \frac{1}{1+e^{-x}} \tag{4}$$

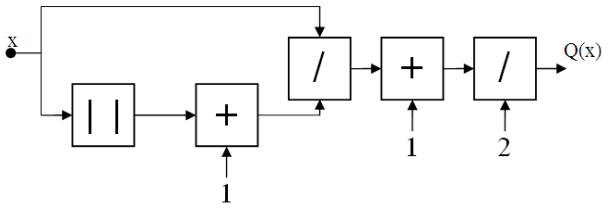
Logaritmik sigmoidal ifadeyi donanımsal gerçeklemek için, (5)'de verilen yaklaşım [12] kullanılmıştır.

$$Q_{logsig}(x) = \frac{1}{2}\left[1 + \frac{x}{1+|x|}\right] \tag{5}$$

Şekil 4'de (4) ve (5)'te verilen logaritmik sigmoidal ve yaklaşımı ile [7]'de kullanılan logaritmik sigmoidal doğrusal parçalı yaklaşımlarının [-10 10] aralığında davranışları karşılaştırmalı olarak gösterilmiştir. Şekil 5'te logaritmik sigmoidal fonksiyonunun matematiksel yaklaşımına ait blok diyagram, Şekil 6'de ise bu yaklaşımın gerçeklendiği VHDL kodu verilmiştir. Görüleceği üzere yaklaşımın gerçeklenebilmesi için 1 mutlak değer, 2 toplama ve 2 bölme modülü gerekmektedir. Mutlak değer alma işlemi sayının işaret bitinin '0' yapılmasıyla gerçeklenir. 2'ye bölme işlemi ise sayının üs (*e*) kısmının 1 azaltılmasıyla gerçeklenir. Şekil 6'dan anlaşılacağı üzere, toplama, çarpma ve bölme işlemleri için önceki çalışmalarımızda [6, 18] oluşturulan sırasıyla "*add*", "*mul*" ve "*divide*" kütüphaneleri kullanılmıştır.



Şekil 4. logisg(x), Q_{logsig}(x) ve [7]'de kullanılan parçalı doğrusal fonksiyonlarının kıyaslaması



Şekil 5. Logaritmik sigmoidal fonksiyonu yaklaşımının FPGA'da gerçekleme blok yapısı

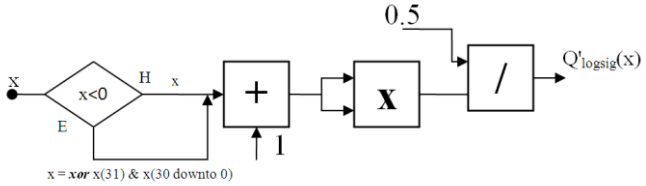
```
case kont is
  when "00000" =>
     toplam_1 <= add(bir, '0' & giris(30 downto 0));
     kont <= "00001";

  when "00001" =>
     bolme_sonuc <= divide(giris, toplam_1)
     kont <= "00010";

 when "00010" =>
    toplam2 <= add(bir, bolme1_sonuc);
    kont <= "00011";

 when "00011" =>
   cikis <= toplam_2(31) & toplam_2(30  downto 23) - 1 &
    toplam2(22 downto 0);
 when others => null;
end case;
```

Şekil 6. Logaritmik sigmoidal fonksiyonu yaklaşımının FPGA'da gerçekleme VHDL kodu

Gradyen tabanlı eğitimde kullanılması açısından, logaritmik sigmoidal ve yaklaşımının türevleri sırasıyla (6) ve (7)'de verilmiştir. Eşitlik 7'de verilen ifadenin FPGA'de gerçeklenmesi blok yapısı Şekil 7'de, bu blok yapının VHDL dili program kodu da Şekil 8'de verilmiştir.
Şekil **9**'da ise orijinal logsig, önerilen yaklaşım ve [7]'de kullanılan yaklaşımın türevleri karşılaştırmalı olarak gösterilmiştir.

$$logsig'(x) = logsig(x)(1 - logsig(x)) \tag{6}$$

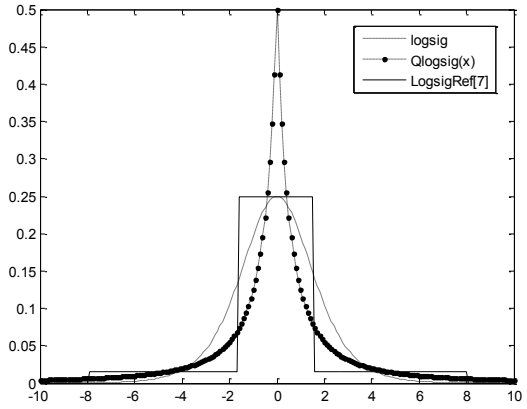
$$Q'_{logisg}(x) = \begin{cases} \frac{0.5}{(1-x)^2}, & x < 0 \\ \frac{0.5}{(1+x)^2}, & x \geq 0 \end{cases} \tag{7}$$



Şekil 7. Logsig yaklaşım fonksiyonu türevinin FPGA'de gerçekleme blok yapısı

```
case kont is
  when "00000" =>
    if giris(31) = '1' then
      toplam_1 <= add(bir, (not giris(31)) & giris(30
downto 0));
      kont <= "00001";
    else
      toplam_1 <= add(bir, giris);
      kont <= "00001";
    end if;

  when "00001" =>
    carpma_1 <= mul(toplam_1, toplam_1);
    kont <= "00010";

  when "00010" =>
    cikis <= divide(yarim, carpma_1)

  when others => null;
end case;
```

Şekil 8. Logsig yaklaşım fonksiyonu türevinin FPGA'de gerçekleme VHDL kodu



Şekil 9. logisg(x), $Q_{logsig}(x)$ ve [7]'de kullanılan parçalı doğrusal fonksiyonlarının türevlerinin kıyaslaması

3.2 Tanjant Hiperbolik

Tanjant hiperbolik fonksiyonu, türevi alınabilir, sürekli ve doğrusal olmayan bir fonksiyon olması nedeniyle doğrusal olmayan problemlerin çözümünde kullanılan YSA'larda tercih edilir. Tanjant hiperbolik fonksiyonunun matematiksel ifadesi (8)'de verilmiştir.

$$tanh(x) = \frac{1-e^{-x}}{1+e^{-x}} \qquad (8)$$

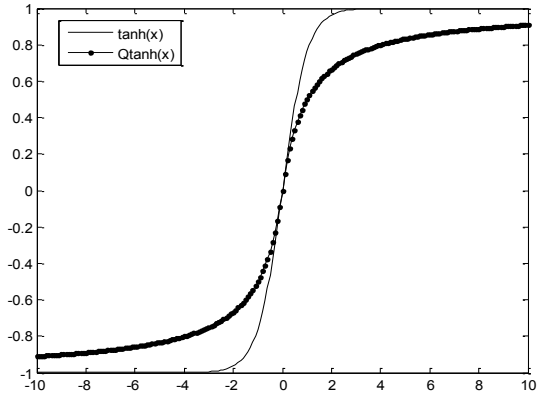
Eşitlikteki üstel ifade donanımsal gerçeklenemediğinden yerine, (9)'da verilen matematiksel yaklaşım [11] kullanılmıştır.

$$Q_{tanh}(x) = \frac{x}{1+|x|} \qquad (9)$$

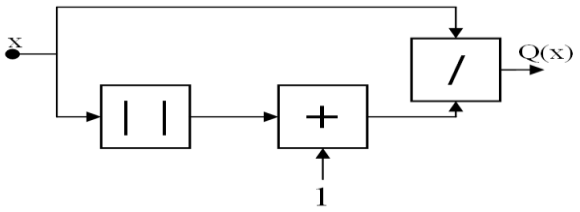
Şekil 10'da *tanh(x)* ve $Q_{tanh}(x)$ fonksiyonlarının [-10 10] aralığında davranışları verilmiştir. Şekil 11'de ise (9)'da verilen yaklaşımın gerçekleme blok yapısı verilmiştir. Gradyen tabanlı eğitimde kullanılması açısından, tanjant hiperbolik ve yaklaşımının türevleri (10) ve (11)'de verildiği gibidir. Şekil 12'de bu yaklaşımın türevinin orijinalinin türevi ile karşılaştırmalı davranışı verilmiştir. Şekil 13'de ise (11)'i gerçekleme blok yapısı verilmiştir.

$$tanh'(x) = 1 - tanh^2(x)\,) \qquad (10)$$

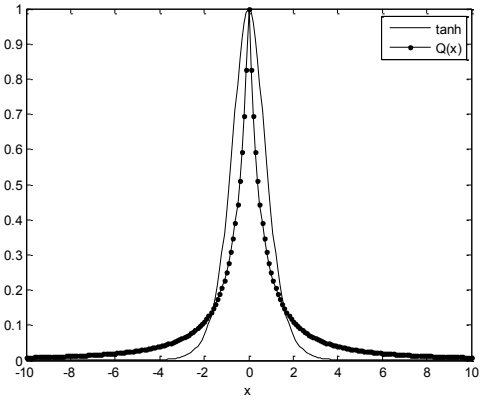
$$Q_{tanh}{}'(x) = \begin{cases} \frac{1}{(1-x)^2}, & x < 0 \\ \frac{1}{(1+x)^2}, & x \geq 0 \end{cases} \qquad 11)$$



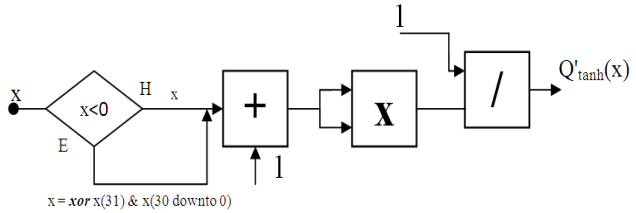
Şekil 10. tanh(x) ve $Q_{tanh}(x)$ fonksiyonlarının kıyaslaması



Şekil 11. Tanjant hiperbolik fonksiyonunun matematiksel yaklaşımının FPGA'da gerçekleme blok yapısı



Şekil 12. tanh(x) ve Qtanh(x) fonksiyonları türevlerinin kıyaslaması



Şekil 13. Tanjant hiperbolik yaklaşımı fonksiyonu türevinin FPGA'da gerçekleme blok yapısı

3.3 Gauss-1

Gauss fonksiyonu hem RBF tipi YSA ve hem de bulanık-nöral ağlarda kullanılanılır. Merkez (m), ve standart sapma (σ) parametreleri ile (12)'deki gibi tanımlanır.

$$Gauss(x) = e^{-(\frac{x-m}{\sigma})^2} \tag{12}$$

Gauss fonksiyonunu donanımsal gerçeklemek için, (9)'da verilen matematiksel yaklaşım [13] kullanılmıştır.

$$Q_{gauss}(x) = \frac{1}{1+\left|\frac{x-m}{\sigma}\right|^2} \tag{13}$$

Eşitlik 13'de verilen ifade 2 toplama, 1 çarpma ve 2 bölme modülü ile gerçeklenebilmektedir. Bölme modülünün FPGA'de gerçeklenme süresi, diğer modüllere göre daha uzun olduğu için, ifade sadece bir bölme modülü içerecek şekilde (14)'deki gibi düzenlenebilir.

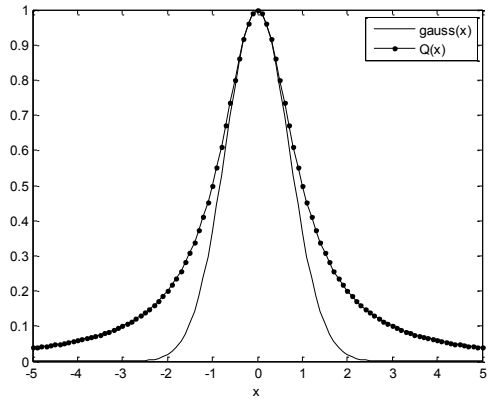
$$Q_{gauss}(x) = \frac{\sigma^2}{\sigma^2+(x-m)^2} \tag{14}$$

Şekil 14'de *Gauss(x)* ve $Q_{gauss}(x)$ fonksiyonlarının m = 0, σ=1 için [-5 5] aralığında davranışları verilmiştir. Şekil 15'da Gauss fonksiyonu yaklaşımının FPGA'de gerçekleme blok yapısı verilmiştir. Şekilden de görüleceği üzere matematiksel yaklaşım 2 toplama, 2 çarpma ve 1 bölme modülü ile gerçeklenebilmektedir. Gauss fonksiyonu için (14)'de verilen yaklaşımın türevi (15)'de verilmiştir. Bu türev ifadesinin orijinal fonksiyonun türevi ile karşılaştırması Şekil 16'dan yapılabilir. Gerçeklenmesi ise Şekil 17'de verilen blok yapı ile gösterilmiştir.

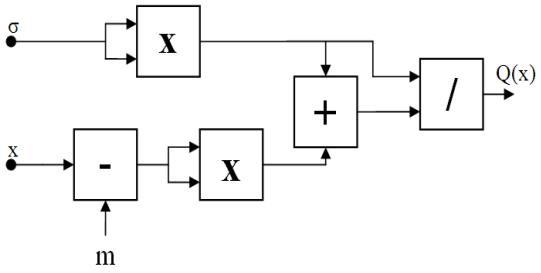
$$Q'_{gauss} = -\frac{2\sigma^2(x-m)}{(\sigma^2+(x-m)^2)^2} \tag{15}$$

3.4 Gauss-2

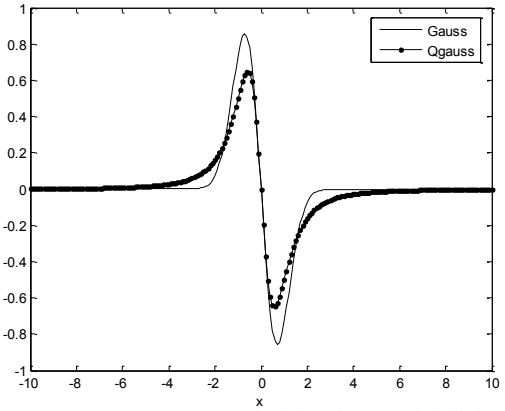
Yukarıda verilen Gauss aktivasyonu veya ÜF yaklaşımına alternatif olarak (16) ile tanımladığımız yaklaşım da kullanılabilir. Bu yaklaşımın en önemli özelliği çarpma modülü kullanmadan gerçeklenebilmesidir. Eşitlikten de görüleceği üzere 3 toplama ve 1 bölme modülü ile fonksiyon gerçeklenebilmektedir.



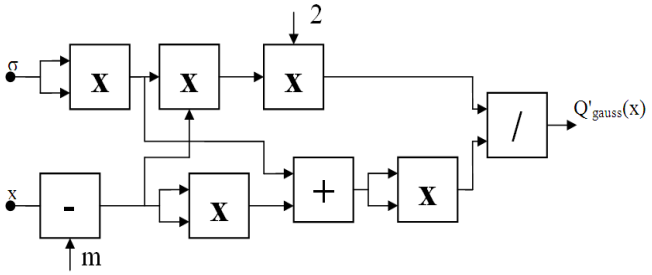
Şekil 14. Gauss(x) ve $Q_{gauss}(x)$ fonksiyonlarının kıyası



Şekil 15. Gauss fonksiyonu yaklaşımının FPGA'da gerçekleme blok yapısı



Şekil 16. Gauss(x) ve $Q_{gauss}(x)$ fonksiyonları türevlerinin kıyası

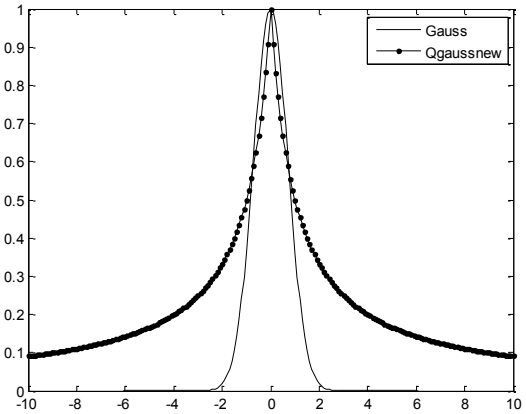


Şekil 17. Gauss yaklaşım fonksiyonunun türevinin FPGA'da gerçekleme blok yapısı

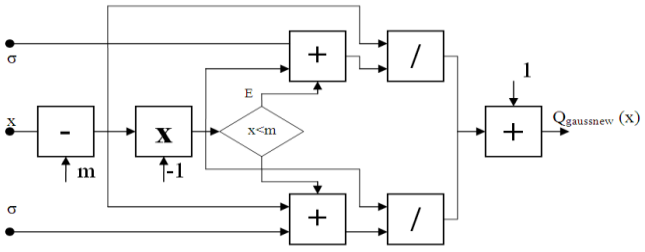
$$Q_{gaussnew}(x) = \begin{cases} 1 + \frac{(x-m)}{\sigma+(m-x)} & , \ x < m \\ 1 + \frac{(m-x)}{\sigma+(x-m)} & , \ x \geq m \end{cases} \quad (16)$$

Şekil 18'de *Gauss(x)* ve $Q_{gaussnew}(x)$ fonksiyonlarının m = 0, σ=1 için [-10 10] aralığında davranışları verilmiştir. Şekil 19'da da önerilen bu Gauss yaklaşım fonksiyonunun FPGA'de gerçekleme blok yapısı verilmiştir. Gauss fonksiyonu için (16)'da verilen yaklaşımın türevi (17)'de verilmiştir. Bu türev ifadesinin orijinal fonksiyonun türevi ile karşılaştırması Şekil 20'den yapılabilir. Gerçeklenmesi ise Şekil 21'de verilen blok yapı ile gösterilmiştir.

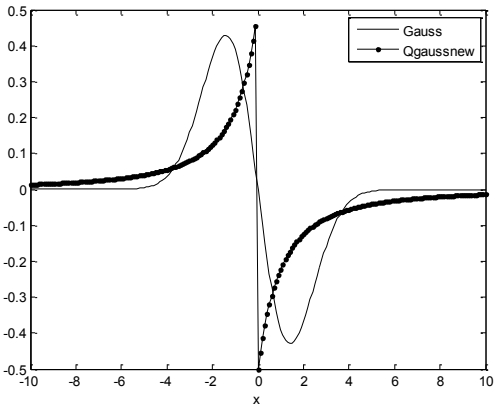
$$Q'_{gaussnew}(x) = \begin{cases} \frac{\sigma}{(\sigma+m-x)^2} & , \ x < m \\ -\frac{\sigma}{(\sigma+x-m)^2} & , \ x \geq m \end{cases} \quad (17)$$



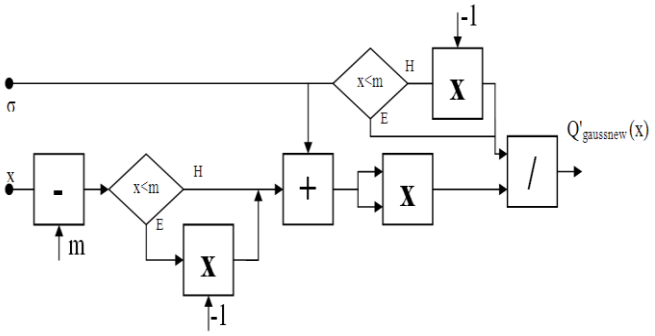
Şekil 18. $Q_{gaussnew}(x)$ fonksiyonun m = 0, σ=1 için karşılaştırmalı davranışı



Şekil 19. Alternatif Gauss yaklaşımının FPGA'da gerçekleme blok



Şekil 20. $Q_{gaussnew}(x)$ fonksiyonu türevinin m = 0, σ=1 için [-10 10] aralığında karşılaştırmalı davranışı



Şekil 21. $Q_{gaussnew}(x)$ Gauss yaklaşımı türevinin FPGA'da gerçekleme blok şeması

4. DENEYSEL SONUÇLAR

Bölüm 3'de verilen aktivasyon fonksiyon ve MF yaklaşımları, Xilinx Virtex 5 xc5vlx110-3ff1153 FPGA'sında gerçeklenmiştir. Bu yaklaşımların donanımsal gerçeklenmesinde kullanılan işlem modülleri Tablo 1'de verilmiştir. Tablodan da görüleceği üzere, tanjant hiperbolik yaklaşım 1 toplama ve 1 bölme, logaritmik sigmoidal matematiksel yaklaşımı 2 toplama ve 1 bölme, gauss matematiksel yaklaşımı 2 toplama, 2 çarpma ve 1 bölme modülü ile gerçeklenmiştir.

Tablo 1. Donanımsal gerçeklemede kullanılan aritmetik işlem modül sayıları

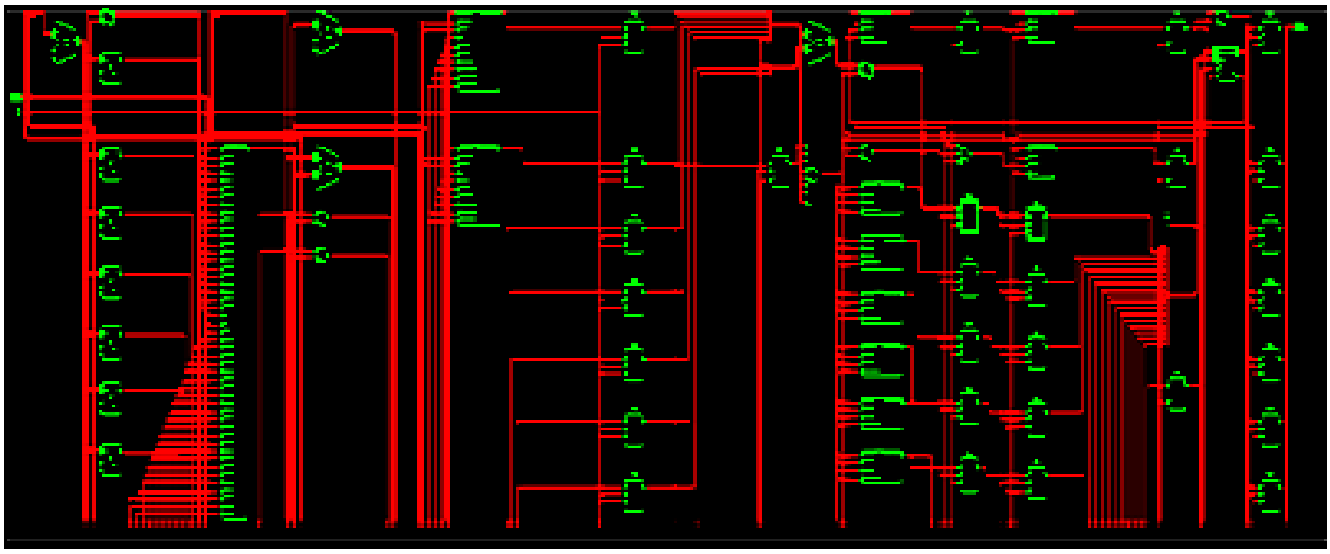
	Toplama	**Çarpma**	**Bölme**
Q_{tanh}	**1**	**-**	**1**
Q_{logsig}	**2**	**-**	**1**
Q_{gauss}	**2**	**2**	**1**
$Q_{gaussnew}$	**3**	**-**	**1**
Q_{tanh}'	**1**	**1**	**1**
Q_{logsig}'	**1**	**1**	**1**
Q_{gauss}'	**2**	**4**	**1**
$Q_{gaussnew}'$	**2**	**1**	**1**

Tablo 2'de tanjant hiperbolik, logaritmik sigmoidal ve Gauss fonksiyonları yaklaşımlarının FPGA üzerinde donanımsal gerçeklenmesine ait sentez sonuçları verilmiştir. Tablodan da görüleceği üzere önerilen yaklaşımlar %0.9 - %2.7 aralığında donanımsal kaynak tüketimi ile gerçeklenebilmiştir.

Şekil 22'de örnek olması açısından tanjant hiperbolik fonksiyonu yaklaşımı gerçeklemesi için sentezlenen devre verilmiştir.

Tablo 2. Sentez sonuçları

	Slice Registers (69120)	Slice LUTs (69120)
Q_{logsig}	392 (%0.476)	1406 (%2.034)
Q_{tanh}	328 (%0.474)	615 (%0.890)
Q_{gauss}	398 (%0.575)	1502 (%2.173)
$Q_{gaussnew}$	363 (%0.525)	2479(%3.586)
Q_{logsig} '	315(%0.455)	640 (%0.925)
Q_{tanh} '	315(%0.455)	640 (%0.925)
Q_{gauss} '	512(%0.740)	1593(%2.304)
$Q_{gaussnew}$ '	352(%0.509)	1823(%2.637)



Şekil 22. Tanjant hiperbolik fonksiyonu yaklaşımı gerçeklemesi için sentezlenen devre

5. SONUÇ ve TARTIŞMA

Bölüm 1'de belirtildiği üzere literatürde aktivasyon fonksiyonları yaklaşımı için parçalı doğrusal yaklaşım, parabolik yaklaşım ve bakma tablosu yaklaşımları, ÜF için ise ikinci dereceden fonksiyonel yaklaşım ve bazı bakma tablosu tabanlı yaklaşımlar önerilmiştir. Parçalı doğrusal yaklaşımla iyi bir aktivasyon fonksiyonu gerçeklemesi elde etmek için, çok sayıda parçalı doğrusal kontrol ifadesi kullanmak gereklidir. Bakma tablosu kullanılan yaklaşımlarda ise iyi bir yaklaşım için fazla hafıza birimi kullanımı gerekir. Benzer durum ÜF yaklaşımları için de geçerlidir.

Bu çalışmada önerilen yaklaşımlar, bakma tablosu yöntemindeki gibi hafıza gerektirmemeleri ve parçalı doğrusal yaklaşım yöntemindeki gibi kontrol ifadelerine ihtiyaç duymamaları üstünlüklerine sahiptir. Bu sayede donanım kaynaklarının daha etkin kullanılmasına olanak sağlarlar. Üstelik önerilen yaklaşımların türevlerinin de her hangi bir kısıtlı tanım aralığı olmaksızın orijinallerine yakın doğrusal olmayan seyir sergilemesi, nöral ve bulanık sistemlerin türev tabanlı eğitimleri ile birlikte FPGA'da gerçeklenmelerini de -düşük donanım kaynağı kullanımıyla- mümkün kılmaktadır. Tablo 2'de verilen sentez sonuçları önerilen yaklaşımların düşük donanım maliyetleriyle bahsedilen üstünlükleri sağlayacaklarının göstergesidir.

Bu çalışmanın bir diğer farklılığı ve katkısı da önerilen tüm yaklaşımların kayan noktalı sayı formatı kullanılarak gerçeklenmiş olmasıdır. Kayan noktalı sayılar hassas hesaplama yapmaya uygun dinamik yapıları sebebiyle özellikle tercih edilmiştir.

KAYNAKLAR

[1]. Martinez, J., Toledo, F.J., Fernandez, E. ve Ferrandez, J.M., "A retinomorphic architecture based on discrete-time cellular neural networks using reconfigurable

computing", **Neurocomputing**, Cilt 71, No 4-6, 766-775, 2008.

[2]. Nedjah, N, Silva, R.M.D., Mourelle, L.M.M, ve Silva, M.V.C.D., "Dynamic MAC-based architecture of artificial neural networks suitable for hardware implementation on FPGAs", **Neurocomputing**, Cilt 72, No 10-12, 2171-2179, 2009.

[3]. Ferreira, P., Ribeiro, P., Antunes, A. , ve Dias, F.M., "A high bit resolution FPGA implementation of a FNN with a new algorithm for the activation function", **Neurocomputing**, Cilt 71, No 1-3, 71-77, 2006.

[4]. Won E., "A hardware implementation of articial neural nertworks using field programmable gate arrays", **Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment**, Cilt 581, No 3, 816-820, 2007.

[5]. Ferrer, D., Gonzalez, R., Fleitas, R., Acle, J.P., ve Canetti, R., "NeuroFPGA - Implementing Artificial Neural Networks on Programmable Logic Devices", **Proceedings of the Design, Automation and Test in Europe Conference and Exhibition Designers' Forum (DATE'04)**, Cilt 3, 218-223, 2004.

[6]. Çavuşlu, M.A., Karakuzu, C., Şahin, S., *"Neural Networkwork Hardware Implementation Using FPGA",* **ISEECE 2006 3rd International Symposium on Electrical, Electronic and Computer Engineering Symposium Proceedings**, Nicosia, TRNC, 287-290, 2006.

[7]. Savich, A.W., Moussa, M., ve Areibi, S., *"The Impact of Arithmetic Representation on Implementing MLP-BP on FPGAs: A Study",* **IEEE Transactions on Neural Networks**, Cilt 18, No 1, 240 – 252, 2007.

[8]. Farmahini-Farahani, A., Fakhraie, S.M., ve Safari, S., "Scalable Architecture for on-Chip Neural Network Training using Swarm Intelligence", **Proc. of the Design, Automation and Test in Europe Conf. (DATE'08)**, Munich, Germany, 1340-1345, 2008.

[9]. Lin , C.J. ve Lee, C.Y. "FPGA Implementation of a Recurrent Neural Fuzzy Network with On-Chip Learning for Prediction and Identification Applications" , **Journal of Information Science and Engineering**, Cilt 25, No. 2, 575-589, 2009.

[10]. Blake, J. J. ve Maguire, L. P., "The implementation of fuzzy systems, neural Networks and fuzzy neural networks using FPGAs", **Information Sciences**, Cilt 112, No 1-4, 151-168, 1998.

[11]. Lin, C.-J. ve Tsai, H.-T. " FPGA implemengtation of a wavelet neural network with particle swarm optimization learning", **Mathematical and Computer Modeling**, Cilt 47, 982–996, 2008.

[12]. Juang, C.-F. ve Hsu, C.-H., "Temperature control by chip-implemented adaptive recurrent fuzzy controller designed by evolutionary algorithm," **IEEE Transactions on Circuits and Systems I: Regular Papers**, Cilt. 52, No. 11, 2376–2384, 2005.

[13]. Glackin, B., Maguire, L. P., ve McGinnity, T. M., "Intrinsic and extrinsic implementation of a bio-inspired hardware system", **Information Sciences**, Cilt 161, No 1-2, 1-19, 2004.

[14]. Campo, I., Echanobe, J., Bosque, G., ve Tarela, J. M., "Efficient hardware/software implementation of an adaptive neuro-fuzzy system," **IEEE Transactions on Fuzzy Systems**, Cilt 16, No.3, 761-778, 2008.

[15]. Mahyuddin, M.N., Wei, C.Z. ve Arshad, M.R., "FPGA as an Embedded System of a Mobile Robot with incorporated Neuro-Fuzzy Algorithm for Obstacle Avoidance Mission", **MASAUM Journal of Basic and Applied Sciences** (MJBAS), Cilt 1, No 3, 361-367, 2009.

[16]. Echanobe, J., del Campo, I., ve Bosque, G., "An adaptive neuro-fuzzy system for efficient implementations", **Information Sciences**, Cilt 178, No 9, 2150-2162, 2008.

[17]. Jang, J.-S. R., "ANFIS: adaptive-network-based fuzzy inference system", **IEEE Trans. on Systems, Man, and Cybernetics**, Cilt 23, No 3, 665-684, 1993.

[18]. Çavuşlu, M. A. , Karakuzu, C., Şahin, S., Yakut, M., "Neural Network Training Based on FPGA with Floating Point Number Format and It's Performance", **Neural Computing & Applications**, Cilt 20, No 2, 195–202, 2011.