SAÜ Fen Bil Der 20. Cilt, 2. Sayı, s. 141-146, 2016

Yapay sinir ağlarının otomatik olarak donanım ortamında gerçeklenmesi

Namık Kemal Sarıtekin[1*], İbrahim Şahin[2]

21.08.2015 Geliş/Received, 22.12.2015 Kabul/Accepted

ÖZ

Yapay Sinir Ağları (YSA) FPGA tabanlı sistemlerde gerçeklenirken; öncelikle istenen YSA için bir sayısal tasarım yapılır, ardından yapılan tasarım, bir donanım tanımlama dilinde kodlanarak hedef FPGA için sentezlenir. Bu işlemler zaman alan, uzman gerektiren ve hataya açık işlemlerdir. Bu çalışmada, yapay sinir ağlarının FPGA tabanlı sistemler de gerçekleme işlemini otomatikleştirmek, bu işlem için uzman gereksinimini azaltmak ve gerçeklenme sürecini kısaltmak amacıyla, bir otomatik tasarım aracı (Yapay sinir ağı Tasarım Aracı (YTA)) geliştirilmiştir. YTA değişik test durumları ile başarılı bir şekilde test edilmiştir. YTA sayesinde istenen YSA için veri yolu saniyeler içinde otomatik olarak tasarlanmakta ve HDL kodu üretilebilmektedir.

Anahtar Kelimeler: tasarım otomasyonu, FPGA, HDL, yapay sinir ağı

Automatical implementation of artificial neural networks on hardware

ABSTRACT

While Artificial Neural Networks (ANNs) are implemented on FPGAs, first, a logic design is made for the desired ANN. Second, this design is coded in a hardware description language and is synthesized for a target FPGA chip. These procedures are time consuming, error prune processes and requires expert personal. In this study, an ANN data paths design tool (YTA) was developed to help automate the application of ANNs to FPGAs, to reduce the design and implementation time, and to minimize the expert requirements while mapping ANNs to FPGAs. YTA was tested with several test cases successfully. Using YTA, data paths can be designed and HDL codes can be produced automatically for given ANN in seconds.

Keywords: design automation, FPGA, HDL, artificial neural networks

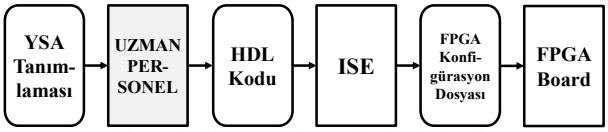
[*] Sorumlu Yazar / Corresponding Author
[1]Düzce Fen Lisesi, Düzce – tekinfizikster@gmail.com
[2]Düzce Üniversitesi, Teknoloji Fakültesi, Bilgisayar Mühendisliği Bölümü, Düzce- ibrahimsahin@duzce.edu.tr

1. GİRİŞ (INTRODUCTION)

İnsanoğlu doğadan ilham alarak değişik model ve modellemeler geliştirmiştir. İnsan beyninde doğal olarak var olan sinir hücrelerinin ve bu hücreler tarafından oluşturulmuş ağların matematiksel olarak taklit edilmesiyle üretilen modele, yapay sinir ağı denir. Yapay zekânın bir uygulaması olan yapay sinir ağları, son yıllarda geleceği öngörmede, örüntü tanımada, verilerin yorumlanmasında, optimizasyon işlemlerinde, doğrusal olmayan fonksiyon yaklaşımlarında ve birçok değişkene bağımlı zaman serilerinin tahmin edilmesinde önemli bir araç olarak kullanılmaktadır [1]. YSA'lar, elektrokimyasal olarak çalışan sinir hücrelerinden çok daha hızlı çalışabilme, yorulmama gibi avantajları nedeniyle odaklanmış olduğu problem için sonuç üretmede çok daha başarılıdırlar [2,3]. YSA'lar da insan beyni gibi hata yaparak öğrenir, hatasını minimum değere indirmeye çalışır ve bir problem için bir kez eğitildikten sonra çok hızlı şekilde sonuca ulaşırlar [4,5].

YSA'lar genellikle yazılım (software) olarak modellenmekte ve kullanılmaktadırlar. Diğer taraftan, anında YSA çıkışının hesaplanması gerektiği gerçek zamanlı uygulamalarda yazılımsal modeller yetersiz kalmaktadır. Bu gibi durumlarda YSA'lar Application Specific Integrated Circuits (ASIC) ya da FPGA teknolojileri [6-9] kullanılarak donanımsal olarak gerçeklenirler. ASIC yerine FPGA'lar paralel işlem yapabilme, kolay ve hızlı bir şekilde yeniden yapılandırılabilme özelliklerinden dolayı birçok YSA uygulamasında, gerçekleme ortamı olarak tercih edilmiştirler [10-13].

Bir YSA donanımsal olarak gerçekleştirilmek istendiğinde, YSA'nın önce sayısal olarak tasarlanması ardından da bir donanım tanımlama dilinde tanımlanması gerekir. Donanım tanımlama işlemi programlama işleminden farklıdır. Burada sanki program yazılıyor gibi tasarlamak istenen sayısal devre tanımlanır. Bu amaçla kullanılan en yaygın iki tanımlama dili Very High Speed İntegrated Circuit Hardware Description Language (VHDL) ve Verilog'dur. Donanım tanımlaması VHDL yada Verilog'da yapıldıktan sonra, bu tanımlama yine otomatik tasarım araçları (yazılımları) tarafından istenen donanım teknolojisine göre (ASIC yada FPGA) sentezlenir ve o teknolojide istenen YSA gerçeklenmiş olur. Aynen programlamada olduğu gibi donanım tanımlamada da tanımlama diline hakim uzman personele ihtiyaç duyulur. Bu işlem hataya açıktır ve uzun zaman alır. Şekil 1'de, bir YSA tanımlamasının FPGA'da donanımsal olarak gerçeklenmesi için izlenecek adımlar görülmektedir. Burada en önemli adım, YSA'nın bir sayısal devre olarak tasarlanması ve uygun bir donanım tanımlama dilinde kodlanmasıdır.



Şekil 1. Klasik tasarım akışı (Classic design flow)

Bu çalışmada, yapay sinir ağlarının FPGA'ya uygulamasında veri yolu tasarımını ve bu tasarımın bir HDL dilinde kodlanması işlemini otomatikleştirmek, böylece uzman gereksinimini azaltmak ve uygulama sürecini kısaltmak amaçlanmıştır. Bu amaçlar doğrultusunda, yapay sinir ağlarının otomatik olarak FPGA'lara uygulanmasına yardımcı olacak bir tasarım aracı (Yapay sinir ağı Tasarım Aracı (YTA)) geliştirilmiştir. YTA uygun formatta verilen bir YSA tanımlaması için otomatik olarak çok kısa bir sürede ve hatasız bir şekilde VHDL dilinde kodlama yapar. Uzman tarafından günler hatta haftalar alan bu işlem basamağı YTA tarafından otomatik olarak saniyeler içinde hatasız bir biçimde gerçekleştirilir.

Literatürde bu konuda yapılmış çok fazla çalışma bulunmamaktadır. Leonardo Reis ve arkadaşları yaptıkları bir çalışmada otomatik olarak yapay sinir hücrelerini değişik bit genişliklerinde oluşturabilecek bir tasarım aracı geliştirmişlerdir [14,15]. Araç tarafından tasarlanan hücrelerin hesaplama doğrulukları seçilen bit genişliğine göre değişmektedir. Hücreler bit genişliğine bağlı olarak belli oranlarda hesaplama hatası yapmaktadırlar. Bu hücreler kullanılarak oluşturulan yapay sinir ağlarında her katmada hata oranı katlanarak artmaktadır. Diğer taraftan bu çalışma kaspamında geliştirilen YTA daha önceden tasarımı yapılmış, kütüphanesinde bulunan hazır yapay sinir hücresi mödüllerini kullanarak yapay sinir ağıları oluşturabilmektedir. Hali hazırda YTA kütühanesinde 6 tane yapay sinir hücresi tasarımı bulunmaktadır. Bu hücrelerin tamamı IEEE-754 floating-point standardında işlem yapacak şekilde tasarlandıklarından, 32-bit yazılım versiyonu ile birebir uyumlu hesaplama yapabilmektedirler. İstenildiği takdirde yeni hücre tasarımları yapılarak YTA kütüphanesine eklenebilmektedir. YTA bu yeni hücreleri otomatik olarak algılamakta ve YSA oluşturmakta kullanabilmektedir.

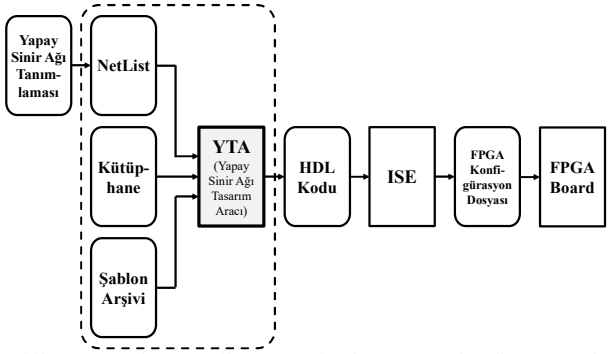
2. MATERYAL VE YÖNTEM (MATERIAL AND METHOD)

Şekil 2'de, geliştirilen tasarım aracının (YTA), FPGA'da YSA tasarımı akış diyagramındaki yeri görülmektedir. YTA girdi olarak üç farklı dosyaya ihtiyaç duyar. Bunlar metin tabanlı YSA tanımlaması (NetList), sinir hücresi kütüphanesi ve şablon arşivi dosyalarıdır.

NetList, oluşturulacak YSA'nın metin tabanlı tanımlamasını içeren bir metin dosyasıdır. Bu dosya içinde hiyerarşik bir yapıda oluşturulmak istenen YSA ile ilgili bilgiler yer alır. Bu yapıda öncelikle YSA'nın kaç katmandan oluştuğu belirtilir. Ardından bloklar halinde katmanlar tanımlanır. Her bir katman tanımlamasında o katmanda yer alan hücrelerin türleri, girişleri, eşik durumu, ağırlık bilgileri ve diğer hücrelerle olan bağlantıları tanımlanır. Giriş, Neuron ve Çıkış olmak üzere üç değişik katman türü tanımlanabilmektedir.

Kütüphane dosyası da yine metin tabanlı hâlihazırda var olan yapay sinir hücrelerinin ve özelliklerinin listelendiği bir dosyadır. Bu dosyada hücrelerin isimleri, eşik durumları ve giriş adetleri listelenmiştir. Eşik durumu bilgisinin 0 olması hücrenin eşik değersiz yani normal olduğunu, 1 olması ise eşik değerli olduğunu belirtmektedir. Otomatik YSA tasarımı sırasında YTA bu dosyadan hâlihazırda tanımlaması yapılmış kullanabileceği hücrelerin bilgilerini alır.

Şablon dosyasında, YTA'nın HDL kodu yazımı sırasında kullandığı çeşitli şablonlar (HDL kod parçaları) ve ayrıca kütüphanede listelenmiş hücrelerin HDL kodları yer almaktadır. Bu bilgiler şablon dosyasına sistematik bir şekilde yerleştirilmiştir.



Şekil 2. FPGA'da otomatik YSA veri yolu tasarımı akış diyagramında YTA'nın konumu (Location of YTA in the design flow of automatic ANN data path design on FPGA)

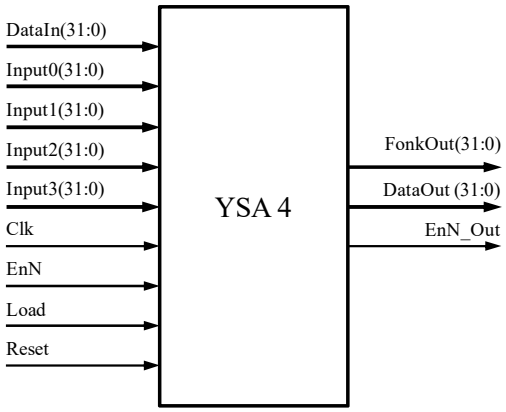
YTA'ya yeni bir hücre tanıtılmak istendiğinde bu hücre ile ilgili tanımlama bilgisinin Kütüphane dosyasına ve hücrenin HDL kodunun Şablon dosyasına eklenmesi yeterlidir. YTA otomatik olarak yeni hücreyi tanır ve gerektiğinde kullanır. YTA'nın hali hazırdaki sürümü HDL dili olarak VHDL kullanmaktadır.

2.1. YTA'nın Kullandığı Yapay Sinir Hücrelerinin Yapısı (Structure of the Artificial Neurons Used by YTA)

Şekil 3'te YTA'nın yapay sinir ağlarını oluşturmada kullandığı 4 girişli örnek bir yapay sinir hücresinin en üst seviye blok diyagramı görülmektedir. Hücre üzerinde 32-bitlik DataIn ve DataOut giriş ve çıkışları, hücrenin diğer hücrelerle zincirleme birbirlerine bağlanması ve hücre içindeki ağırlık (Weight) ve eşik (Bias) değerlerinin tutulduğu registerlerin ilk değerlerinin yüklenmesi için tasarlanmıştır. Input0, Input1, Input2 ve Input3 hücreye giden girdi sinyalleridir. EnN ve EnN_Out sinyalleri ise hücreler arasında veri akışını senkronize etmek için kullanılmıştır. Hücreler EnN sinyali aktif olduğunda girişlerdeki Input sinyallerini almakta ve işlemeye başlamakta, çıkışta sonuç hazır olduğunda ise kendinden sonra gelen hücreyi uyarmak için EnN_Out sinyalini aktif etmektedirler. Hücre hesapladığı en son değeri FonkOut çıkışı ile kendinden sonraki hücreye iletmektedir.

Bu çalışma kapsamında giriş adedine, eşik durumuna ve transfer fonksiyonuna bağlı olarak altı değişik yapay sinir hücresi lego parçaları gibi birbirine bağlanabilecek şekilde tasarlanmıştır. Bu hücreleri kullanarak *i* girişli, *z* çıkışlı ve *m* katmanlı istenilen büyüklükte YSA'lar oluşturulabilmektedir. Yeni hücreler tanımlanarak kolaylıkla sisteme dâhil edilebilmektedir.

Tablo 1'de tasarlanan hücrelerin Xilinx'in Virtex-6 FPGA yongası için sentezlenmesi sonucunda elde edilen çip istatistikleri görülmektedir. Bu istatistiklere göre en yavaş hücre 367 MHz (Megahertz) de çalışabilmektedir. Bunun anlamı bu hücrelerle oluşturulacak herhangi bir YSA saniyede 367 milyon çıktı üretebilmektedir. Hücrelerin yonga içinde kapladıkları alana bakılırsa dört girişli hücreler en fazla %10, iki girişli hücreler ise en fazla %4 yer kaplamaktadır. Bu yüzde değerlerine göre bir çip içerisine en fazla 10 tane 4 girişli hücre veya en fazla 25 tane 2 girişli hücre sığdırılabilmektedir. Bu sayılar Virtex-6 ailesinin en küçük yongası olan XC6VLX75T sürümü için geçerlidir. Bu ailenin en büyük yongası XC6VLX760 10 kat daha fazla donanım içerdiğinden içerisine 100 adet dört girişli hücrelerden veya 250 adet iki girişli hücrelerden sığmaktadır. Bu adet değerleri teorik üst sınırlardır.



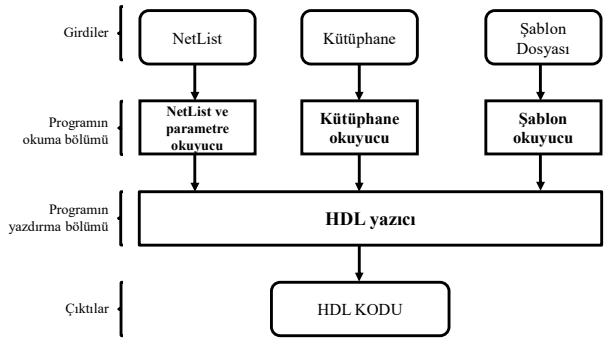
Şekil 3. 4 girişli örnek yapay sinir hücresi en üst seviye blok diyagramı (Top level block diagram of an example 4-input neural cell)

Tablo 1. Kütüphanede bulunan Yapay sinir hücrelerinin Xilinx'in Virtex-6 (XC6VLX75T) FPGA yongası için sentezlenmesi sonucunda oluşan istatistiki bilgiler (Statistical information collected after synthesizing artificial neural cells in the library for Xilinx's Virtex-6 (XC6VLX75T) chip)

Neuron Türü (Neuron Type)	Giriş Sayısı (Input Cnt.)	Slice kayıtcısı (Slice Regs. (%))	LUT (LUTs (%))	Kullanılan Slice (Occup. Slices (%))	Gecikme (Latency (Clk C.))	Neuron Sayısı (Num.of Neurons)	Maks. Frekans (Max Freq.) (MHz)
HLIM	*2*	1	3	4	17	25	367
HLIM	*4*	4	8	10	25	10	441
HLMS	*2*	1	3	4	17	25	388
HLMS	*4*	4	8	10	25	10	402
PLIN	*2*	2	3	4	17	25	446
PLIN	*4*	4	8	10	25	10	457

2.2. Otomatik Yapay Sinir Ağı Tasarım Aracı (YTA) (Artificial Neural Network Design Tool (YTA))

Şekil 4'te görüldüğü gibi YTA dört temel bileşenden oluşmaktadır. Bunlar sırası ile NetList ve Parametre okuyucu, Kütüphane okuyucu, Şablon okuyucu ve HDL yazıcıdır.



Şekil 4. YTA'nın genel yapısı (General Structure of YTA)

Şekil 5'te YTA'nın temel çalışma algoritması görülmektedir. Algoritmaya göre önce isimleri verilen NetList, Kütüphane ve Şablon dosyaları okunur ve okunan bu bilgiler ilgili bölüm için tanımlanmış veri yapısında tutulur. Ardından oluşturulmak istenen YSA veri yolunun tanımlandığı NetList'teki hücrelerinin tamamının kütüphanede var olup olmadığı kontrol edilir. Bu aşamada eğer NetList'teki herhangi bir sinir hücresinin tanımlaması kütüphanede bulunamazsa, YTA bir uyarı mesajı vererek işlemi sonlandırır. Eğer bütün hücrelerin tanımlaması varsa YTA NetList için HDL kodunu üreterek sonlanır.

NetList okuyucunun görevi, verilen NetList dosyasındaki ağ yapısını okumak, parse etmek ve NetList için oluşturulmuş veri yapısında saklamaktır. Kütüphane okuma fonksiyonu verilen kütüphane dosyasında hâlihazırda var olan ve veri yolu oluşturmada kullanılabilecek yapay sinir hücreleri hakkındaki bilgileri okuyarak Kütüphane türünde tanımlanmış veri yapısına aktarır. Şablon okuma fonksiyonu, adı verilen Şablon dosyasını açarak dosya içinde bulunan ve HDL kod yazımında kullanılacak şablon bilgileri ile yapay sinir hücrelerine ait HDL kod tanımlamalarını okur ve Şablon türünde tanımlanmış veri alanına düzenli bir şekilde aktarır. Her üç okuyucuda dinamik veri yapısı kullanmaktadır. Dinamik veri yapısı YTA'ya esneklik kazandırmakta ve girdi dosyalarına sonradan yapılacak eklentilerin YTA tarafından otomatik olarak tanınmasına yardımcı olmaktadır [16].

01 Başla.
02 NetList'i oku.
03 Kütüphaneyi oku.
04 Şablon Dosyasını oku.
05 **Eğer** (NetList'teki bütün hücreler kütüphanede varsa)
  HDL kodunu yaz.
  Değilse
  Uyarı mesajı yaz.
06 Son.

Şekil 5. YTA algoritması (YTA Algorithm)

YTA bütün girdi dosyalarını okuduktan sonra istenen YSA veri yolunu oluşturmadan önce verilen NetList'teki bütün nöronların hâlihazırda kütüphanede olup olmadığını kontrol eder. Bu amaçla bir sinir hücresi kontrol fonksiyonu yazılmıştır. Eğer oluşturulmak istenen YSA'daki bütün yapay sinir hücrelerine ait tanımlama kütüphane ve şablon dosyalarında mevcut ise sinir hücresi kontrol fonksiyonu "Doğru" değeri gönderir.

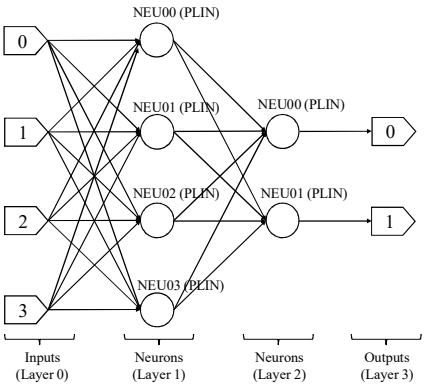
YTA sinir hücresi kontrol fonksiyonundan "Doğru" sonucu gelmiş ise NetList de belirtilen YSA için gerekli HDL kodunu üretmek üzere HDL yazıcı fonksiyonunu (HDLYazıcı) çalıştırır. Eğer "Yanlış" gelmiş ise bir uyarı mesajı vererek sonlanır.

HDLYazıcı fonksiyonu ilk olarak şablonları kullanarak HDL kodu içinde bir açıklama kısmı oluşturur. Bu açıklama kısmında klasik bir HDL kodunda görülebilecek açıklamaların başlıkları yazılır ve geri kalanı kullanıcı tarafından doldurulmak üzere boş bırakılır. Ayrıca tarih bilgisi sistemden alınarak buraya eklenir. HDLyazıcı, okuyucu fonksiyonları tarafından oluşturulmuş dinamik veri yapılarını kullanarak istenen YSA veri yolu için gerekli tasarımı yapar ve HDL kodunu üretir. YTA, Visual Studio.Net 2010 ortamında C++ dili [17] kullanılarak yazılmıştır.

3. YTA'NIN TEST EDİLMESİ (TESTING YTA)

Bu çalışmada geliştirilen YTA'yı test etmek amacıyla birçok test durumu oluşturulmuş ve YTA bu test durumları ile başarılı bir şekilde test edilmiştir. Bu kısımda YTA'yı test etme için kullanılan test

durumlarından bir tanesi örnek olarak verilmiştir. Örnek test durumunda, dört giriş, iki çıkış ve iki gizli katmandan oluşan bir YSA tanımlaması kullanılmıştır. Gizli katmanlarda doğrusal aktivasyon fonksiyonuna sahip yapay sinir hücreleri tercih edilmiştir. Şekil 6 a) ve b) de örnek test durumu için oluşturulan YSA'nın yapısı ve bu yapıyı tanımlamak için oluşturulmuş Netlist görülmektedir.



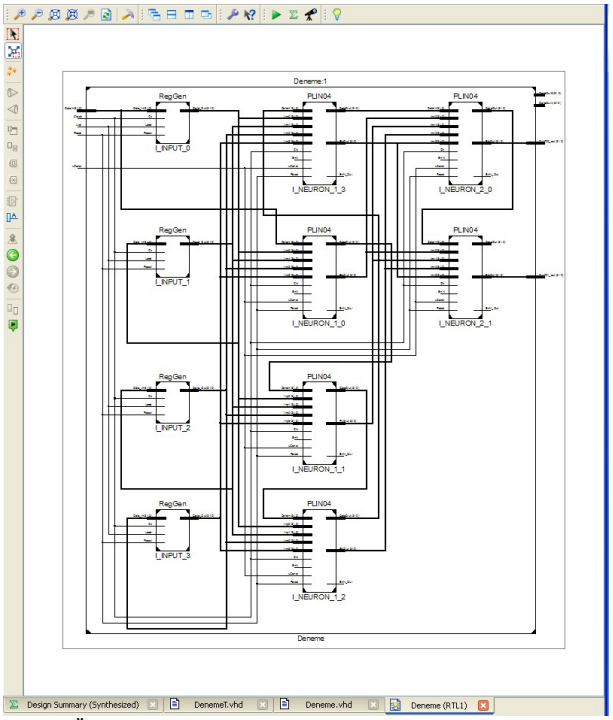
(a)

```
NETLIST 4
[
LAYER 0 INPUT 4
     [
          INP00 INP01 INP02
INP03
     ]
     LAYER 1 NEURON 4
     [
          NEU00 PLIN 0 0.0 4
               0 INP00 1.1
               0 INP01 1.1
               0 INP02 1.1
               0 INP03 1.1
          NEU01 PLIN 0 0.0 4
               0 INP00 1.1
               0 INP01 1.1
               0 INP02 1.1
               0 INP03 1.1
          NEU02 PLIN 0 0.0 4
               0 INP00 1.1
               0 INP01 1.1
               0 INP02 1.1
               0 INP03 1.1
          NEU03 PLIN 0 0.0 4
               0 INP00 1.1
               0 INP01 1.1
               0 INP02 1.1
               0 INP03 1.1
     ]
     LAYER 2 NEURON 2
     [
          :
          :
     ]
     LAYER 3 OUTPUT 2
     [
          OUT00 2 NEU00
          OUT01 2 NEU01
     ]
]
PARAMETERS 4
[
     DataType float
     DataWidth 32
     AddressWidth 32
     VHDLName Deneme
]
```

(b)

Şekil 6. Örnek test durumu için oluşturulan a) YSA yapısı b) Netlist (For the example test case a) ANN structure b) Netlist)

Normal şartlarda yukarıda belirtilen YSA'lar için kodun üretimi zaman alan ve uzman gerektiren bir işlemdir. YTA her örnek test durumu için verilen NetList ile çalıştırılmış ve saniyeler içinde HDL kodunu üretmiştir. Üretilen kodların doğruluğunu test etmek amacıyla Xilinx'in ISE [18] tasarım aracında VHDL projesi tanımlanmıştır. Tanımlanan bu projede YTA tarafından oluşturulan kodlar eklenerek ISE aracılığıyla önce yazım hatası kontrolü yapılmış ardından devre sentezlenerek örnek YSA için RTL (Register Transfer Level) görüntüsü oluşturulmuştur. Oluşturulan RTL, tasarlanmak istenen YSA ile karşılaştırılarak YTA'nın doğruluğu teyit edilmiştir. Bu işlemlerin tamamı dakikalar içinde tamamlanmıştır. Şekil 7 de örnek test durumunda YTA'nın ürettiği HDL kodları ile oluşturulmuş ISE projesi görülmektedir.



Şekil 7. Örnek test durumu için RTL görüntüsü (RTL block diagram for the example test case)

4. SONUÇ VE TARTIŞMA (CONCLUSION AND DISCUSSION)

Bu çalışmada yapay sinir ağlarının otomatik, hızlı ve hatasız bir şekilde FPGA tabanlı sistemlerde gerçeklenebilmesi amacıyla bir otomatik tasarım aracı (YTA) geliştirilmiştir. YTA'yı test etmek amacıyla bir çok test durumları oluşturulmuş ve araç bu test durumları ile test edilmiştir. Test sonucunda YTA'nın saniyeler içinde istenen YSA'lar için gerekli veri yolu tasarımlarını yaptığı ve bu tasarımları gerçeklemek için gereken HDL kodlarını ürettiği görülmüştür. YTA'nın ürettiği HDL kodlarını test etmek için Xilinx'in ISE aracı kullanılmıştır. Bu çalışma kapsamında geliştirilen YTA ile NetList verildiğinde istenen YSA veri yollarının saniyeler içinde otomatik olarak FPGA tabanlı sistemlerde gerçeklenebilecek şekilde tasarlanabildiği, gerçekleme süresinin çok kısalabildiği, yapılan tasarımlarda hata ayıklama aşamasının ortadan kaldırılabildiği ve ayrıca uzman personel ihtiyacının minimum seviyeye indirilebildiği görülmüştür.

KAYNAKÇA (REFERENCES)

[1] Z. Şen, "Yapay sinir ağları ilkeleri", İstanbul, Su Vakfı, 2004.

[2] Ç. Elmas, "Yapay zekâ uygulamaları", Ankara, Seçkin Yayıncılık, 2007.

[3] E. Kandel, "Principles of neural science", U.S.A., Elsevier Science Publishing Co., 1991.

[4] Ö. Yıldız, "Döviz kuru tahmininde yapay sinir ağlarının kullanımı", Eskişehir, Anadolu Üniversitesi (Yüksek Lisans Tezi), 2006.

[5] J. Tebelkıs, "Speech recognition using neural networks", Pennsylvania, School of Computer Science Carnegie Mellon University (Doctor of Philosophy Thesis), 1995.

[6] İ. Şahin ve C. S. Gloster, "FPGA tabanlı CCM'ler için genel amaçlı bir ara yüz", Kocaeli, Bilimde Modern Yöntemler Sempozyumu, 2005.

[7] C. S. Gloster ve İ. Şahin, "Floating-Point modules targeted for use with RC compilation tools", in Earth Science Technology Conference (ESTC), College Park, MD., 2001.

[8] İ. Şahin, "A 32-Bit Floating-Point module design for 3D graphic transformations", Scientific Research and Essays, cilt 5, no. 20, pp. 3070-3081, 2010.

[9] İ. Koyuncu, "A matrix multiplication engine for graphic systems designed to run on FPGA devices", Düzce-Türkiye: Düzce Üniversitesi (Doktora Tezi), 2008.

[10] M. A. Çavuşlu, C. Karakuzu, S. Şahin ve F. Karakaya, "Yapay sinir ağı eğitiminin IEEE 754 kayan noktalı sayı formatı ile FPGA tabanlı gerçeklenmesi", İstanbul: İstanbul Teknik Üniversitesi (GOMSİS), 2008.

[11] M. A. Çavuşlu, H. Altun ve F. Karakaya, "Plaka yeri tespiti için kenar bulma, bit tabanlı öznitelik çıkartma ve YSA sınıflandırıcısının FPGA üzerine uyarlanması", İstanbul, İstanbul Teknik Üniversitesi (GOMSİS), 2008.

[12] F. Benrekia, M. Attari, A. Bermak ve K. Belhout, "FPGA implementation of a neural network classifier for gas sensor array applications," in 6th International Multi-Conference on Systems, Signals and Devices, Djerba,1-6, 2009.

[13] A. Uçar, "Türkçe fonemlerin sınıflandırılmasında kullanılan sinir ağının FPGA uygulaması", Ankara, Hacettepe Üniversitesi (Yüksek Lisans Tezi), 2007.

[14] L. Reis, L. Aguiar, D. Baptista and F. Morgado-Dias, "A software tool for automatic generation of neural hardware", The International Arab Journal of Information Technology, cilt 11, no. 3, pp. 229-235, 2014.

[15] L. Reis, L. Aguiar, D. Baptista and F. Morgado-Dias, "ANGE – Automatic Neural Generator", Artificial Neural Networks and Machine Learning, Espoo, Finland, Springer, 2011, pp. 446-453.

[16] J. Freeman ve D. Skapura, "Neural networks algorithms, applications and programming techniques", Boston, USA, Addison-Wesley Publishing Company, 1991.

[17] Microsoft, "MSD Developer Network", Microsoft, 2010. [Çevrimiçi]. Available: https://msdn.microsoft.com/en-us/library/dd831853(v=vs.100).aspx. [Erişildi: Pazartesi Aralık 2015].

[18] Xilinx, ISE In-Depth Tutorial, Xilinx Corp. (http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/ise_tutorial_ug695.pdf), 2012.

[19] O. İnan, "Veri madenciliği", Konya, Selçuk Üniversitesi, Fen Bilimleri Enstitüsü, 2003.

[20] A. Kalikov, "Veri madenciliği ve bir e-ticaret uygulaması", Yüksek Lisans Tezi, Gazi Üniversitesi, Fen Bilimleri Enstitüsü, 2006.

[21] M. Roberti, "RFID journal home linda", 16 Ekim 2013. [Çevrimiçi]. Available: http://www.rfidjournal.com/articles/view?926#sthash.aNglf57U.dpuf.

[22] Ü. Yarımağan, "Veritabanı sistemleri", Ankara, Akademi Yayınevi, 2000.

[23] P. Adriaans ve D. Zantinge, "Data Mining", Boston, MA, Addison Wesley Longman Publishing, 1997.