SDU International Journal of Technological Science
Vol. 8, No 1, April 2016
pp. 34-52

YAPAY SİNİR AĞLARININ OTOMATİK OLARAK FPGA ÇİPİNE UYGULANMASI İÇİN OTOMATİK DENETLEYİCİ TASARIM ARACI

İbrahim ŞAHİN, Günay TEMÜR*

Geliş Tarihi/ Received: 15.01.2016, Kabul tarihi/Accepted: 10.03.2016

Özet

Yapay Sinir Ağları (YSA), insan beynindeki sinir hücrelerini ve bunların oluşturduğu sinir ağlarını örnek alarak oluşturulmuş matematiksel modellerdir. YSA'lar, paralel dağıtılmış veri işlemeye ihtiyaç duyduklarından dolayı yazılımın yetersiz kaldığı durumlarda donanım olarak gerçeklenirler. FPGA (Field Programmable Gate Arrays) çipleri, YSA'ları donanım olarak gerçeklemede paralel işlem yapabilme ve tekrar programlanabilme özelliklerinden dolayı en uygun seçenektir. Fakat bu gerçekleme işlemlerinin sorunsuz ve verimli bir şekilde yapılabilmesi için uzman gereksinimine ihtiyaç vardır. İnsan tarafından yapılan tasarımlar ve bunların gerçeklenmesi her zaman için hataya açıktır ve muhtemel hatalar neticesinde tasarımın en başına dönülmesine neden olmaktadır. Bu çalışmada, daha önceden geliştirilen FPGA tabanlı otomatik YSA tasarım sisteminin bir parçası olarak ANNCONT (**A**rtifical **N**eural **N**etwork **CONT**roller Tasarım Aracı) geliştirilmiştir. Bu sistem içinde ANNCONT, veri yolu tasarlanmış YSA için otomatik olarak denetleyici tasarımını yapmakta ve tasarladığı denetleyiciyi veri yolu ile birleştirerek YSA sistemini üretmektedir. ANNCONT çeşitli test durumları ile test edilmiş ve istenen YSA'lar için çok kısa bir sürede, hatasız bir şekilde YSA denetleyicilerini ve sistemlerini tasarladığı gözlenmiştir. ANNCONT sayesinde FPGA üzerinde YSA tasarım ve gerçekleme süreci kısalmış, uzman gereksinimi en aza inmiştir. Ayrıca ANNCONT hatasız kod ürettiğinden hata ayıklama süreci ortadan kalkmıştır.

Anahtar kelimeler: YSA, Denetleyici, FPGA, Tasarım Otomasyonu.

A CONTROLLER DESIGN TOOL DEVELOPMENT FOR AUTOMATICALLY MAPPING ARTIFICIAL NEURAL NETWORKS ONTO FPGAS

Abstract

Artificial Neural Networks (ANNs) are the mathematical models which are based on neurons (brain cells) and the network of the neurons. Since ANNs require parallel distributed calculations, they usually are implemented on hardware when software implementations do not provide sufficient performance. Field Programmable Gate Array (FPGAs) chips are the best implementation option for ANNs due to their parallel processing and reconfiguration ability. On the other hand, implementing ANNs on FPGAs is a time consuming process and requires expert personal. Design and implementations made by humans are error prone and errors make the designers go back and reconsider their design from the beginning. In this study, ANNCONT has been developed as a part of a beforehand developed FPGA based automatic ANN design system. In this system, ANNCONT is responsible for designing a controller for a given ANN data-path and forming an ANN system by integrating this controller with the given data-path. ANNCONT has been tested with several test cases. Our observations show that it is able to design controllers and generate VHDL code for both the controller and the ANN system in less than a second without any errors in the code. Using ANNCONT, design and implementation processes can be shortened in terms of time, and expert requirement is minimized. Moreover, since ANNCONT produces error free code, debugging stage is eliminated.

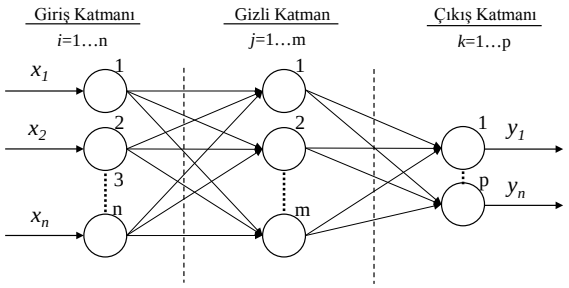
Key Words: ANN, Controller, FPGA, Design Automation.

* Kaynaşlı Meslek Yüksekokulu, Düzce Üniversitesi, Düzce, TÜRKİYE
E-posta: gunaytemur@duzce.edu.tr

1. Giriş

Canlı türlerinin beyinleri iki tür hücreden oluşmaktadır. Bu hücre türleri gail ve nöron hücreleri olarak adlandırılırlar. Gail hücreleri yapısal ve metabolik destek, yalıtım ve gelişme gibi birçok hayati görevi üstlenmiştir. Bunun yanında nöron hücreleri ise algılama, karar verme, hafızaya alma gibi asıl beyin aktivitelerinin gerçekleştirildiği hücrelerdir [1]. Bu hücreler birbirlerine akson'lar ve sinaps'lar aracılığıyla bağlanarak sinir hücresi ağlarını oluştururlar. Yapay Sinir Ağları (YSA) canlılardaki sinir hücrelerinin ve ağlarının matematiksel olarak modellenmesi ile oluşturulmuş yapılardır. Bu yapılar günümüzde karar verme, sınıflandırma, tahminde bulunma, kontrol gibi birçok alanda kullanılmaktadır.

Bilgi işlem yönünden bakıldığında YSA'lar ağırlıklı bağlantılar aracılığıyla birbirine bağlanan ve her biri kendi belleğine sahip işlem elemanlarından oluşan paralel ve dağıtılmış bilgi işleme yapılarıdır [2]. YSA'lar da bulunan her düğüm, n. dereceden tercihen lineer olmayan bir işlem birimidir. Düğümler arasında bağlantılar bulunmakla birlikte her bağlantı tek yönlü iletim yoludur. Bir düğüm birden fazla düğüme veri aktarabilir. İşlenen bilgiler (sonuç belirleme süreci) bir sonraki katmandaki bir veya birden fazla düğüme iletilir [3]. Sadece giriş ve çıkış katmanlarından oluşan ağlar, karmaşık işlemleri hesaplama yeteneklerine sahip değildirler. Bu sebeple karmaşık hesaplamalar için en az bir ara (gizli) katman olmalıdır. Şekil 1.1'de gizli katmana sahip 3 katmanlı bir yapay sinir ağı görülmektedir [4].



Şekil 1.1 Çok Katmanlı Yapay Sinir Ağı.

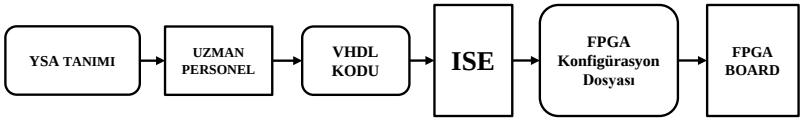
YSA'lar genellikle yazılım olarak modellenmekte ve kullanılmadan önce bir eğitim aşamasından geçmeleri gerekmektedir. Bu eğitim aşaması genellikle bir defaya mahsus olarak gerçekleştirilir. YSA'ların kullanıldığı bazı gerçek zamanlı sistemlerde, yazılımsal olarak gerçeklenen YSA yapıları istenen performansı verememektedirler. Bu gibi durumlarda daha hızlı çalışabilecek donanımsal YSA yapılarına ihtiyaç duyulur. Donanım olarak YSA'lar üç faklı şekilde gerçeklenebilir. Bunlar; ASIC (**A**pplication **S**pecific **I**ntegrated **C**ircuit; Uygulamaya Özel Tümleşik Devre) çiplerini, DSP (**D**igital **S**ignal **P**rocessing; Sayısal İşaret İşleme) işlemcilerini ya da FPGA çiplerini kullanarak gerçeklenmesidir. YSA'ları gerçeklemede ASIC kullanımı çok büyük çaplı üretimler için geçerlidir. Tasarımda yapılan bir hata çok büyük maliyetlere ve zaman kayıplarına neden olur. ASIC yaklaşımının en önemli dezavantajı da tasarımın sabit kalması yani sonradan değişiklik yapılamamasıdır. DSP çipleri ise sinyal işlemeye özel donanımlar içerir. DSP üzerinde gerçeklenen YSA'lar yine bir mikro işlemci mantığı ile komutlar çalıştırılarak gerçekleştirilebilir. YSA'lardaki paralel işlemler DSP'de tam olarak gerçeklenemez. Bu sebeplerden dolayı ASIC ve DSP'ye alternatif olarak FPGA çipleri kullanılmaktadır. Yeniden programlanabilme, paralel çalışabilme özellikleri sayesinde FPGA çipleri YSA'lar için en uygun gerçekleme ortamıdır [5]. Bir

FPGA'nın genel yapısı, programlanabilir dahili mantık bloğu dizisi, giriş-çıkış blokları ve bunlar arasında bağlantıları sağlayan yine programlanabilir ara bağlantılardan oluşmaktadır [6].

FPGA'lar genellikle yüksek seviyeli donanım tanımlama dilleri (Hardware Description Language, (HDL)) kullanılarak programlanırlar. Donanım tasarımı için yazılan program kodu derlenerek simülasyonu yapılır. Simülasyon aşamasında beklenilen sonuçlar alındıktan sonra sentezleme aşamasına geçilir. Sentezleyici, kullanıcının tasarlamış olduğu devreyi istenen FPGA elemanına göre sentezler. Bu aşamadan sonra yerleştirme ve yollandırma işlemleri yapılarak FPGA'ın programlanması için kullanılacak bit dizisi üretilir. FPGA'ın uygun donanımlar kullanılarak programlanmasıyla gerçekleme işlemi tamamlanır. HDL kodu yazıldıktan sonraki bütün bu işlemler üretici firma tarafından sağlanan yazılım araçları aracılığıyla yapılır [7] [8].

YSA'ların FPGA çipleri kullanılarak gerçeklenmesi konusunda literatürde birçok çalışma bulmak mümkündür [9] [10] [11] [12]. Bu çalışmalarda genellikle belirli bir amaca yönelik tasarlanan YSA yapısı performans kazancı sağlamak amacıyla FPGA çiplerine uygulanmıştır. FPGA'ler ile YSA'ların gerçeklenmesinde öncelikle her bir yapay sinir hücresinin bir donanım tanımlama dilinde (Hardware Description Language (HDL)) tanımlanması ve test edilmesi gereklidir. Ardından bu hücreler birbirine bağlanarak ağ oluşturulmalıdır. Oluşturulan ağın istenildiği gibi çalışması için ayrıca bir kontrol ünitesine ihtiyaç duyulur. Bu ünite kontrol edilecek her bir YSA için özel olarak tasarlanmalı ve YSA'ya entegre edilmelidir. Eğer YSA bir hafıza üzerinden veri okuyarak işlem yapacak ise bu işlemlere ek olarak uygun bir adres ünitesinin de tasarlanması ve sisteme dahil edilmesi gerekir. Tüm bu işlemler ASIC tasarımı kadar uzun sürmese de zaman alan, uzman gerektiren ve hataya açık süreçlerdir.

Şekil 1.2'de YSA'ların FPGA'de gerçeklenmesinde klasik tasarım akış diyagramı görülmektedir [13]. Bu akış diyagramında ISE, Xilinx firması tarafından geliştirilmiş bir tümleşik sentez ortamıdır. Uzman kişiler tarafında yazılan VHDL kodunu sentezleyerek seçilen FPGA çipini yapılandırmak için gerekli konfigürasyon dosyasını oluşturur. Bu dosya bir bord üzerinde yerleştirilmiş FPGA çipine yüklenerek tarasarım ve gerçekleme işlemi tamamlanmış olur.



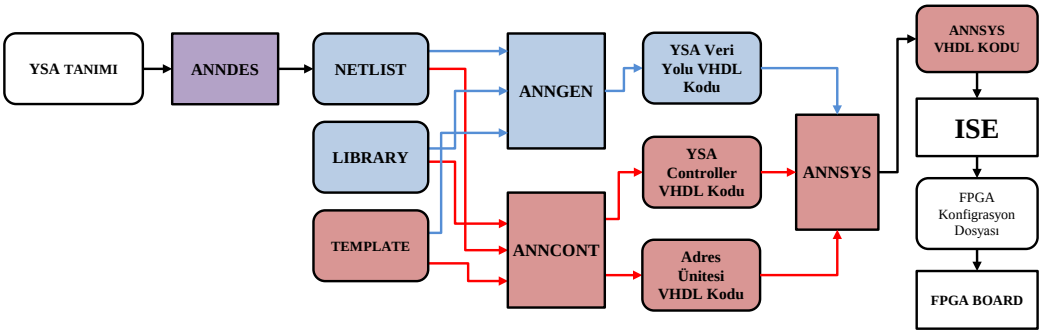
Şekil 1.2 Klasik Tasarım ve Gerçekleme Akış Diyagramı

Reis ve diğerleri yaptıkları bir çalışmada uzman personel tarafından yapılan VHDL kod yazım sürecini otomatikleştirmek için bir yazılım geliştirmişlerdir. Geliştirdikleri yazılım sayesinde istenen bit genişliğinde veri işleyebilen bir YSA yapısı otomatik olarak tasarlanabilmekte ve FPGA çiplerine uygulanabilmektedir. Yine çalışmada gerçeklenen YSA yapıları sadece sabit noktalı yada tamsayı türünde veriler ile çalışmakta, standart veri türleri olan IEEE 754 floating-point (kayan noktalı) verileri işleyememektedir. Bu durum oluşturulan YSA'ların işleyebileceği veri aralığını kısıtlamaktadır. Ayrıca sadece YSA yapısı oluştrulmakta ama YSA'yı kontrol edecek bir mekanizma bulunmamaktadır [14].

Bu çalışmada standart IEEE-754 kayan noktalı sayı formatında veri işleyebilen YSA yapılarını denetleyecek ve hafıza ile YSA arasında veri akışını düzenleyecek bir denetleyici tasarım aracı geliştirilmiştir.

2. Geliştirilen Otomatik YSA Denetleyici Tasarım Aracı

FPGA tabanlı YSA'ların elle yapılan tasarımlarında karşılaşılan dezavantajları ortadan kaldırmak amacıyla bu çalışmada yeni bir denetleyici tasarım aracı olan ANNCONT gelişitirilmiştir. ANNCONT'un amacı, verilen bir YSA veri yolunu denetleyerek, onun uygun bir şekilde hafızadan giriş verilerini okumasını, ardından da ürettiği sonuçları yine hafızaya yazmasını sağlayacak bir denetleyiciyi otomatik olarak oluşturmaktır. ANNCONT daha önceki çalışmalar olan ANNDES (YSA grafik tasarım ve test aracı [15]) ve ANNGEN (YSA'lar için otomatik veri yolu tasarımı aracı [9])' in devamı olarak geliştirilmiş olup sistemin son basamağını oluşturmaktadır. Şekil 2.1'de ANNCONT'ın otomatik YSA tasarım akış diyagramı içindeki yeri görülmektedir. ANNCONT denetleyici tasarlamanın yanında tasarladığı denetleyiciyi ANNGEN tarafından oluşturulmuş olan veri yolu ile birleştirerek YSA sistemini de oluşturmaktadır. Akış diyagramında yer alan bileşenlerin detaylı açıklaması 4. bölümde yapılmıştır.

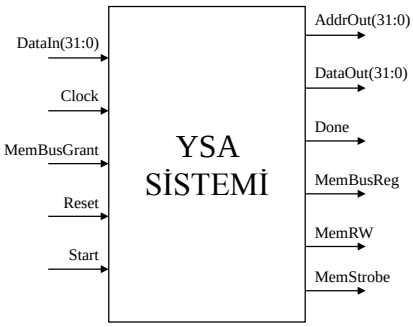


Şekil 2.1 Geliştirilen Tasarım Araçlarının Tasarım Akış Diyagramı [16]

Çalışma kapsamında geliştirilen ANNCONT diğer bileşenler ile birleştirilerek çeşitli test durumları üzerinde denenmiştir. ANNCONT tarafından otomatik olarak tasarlanan YSA denetleyicileri ve sistemleri Xilinx'in ISE tasarım aracı ile sentezlenerek ANNCONT'un doğruluğu test edilmiştir. Deney sonuçları göstermiştir ki ANNCONT sistemin diğer elemanları ile uyumlu bir şekilde çalışmakta ve çok kısa sürede verilen tanımlamaya göre hatasız bir şekilde YSA denetleyici tasarımını yapmaktadır.

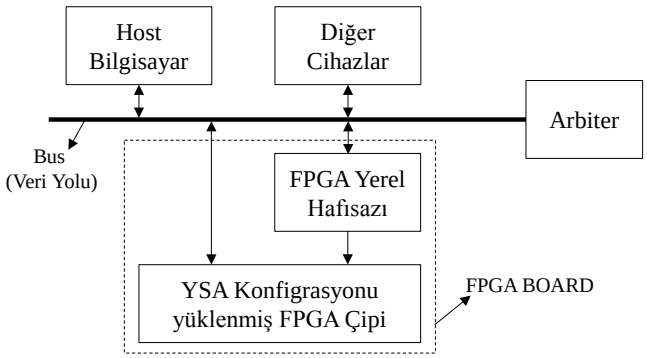
3. Otomatik Olarak Tasarlanan YSA Sisteminin Yapısı

Şekil 3.1'de otomatik olarak tasarlanan bir YSA sisteminin en üst seviye blok diyagramı görülmektedir. Girişteki 32-bitlik DataIn sinyali iki amaç için kullanılır. Birinci olarak YSA çalışmaya başlamadan önce bir birine seri bağlı ağırlık (weight) ve eşik (bias) değerlerinin tutulduğu kayıtçıların ilk değerlerinin sistem içine aktarılmasında kullanılır. İkinci olarak ise, çalışma durumunda YSA girişlerinin seri bir şekilde sisteme girilmesinde kullanılır. 32-bitlik DataOut sinyali ile YSA sisteminin hesapladığı sonuç değeri çıkışa iletilir.



Şekil 3.1 YSA Sistemi En Üst Seviye Blok Diyagramı [16]

MemBusGrant, MemBusReg, MemRW ve MemStrobe sinyalleri, YSA sisteminin hafıza erişimini senkronize etmede kullanılan giriş/çıkış sinyalleridir. AddrOut sinyali YSA sistemi içinde denetleyicinin bir parçası olan adres ünitesi tarafından oluşturulan adres bilgisinin, hafızaya iletilmesinde kullanılır. Sistem, girdileri hafızadan okuyarak sonuçlarını yine hafızaya yazacak şekilde tasarlanır. Clock, Start, Reset ve Done sinyalleri ise YSA sisteminin bir host bilgisayar tarafından kontrol edilmesini sağlayan sinyallerdir. YSA sistemi ile host bilgisayar ilişkisi Şekil 3.2'de gösterildiği gibidir.

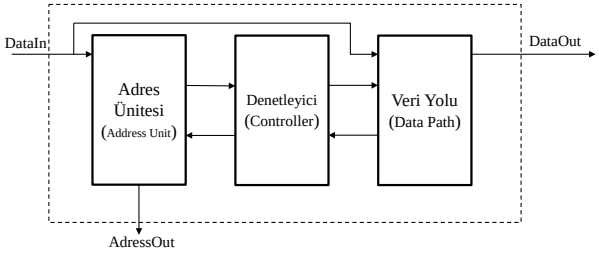


Şekil 3.2 YSA Host Hafıza Erişim İlişkisi

Otomatik olarak tasarlanan YSA'nın çalışması şu şekilde planlanmıştır. Öncelikle host bilgisayar YSA konfigürasyonunu FPGA çipine yükler. Ardından FPGA'ın yerel hafızasına erişerek YSA tarafından işlenecek verileri ve verilerle ilgili gerekli parametreleri aktarır. Bu aşamadan sonra host bilgisayar tarafından FPGA'de yüklü YSA sistemine bir Başla sinyali gönderilerek veri işlemesi sağlanır. YSA Başla sinyalini aldıktan sonra hafızaya erişmek için MemBusRequest ile Bus'ı Arbiter'den ister. Arbiter MemBusGrant ile Bus'ı YSA'ya verir. YSA hafızadan önce ağ parametrelerini (Weight, Bias), giriş veri alanı başlangıç adresini, çıkış veri alanı başlangıç adresini ve giriş veri boyutunu okur. Daha sonra, YSA giriş verilerini sıra ile hafızadan okur, işler ve sonuçlarını hafızaya yazar. Bütün veriler işlenince host bilgisayara Done (Bitti) sinyalini gönderir. Host bilgisayar YSA'yı resetler ve sonuçları hafızadan alır.

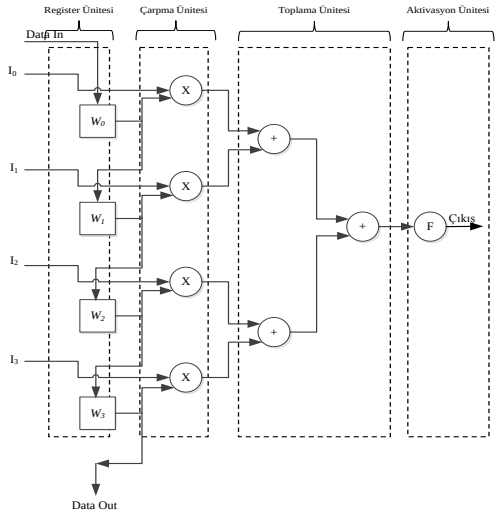
3.1 Otomatik olarak oluşturulan YSA veri yolu

Şekil 3.3'te bu araştırma çalışması kapsamında geliştirilen tasarım araçları ile otomatik olarak tasarlanan YSA sistemlerinin genel ikinci seviye blok diyagramı görülmektedir. Sistemler, AdressUnit, DataPath ve Controller olmak üzere ayrı ayrı tasarlanan üç elemanın birleştirilmesi ile oluşturulur. DataPath bölümü ANNGEN aracılığıyla oluşturulmuş YSA veri yoludur. AdressUnit ve Controller ise bu çalışma kapsamında geliştirilen tasarım aracı (ANNCONT) ile otomatik olarak tasarlanan ünitelerdir.



Şekil 3.3 İkinci Seviye Blok Diyagram [16]

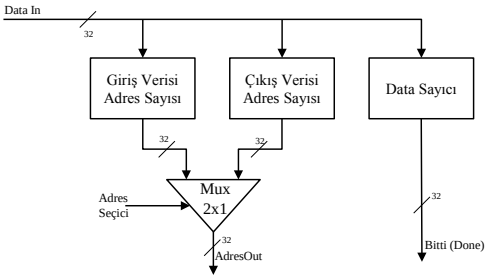
Şekil 3.4'te ANNGEN tarafından yapay sinir hücrelerinin otomatik olarak birbirine bağlanması ile oluşturulmuş YSA veri yolunun genel yapısı görülmektedir. Giriş katmanından gelen verilerin geçici olarak saklanması için girişlere kayıtçılar yerleştirilmiştir. Hücre içindeki weight (ağırlıklar) ve eğer varsa bias (eşik) değerlerini tutan kayıtçılara ilgili değerleri yükleyebilmek için, tek bir kanaldan gelen DataIn sinyali ilk katmanın ilk hücresinin data girişine bağlanmıştır. Bu hücreden çıkan DataOut sinyali bir sonraki hücrenin DataIn girişine bağlanarak veri yolundaki bütün hücrelerin weight ve bias kayıtçıları aynı zincire dahil edilir. Bu zincir sayesinde tek bir veri girişinden bütün kayıtçılara ilk değerlerini seri olarak yüklemek mümkündür [9].



Şekil 3.4 Dört Girişli Yapay Sinir Hücresi Tasarımı [9]

3.2 Adres ünitesi

Adres ünitesi, hafızadan okunacak giriş verilerinin ve yine hafızaya yazılacak sonuç verilerinin adreslerinin tutulduğu adres sayaçlarından ve bir adres seçici multiplexerden oluşur. Bu ünite hafıza erişiminde denetleyici tarafından kullanılır. Şekil 3.5'te görüldüğü gibi Giriş Verisi Adres Sayıcıdan ve Çıkış Verisi Adres Sayıcıdan gelen adres bilgileri, adres seçici tarafından seçilerek AdresOut olarak çıkışa aktarılır. Bu ünite ayrıca bir Data Sayıcı içerir. Data sayıcı giriş verilerinin tamamı işlendiğinde çıkışa bir Bitti(Done) sinyali gönderen özel bir sayaçtır. Bu sinyal ile Denetleyiciye işlemin bittiği bilgisi iletilir.

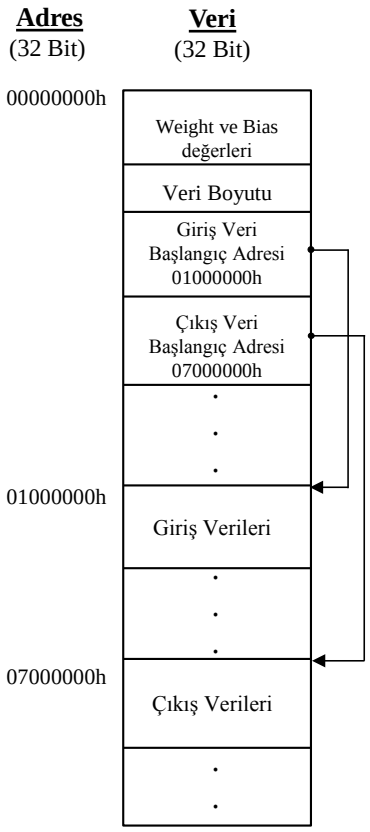


Şekil 3.5 Adres Ünitesi [16]

3.3 Hafıza haritası

Şekil 3.6'da bu çalışma kapsamında otomatik olarak tasarlanan YSA'ların kullandığı hafıza haritası görülmektedir. Hafıza haritası üç bölümden oluşur. Bu bölümler, Parametre alanı, Giriş Verileri alanı ve Çıkış Verileri alanıdır.

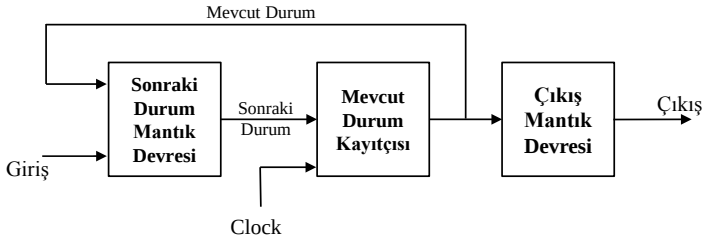
Bir YSA donanım olarak tasarlanmadan önce ANNDES kullanılarak yazılım tabanlı olarak tasarlanır. ANNDES tarafından tasarlanan bu yazılım tabanlı YSA, istenilen veriler ile eğitilerek ağda kullanılması gereken weight ve bias değerleri belirlenir. Parametre alanında, YSA'nın eğitilmesinden elde edilen weight ve bias değerleri tutulmaktadır. Bu alanda ayrıca YSA sisteminin işleyeceği giriş verilerinin ve üreteceği çıkış verilerinin hafızadaki başlangıç adresleri ve giriş verilerinin boyutu tutulur. İkinci ve üçüncü bölümler ise giriş ve sonuç verileri için ayrılmıştır.



Şekil 3.6 Hafıza Haritası [16]

3.4 ANNCONT ile otomatik olarak oluşturulan YSA denetleyicisi

Denetleyici (Controller) ünitesi; YSA sisteminin tüm işleyişini denetleyen ve hafıza erişimini sağlayan ünitedir. Şekil 3.7'de blok diyagramı görülen ve otomatik olarak oluşturulan denetleyici bir Sonlu Durum Makinesi (Finite State Machine) modeli olan Moore Makinesi (Moore Machine) modelinde çalışmak üzere ANNCONT tarafından otomatik olarak tasarlanır.



Şekil 3.7 Denetleyici Ünitesi Blok Diyagramı [16]

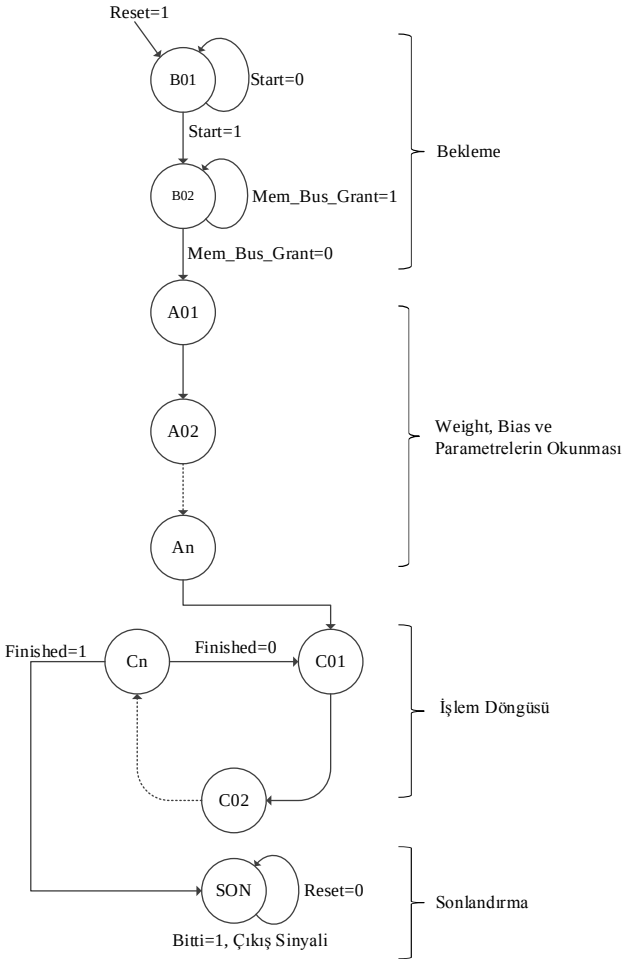
Şekil 3.8'de denetleyicinin genel durum (state) diyagramı görülmektedir. Durum diyagramı toplam dört bölümden oluşur. Birinci bölümün yapısı sabittir ve iki durumdan oluşur. Birinci durum olan B01 durumu, host bilgisayardan gelecek start sinyalini bekleyecek şekilde tasarlanır. Beklenen bu sinyal 1 olduğunda makine B02 duruma geçer. B02 durumu Bus

Arbiter'a MemBusRequest sinyali gönderecek ve Arbiter'den gelecek MemBusGrant sinyalini bekleyecek şekilde oluşturulur. MemBusGrant sinyali geldiğinde ise makine ikinci bölüme geçer.

İkinci bölümün görevi, hafızadan ağın çalışmaya başlamadan önce okunması gereken weight ve bias değerlerinin hücre içinde bulunan ilgili yazaçlara okunmasını sağlamak, ayrıca adres ve veri boyutu sayaçlarına ilk değerleri okumaktır. Bu bölüm, esnek bir yapıya sahip olup, buradaki durumların adedi kontrol edilen YSA veri yolunun giriş çıkış sayısına ve ağda kullanılan sinir hücrelerinin yapısına bağlı olarak değişmektedir. Değişikliğin nedeni ağın yapısına bağlı olarak okunacak veri miktarının (weight ve bias adedinin) farklı farklı olmasıdır. Bu bölüme yerleştirilen durumlar şartsız bir şekilde bir durumdan diğerine geçecek şekilde tasarlanırlar.

Üçüncü bölüme gelindiğinde bütün parametreler okunmuş ve YSA sistemi, giriş verilerini işlemek üzere hazır hale gelmiş durumdadır. Bu bölüm, verilen n büyüklüğünde veriyi işlemek üzere durumlardan oluşan bir döngüyü içerecek şekilde tasarlanır. Döngü, her tekrarlanışında öncelikle bir giriş veri grubu hafızadan okunur. Bu okunan veriler YSA veri yoluna işlenmek üzere verilir, ardından eğer YSA çıkışında bir çıkış varsa (sonuç varsa) bu çıkış değeri hafızaya yazılır. YSA veri yolu kanallı (pipelined) olarak tasarlandığından kanal derinliğine bağlı olarak değişmekle beraber giriş verilerinin sonuca ulaşması belli bir süreç almaktadır. Kanal yapısı içinde kayıtçılar kullanıldığı için YSA veri yolundan aynı anda birden fazla giriş verisi değişik aşamalarda işlenerek (tıpkı yürüyen merdivenlerde olduğu gibi) YSA çıkışına doğru hareket eder. Denetleyicinin bu bölümünde oluşturulan döngü, sırasıyla üç değişik modda çalışır. Birinci modda hafızadan sürekli olarak giriş verilerini okur ve YSA veri yoluna girdi olarak iletir. Bu modda veriler YSA veri yolunda işlenerek adım adım çıkışa doğru ilerler. İlk çıkış üretildiği andan itibaren döngü ikinci modda çalışmaya başlar. Bu modda bir yandan giriş verileri hafızadan okunurken diğer yandan da oluşan sonuçlar hafızada uygun adrese yazılır. Giriş verilerinin tamamının okunması bittiğinde döngü üçüncü moda girer. Bu modda artık okuma işlemi yapılmaz. YSA, veri yolundaki bütün veriler işlenip, sonuca ulaşana kadar (kanal boşaltılana kadar) döngü tekrar eder ve oluşan çıkış değerlerini hafızaya yazar. Son çıkış ta hafızaya yazıldıktan sonra döngüden çıkılarak son bölüme geçilir. Döngü çalışırken, adres ünitesinden gelen bilgilere göre modlar arasında geçiş yapılır. Döngüyü oluşturan durumlar her modda uygun kontrol sinyallerini üretecek şekilde tasarlanır.

Döngüden çıkıldıktan sonra, işlemin bittiğini host'a bildirmek ve host bilgisayara bir kesme sinyali gönderebilmek üzere denetleyicinin dördüncü bölümünde bir Son durumu mevcuttur. Denetleyici bu Son durum'unda reset sinyali gelene kadar bekler. Reset sinyali geldiğinde ise bütün işlemi baştan başlatmak üzere birinci bölümün ilk state'i olan B01 durumuna geçilir.



Şekil 3.8 YSA Sistem Kontrolü Durum Diyagramı [16]

3.5 Otomatik oluşturulan denetleyici durum sayısının belirlenmesi

Yukarıda belirtildiği gibi ANNCONT tarafından otomatik oluşturulan denetleyici toplam dört bölümden meydana gelmektedir. Birinci bölüme toplam iki durum yerleştirilmektedir ve bu bölümün yapısı sabittir. İkinci bölüme yerleştirilen durum sayısı denetlenecek YSA'nın yapısına göre değişmektedir. Bu bölümün durum sayısını, YSA yapısında kullanılan toplam ağırlık adedi, her bir YSA için sabit olan giriş verisi başlangıç adresi, çıkış verisi başlangıç adresi ve veri boyutu parametrelerini okumak için gerekli durumlar belirler.

Üçüncü bölümde döngüye yerleştirilen durum sayısı, yine YSA'nın yapısına göre (giriş ve çıkış adedine göre) değişmektedir. j adet girişli, k adet çıkışlı bir veri yolu için döngüye j+k+2 adet durum yerleştirilir. Burada j+2 adet durum, her seferinde bir giriş veri setini hafızadan okumak için, k adet durum ise sonuçta üretilen k adet çıkışı hafızaya yazmak için kullanılır. Hafızadan veri okuma işlemi belli bir gecikme ile yapılabildiğinden okunacak j adet veri için bu bölüme j+2 adet durum yerleştirilir. Döngünün çalışması sırasında birinci modda (sadece veri okumasının yapıldığı mod), k yazma durumlarında sistem herhangi bir işlem yapmadan boş geçer. İkinci modda döngünün bütün durumlarında işlem yapılır. Üçüncü modda ise sadece k durumlarında hafızaya yazma işlemi yapılır. Son bölümde ise kesme sinyalini gönderebilmek üzere tek bir durum yerleştirilir.

j adet girişli, k adet çıkışlı ve içinde wb adet weight ve bias içeren bir YSA veri yolunu kontrol etmek için ANNCONT tarafından oluşturulan denetleyicinin içereceği toplam durum sayısı eşitlik (1)'de verilmiştir. Toplam durum sayısı (DS);

$$DS = \sum_{i=1}^{4} BölDS_i \tag{1}$$

Birinci bölüm ve son bölümün yapısı sabittir ve içerdikleri durum sayıları sırasıyla 2 ve 1'dir. İkinci bölümün durum sayısı;

$$BölDS_2 = wb + parametre\ adedi + 2 \tag{2}$$

Burada sabit olarak 3 parametre okunmaktadır. Dolayısıyla;

$$BölDS_2 = wb + 3 + 2 \tag{3}$$

Üçüncü bölümün durum sayısı j+k+2 dir. Burada j giriş ve k çıkış sayısıdır. Sonuç olarak bölümlerin durum sayılarına göre eşitlik (1)'i yeniden düzenlersek;

$$DS = \sum_{i=1}^{4} BölDS_i = wb + j + k + 10 \tag{4}$$

olarak belirlenir. İkinci ve üçüncü bölümün durum sayılarındaki +2 değerleri hafızadan okuma işlemindeki iki saat darbelik gecikmeye karşılık olarak kullanılmıştır.

4. ANNCONT (Otomatik YSA Denetleyicisi Tasarım Aracı)

ANNCONT, verilen metin tabanlı Neural Network tanımlamasına göre oluşturulmuş bir YSA veri yolunu denetleyecek denetleyici tasarımını otomatik olarak gerçekleyecek VHDL kodunu sunan bir yazılım aracıdır. Şekil 2.1'de otomatik YSA sistemi tasarım akışı diyagramı içinde ANNDES, ANNGEN ve ANNCONT'un yerleri görülmektedir [16]. Otomatik olarak YSA sisteminin oluşturulabilmesi için ANNGEN ve ANNCONT birlikte bazı dosyalara ihtiyaç duymaktadırlar. Bu dosyaların içerikleri ve yapıları bu bölümde sırasıyla verilmiştir.

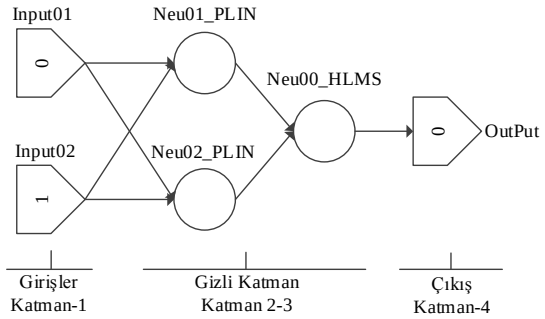
4.1 NETLIST dosyası

NetList'ler genellikle bir elektronik devrenin yapısını tanımlamada kullanılan veri formatlarıdır. Çok çeşitli devre türlerini ifade edebilmek için endüstride kullanılan bazı NetList formatları EDIF (Electronic Data Interchange Format), XNF (Xilinx Netlist Format)'tir. Daha önceden yapılan ANNDES ve ANNGEN çalışmalarında, yapay sinir ağlarını daha kolay bir şekilde tanımlayabilmek için yeni bir NetList formatı geliştirilmiştir. Bu çalışmada da aynı NetList formatı kullanılmıştır. Şekil 4.1'de, NetList dosyasının genel formatı görülmektedir.

```
NETLIST <Katman Sayısı>
[
        LAYER <KatmanNo> <KatmanTürü> <Katmandaki Element Sayısı>
        [
                <Giris0> <Giris1> … <Girisn>   (Giriş Katmanı)
        ]
        LAYER 1 NEURON 4                                (Gizli Katmanı)
        [
                <NeuronİSim> <TransferTürü> <Esik> <EsikDegeri> <GirişSayısı>
                        <GirişAlınan KatmanNo> <Neuron Ismi> <Ağırlık>
                        <GirişAlınan KatmanNo> <Neuron Ismi> <Ağırlık>
                        <GirişAlınan KatmanNo> <Neuron Ismi> <Ağırlık>
                        :
        ]
        :
        PARAMETERS <Parametre Sayisi>
        [
                :
        ]
]
```

Şekil 4.1 NetList Formatı [9]

NetList iç içe bloklardan oluşur. Blok başlangıç ve bitişleri özel karakterler (‘[‘ ve ‘]’) ile sembolize edilirler. Her bir blok içerisinde ya bir katman ya da gerekli parametreler tanımlanır. LAYER ve PAREMETERS kelimeleri blok türünü belirler. LAYER kelimesinden sonra gelen INPUT/NEURON/OUTPUT kelimeleri o katmanın giriş, gizli ya da çıkış katmanı olduğunu belirler. Her bir katmanın yapısı diğerinden farklıdır. Giriş ve çıkış katmanları içinde sadece girişler ve çıkışlar ismen tanımlanır. Nöron katmanında ise o katmanda bulunan nöronların türleri, girişleri, giriş bağlantıları ve ağırlık değerleri tanımlanabilir. Şekil 4.2’de görülen ağı tanımlamak üzere oluşturulmuş NetList dosyası Şekil 4.3’te verilmiştir.



Şekil 4.2 Kontrol Edilmek İstenen Yapay Sinir Ağı

```
NETLIST 4
[
     LAYER 0 INPUT 2
     [
          INP00 INP01
     ]
     LAYER 1 NEURON 2
     [
          NEU00 PLIN 0 0.0 2
               0 INP00 1.1
               0 INP01 1.1
          NEU01 PLIN 0 0.0 2
               0 INP00 1.1
               0 INP01 1.1
     ]
     LAYER 2 NEURON 1
          [
          NEU00 HLIM 0 0.0 2
               1 NEU00 1.1
               1 NEU01 1.1
          ]
     LAYER 3 OUTPUT 1
          [
               OUT00 2 NEU00
          ]
     ]
     PARAMETERS 4
     [
          DataType float
          DataWidth 32
          AddressWidth 32
          VHDLName Deneme
     ]
```

Şekil 4.3 Kontrol Edilmek İstenen Yapay Sinir Ağı Netlist’i

4.2 Kütüphane dosyası

Bu dosyada, tanımlaması yapılmış ANNGEN ve ANNCONT tarafından kullanılabilecek yapay sinir hücrelerinin listesi ve bu hücrelerle ilgili bazı ek bilgiler tutulur. Yapay sinir ağı oluşturulmadan önce ağdaki bütün hücre türlerinin bu dosyada listelenip listelenmediği kontrol edilir. Eğer herhangi bir hücre bu dosyada yer almıyorsa ağ üretilmez. Dosya çok basit bir formata sahiptir. Tablo 4.1'de görüldüğü gibi bu dosyada hücrelerin isimleri, eşik durumları ve giriş adetleri gösterilmiştir.

Tablo 4.1 Kütüphane (Library) Dosyası Formatı [16]

Sinir Hücresinin (Nöron) İsmi	Eşik (Bias) Durumu	Giriş Adedi
LOGS	0	4
LOGS	1	4
TANS	0	2
TANS	1	2

4.3 Şablon dosyası

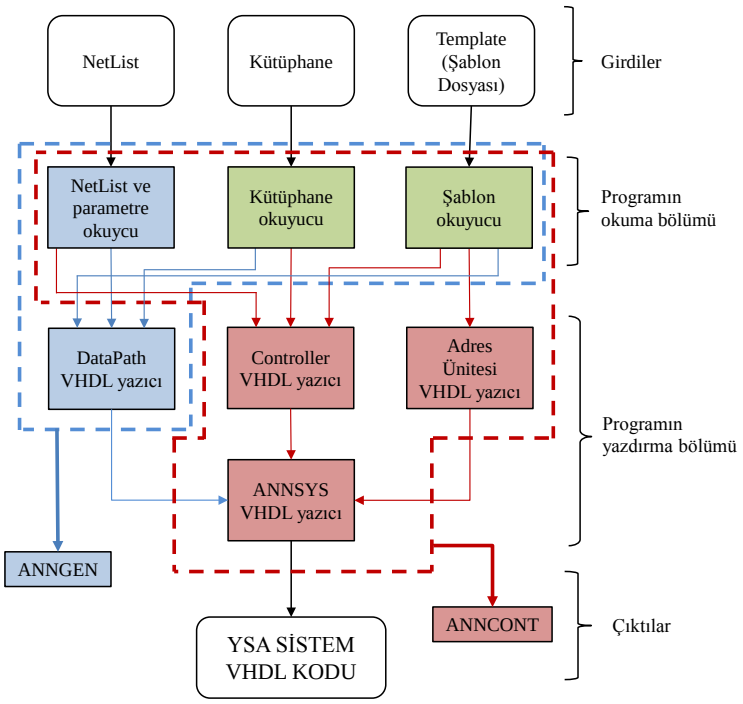
Bu dosyada, ANNGEN ve ANNCONT'un VHDL kodu yazımında kullandığı çeşitli şablonlar (VHDL kod parçaları) ve ayrıca kütüphanede tanımlanmış hücrelerin VHDL kodları yer almaktadır. Bu bilgiler şablon dosyasına sistematik bir şekilde yerleştirilmiştir. Dosyadaki ilk bilgi, kaç tane şablonun olduğudur. Ardından sırası ile şablonlar yerleştirilmiştir. Şablon okuyucu bu bilgiye göre şablonları okur. Her bir şablonun formatı ise şablonun ismi, satır sayısı ve şablonun içeriği şeklindedir. Şekil 4.4'te dosya formatı görülmektedir.

```
<Şablon Adedi>
<Şablon İsmi> <Satır Sayısı>
library IEEE;
use IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;
USE IEEE.std_logic_unsigned.all;
entity COUNTER is
:
end entity COUNTER;
architecture RTL
 of COUNTER is
:
end architecture RTL;
<Şablon İsmi> <Satır Sayısı>
:
```

Şekil 4.4 Şablon (Template) Formatı [9]

5. ANNCONT'un Çalışması

ANNCONT ve ANNGEN entegre bir şekilde çalışmaktadır. Bu çalışmada hem ANNCONT geliştirilmiş hem de ANNGEN ve ANNCONT'un çıkışları birleştirilerek YSA sistemini oluşturan ANNSYS geliştirilmiştir. Şekil 5.1'de ANNGEN, ANNCONT ve ANNSYS'in konumları görülmektedir.



Şekil 5.1 Otomatik YSA sistemi Tasarım Aracının Genel Yapısı [16]

Otomatik YSA tasarım aracının temel algoritmasına ait main fonksiyonu Şekil 5.2'de verilmiştir. Öncelikle üç okuyucu modül aracılığıyla verilen NetList, Kütüphane ve Şablon dosyaları okunarak ilgili veri yapısına kaydedilir. Ardından CheckModulResult() fonksiyonu ile NetList'te belirtilen bütün hücre türlerinin Kütüphane' de tanımlamasının olup olmadığı kontrol edilir. Eğer sonuç pozitifse, istenen YSA için önce Sarıtekin tarafından geliştirilen WriteVHDL() (ANNGEN) fonksiyonu ile veri yolu oluşturularak bir VHDL dosyasına yazılır. Ardından bu çalışmada geliştirilen WriteController() (ANNCONT) ve WriteAddressUnit() aracılığıyla Denetleyici ve Adres Üniteleri tasarlanarak aynı VHDL dosyasına eklenir. Son aşamada ise WriteSystem() fonsiyonu ile Veri Yolu, Denetleyici ve Adres üniteleri birleştirilerek, istenen YSA sistemini oluşturmak için gerekli VHDL kodu aynı dosyaya eklenerek otomatik tasarım işlemi sonlandırılır.

```
Void Main()
    Begin
        Net=ReadNetList("NetList.Txt")
        Lib=ReadLibrary("Library.Txt")
        T=ReadTemplate("Template.Txt")
        if (CheckModulResult(Net,Lib,T))
            WriteVHDL(VHDLName,Net,Lib,T)
            WriteController(VHDLName,Net,Lib,T)
            WriteAddressUnit(VHDLName,Net,Lib,T)
            WriteSystem(VHDLName,Net,Lib,T)
        else
            Msg("Nöron Library'de bulunamadı")
    End
```

Şekil 5.2 Main Fonksiyonu [16]

5.1 WriteController(): ANNCONT'un temel fonksiyonu

Fonksiyon öncelikle oluşturulacak denetleyici ünitesi ile ilgili açıklama satırlarını çıkış dosyasına yazarak başlar. Ardından WriteControllerENTITY() ve WriteControllerARCHITECTURE() fonksiyonlarını çağırarak hedef denetleyici ünitesi VHDL kodunun entity ve architecture bölümlerini iki ayrı aşamada oluşturur [16].

Denetleyicinin entity bölümü sabit bir yapıya sahiptir. WriteControllerENTITY() denetleyicinin entity kısmını ve entity içindeki gerekli portmap'i uygun bir şekilde tanımlar. Denetleyicinin en karmaşık kısmı architecture kısmıdır. Şekil 5.3'te denetleyicinin architecture kısmını oluşturan WriteControllerARCHITECTURE() fonksiyonunun basitleştirilmiş algoritması görülmektedir. Architecture kısmı yapısal olarak sabit bölümlerden oluşur fakat denetlenecek YSA veri yoluna göre her bölümün içyapısı değişiklik arz edebilir. WriteControllerARCHITECTURE() ilk önce denetlenecek veri yolunu analiz ederek veri yolu içinde kullanılmış weight/bias kayıtçı adedini ve veri yolunun giriş adedini tespit eder. Bu sayıların tespit edilmesi oluşturulacak denetleyici için önemlidir. Çünkü bu sayılara göre denetleyicinin sonlu durum makinesi şekil alacaktır. Bu sayılar belirlendikten sonra sonlu durum makinesinde kullanılacak durum değişkeni ve bu durum değişkeninin alacağı değerler tanımlanır. Bu tanımlamalarda yukarıda belirlenen kayıtçı adetleri kullanılır. Denetleyicinin sonlu durum makinesi iki aşama halinde oluşturulur. Birinci aşama sonlu durum makinesinin kendisi, ikinci aşama ise hem sonraki durum belirleyen hem de denetleyici çıkışını belirleyen kombinasyonel lojik aşamasıdır. Bu iki aşamanın oluşturulmasında da yine algoritmanın başında belirlenen kayıtçı adetleri kullanılır.
WriteControllerARCHITECTURE() fonksiyonunun tamamlanması ile denetleyicini tasarımı ve yazımı bitmiş olur.

```
Void WriteControllerArchitecture(char *FileName, NetListType *N, LibraryType *L, TempType *T)
    Begin
      YSA sistemindeki toplam weight ve bias kayıtçı sayısının bulunması,
      YSA sistemindeki toplam giriş ve çıkış kayıtçı sayısının bulunması,
      State turunun ve değerlerinin tanımlanması,
      FSM aşamasının oluşturulması,
      CLogic aşamasının oluşturulması,
    End
```

Şekil 5.3 WriteControllerArhitecture() Fonksiyonu Algoritması [16]

5.2 WriteAddressUnit(): Adres ünitesi tasarım fonksiyonu

Bu fonksiyon, sistemin hafızadan veri okuma, yazma ve sonlandırma işlemleri için hafıza adreslerinin tutulduğu adres ünitesini oluşturmak için WriteController() fonksiyonundan sonra çalışmaktadır. İlk olarak üniteye ait açıklama satırları yazılır. Ardında fonksiyon, şablon dosyası içerisinden gerekli parametreleri okuyup adres ünitesini tasarlar. Tasarım, ilgili VHDL dosyasına yazılır ve fonksiyon sonlanır.

5.3 WriteSystem(): ANNSYS fonksiyonu

Son aşamada, WriteSystem() fonksiyonu devreye girer. Şekil 5.4'te de görüldüğü gibi bu fonksiyon bir top-level entity-architecture bloğu içinde Veri yolunu, denetleyiciyi ve adres ünitelerini, tanımladığı dahili sinyalleri kullanarak birleştirir ve böylece otomatik YSA sistemi tasarım işlemi tamamlanmış olur.

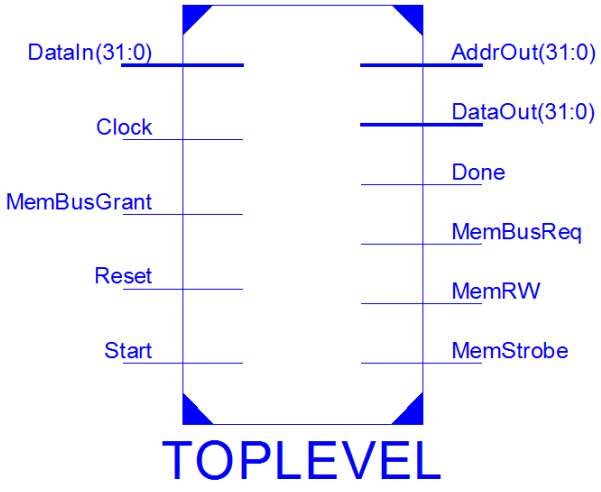
```
Void WriteANNSYS(char *FileName, NetListType *N, LibraryType *L, TempType
*T)
    Begin
      En Üst Seviye ANNSYS Entity oluştur,
      En Üst Seviye ANNSYS Architecture oluştur,
      Bağlantı sinyallerini tanımla,
      YSA veri yolunu instantiate et,
      Adres Ünitesini instantiate et,
      YSA Denetleyicisini instantiate et,
    End
```

Şekil 5.4 ANNSYS Algoritması [16]

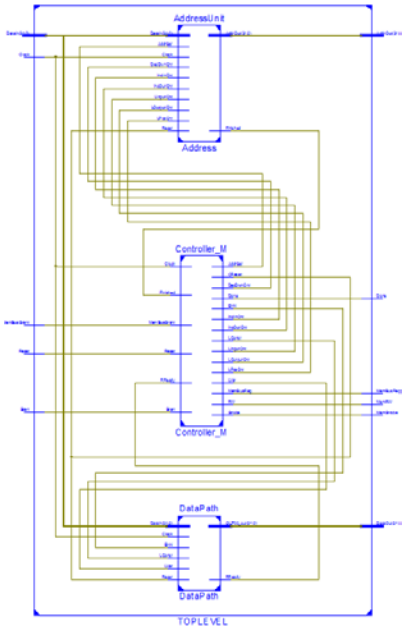
6. Test Sonuçları ve Sonuçların Yorumu

Bu çalışmada, geliştirilen ANNCONT'un test edilmesi amacıyla oluşturulmuş örnek bir YSA yapısı ve bu yapıyı tanımlamada kullanılan NetList Şekil 4.2 ve 4.3'de görülmektedir. Bu YSA yapısında giriş, çıkış ve 2 gizli katman olmak üzere toplam 4 katman mevcuttur. İkinci ve üçüncü gizli katman toplam dört adet iki girişli purelineer aktivasyon fonksiyonlu hücrelerden oluşmaktadır.

ANNCONT, test NetList'i ile birlikte çalıştırılarak, istenen YSA için VHDL kodlarının saniyeler içinde başarılı bir şekilde üretilmesi sağlanmıştır. ANNCONT tarafından otomatik olarak tasarlanan YSA sisteminin işlerliliğini göstermek amacıyla üretilen VHDL kodlarının doğruluğunu tespit etmek için, kodlar Xilinx ISE 14.5 yazılımı kullanılarak Virtex 6 XC6VLX75T FPGA çipinde sentezlenmiştir. ISE başarılı bir şekilde önce yazılım hatası denetimini yapmış, ardında da VHDL kodlarını sentezleyerek Şekil 6.1 ve Şekil 6.2'de görülen RTL diyagramları oluşturulmuştur.



Şekil 6.1 YSA Sistem Üst Seviye RTL Yapısı



Şekil 6.2 YSA Sistem İkinci Seviye RTL Yapısı

Bu şekilde çalışmamızda hedeflenen amaçlara ulaşılmıştır. Sentezleme sonucunda çip üzerinde kullanılan hafıza blokları ve mantıksal blokların kullanım yüzdeleri Tablo 6.1'de verilmiştir. Ayrıca oluşturulan YSA'nın çalışma saat frenkansı 517MHz ve periyodu 1.932 ns olarak ISE tarafından tespit edilmiştir. Böylece YSA sistemi saniyede 517 milyon adet giriş verisini işleyebilmektedir. YSA sistemleri pipeline (kanal) yapısında tasarlandığından ilk sonucun çıkışa ulaşması katman sayısına bağlı olarak değişmektedir. Fakat ilk sonuçtan sonra her saat darbesinde yeni bir sonuç üretilebilmektedir. Bu örnekte verilen YSA yapısında ilk sonucun çıkışa ulaşması için 41 saat darbesine ihtiyaç duyulmaktadır. Bunun anlamı ilk sonucun çıkışa ulaşması 79 ns almaktadır.

Tablo 6.1 Virtex-6 XC6VLX75T FPGA Çip İstatistikleri

Kullanılan Kaynak Türü	Kullanılan	Mevcut	Yüzde
Kayıtcı Adedi (Number of Slice Register)	91	93120	< 1%
LUT adedi (Number of Slice LUTs)	84	46560	< 1%
Giriş/Çikiş Pin Adedi (Number of bonded IOBs)	104	240	43%

7. Sonuç ve Öneriler

Özellikle gerçek zamanlı uygulamalarda yazılım olarak tasarlanan YSA'lar istenen performansı gösterememektedirler. Bu sebepten dolayı alternatif yöntemler geliştirilmeye

çalışılmıştır. Bu aşamada FPGA çiplerinin kullanımı alternatif bir çözüm olmuştur. Ancak YSA'ların FPGA çiplerine uygulanması zaman alan ve uzman gerektiren bir iştir. Aynı zamanda uzman tarafından gerçekleştirilen tüm işlemler ve tasarımı kodlama sırasında oluşabilecek kod hataları, yeniden düzenleme (debug) sorununu da beraberinde getirmektedir.

Bu problemlerden yola çıkarak, bu çalışmada Yapay Sinir Ağlarının FPGA'lara uygulanmasında ANNGEN tarafından otomatik olarak tasarlanan veri yolları için, yine otomatik olarak denetleyici tasarımı yapan ve bu iki üniteyi birleştirerek YSA sistemini oluşturan ANNCONT ve ANNSYS geliştirilmiştir.

ANNCONT'un etkinliğini test edebilmek ve hedeflenen amaçlara ulaşılabildiğini göstermek amacıyla çeşitli test durumları oluşturulmuştur. ANNCONT bu çalışmanın öncesinde geliştirilen ANNGEN ile birlikte test durumları üzerinde denenmiş ve saniyeler içerisinde hatasız bir şekilde istenen YSA için denetleyici tasarımını yaparak VHDL kodunu ürettiği gözlenmiştir.

Bu çalışmada sadece feed-forward (ileri beslemeli) YSA'lar dikkate alınmıştır. Çalışmanın devamında diğer YSA topolojileri içinde sistem geliştirilebilir. Ayrıca burada tasarlanan YSA sistemi sadece hafıza üzerinden veri okuyabilecek şekilde geliştirilmiştir. Çalışmaya yapılacak ufak bir eklenti ile otomatik olarak tasarlanan YSA'ların anlık olarak girişlerden aldığı verileri okuyup çıkış aktaracak şekilde tasarlanmaları da mümkündür.

8. Kaynaklar

[1] J. E. Hall, Guyton Tıbbi Fizyoloji, İstanbul: Nobel Tıp Kitabevi, 2013.

[2] Ç. Elmas, Yapay Zeka Uygulamaları, Ankara: Seçkin Yayıncılık, 2007.

[3] N. Yılmaz, Sahada Programlamalı Kapı Dizileri (Fpga) Üzerinde Bir Ysa'nın Tasarlanması Ve Donanım Olarak Gerçekleştirilmesi, Konya: Selçuk Üniversitesi, 2008.

[4] S. Uzun, XC95XX CPLD' sinin İncelenmesi ve Programlama Kartı Tasarımı, Ankara: Gazi Üniversitesi, 2008, pp. 32-34.

[5] M. A. Çavuşlu, C. Karakuzu, S. Şahin ve F. Karakaya, Yapay Sinir Ağı Eğitiminin IEEE 754 Kayan Noktalı Sayı Formatı İle FGPA Tabanlı Gerçeklenmesi, GOMSİS, İstanbul, 2008.

[6] S. Acar, Eliptik Eğri Kriptografisinde Skaler çarpma bloğunun VHDL ile Tasarımı, İstanbul: İstanbul Teknik Üniversitesi, 2005.

[7] İ. H. Topçu, Sahada Programlanabilir Kapı Dizileri Kullanılarak Sayısal Tasarım Kartı Gerçeklenmesi, İstanbul: İstanbul Teknik Üniversitesi, 2002.

[8] E. Öztürk, Fpga Kullanarak 16 Bitlik Mikroişlemci Tasarımı, İstanbul: İstanbul Teknik Üniversitesi, 2010.

[9] K. N. Sarıtekin, Yapay Sinir Ağlarının Otomatik Olarak Fpga'ya Uygulanması İçin Veri Yolu Tasarım Aracı, Düzce: Düzce Üniversitesi, 2011.

[10] Q. N. Le ve J.-W. Jeon, Neural-Network-Based Low-Speed-Damping Controller for Stepper Motor With an FPGA, IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS,, cilt 9, no. 57, 2010.

[11] A. Gomperts, A. Ukil, S. Member, IEEE ve F. Zurfluh, Development and Implementation of Parameterized FPGA-Based General Purpose Neural Networks for Online Applications, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, cilt 1, no. 7, 2011.

[12] İ. Şahin ve İ. Koyuncu, Design and Implementation of Neural Networks Neurons with RadBas, LogSig and TanSig Function on FPGA, Elektronika Ir Elektrotechnika, cilt 4, pp. 51-54, 2012.

[13] İ. Şahin, A Compilation Tool for Automated Mapping of Algorithms onto FPGA Based Custom Computing Machines, Raleigh-USA: NC State University, 2002.

[14] L. Reis, L. Aguiar, D. Baptista ve F. Morgado-Dias, A Software Tool for Automatic Generation of Neural Hardware, The International Arab Journal of Information Technology, cilt 3, no. 11, 2014.

[15] İ. Şahin ve A. Akkaya, ANNDES : Bir Yapay Sinir Ağı Tasarım ve Eğitim Yazılımı, 5th International Computer & Instructional Technologies Symposium, ELAZIĞ-, 2011.

[16] G. Temür, Yapay Sinir Ağlarının Otomatik Olarak Fpga'ya Uygulanması İçin Denetleyici Tasarım Aracı, Düzce: Düzce Üniversitesi, 2013.