BMBAD BJESR Mühendislik Bilimleri ve Araştırmaları Dergisi
Journal of Engineering Sciences and Researches

FPGA Tabanlı Eğitilebilir Sinirsel Ağ Modeli Tasarımı ve Benzetim Tabanlı Doğrulaması

FPGA-Based Trainable Neural Network Model Design and Simulation-Based Verification

[1]Abdelkader LAZZEM , [2]Halit ÖZTEKİN , [3]Feyzullah TEMURTAŞ , [4]Onursal ÇETİN

[1]Sakarya Uygulamalı Bilimler Üniversitesi, Elektrik-Elektronik Mühendisliği Bölümü, Serdivan/Sakarya, Türkiye
[2]Sakarya Uygulamalı Bilimler Üniversitesi, Bilgisayar Mühendisliği Bölümü, Serdivan/Sakarya, Türkiye
[3]Ostim Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü, Merkez/Ankara, Türkiye
[4]TÜRKSAT A.Ş., Uydu Teknolojileri Direktörlüğü, Uydu Ar-Ge Müdürlüğü, Gölbaşı/Ankara, Türkiye

[1]22500409008@subu.edu.tr,[2]halitoztekin@subu.edu.tr
[3]feyzullah.temurtas@ostimteknik.edu.tr, [4] onursal.cetin@turksat.com.tr

Araştırma Makalesi/Research Article

ARTICLE INFO

Article history

Received : 08 July 2026
Accepted : 05 October 2026

Keywords:

FPGA-Based Artificial Neural Network (ANN), Trainable Neural Network, 16-bit Floating-Point Arithmetic

ABSTRACT

This study presents a trainable ANN designed for a FPGA. The model uses a 2-2-1 topology comprising two inputs, two hidden neurons, and one output neuron. It employs 16-bit floating-point arithmetic and a Sigmoid activation function approximated by three linear segments and two saturation regions. Training uses backpropagation with momentum. The architecture follows a bottom-up design methodology and is implemented in VHDL using FSM-based control. Targeting the Altera DE2-70 development board, verification is performed eSxclusively through ModelSim VHDL simulations, without physical FPGA programming or measurements. The model is evaluated using the XOR classification problem. Results show that forward-propagation outputs agree with a software reference model and that backpropagation and weight updating operate at the hardware-description level. Synthesis, resource utilization, operating frequency, and power measurements are excluded; therefore, no quantitative claims about energy efficiency or processing performance are made.

© 2026 Bandirma Onyedi Eylul University, Faculty of Engineering and Natural Science. Published by Dergi Park. All rights reserved.

MAKALE BİLGİSİ

Makale Tarihleri

Gönderim : 8 Temmuz 2026
Kabul : 05 Ekim 2026

Anahtar Kelimeler:

FPGA Tabanlı Yapay Sinir Ağı (YSA), Eğitilebilir Sinir Ağı, 16-bit Kayan Noktalı Aritmetik

ÖZET

Bu çalışmada, FPGA üzerinde çalışmak üzere tasarlanmış eğitilebilir bir YSA modeli sunulmaktadır. Model, iki giriş, iki gizli nöron ve bir çıkış nöronundan oluşan 2-2-1 topolojisine sahiptir. Aritmetik işlemlerde 16-bit kayan noktalı sayı gösterimi, aktivasyon fonksiyonu olarak ise üç doğrusal segment ve iki doyum bölgesinden oluşan parçalı doğrusal Sigmoid yaklaşımı kullanılmaktadır. Ağ, momentum terimi içeren geri yayılım algoritmasıyla eğitilmektedir. Aşağıdan yukarıya tasarım yaklaşımıyla geliştirilen modüler mimari, FSM tabanlı kontrol kullanılarak VHDL dilinde gerçeklenmiştir. Altera DE2-70 geliştirme kartı hedeflenmiş, ancak doğrulama fiziksel FPGA üzerinde değil, yalnızca ModelSim ortamındaki VHDL benzetimleriyle gerçekleştirilmiştir. Model, XOR sınıflandırma problemi üzerinde doğrulanmıştır. Sonuçlar, ileri yayılım çıktılarının yazılım referans modeliyle uyumlu olduğunu ve geri yayılım/ağırlık güncelleme mekanizmasının çalıştığını göstermektedir. Sentez, kaynak kullanımı, çalışma frekansı ve güç tüketimi değerlendirilmemiş; enerji verimliliği veya işlem performansı hakkında nicel bir iddia sunulmamıştır.

© 2026 Bandırma Onyedi Eylül Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi. Dergi Park tarafından yayınlanmaktadır. Tüm Hakları Saklıdır.

ORCID ID: [1]0000-0003-0136-356X
[2]0000-0001-8598-4763
[3]0000-0002-3158-4032
[4]0000-0001-5220-3959

1. GİRİŞ

İnsan sinir sistemi, hastalıkların tanı ve tedavisine yönelik tıbbi araştırmaların yanı sıra biyolojik nöronların davranışlarının modellenmesi amacıyla uzun yıllardır bilimsel çalışmaların odağında yer almaktadır [1-4]. Biyolojik nöronların matematiksel olarak modellenmesi, hesaplamalı sinirbilim ve yapay zekâ (YZ) alanlarında önemli gelişmelere katkı sağlamıştır. Bu kapsamda Yapay Sinir Ağları (YSA), insan beyninin bilgi işleme mekanizmasından esinlenerek geliştirilen ve günümüzde geleneksel hesaplama yöntemlerine güçlü bir alternatif olarak kabul edilen YZ yaklaşımlarından biridir [5]. YSA'lar, biyolojik sinir sistemindeki nöronların çalışma prensibini taklit eden yapay nöronlardan oluşmaktadır. Birbirleriyle bağlantılı bu yapay nöronlar, giriş verilerini işleyerek öğrenme, sınıflandırma, tahmin ve karar verme gibi karmaşık görevleri yerine getirebilmektedir. Bu özellikleri sayesinde YSA'lar, YZ tabanlı bilgi işlem sistemlerinin temel bileşenlerinden biri hâline gelmiştir [6].
Son yıllarda, biyolojik sinir sistemlerinin yüksek hesaplama kapasitesi ve enerji verimliliğinden esinlenerek daha hızlı ve daha verimli YZ sistemlerinin geliştirilmesine yönelik çalışmalar önemli ölçüde artmıştır. Bu doğrultuda YSA'lar; finansal tahmin, endüstriyel süreç ve kalite kontrolü, tıbbi tanı, görüntü işleme ve örüntü tanıma gibi birçok uygulama alanında yaygın olarak kullanılmaktadır [7-11]. Geliştirilen sinir ağı modelleri çoğunlukla eğitim ve doğrulama aşamalarında Merkezi İşlem Birimi (CPU) veya Grafik İşlem Birimi (GPU) tabanlı platformlarda çalıştırılmaktadır. Ancak gerçek zamanlı uygulamalarda yüksek işlem hızı, düşük gecikme süresi ve enerji verimliliği gereksinimleri, donanım tabanlı gerçekleştirimlere olan ilgiyi artırmıştır [12]. Yazılım tabanlı çözümlerde bellek–işlemci veri aktarımından kaynaklanan darboğazlar literatürde tartışılmakta [13]; ancak bu çalışmada CPU/GPU platformlarıyla nicel bir karşılaştırma yapılmadığından, bu husus yalnızca donanım tabanlı gerçekleştirimlere yönelimin genel bir motivasyonu olarak anılmaktadır.
Bu sınırlamaların aşılabilmesi amacıyla YSA mimarilerinin donanım üzerinde gerçekleştirilmesine yönelik çalışmalar hız kazanmıştır. Donanım tabanlı yaklaşımlar, sinir ağlarının doğal paralellik özelliğinden yararlanarak öğrenme ve çıkarım işlemlerinin eş zamanlı yürütülmesine olanak sağlamaktadır [14,15]. Bu amaçla kullanılan donanım platformları arasında Alan Programlanabilir Kapı Dizileri (FPGA), yeniden yapılandırılabilir yapıları, paralel işlem yetenekleri, düşük güç tüketimleri ve farklı YSA mimarilerine kolaylıkla uyarlanabilmeleri nedeniyle öne çıkmaktadır [16]. Ayrıca FPGA tabanlı sistemler, donanım kaynaklarının uygulamaya özgü olarak optimize edilebilmesine olanak tanımaları nedeniyle YZ uygulamalarında yaygın olarak tercih edilmektedir [17,18].
Literatürde FPGA tabanlı YSA mimarilerine yönelik çok sayıda çalışma bulunmaktadır [12,19-24]. Bu çalışmalar, FPGA platformlarının paralel işlem yetenekleri, yeniden yapılandırılabilir yapıları ve yüksek işlem performansı sayesinde YSA'ların donanımsal olarak etkin bir şekilde gerçekleştirilebildiğini göstermektedir. Güncel literatürde bu yönelim; FPGA üzerinde derin öğrenme hızlandırıcıları [16,29], vurmalı (spiking) sinir ağı gerçekleştirimleri [21], eğitilebilir FPGA tabanlı denkleştiriciler [22], yeni nesil donanım hızlandırıcı mimarileri [23] ve donanımsal sinir ağı gerçekleştirimlerine ilişkin 2021–2026 dönemi çalışmalarıyla [24] sürdürülmektedir. Ayrıca düşük duyarlıklı eğitim üzerine yürütülen karma duyarlıklı eğitim [30], bfloat16 [31], 8 bit kayan nokta ile eğitim [32] ve FP8 formatları [33] gibi çalışmalar, dar bit genişlikli aritmetiğin öğrenme sürecindeki fırsat ve kısıtlarını ortaya koymaktadır. Geri yayılım algoritmasının bütünüyle çip üzerinde gerçekleştirilmesine ilişkin öncü çalışmalardan biri Ortega-Zamorano ve arkadaşlarının FPGA/mikrodenetleyici gerçekleştirimidir [34].
Bununla birlikte, eğitilebilir YSA modellerinin geliştirilmesinde, geri yayılım algoritmasının yoğun aritmetik işlem gereksinimi ve kayan noktalı hesaplamaların etkin biçimde gerçekleştirilmesi ihtiyacı nedeniyle eğitim süreci çoğunlukla CPU veya GPU tabanlı platformlarda yürütülmektedir. FPGA üzerinde geri yayılım algoritmasının gerçekleştirilmesi ise yüksek hesaplama karmaşıklığı, kayan noktalı aritmetik birimlerinin donanım maliyeti ve ağırlık güncelleme mekanizmasının donanım seviyesinde uygulanmasındaki tasarım zorlukları nedeniyle daha karmaşık bir süreç oluşturmaktadır [25]. Bu nedenle, eğitim mekanizmasını destekleyen modüler ve donanım kaynaklarını verimli kullanan FPGA tabanlı YSA mimarilerinin geliştirilmesi güncelliğini koruyan önemli araştırma konularından biri olmaya devam etmektedir.
Bu çalışmada, FPGA tabanlı eğitilebilir bir YSA mimarisi tasarlanmış ve VHDL dili kullanılarak gerçeklenmiştir. Tasarım, üzerinde Altera Cyclone II ailesinden bir FPGA bulunduran Altera DE2-70 geliştirme kartı hedef alınarak geliştirilmiş; doğrulama ise fiziksel kart üzerinde koşturma yapılmaksızın ModelSim ortamındaki VHDL benzetimleriyle sınırlı tutulmuştur. Geliştirilen mimari, 16-bit kayan noktalı sayı gösterimi kullanılarak tasarlanmış olup, Sigmoid aktivasyon fonksiyonu ve geri yayılım algoritmasını destekleyen modüler bir donanım yapısına sahiptir. Önerilen mimarinin işlevselliği, doğrusal olmayan sınıflandırma problemlerinin temel örneklerinden biri olan XOR problemi kullanılarak benzetim düzeyinde doğrulanmıştır.
Bu çalışmanın katkıları şu şekilde özetlenebilir:

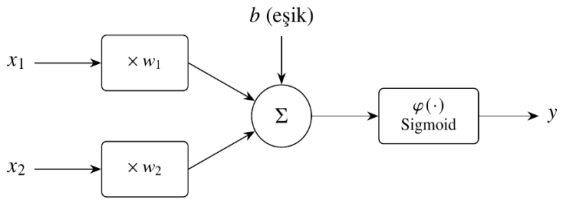
- İleri yayılım ile birlikte geri yayılım ve momentumlu ağırlık güncelleme mekanizmasının tamamının, harici bir işlemci desteği olmaksızın, FSM tabanlı özgün VHDL modülleriyle çip üzerinde (on-chip) gerçekleştirilmesi;
- 16-bit (IEEE 754 yarı duyarlık biçimli) kayan nokta toplayıcı, çarpıcı ve parçalı doğrusal Sigmoid aktivasyon birimlerinin, hazır IP çekirdeği kullanılmadan temel mantık bileşenlerinden (taşıma öngörülü toplayıcı, kaydır-topla çarpıcı) aşağıdan yukarıya inşa edilmesi;
- 2-2-1 topolojili eğitilebilir ağın XOR problemi üzerinde, yazılım referans modeliyle karşılaştırmalı olarak benzetim düzeyinde doğrulanması ve donanım–yazılım çıktı sapmalarının kaynaklarının tartışılması.

2. YÖNTEM

Bu bölümde, önerilen FPGA tabanlı YSA mimarisinin tasarım yöntemi sunulmaktadır. Aşağıdan yukarıya (bottom-up) tasarım yaklaşımı kapsamında, sistemi oluşturan temel bileşenler öncelikle bağımsız modüller hâlinde tasarlanmış ve doğrulanmış, ardından bu modüller bir araya getirilerek bütünleşik sistem mimarisi oluşturulmuştur. Tüm benzetimler ModelSim ortamında, davranışsal (RTL) düzeyde gerçekleştirilmiştir; VHDL kaynak kodları yazarlardan talep üzerine temin edilebilir.

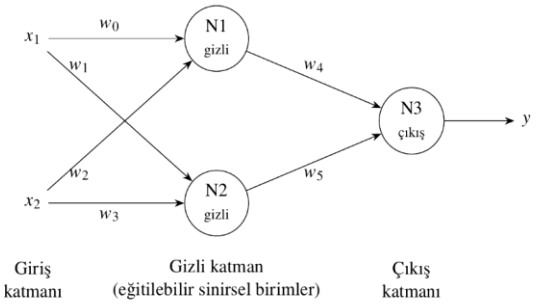
2.1. Önerilen Model Mimarisi

Donanım tasarımında aşağıdan yukarıya tasarım yaklaşımı benimsenmiştir. Bu yaklaşımda öncelikle yapay sinir ağının temel yapı taşlarını oluşturan tekil nöronlar ve bunlar arasındaki bağlantılar tasarlanmış, ardından bu bileşenler bir araya getirilerek sistem mimarisi oluşturulmuştur. Önerilen modele ait tek bir yapay nöronun blok diyagramı Şekil 1'de gösterilmektedir. Şekilde x_1 ve x_2 nöronun 16-bit kayan noktalı girişlerini, w_1 ve w_2 bu girişlere ait bağlantı ağırlıklarını, b bias (eşik) değerini, Σ ağırlıklı toplam işlemini, $\varphi(\cdot)$ Sigmoid aktivasyon fonksiyonunu ve y nöron çıkışını göstermektedir. Gerçeklenen tasarımda bias değeri nöron modülü içinde b = 0,5 (onaltılık 0x3800) sabitiyle tanımlıdır. Eşik değerinin sabit tutulması, her nöron için ek bir eşik güncelleme veri yolu ve kayıt gereksinimini ortadan kaldırarak donanım karmaşıklığını azaltmaya yönelik bilinçli bir tasarım sadeleştirmesidir; XOR problemi için gerekli karar sınırları yalnızca bağlantı ağırlıkları ayarlanarak da oluşturulabildiğinden bu sadeleştirme benimsenmiştir (bkz. Sınırlılıklar, 6. madde).



Şekil 1. Önerilen model için kullanılan tek bir nöronun blok diyagramı (x_1, x_2: girişler; w_1, w_2: bağlantı ağırlıkları; b: eşik değeri; Σ: ağırlıklı toplam; φ: Sigmoid aktivasyon fonksiyonu; y: nöron çıkışı).

FPGA tabanlı YSA modelinin genel blok diyagramı Şekil 2'de sunulmaktadır. Önerilen mimari; 2 giriş, 2 gizli nöron ve 1 çıkış nöronundan oluşan 2-2-1 topolojisine sahiptir. Gizli katmandaki N1 ve N2 nöronları her iki girişi de almakta, çıkış nöronu N3 ise gizli nöron çıkışlarını giriş olarak kullanmaktadır. Ağda toplam altı bağlantı ağırlığı (w_0, …, w_5) bulunmaktadır. Modüler tasarım yaklaşımı sayesinde nöron sayısı uygulama gereksinimlerine göre artırılabilmekte veya azaltılabilmektedir.

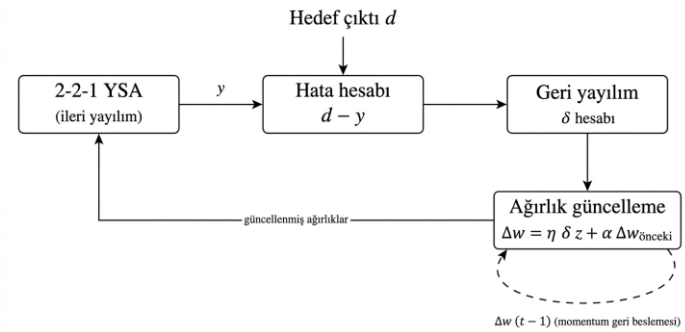


Şekil 2. Önerilen 2-2-1 topolojili modelin blok diyagramı.

Önerilen modelin eğitim mekanizması Şekil 3'te sunulmaktadır. Eğitim sürecinde, ağ çıktısı ile hedef çıktı arasındaki hata hesaplanmakta ve elde edilen hata değeri, öğrenme oranı (η = 0,5) ve momentum katsayısı (α = 0,8) dikkate alınarak geri yayılım algoritması ile ağırlıkların güncellenmesinde kullanılmaktadır.

Önerilen mimarinin donanımsal gerçekleştirilmesinde VHDL donanım tanımlama dili kullanılmıştır. Tasarım sürecinde, sistemin modüler, ölçeklenebilir ve yeniden kullanılabilir bir yapıda geliştirilebilmesi amacıyla modüler programlama yaklaşımı benimsenmiştir. Kontrol işlemlerinin gerçekleştirilmesinde sonlu durum makineleri (FSM) kullanılarak işlem akışı sistematik bir yapıda düzenlenmiştir. Aritmetik alt modüllerde bit genişliği VHDL generic parametreleriyle tanımlanmıştır (örneğin taşıma öngörülü toplayıcıda N parametresi); buna karşılık nöron modülünün giriş sayısı bu sürümde iki giriş için sabitlenmiş olup giriş sayısı bakımından parametrik değildir. Farklı topolojiler (örn. 2-4-1) için kaynak kullanımı ölçekleme verisi, sentez yapılmadığından bu çalışmada sunulamamaktadır; bu husus Bölüm 4'te sınırlılık olarak belirtilmiştir.

Modüllerin giriş/çıkış arayüzleri Tablo 1'de özetlenmiştir. VHDL kaynak kodundaki bazı sinyal adları Türkçe–İngilizce karışık biçimdedir (örn. hata_deger, hid_out, roc); okunabilirlik için bu adlar tabloda açıklanmıştır.



Şekil 3. Önerilen modelin eğitim mekanizması.

roc (rate of change) sinyali bir önceki yinelemedeki ağırlık değişimini (Δw(t−1)) tutmakta, updated_roc çıkışı ise momentum teriminde kullanılmak üzere hesaplanan güncel ağırlık değişimini (Δw(t)) bir sonraki yinelemeye aktarmaktadır.

Tablo 1. Tasarlanan VHDL modüllerinin arayüz özeti

Modül	Girişler	Çıkışlar / Açıklama
fp_adder	A, B (16 bit), op_sel (0: toplama, 1: çıkarma), clk, reset, start	sum (16 bit), done. FP16 toplama/çıkarma
multiplier	A, B (16 bit), clk, reset, start	sum (16 bit), done. FP16 çarpma; mantis çarpımı 11 bit kaydır-topla birimiyle.
activation	X (16 bit), clk, reset, start	Y (16 bit), done. Parçalı doğrusal Sigmoid
neuron	input1, input2, weight1, weight2, clk, reset, start	output (16 bit), done. İleri yayılım; bias dahili sabit (0,5)
update_weight (çıkış/gizli katman türevleri)	weight, output, hid_out (gizli katman çıkışı), expected (hedef), roc (Δw(t−1)), hata_deger (çıkış katmanı hata terimi δo), next_weight (bir sonraki katman ağırlığı), clk, start, reset	updated_weight (güncel ağırlık), updated_roc (Δw(t); momentum için), done. Geri yayılım ve ağırlık güncelleme
XOR_problem	input1, input2, clk, start, reset	output, done. Üst düzey modül: ileri yayılım + 10 eğitim yinelemesi

2.2. Gerçeklenen Ağın Matematiksel Modeli

Bu bölümde, VHDL kodunda gerçeklenen işlemlerle bire bir örtüşen matematiksel model verilmektedir. Gizli katmandaki j nöronunun (j ∈ {1, 2}) ağırlıklı toplamı ve çıkışı,

$$Net_j = w_{(1j)x_1} + w_{(2j)x_2} + b \tag{1}$$

$$h_j = \varphi(net_j) \tag{2}$$

şeklindedir; burada x_i girişleri, w_{ij} bağlantı ağırlıklarını ve b = 0,5 nöron modülünde sabit tanımlı bias değerini göstermektedir. Çıkış nöronu için

$$y = \varphi(w_{4h_1} + w_{5h_2} + b) \tag{3}$$

yazılır.
Sigmoid fonksiyonu $\varphi(x) = 1/(1+e^{-x})$, donanımda parçalı doğrusal (piecewise linear) yaklaşıklık ile gerçeklenmiştir (bkz. Bölüm 2.6, Denklem 10). Geri yayılımda Sigmoid türevi ayrı bir yaklaşıklıkla hesaplanmamakta; analitik özdeşlik olan

$$\varphi'^{(x)} = \varphi(x)(1 - \varphi(x)) \approx y(1 - y) \tag{4}$$

biçimi, nöron çıkışı üzerinden doğrudan FP16 çarpıcı ve toplayıcı birimleriyle hesaplanmaktadır (kodda 1 − y terimi, fp_adder birimine 0x3C00 = 1,0 sabiti ve çıkarma kipi op_sel = 1 verilerek elde edilir).
Eğitimde örtük kayıp fonksiyonu karesel hatadır:

$$E = \frac{1}{2}(d - y)^2 \tag{5}$$

burada d hedef (beklenen) çıktıdır. Çıkış katmanı hata terimi (delta),

$$\delta_o = y(1 - y)(d - y) \tag{6}$$

olarak; gizli katman j nöronunun hata terimi ise çıkış katmanından geriye iletilen hata kullanılarak

$$\delta_{h_j} = h_j(1 - h_j)(\delta_o w_{j\to o}) \tag{7}$$

olarak hesaplanır; wj→o, j gizli nöronunu çıkış nöronuna bağlayan ağırlıktır (kodda next_weight girişi). Ağırlık güncelleme, momentum terimli gradyan iniş kuralıyla yapılır:

$$\Delta w_{ij}(t) = \eta \delta_j z_i + \alpha \Delta w_{ij}(t-1) \tag{8}$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \tag{9}$$

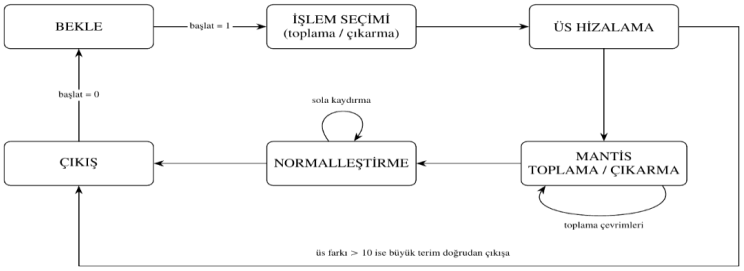
Burada z_i ilgili bağlantının giriş işareti (çıkış katmanı için gizli nöron çıkışı h_j, gizli katman için ağ girişi x_i), η öğrenme oranı ve α momentum katsayısıdır. Gerçeklenen tasarımda bu sabitler VHDL generic tanımlarıyla η = 0,5 (0x3800) ve α = 0,8 (0x3A66; FP16 gösteriminde 0,79980…) olarak belirlenmiş, tüm $\Delta w_{ij}(0)$ değerleri sıfırla ilklendirilmiştir. Denklem (1)–(9), ekli VHDL modüllerindeki işlem sırasıyla bire bir örtüşmektedir: fark sinyali (d − y)'yi, error sinyali δ'yı, roc/updated_roc sinyalleri Δw(t−1)/Δw(t)'yi taşımaktadır.

2.3. Tam Toplayıcı

Aritmetik işlem birimlerinin gerçekleştirilmesinde temel yapı taşı olarak kullanılmak üzere ilk aşamada bir tam toplayıcı modeli tasarlanmıştır. Tek bitlik kısmi tam toplayıcı hücreleri, taşıma öngörülü (carry look-ahead) yapıda birleştirilerek N bitlik parametrik bir toplayıcı oluşturulmuş ve bu birim, 16-bit kayan noktalı toplayıcının gerçekleştirilmesinde temel bileşen olarak kullanılmıştır.

2.4. 16-bit Kayan Nokta Toplayıcı

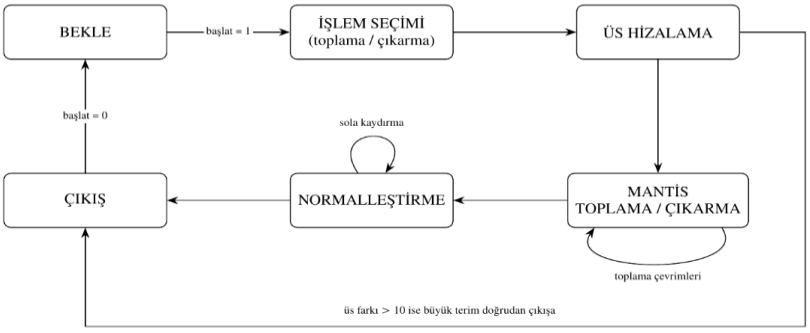
Toplama işlemlerinin gerçekleştirilmesi amacıyla FSM tabanlı 16-bit kayan noktalı bir toplayıcı tasarlanmıştır. Geliştirilen birim, iki 16-bit kayan noktalı girişin (A ve B) IEEE 754 yarı duyarlıklı gösterime uygun olarak toplanmasını (op_sel girişiyle çıkarılmasını) gerçekleştirmektedir. Toplama işlemi; üslerin hizalanması, mantislerin toplanması veya çıkarılması, sonucun normalleştirilmesi ve nihai çıktının oluşturulması adımlarından oluşmaktadır. Tasarlanan birime ait durum diyagramı Şekil 4'te verilmiştir.



Şekil 4. 16-bit kayan noktalı toplama/çıkarma modelinin durum diyagramı.

2.5. 16-bit Kayan Nokta Çarpma

Çarpma işlemlerinin gerçekleştirilmesi amacıyla FSM tabanlı 16-bit kayan noktalı bir çarpıcı tasarlanmıştır. Önerilen birim, iki 16-bit kayan noktalı girişin işaret, üs ve mantis bileşenlerini işleyerek kayan noktalı çarpma işlemini ardışık durumlar hâlinde gerçekleştirmektedir. İşlem süreci, başlangıç sinyalinin alınmasının ardından giriş verilerinin değerlendirilmesiyle başlamakta ve öncelikle sıfır giriş durumları kontrol edilmektedir. Ardından üs değerleri toplanmakta, mantis çarpımı 11 bitlik kaydır-topla (shift-add) çarpma birimiyle gerçekleştirilmekte ve elde edilen sonuç normalleştirilerek uygun kayan noktalı formatta çıkışa aktarılmaktadır. Tasarlanan çarpıcıya ait durum diyagramı Şekil 5'te gösterilmektedir.



Şekil 5. 16-bit kayan noktalı toplama/çıkarma modelinin durum diyagramı.

2.6. Aktivasyon Fonksiyonu

Aktivasyon fonksiyonu, yapay sinir ağlarında nöron çıkışına doğrusal olmayan bir dönüşüm uygulanmasını sağlayarak ağın karmaşık ilişkileri öğrenebilmesine olanak tanımaktadır. Bu kapsamda, Sigmoid aktivasyon fonksiyonunu gerçekleyen FSM tabanlı bir donanım modülü geliştirilmiştir.

VHDL gerçekleştirimi incelendiğinde kullanılan yöntem; arama tablosu (LUT), Taylor açılımı veya CORDIC değil, üç doğru parçası ve iki doyum bölgesinden oluşan parçalı doğrusal (piecewise linear) yaklaşıklıktır. Aktivasyon modülü, 16-bit kayan noktalı giriş x'in işaret, üs ve mantis alanlarını karşılaştırarak girişin hangi aralıkta olduğunu belirlemekte; doyum bölgelerinde çıkışa doğrudan sabit değer atamakta, ara bölgelerde ise $\varphi pd(x) = a_i x + b_i$ biçimindeki doğru denklemini FP16 çarpıcı ve toplayıcı modülleriyle hesaplamaktadır. Segment katsayıları VHDL kodunda FP16 sabitleri olarak gömülüdür ve yaklaşıklık şu şekildedir:

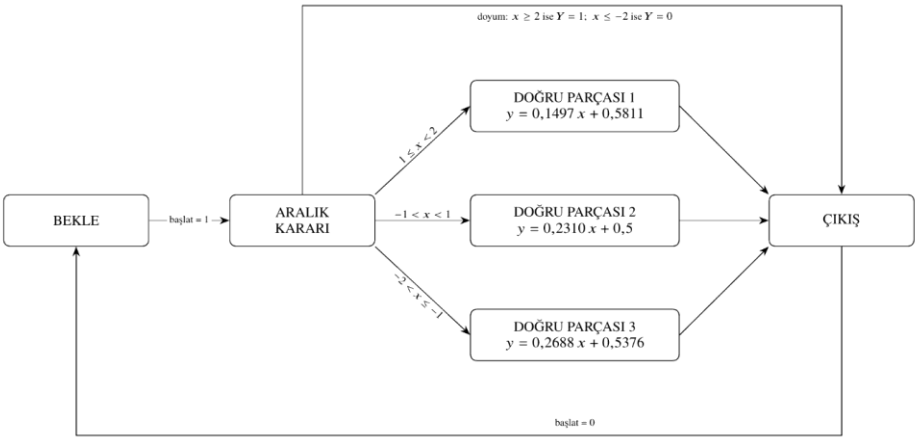
$$\varphi_{pd}(x) = \begin{cases} 0, & x \le -2, \\ 0{,}2688x + 0{,}5376, & -2 < x \le -1, \\ 0{,}2310x + 0{,}5000, & -1 < x < 1, \\ 0{,}1497x + 0{,}5811, & 1 \le x < 2, \\ 1, & x \ge 2 \end{cases} \tag{10}$$

Segment sınırları |x| = 1 ve |x| = 2 eşikleri olup, bu eşikler FP16 üs–mantis karşılaştırmasıyla (0x3C00 = 1, 0x4000 = 2) belirlenmektedir. Gerçek Sigmoid fonksiyonuna göre yaklaşıklık hatası (katsayıların FP16 çözünürlüğündeki değerleri üzerinden analitik olarak hesaplanmıştır): |x| < 1 bölgesinde en büyük mutlak hata ≈ 0,007, 1 ≤ x < 2 bölgesinde ise ≈ 0,012'dir. Negatif taraftaki doğru parçası (−2, 0) ve (−1, 0,2688) noktalarından geçecek biçimde seçildiğinden ve |x| ≥ 2 bölgesi 0/1 değerlerine doyurulduğundan, x = ±2 sınırlarında gerçek Sigmoid ile yaklaşık fonksiyon arasındaki en büyük mutlak hata yaklaşık 0,119 değerine ulaşmaktadır. Özellikle x = 2'de yaklaşık fonksiyonun doyum değeri 1 olduğundan hata |1 − ϕ(2)| ≈ 0,119 olurken, x = 2'ye soldan yaklaşırken doğrusal parçanın değeri 0,8805 olup gerçek Sigmoid değerine oldukça yakındır. Benzer şekilde, x = −2'de yaklaşık fonksiyon 0 değerine doyarken gerçek Sigmoid değeri yaklaşık 0,1192'dir. [−4, 4] aralığı üzerinden hesaplanan ortalama mutlak hata ise yaklaşık 0,036'dır. Bu hata düzeyi, XOR probleminde sınıf ayrımı için yeterli olmakla birlikte, Tablo 4'teki donanım–yazılım sapmalarının bileşenlerinden biridir (bkz. Bölüm 3).

Yaklaşık fonksiyonun doyum geçişleri, pozitif ve negatif tarafta aynı süreklilik davranışını göstermemektedir. Negatif tarafta üçüncü doğru parçası (−2, 0) noktasından geçtiğinden, x = −2'de yaklaşık fonksiyon ile doyum bölgesi arasında süreklilik sağlanmaktadır; ancak gerçek Sigmoid değeri ile yaklaşık değer arasındaki hata yaklaşık 0,119'dur. Pozitif tarafta ise doğru parçası x = 2'ye soldan yaklaşırken yaklaşık 0,8805 değerine ulaşmakta ve x = 2'de doyum değeri olan 1'e geçmektedir. Bu geçiş, yaklaşık fonksiyon açısından 0,1195 büyüklüğünde bir sıçramaya neden olmaktadır. Tablo 2'nin son sütununda verilen en büyük hata değerleri bu sınır davranışından kaynaklanmaktadır. Aktivasyon fonksiyonu modeline ait durum diyagramı Şekil 6'da gösterilmektedir.

Tablo 2. Farklı Parçalı doğrusal Sigmoid yaklaşıklığının segmentleri ve gerçek fonksiyona göre analitik hataları

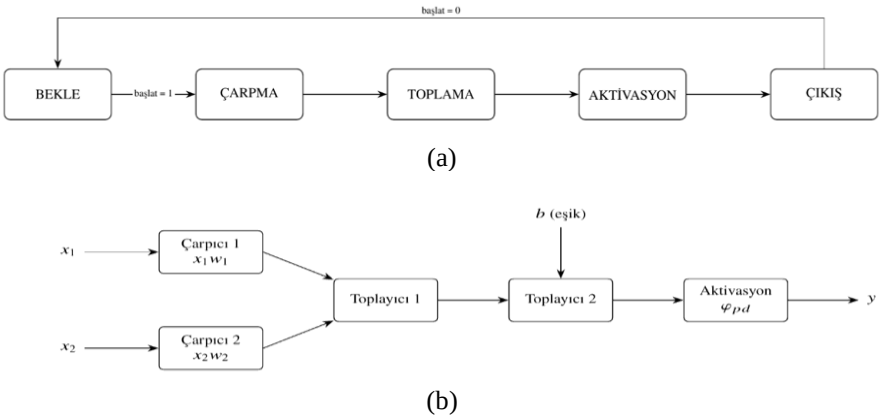
Aralık	İfade	FSM durumu	En büyük mutlak hata
x ≤ −2	0	doyum (sabit)	0,119 (x → −2)
−2 < x ≤ −1	0,2688x + 0,5376	FUNCTION3	0,119 (x → −2)
−1 < x < 1	0,2310x + 0,5000	FUNCTION2	0,007
1 ≤ x < 2	0,1497x + 0,5811	FUNCTION1	0,012
x ≥ 2	1	doyum (sabit)	0,119 (x → 2)
x ≤ −2	0	doyum (sabit)	0,119 (x → −2)



Şekil 6. Aktivasyon fonksiyonu modelinin durum diyagramı.

2.7. Aktivasyon Fonksiyonu

Temel donanım bileşenlerinin tasarlanmasının ardından, bu bileşenler bütünleşik bir yapay nöron modeli oluşturacak şekilde bir araya getirilmiştir. Geliştirilen nöron modeli, iki giriş (input1, input2), bu girişlere ait ağırlıklar (weight1, weight2) ile bir bias değerini kullanarak ileri yayılım işlemini gerçekleştirmektedir. İlk aşamada girişler, kayan noktalı çarpıcı modülleri kullanılarak ilgili ağırlıkları ile çarpılmaktadır. Bu iki çarpma işlemi, iki ayrı çarpıcı örneği üzerinde eş zamanlı olarak başlatılmaktadır; modül içi işlemler ise FSM denetiminde çok çevrimli ardışık adımlarla yürütülmektedir. Elde edilen ara sonuçlar 16-bit kayan noktalı toplayıcı yardımıyla toplanmakta ve bias değeri bu toplama işlemine dâhil edilmektedir. Ardından hesaplanan toplam, aktivasyon fonksiyonuna uygulanarak nöronun çıkış değeri elde edilmektedir. Hesaplama tamamlandığında done sinyali etkinleştirilerek nöron çıkışının hazır olduğu bildirilmektedir. Nöron modeline ait durum ve blok diyagramları sırasıyla Şekil 7(a) ve (b)'de gösterilmektedir.



Şekil 7. Nöron hücre modelinin durum (a) ve blok (b) diyagramı.

2.7. Geri Yayılım Algoritması

16-bit kayan nokta seçimi ve sayısal davranış: Aritmetikte IEEE 754 yarı duyarlık (1 işaret + 5 üs + 10 mantis biti) alan düzeni kullanılmıştır. Bu seçim; 32-bit kayan noktaya göre çarpıcı ve toplayıcı veri yollarını yarıya indirerek kaynak maliyetini düşürmesi, sabit noktalı gösterime göre ise geniş dinamik aralık sunarak ölçek seçimi gereksinimini ortadan kaldırması nedeniyle tercih edilmiştir. Literatürde 16-bit ve altı duyarlıkla eğitimde bilinen kısıtlar (dar dinamik aralık, küçük gradyanların alt taşması ve uzun eğitimlerde hata birikimi) raporlanmıştır [30-33]; bu kısıtlar XOR gibi küçük ölçekli bir problemde sınırlı etkiye sahip olmakla birlikte, daha büyük ağlara geçişte kayıp ölçekleme gibi tekniklerin gerekebileceği değerlendirilmektedir. Gerçeklenen birimlerde denormal sayılar, NaN ve sonsuz özel durumları desteklenmemekte (mantiste örtük "1" her durumda varsayılmakta), sıfır girişler çarpıcıda açık biçimde denetlenmekte, üs taşması durumunda üs alanı doyuma götürülmektedir. Yuvarlama, mantisin 10 bite kırpılması (sıfıra doğru yuvarlama) ile yapılmaktadır. Tablo 4'teki 0,742232 → 0,742 türü sapmaların baskın kaynağı bu mantis kırpmasıdır; yazılım referansı, Denklem (10)'daki parçalı doğrusal modeli tam duyarlıkla hesapladığından aktivasyon yaklaşıklığı bu farka ayrıca katkı vermemektedir (her iki taraf da aynı yaklaşıklığı kullanmaktadır).

3. ANALİZ VE BULGULAR

Önerilen FPGA tabanlı YSA mimarisinin işlevselliğini benzetim düzeyinde değerlendirmek amacıyla doğrusal olmayan sınıflandırma problemlerinin temel örneklerinden biri olan XOR problemi kullanılmıştır. XOR problemi, doğrusal olarak ayrıştırılamayan yapısı nedeniyle yapay sinir ağlarının öğrenme yeteneklerinin değerlendirilmesinde yaygın olarak kullanılan bir kıyaslama problemidir [27]. Tüm sonuçlar ModelSim ortamında yürütülen RTL benzetimlerinden elde edilmiştir; tasarım fiziksel FPGA kartına yüklenmemiş ve kart üzerinde herhangi bir ölçüm yapılmamıştır.

Benzetimlerde kullanılan başlangıç ağırlıkları ve bias değerleri, VHDL üst modülünde tanımlı FP16 sabitleridir ve Tablo 3'te hem onaltılık kodlarıyla hem de ondalık karşılıklarıyla eksiksiz olarak verilmiştir. Öğrenme oranı $\eta = 0{,}5$, momentum katsayısı $\alpha = 0{,}8$ ve tüm $\Delta w(0) = 0$'dır.

Eğitim döngüsü üst düzey XOR_problem modülündeki FSM tarafından yürütülmektedir: uygulanan her giriş kombinasyonu için önce bir ileri yayılım yapılmakta, ardından çıkış ve gizli katman ağırlık güncelleme modülleri sırayla çalıştırılarak bu adımlar toplam 10 eğitim yinelemesi boyunca tekrarlanmaktadır. Durdurma ölçütü bir hata eşiği değil, donanımda sabitlenmiş bu yineleme sayısıdır. Tam yakınsamaya (çıktıların 0/1 hedeflerine istenen toleransla yaklaşmasına) kadar geçen epok sayısına ilişkin bir ölçüm bu çalışmada yapılmamıştır; bu husus Bölüm 4'te sınırlılık olarak belirtilmiştir.

Tablo 3. Başlangıç ağırlık ve bias değerleri (FP16)

Parametre	Bağlantı	FP16 (onaltılık)	Ondalık değer
w_0	x_1 → gizli N1	0x3028	+0,1299
w_1	x_1 → gizli N2	0x3890	+0,5703
w_2	x_2 → gizli N1	0xBB62	−0,9229
w_3	x_2 → gizli N2	0xB543	−0,3289
w_4	gizli N1 → çıkış N3	0x3145	+0,1647
w_5	gizli N2 → çıkış N3	0x3A05	+0,7524
b	tüm nöronlarda (modül içi sabit)	0x3800	+0,5

Bu çalışmada özgün gönderimde yer alan XOR doğruluk tablosu, standart ders bilgisi niteliğinde olduğundan çıkarılmıştır; XOR hedef değerleri Tablo 4'ün "XOR hedefi" sütununda verilmektedir.
Tablo 4'te, başlangıç ağırlıkları kullanılarak gerçekleştirilen *ilk* ileri yayılım işlemine ait sonuçlar sunulmaktadır. Özgün gönderimde "Beklenen sonuç" olarak adlandırılan sütun, XOR hedef değeri değil, aynı parçalı doğrusal modeli tam duyarlıkla hesaplayan yazılım referans çıktısıdır; tabloda bu sütun yeniden adlandırılmış, XOR hedefi ve hata için ayrı sütunlar eklenmiştir.

Tablo 4. Eğitim öncesi (başlangıç ağırlıklarıyla) ilk ileri yayılım sonuçları. Ağırlıkların tamamı Tablo 3'te verilmiştir; Hata sütunu, XOR hedefi ile benzetim çıktısı arasındaki farktır.

Deneme	Giriş 1	Giriş 2	XOR hedefi	Yazılım referansı	Benzetim çıktısı	Hata	|Yazılım − Benzetim|
1	1	1	0	0,742232	0,742	−0,742	0,000232
2	1	0	1	0,755826	0,755	+0,245	0,000826
3	0	1	1	0,724548	0,7242	+0,276	0,000348
4	0	0	0	0,740461	0,74	−0,740	0,000461

Tablo 4'teki çıktıların 0 veya 1 yerine 0,742 gibi ara değerler almasının iki nedeni vardır: (i) ağın çıkışı, ikili bir karar değil, Sigmoid aktivasyonunun [0, 1] aralığındaki sürekli değeridir; sınıf kararı ancak bir eşikleme (örn. $y > 0,5$) ile elde edilir; (ii) tablo, ağın henüz eğitilmemiş hâlinin, yani rastgele seçilmiş başlangıç ağırlıklarıyla yapılan ilk ileri yayılımın (eğitim öncesi tek geçişin) çıktısını göstermektedir. Bu aşamada dört girişin çıktılarının birbirine yakın ve $\approx 0,74$ civarında olması beklenen bir durumdur; model bu tabloda "öğrenmiş" olduğu iddiasıyla değil, ileri yayılım veri yolunun yazılım referansıyla tutarlılığını göstermek amacıyla sunulmaktadır. Sayı gösterimi ve FP16 duyarlılığına ilişkin sınırlamalar için 4. SINIRLILIKLAR bölümünün 7. maddesine bakınız.
İlk ileri yayılım işleminden elde edilen sonuçlar, hedef değerlerle karşılaştırılarak hata değerleri hesaplanmıştır. Hesaplanan hata, geri yayılım algoritması kullanılarak ağ boyunca geriye iletilmiş ve bağlantı ağırlıkları öğrenme oranı ile momentum katsayısı dikkate alınarak güncellenmiştir. Benzetim çalışmalarında ağırlık güncelleme modüllerinin Denklem (6)–(9)'a uygun işlem yaptığı gözlemlenmiş; ancak 10 yineleme sonundaki çıktı ve ağırlık değerleri ile hata–yineleme eğrisi sayısal olarak kayıt altına alınmamıştır. Bu nedenle bu çalışmada "çıktıların hedef değerlere yakınsadığı" yönünde nicel bir iddia ileri sürülmemekte; eğitim sonrası davranışın nicel olarak raporlanması ileriki çalışmaya bırakılmaktadır.
Paralellik üzerine not. Tasarımda gizli katmandaki iki nöron, ayrı donanım örnekleri olarak eş zamanlı başlatılmakta ve birbirine paralel çalışmaktadır; benzer biçimde gizli katman ağırlık güncelleme modülleri de eş zamanlı yürütülmektedir. Bununla birlikte her modülün kendi içindeki işlemler FSM denetiminde çok çevrimli ardışık adımlarla gerçekleştirilmektedir. İleri yayılım ve eğitim yinelemesi başına çevrim sayısı ölçümü yapılmadığından, bu çalışmada donanımsal paralellikten kaynaklanan bir hızlanma iddiası ileri sürülmemektedir.
Önerilen tasarımın literatürdeki FPGA tabanlı XOR/eğitilebilir YSA çalışmalarıyla niteliksel karşılaştırması Tablo 5'te verilmiştir. Karşılaştırma; bit genişliği, eğitimin çip üzerinde yapılıp yapılmadığı, doğrulama problemi ve hedef cihaz eksenlerinde yapılmıştır. Bu çalışmada sentez gerçekleştirilmediğinden kaynak kullanımı (LUT, FF, DSP, BRAM), çalışma frekansı, saniyedeki ileri yayılım sayısı (PS) ve FPS/DSP, FPS/1K-LUT türü verimlilik metrikleri üzerinden nicel bir karşılaştırma yapılamamaktadır; ilgili sütunlar bilinçli olarak boş ("–") bırakılmış, değer uydurulmamıştır. Klasik (yazılım tabanlı) YSA ile nicel karşılaştırma da aynı nedenle kapsam dışıdır ve Bölüm 4'te sınırlılık olarak açıklanmıştır.

4. SINIRLILIKLAR

Bu çalışmanın sınırlılıkları şunlardır:
1. Fiziksel gerçekleştirim yapılmamıştır. Tasarım Altera DE2-70 kartı hedeflenerek geliştirilmiş, ancak sentez/yerleşim-rotalama akışı tamamlanmamış ve kart üzerinde koşturulmamıştır. Bu nedenle kaynak kullanımı (LUT, FF, DSP, BRAM), en yüksek çalışma frekansı, çevrim sayıları ve güç tüketimi verileri raporlanamamaktadır; enerji verimliliği ve işlem performansına ilişkin nicel bir iddia bulunmamaktadır.
2. Doğrulama yalnızca XOR problemiyle sınırlıdır. MNIST gibi gerçek veri kümeleri üzerinde deney yapılmamıştır; XOR, işlevsel doğruluğun ilk kanıtı olarak sunulmaktadır ve mimarinin genel başarımına ilişkin

Tablo 5. FPGA tabanlı YSA çalışmalarıyla niteliksel karşılaştırma

Çalışma	Sayı formatı	Eğitim çip üzerinde mi?	Doğrulama problemi	Hedef cihaz	Sonuç türü
Nichols vd. (2003) [25]	kayan nokta / sabit nokta karşılaştırması	evet (geri yayılım)	XOR	Xilinx FPGA	sentez + benzetim
Abrol & Mahajan (2015) [28]	–	hayır (yalnız ileri yayılım)	XOR (tek nöron)	Xilinx FPGA	sentez + benzetim
Ortega-Zamorano vd. (2016) [34]	sabit nokta	evet (geri yayılım)	çok sayıda kıyaslama kümesi	Xilinx Virtex-5	donanım ölçümü
Ney & Wehn (2024) [22]	–	evet	haberleşme denkleştirme	FPGA	donanım ölçümü
Bu çalışma	FP16 (1+5+10)	evet (geri yayılım + momentum)	XOR (2-2-1)	Altera DE2-70 (hedef)	yalnızca ModelSim benzetimi

bir genelleme yapılmamaktadır.
3. Eğitim sonrası nicel veriler kayıt altına alınmamıştır. Eğitim döngüsü donanımda 10 yinelemeyle sınırlı olup hata–yineleme eğrisi, eğitim sonrası ağırlık ve çıktı değerleri ile tam yakınsama için gereken epok sayısı raporlanamamaktadır.
4. Klasik (yazılım tabanlı) YSA ile nicel karşılaştırma yapılmamıştır. PS, FPS/DSP, FPS/1K-LUT türü metrikler sentez ve ölçüm gerektirdiğinden kapsam dışıdır.
5. Ölçeklenebilirlik nicel olarak gösterilmemiştir. Nöron modülü iki giriş için sabitlenmiştir; farklı topolojilerde (örn. 2-4-1) kaynak kullanımı ölçeklemesi sentez verisi gerektirdiğinden sunulamamaktadır.
6. Eşik (bias) değerleri eğitilmemektedir. Tüm nöronlarda eşik 0,5 sabitinde tutulmakta, geri yayılım yalnızca bağlantı ağırlıklarını güncellemektedir; bu sadeleştirme donanım karmaşıklığını azaltmakla birlikte ağın ifade gücünü sınırlayabilecek bir tasarım kısıtıdır.
7. Tablo 4'teki benzetim çıktılarının FP16 biçiminde sunulması, FP16 gösteriminin sınırlı duyarlılığı ve yuvarlama etkisi nedeniyle yazılım referansı ile benzetim çıktısı arasındaki farkın sıfır veya anlamsız hâle gelmesine yol açabileceğinden, çıktılar özgün ondalık gösterimleriyle korunmuştur. Gelecek çalışmalarda, sayısal doğruluk ve yuvarlama etkileri FP32 gibi daha yüksek duyarlılığa sahip sayı formatları kullanılarak daha ayrıntılı incelenebilir.

5. SONUÇ

Bu çalışmada, ileri yayılım, parçalı doğrusal Sigmoid aktivasyon, geri yayılım ve momentumlu ağırlık güncelleme mekanizmalarının tamamını FSM tabanlı özgün VHDL modülleriyle çip üzerinde gerçekleyen, 2-2-1 topolojili eğitilebilir bir YSA mimarisi sunulmuştur. Önceki bölümlerin bir özeti yerine, elde edilen somut bulgular şu şekilde değerlendirilmektedir: (i) 16-bit kayan noktalı toplayıcı, çarpıcı ve aktivasyon birimlerinden aşağıdan yukarıya inşa edilen ileri yayılım veri yolu, XOR probleminin dört giriş kombinasyonunda yazılım referans modeliyle en fazla binde birkaç düzeyinde sapma gösteren çıktılar üretmiştir (Tablo 4); bu sapmaların baskın kaynağı FP16 mantis kırpmasıdır. (ii) Geri yayılım modülleri, Denklem (6)–(9)'da verilen hesaplamaları donanım tanımı düzeyinde gerçeklemektedir. (iii) Doğrulama yalnızca ModelSim benzetimleriyle yapıldığından, mimarinin gerçek zamanlı uygulamalara uygunluğu, enerji verimliliği ve işlem performansı hakkında nicel bir sonuç bildirilmemektedir.
Gelecek çalışmalarda; tasarımın Quartus ortamında sentezlenerek, fiziksel bir karta ihtiyaç duyulmayan bu yazılım adımı üzerinden kaynak kullanımı, çalışma frekansı ve güç tüketimi sonuçlarının raporlanması; tasarımın fiziksel DE2-70 FPGA kartı üzerinde gerçeklenmesi; eğitim sürecinin ve eğitim sonrası davranışın hata–iterasyon eğrileri kullanılarak nicel olarak değerlendirilmesi; MNIST gibi yaygın veri kümeleri üzerinde yazılım referans modelleriyle karşılaştırmalı deneylerin gerçekleştirilmesi ve mimarinin daha fazla nöron ve gizli katman içerecek şekilde genişletilmesi planlanmaktadır. Ayrıca, 4. SINIRLILIKLAR bölümünün 7. maddesinde belirtilen sayı gösterimi ve FP16 duyarlılığına ilişkin sınırlamaların değerlendirilmesi amacıyla, FP32 gibi daha yüksek duyarlılığa sahip sayı formatları kullanılarak sonuçların karşılaştırılması planlanmaktadır.

Yazar Katkıları

Abdelkader LAZZEM: Yöntem, donanım/yazılım tasarımı, VHDL kodlaması, benzetim çalışmalarının gerçekleştirilmesi ve makalenin yazımı. Halit ÖZTEKİN: Kavramsallaştırma, danışmanlık, makalenin incelemesi ve düzenlemesi. Feyzullah TEMURTAŞ: Yöntem ve değerlendirme konularında danışmanlık ve makalenin gözden geçirilmesi. Onursal ÇETİN: Yöntem ve değerlendirme konularında danışmanlık ve makalenin gözden geçirilmesi.

Çıkar Çatışması

Makale yazarları aralarında herhangi bir çıkar çatışması olmadığını beyan ederler.

KAYNAKÇA

[1] M. Popescu, C. Ravariu, and F. Bulucea, "Neural circuits and their electronic models," in Proc. 2023 15th Int. Conf. Electronics, Computers and Artificial Intelligence (ECAI), Bucharest, Romania, , pp. 1–4, 2023

[2] T. Riis and J. Kubanek, "Effective ultrasonic stimulation in human peripheral nervous system," IEEE Trans. Biomed. Eng., vol. 69, no. 1, pp. 15–22, Jan. 2022.

[3] N. M. Safiannikov and P. N. Burenev, "Intelligent measuring system for implementation of the method of assessment of functional state of human central nervous system," in Proc. 2017 XX IEEE Int. Conf. Soft Computing and Measurements (SCM), St. Petersburg, Russia, pp. 577–579, 2017.

[4] P. Gahiwad, N. Deshmane, S. Karnakar, S. Mali, and R. Pise, "Brain stroke detection using CNN algorithm," in Proc. 2023 IEEE 8th Int. Conf. for Convergence in Technology (I2CT), Lonavla, India, pp. 1–4, 2023.

[5] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed. Hoboken, NJ, USA: Pearson, 2021.

[6] C. Aggarwal, "An introduction to neural networks," in Neural Networks and Deep Learning. Cham, Switzerland: Springer, 2023.

[7] M. Biswas, A. J. Nova, M. K. Mahbub, S. Chaki, S. Ahmed, and M. A. Islam, "Stock market prediction: A survey and evaluation," in Proc. 2021 Int. Conf. Science & Contemporary Technologies (ICSCT), Dhaka, Bangladesh, pp. 1–6, 2021.

[8] Y. E. Shao, C. H. Wu, B. Y. Ho, and B. S. Hsu, "Integration of multivariate control charts and neural networks to determine the faults of quality characteristic(s) in a multivariate process," in Proc. 11th Int. Conf. Computer Supported Cooperative Work in Design, Melbourne, VIC, Australia, pp. 1028–1031, 2007.

[9] A. V. Milov, V. S. Tynchenko, and A. V. Murygin, "Neural network modeling to control process of induction soldering," in Proc. 2019 Int. Conf. Industrial Engineering, Applications and Manufacturing (ICIEAM), Sochi, Russia, pp. 1–5, 2019.

[10] A. V. Kugaevskikh, "COVID-19 screening based on application of neural network classification of exhale spectra," in Proc. 2022 III Int. Conf. Neural Networks and Neurotechnologies (NeuroNT), St. Petersburg, Russia, pp. 24–27, 2022.

[11] M. S. Chvanova and I. A. Bakalets, "Neural network technology for quality control of the milk packaging process," in Proc. 2022 6th Scientific School Dynamics of Complex Networks and Their Applications (DCNA), Kaliningrad, Russia, pp. 73–74, 2022.

[12] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," Neurocomputing, vol. 74, no. 1–3, pp. 239–255, 2010.

[13] A. Shrestha, H. Fang, Z. Mei, D. P. Rider, Q. Wu, and Q. Qiu, "A survey on neuromorphic computing: Models and hardware," IEEE Circuits Syst. Mag., vol. 22, no. 2, pp. 6–35, 2022.

[14] M. Forssell, "Hardware implementation of artificial neural networks," Information Flow in Networks, course notes, course 18-859E, pp. 1–4, 2014.

[15] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A survey of neuromorphic computing and neural networks in hardware," arXiv:1705.06963, 2017.

[16] S. S. Lingala, S. Bedekar, P. Tyagi, P. Saha, and P. Shahane, "FPGA-based implementation of neural network," in Proc. 2022 Int. Conf. Advances in Computing, Communication and Applied Informatics (ACCAI), Chennai, India, pp. 1–5, 2022.

[17] P. D. Deotale and L. E. Dole, "Design of FPGA-based general-purpose neural network," in Proc. Int. Conf. Information Communication and Embedded Systems (ICICES), Chennai, India, pp. 1–5, 2014.

[18] P. Jawandhiya, "Hardware design for machine learning," Int. J. Artif. Intell. Appl., vol. 9, no. 1, pp. 63–84, 2018.

[19] J. Liu and D. Liang, "A survey of FPGA-based hardware implementation of ANNs," in Proc. 2005 Int. Conf. Neural Networks and Brain, Beijing, China, pp. 915–918, 2005.

[20] Z. Li, Y.-J. Huang, and W.-C. Lin, "FPGA implementation of neuron block for artificial neural network," in Proc. Int. Conf. Electron Devices and Solid-State Circuits (EDSSC), Hsinchu, Taiwan, 2017.

[21] M. Karamimanesh, E. Abiri, M. Shahsavari, K. Hassanli, A. van Schaik, and J. Eshraghian, "Spiking neural networks on FPGA: A survey of methodologies and recent advancements," Neural Networks, vol. 186, Art. no. 107256, 2025.

[22] J. Ney and N. Wehn, "Achieving high throughput with a trainable neural-network-based equalizer for communications on FPGA," in Proc. 2024 27th Euromicro Conf. Digital System Design (DSD), Paris, France, pp. 106–113, 2024.

[23] A. Kumar, "FPGA-based hardware accelerators for ANN and DNN," in Smart Chips for Smart Devices: VLSI Design for Next-Generation IoT Solutions. Hoboken, NJ, USA: Wiley, pp. 31–48, 2026.

[24] D. I. Skrebenkov and D. O. Budanov, "Hardware implementation of neural networks," in Proc. 2024 Int. Conf. Electrical Engineering and Photonics (EExPolytech), St. Petersburg, Russia, 2024.

[25] K. R. Nichols, M. A. Moussa, and S. M. Areibi, "Feasibility of floating-point arithmetic in FPGA-based artificial neural networks", 2003.

[26] S. Jeyanthi and M. Subadra, "Implementation of single neuron using various activation functions with FPGA," in Proc. 2014 IEEE Int. Conf. Advanced Communications, Control and Computing Technologies, pp. 1126–1131, 2014.

[27] S. Haykin, Neural Networks and Learning Machines, 3rd ed. Upper Saddle River, NJ, USA: Pearson Prentice Hall, 2009.

[28] S. K. Abrol and R. Mahajan, "Implementation of single artificial neuron using various activation functions and XOR gate on FPGA chip," in Proc. 2015 2nd Int. Conf. Advances in Computing and Communication Engineering, pp. 118–123, 2015.

[29] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "A survey of FPGA-based neural network inference accelerator," ACM Trans. Reconfigurable Technol. Syst., vol. 12, no. 1, pp. 1–26, 2019.

[30] P. Micikevicius et al., "Mixed precision training," in Proc. Int. Conf. Learning Representations (ICLR), 2018.

[31] D. Kalamkar et al., "A study of BFLOAT16 for deep learning training," arXiv:1905.12322, 2019.

[32] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2018.

[33] P. Micikevicius et al., "FP8 formats for deep learning," arXiv:2209.05433, 2022.

[34] F. Ortega-Zamorano, J. M. Jerez, D. Urda Muñoz, R. M. Luque-Baena, and L. Franco, "Efficient implementation of the backpropagation algorithm in FPGAs and microcontrollers," IEEE Trans. Neural Netw. Learn. Syst., vol. 27, no. 9, pp. 1840–1850, Sep. 2016.