İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

BİR HÜCRESEL YAPAY SİNİR AĞININ SABİT NOKTALI SAYI ARİTMETİĞİYLE SAYISAL TASARIMI VE GERÇEKLENMESİ

YÜKSEK LİSANS TEZİ

Barış KARAKAYA

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

ARALIK 2014

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

BİR HÜCRESEL YAPAY SİNİR AĞININ SABİT NOKTALI SAYI ARİTMETİĞİYLE SAYISAL TASARIMI VE GERÇEKLENMESİ

YÜKSEK LİSANS TEZİ

Barış KARAKAYA
(504121380)

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Müştak Erhan YALÇIN

ARALIK 2014

İTÜ, Fen Bilimleri Enstitüsü'nün 504121380 numaralı Yüksek Lisans Öğrencisi **Barış KARAKAYA**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"BİR HÜCRESEL YAPAY SİNİR AĞININ SABİT NOKTALI SAYI ARİTMETİĞİYLE SAYISAL TASARIMI VE GERÇEKLENMESİ"** başlıklı tezini aşağıdaki imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Prof. Dr. Müştak Erhan YALÇIN** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. İsmail Serdar ÖZOĞUZ** İstanbul Teknik Üniversitesi	.....
	Yrd. Doç. Dr. Nerhun YILDIZ Yıldız Teknik Üniversitesi	.....

Teslim Tarihi : **10 Aralık 2014**
Savunma Tarihi : **22 Aralık 2014**

Aileme ve Eşime,

ÖNSÖZ

Bu tez çalışmasında danışmanlığımı yapan değerli hocam Prof. Dr. Müştak Erhan Yalçın'a teşekkür ederim.

Aileme ve eşime her an yanımda olup bana sağlamış oldukları maddi ve manevi tüm desteklerden dolayı teşekkür ederim.

Tez çalışması süresince yardımlarını benden esirgemeyen Yük. Müh. Ramazan Yeniçeri'ye teşekkürü bir borç bilirim.

Aralık 2014

Barış KARAKAYA
Araştırma Görevlisi

İÇİNDEKİLER

Sayfa

KISALTMALAR

FPGA : Field Programmable Gate Array
CNN : Cellular Neural Network
CLB : Configurable Logic Block
CNN-UM : CNN Universal Machine
HYSA : Hücresel Yapay Sinir Ağı
NPE : Neural Processing Element
CNPN : Cellular Neural Processing Network
VGA : Video Graphics Array
RAM : Random Access Memory
VLSI : Very Large Scale Integration
RO-HYSA : Relaksasyon Osilatörü tabanlı Hücresel Yapay Sinir Ağı

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

BİR HÜCRESEL YAPAY SİNİR AĞININ SABİT NOKTALI SAYI ARİTMETİĞİYLE SAYISAL TASARIMI VE GERÇEKLENMESİ

ÖZET

Bu çalışmada, en kısa yolu bulma problemine çözüm üretebilen bir hücresel yapay sinir ağı tasarlanıp gerçeklenmektedir. Tasarlanan devre daha önceden kayan noktalı sayı aritmetiği kullanılarak gerçeklenmiştir. Bu tez çalışmasında ise sabit noktalı sayı aritmetiği kullanılarak gerçekleme yapılmaktadır. Sabit noktalı sayı aritmetiğini kullanmak kaynak kullanımı ve öykünleme hızı anlamında iyileştirmelere imkan sağlamaktadır.

Çalışmanın içeriğinde ilk olarak hücresel yapay sinir ağları incelenmiştir. Hücre modeli olarak Relaksasyon Osilatörü temelli Hücresel Yapay Sinir Ağı (RO-HYSA) modeli kullanılmaktadır. Modele ilişkin benzetim sonuçları sabit noktalı sayı aritmetiği yardımıyla elde edilmiştir. Benzetimi yapılan 4x4 hücreli RO-HYSA, hedeflenen tasarım olan 128x128 hücreli programlanabilen RO-HYSA'nın temelini oluşturmaktadır. 4x4 hücreli ağ çoğaltılıp düzgün biçimde dizilerek 128x128 hücreli ağ oluşturulmaktadır. Ağın FPGA'da gerçeklenebilmesi için gerekli çevre birimler kullanılmakta ve ağ ile haberleştirilmektedir.

Tasarım aşamasında kullanılacak sayı aritmetiği, sabit noktalı sayı aritmetiğidir. Bu aritmetiğin, kayan noktalı sayı aritmetiğine kıyasla FPGA üzerinde daha az yer kaplayacağı ve daha hızlı öykünleme gerçekleştireceği öngörülmüştür. Bu sayı aritmetiği, hücre modeline ilişkin durum değişken değerleri ve parametre değerleri düşünülerek tasarlanmış ve hücrenin matematiksel modelini gerçekleyecek devre içerisine gömülmüştür.

Bu tasarım Xilinx Virtex-II Pro Development System devre kartındaki XC2VP30 FPGA'sında gerçeklenmiştir. Kayan noktalı sayı aritmetiği kullanılarak elde edilen dalga formlarının çoğu bu çalışma sonucunda da gözlenmiştir.

Sonuç olarak, sabit noktalı sayı aritmetiği ile tasarlanan HYSA'nın FPGA üzerinde gerçeklenmesinin daha az kaynak kullanımı ve daha yüksek öykünleme hızı ile kayan noktalı olarak tasarlanan aynı boyuttaki ağ ile benzer uzay-zaman dalgaları ürettiği gözlenmiştir. Tezde önerilen tasarım ve gerçeklemeler en kısa yol bulma gibi uzay-zaman dalgalarını kullanan uygulamalarda uzay-zaman dalgalarının yüksek hızda üretiminde kullanmak için uygundur.

DIGITAL DESIGN OF A CELLULAR NEURAL NETWORK USING FIXED-POINT NUMBER ARITHMETIC

SUMMARY

In this thesis, a cellular neural network that solves the shortest path finding problem is designed and implemented. The designed circuit has been implemented using floating-point number arithmetic previously. In this study, the implementation is performed by using fixed-point number arithmetic. Using fixed-point number arithmetic enables some improvements in the meaning of source utilization and speed of emulation.

In the content of study, first of all cellular neural network has been researched. The architecture of CNN consists of rectangular cell arrays which are in the *mxn* form. The minimal unit of the architecture is called "cell". These cells are placed to two dimensional space in form of cartesian coordinate system where *m* means the number of rows and *n* means the number of column. C(i,j) is called the cell that it is in the i. row and j. column.

Relaxation Oscillator based Cellular Neural Network (RO-CNN) model is used as model of the cell. In literature, active waves are producted by using this model and some important engineering applications such as path planning and robot navigation are implemented by using these waves. Computer simulation results of the chosen model is analyzed by using fixed-point number arithmetic. 4x4 cellular RO-CNN that is simulated network forms a basis of targeted design of 128x128 cellular programmable RO-CNN. 128x128 cellular network is composed by enhancing and collating the 4x4 cellular network in a formal shape. In order to implement this network on an FPGA, environmental units are required and these units are communicated with the network.

In this thesis, the main focus is to design an alternative RO-CNN with fixed-point number arithmetic among to Yeniceri's study and to observe the same waveforms.

Number representation format that is used on the design stage is fixed-point number arithmetic. Fixed-point number arithmetic is symbolized as $Qm.n$. In this format, m means the whole part of the number and n means the fractional part of the number in decimal. State variables, initial condition values and values of parameters used on the model are in 16-bit width. Therefore, number arithmetic will be used is $Q6.9$ where 1 bit is for sign.

Designed arithmetic has been predicted that consumes less sources of FPGA and performs faster emulation in comparison with floating-point number arithmetic since the very beginning of the study. The arithmetic is embedded inside the circuit that realizes the mathematical model of the cell.

In order to compute new values of the state variables, the mathematical model of the network has summation and multiplication circuits. Designed summation circuit

sums two fixed-point numbers in 2 clock cycle. On the other hand, summation circuit that is designed using floating-point arithmetic needs 9 times more cycle. Also summation circuit consumes %90 fewer resource on the implementation stage. Multiplication circuit with fixed-point arithmetic gives the answer in 2 clock cycle as the floating-point multiplication circuit gives. However, multiplication circuit consumes %80 fewer resource on the implementation stage.

Neural Processing Element (NPE) is the most important unit of the CNN circuit that perform the function of the cell. NPE has registers of the requested computation values but does not keep any value of nework constantly. NPEs is used to get a matrix by obtained themselves in order to make network have parallel processing ability. 4x4 parallel processing units are obtained from these NPEs for 128x128 cellular network.

Cellular Neural Processing Network (CNPN) has 16 NPEs that execute an iteration parallelly. One of the most important duty of the CNPN is to make NPEs begin the iteration and make NPEs finish the iteration. Other one is to deliver parameters, state variables and initial conditions to related NPEs.

CNN emulator digital circuit that is called Wave Computer Core insists on Control Circuit, Parameter Register, Communication Circuit, VGA Driver Circuit, CNPN Circuit and RAM network. The most important duties of the Control Circuit is to make CNPN begin and finish the iteration, RAM read the new values and write the computed values of the network, Parameter Register read and write the values of the network, Communication Circuit and VGA Driver Circuit to operate in order to get network image on the on-line monitor VGA.

New core circuit with fixed-point number arithmetic consumes approximately %28 fewer resource on the implementation stage rather than core circuit with floating-point number arithmetic. The number of used BlockRAM is the same in two studies.

All design is called Autowave Generator (AWG) and new AWG design is faster than previous design because of used fixed-point number arithmetic in NPE Circuit. New AWG design consumes approximately %19 fewer slices of logic, %21 FlipFlop, %21 Look-up-Table and %36 MULT18x18 multiplier element on the FPGA.

This design is implemented on XC2VP30 FPGA chip that is inside Xilinx Virtex-II Pro Development System board. In order to transfer parameters, state variables and initial conditions from controller computer to FPGA, RS-232 communication serial port is used and this port is programmed by using MATLAB platform in computer. When the all design is obtained and programmed by using Xilinx ISE platform, file of the implementation code is generated. Programming file is transferred from controller computer to the FPGA by using Xilinx IMPACT programming platform.

After programming the FPGA, parameters, state variables and initial conditions are transferred to the FPGA and designed core begins the iteration with control signals. While network is emulated, obtained network image can be watched on the VGA monitor. Therefore, wave forms that are obtained by using floating-point number arithmetic are also observed in this study.

All implementation results are compared with previous study. New AWG design is faster than the previous. Therefore, bigger size cellular network can be obtained and implement on the same FPGA. Used number of BlockRAM is the same as previous. So, for bigger size cellular network implementation, external memory may be needed.

As a result, minor resource utilization and higher emulation speed of implementation on FPGA of the CNN that is designed with fixed-point number arithmetic and same sized network designed with floating-point arithmetic were observed that similar spatiotemporal waves are generated. Design and implementations proposed in this thesis is suitable for using in finding the shortest path applications where high speed propagation of spatiotemporal waves is needed.

1. GİRİŞ

Chua ve Yang'ın 1988'de ortaya attığı model [1] ile Hücresel Yapay Sinir Ağları (HYSA) tanımlanmış ve model kullanılarak birçok mühendislik problemine çözüm sağlanmıştır. HYSA'lar kısmi diferansiyel denklem çözümünde, yapay görü konusunda ve özellikle görüntü işleme [2] alanında uygulanmış ve gelişen teknoloji ile birlikte biyolojik temel kazanmıştır.

Bu çalışmanın temeli Yalçın'ın ortaya koyduğu Relaksasyon Osilatörü temelli HYSA modeline [3] dayanmaktadır. Literatürde ortaya konulan bu model kullanılarak aktif dalgalar [4, 5] üretilmiştir ve bu dalgalar kullanılarak yol planlama ve robot yönlendirme gibi mühendislik uygulamaları gerçeklenmiştir [6, 7].

Literatürde birçok yayın ile yerini almış aktif dalgalar, özellikle robot uygulamalarında en kısa yolu bulma probleminde [8, 9] kullanılmıştır. Aktif dalgaların gözlendiği kimyasal ortamlar ile kurulan işlemciler, HYSA'nın analojik devre gerçeklemeleri [10] ve yeniden konfigüre edilebilir sayısal cihazlar kullanılmıştır. ACE16K tümdevresi [11] ve sahada programlanabilir kapı dizileri (FPGA) [12, 13] ile yapılan çalışmalarda HYSA gerçeklemeleri farklı mühendislik problemlerine çözüm sunmaktadır.

Uygulamalarda FPGA'nın kullanılması sayısal tasarımın gerçeklenmesini çok hızlandırmaktadır. Tasarımın bu cihazlarla sentezlenmesi, cihaz üzerine serilmesi ve yolların bağlanma işleminin otomatik olarak yapılabilmesi tasarımcıya büyük kolaylıklar sağlamaktadır.

Bu tez çalışmasının odak noktası, Yeniçeri'nin oluşturduğu RO-HYSA tasarımına [14] ait sayı aritmetiğini değiştirip elde edilen dalga formları ve desenlerini gözlemlemektir.

Bu tez çalışmasında öncelikle RO-HYSA için kullanılan hücre modeline ilişkin benzetimler yapılmıştır. Hücre sayısı artırılarak önce 4x4 sonra 128x128 hücreli ağlar oluşturulmuştur. 4x4 hücreli ağa ilişkin benzetimler yapılarak 128x128 hücreli programlanabilen RO-HYSA devresi kurulmuş ve hedeflenen uzay-zaman dalga formları elde edilmiştir.

Yol bulma problemine çözüm üretebilen 128x128 hücreli RO-HYSA tasarımında [14] yer alan aritmetik devrelerin gerçekleştirdiği işlemler için yarı duyarlı kayan noktalı sayı aritmetiği kullanılmıştır. Bu aritmetik yerine sabit noktalı sayı aritmetiğini kullanmanın, tümdevre üzerinde gerçekleme yapılırken kullanılan kaynak sayısı ve ağın öykünlenme hızı bakımından birtakım gelişmelere yol açacağı öngörülmektedir.

Bu yaklaşımla devre çevre ve kontrol birimleri ile yeniden tasarlanmıştır. Sabit noktalı sayı aritmetiği kullanarak işlem yapacak aritmetik devreler oluşturulup 128x128 hücreli programlanabilen RO-HYSA tasarım devresine gömülmüştür.

2. HÜCRESEL YAPAY SİNİR AĞLARI

2.1 Genel Bilgi

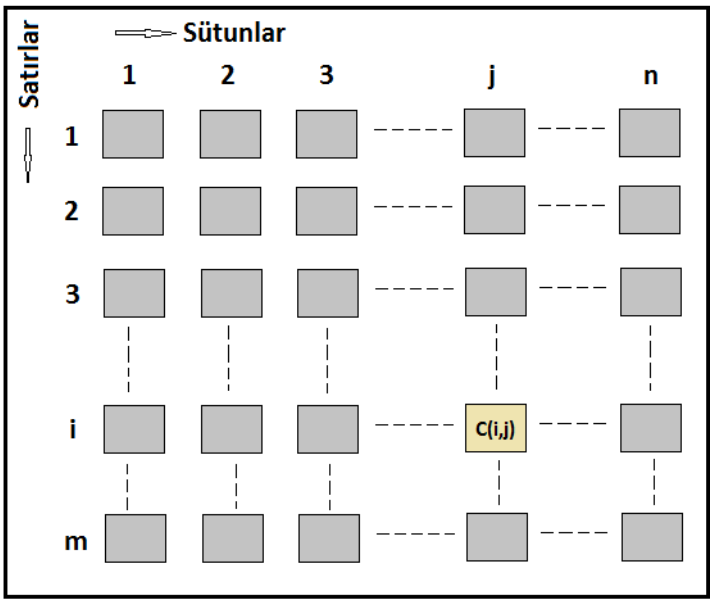
Hücresel yapay sinir ağı, Leon O. Chua ve Lin Yang'ın 1988 yılında yeni bir yapay sinir ağı tasarımı ortaya koydukları çalışmalarında [1,2] ortaya çıkmıştır. Bu çalışmalar ışığında, sinir sistemimizdeki nöronların diğer nöronlarla yerel olarak bağlı olduğu ortaya atılmıştır ve bu HYSA yapısının en önemli özelliğidir. Hopfield sinir ağlarında bütün hücrelerin birbirleriyle doğrudan bağlı olmasına karşın, HYSA yapısında her bir hücre en yakın komşuluğundaki hücrelerle doğrudan, diğer hücrelerle dolaylı olarak bağlıdır. Belirtilen bu yerel bağlantı sonucunda devrenin kapladığı alan azalmakta ve harcanan güç düşük olmaktadır. 1993'de Chua ve Roska tarafından yapılan bir çalışmada, HYSA'nın ortaya atıldığı günden itibaren HYSA ile yapılan çalışmalar özetlenmekte ve HYSA paradigması ortaya koyulmaktadır [15].

Yine aynı yılda, Roska ve Chua tarafından yeni bir bilgisayar mimarisi ortaya atılmıştır. Bu mimari [10], HYSA'yı merkezi işlem birimi olarak kullanmakta olup analog ve sayısal işlem yapma yeteneğine sahiptir. Bu yeteneğinden ötürü mimariye hücresel yapay sinir ağı evrensel makinesi, HYSA-EM (CNN-UM) adı verilmektedir. Bu işlemci, analog ve sayısal olmak üzere yerel ve genel belleğe ve ayrıca HYSA'yı programlamak için gereken şablonların tutulduğu bellek alanlarına sahiptir. HYSA günümüzde retina modellemesi, biyonik göz gibi birçok uygulama alanlarında kullanılmaktadır. Roska ve Rodriguez-Vazguez'in 2002'de sundukları çalışma ile HYSA, görüntü işleme alanında kullanılmış ve bu alandaki becerisinin görsel işlemcilere doğru nasıl yolalınmasına sebep olduğunu göstermektedir [16].

2.2 HYSA Mimarisi

HYSA yapısı *mxn* boyutundaki dikdörtgen hücre dizilerinden oluşur. Yapının en küçük birimine hücre adı verilmektedir. Bu hücreler ağın boyutuna göre iki boyutlu uzayda kartezyen koordinat sistemi düzeninde yerleştirilir. Burada *m*, satır sayısını

n ise sütun sayısını temsil etmektedir. C(i,j), i. satır j. sütundaki hücre olarak ifade edilmektedir (i=1,2,3,...m, j=1,2,3,...n) [17]. Standart HYSA mimarisi Şekil 2.1'de resmedilmektedir.

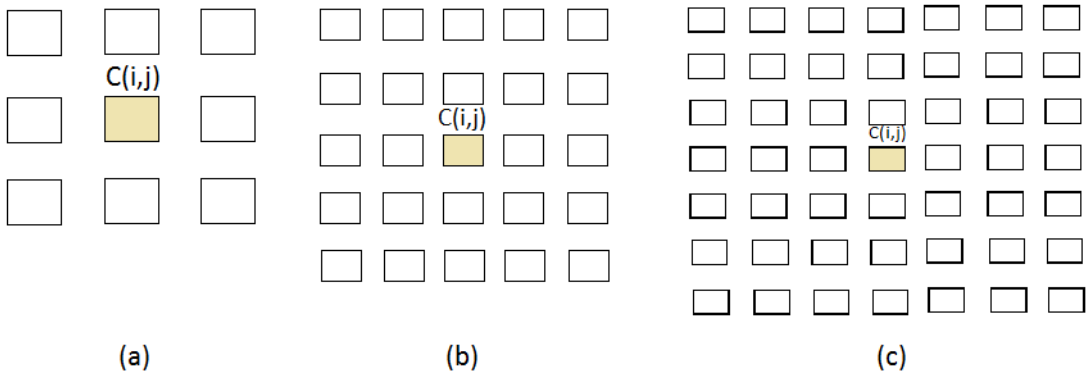


Şekil 2.1: Standart HYSA mimarisi.

C(i,j) hücresinin r yarıçaplı etki küresi $S_r(i,j)$ olarak tanımlanıp;

$$S_r(i,j) = \{C(k,l) | max\{|k-l|, |l-j|\} \leq r\} \quad \textbf{(2.1)}$$

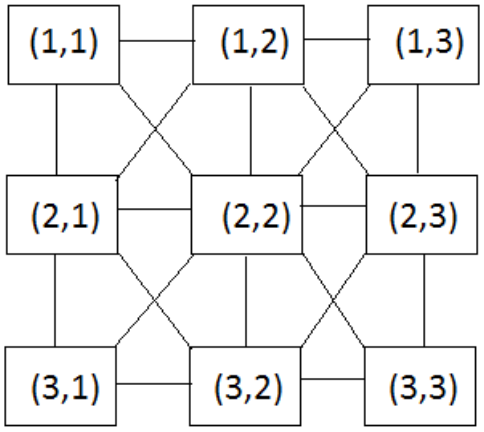
koşulunu sağlayan tüm komşu hücreler topluluğuna denir. Burada r, pozitif bir tam sayı olup Şekil 2.2'de örnek komşuluklar gösterilmektedir.



Şekil 2.2: (a) r=1 (3x3 komşuluk), (b) r=2 (5x5 komşuluk), (c) r=3 (7x7 komşuluk)

Komşuluk sistemi simetri özelliğine sahiptir ve etki küresi içerisinde $(2r+1)^2$ kadar hücre bulunmaktadır. Şekil 2.3'de 3x3 komşuluğuna ait hücreler arası etkileşim gösterilmektedir.

$$C(k,l) \in S_r(i,j) \Leftrightarrow C(i,j) \in S_r(k,l) \tag{2.2}$$



Şekil 2.3: 3x3 komşuluğundaki hücreler arası etkileşim

Chua-Yang modeli olarak adlandırılan bir HYSA'da C(i,j) hücresine ilişkin matematiksel model şu şekildedir;

$$\frac{dx_{ij}}{dt} = -x_{i,j} + \sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)y_{kl} + \sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)u_{kl} + z_{ij} \tag{2.3}$$

Burada;

$x_{ij} \in R$, C(i,j) hücresinin durum değişkeni;

$y_{kl} \in R$, etki küresi içerisindeki hücrelerin çıkışları;

$u_{kl} \in R$, etki küresi içerisindeki hücrelerin girişleri;

$z_{ij} \in R$, eşik değeri;

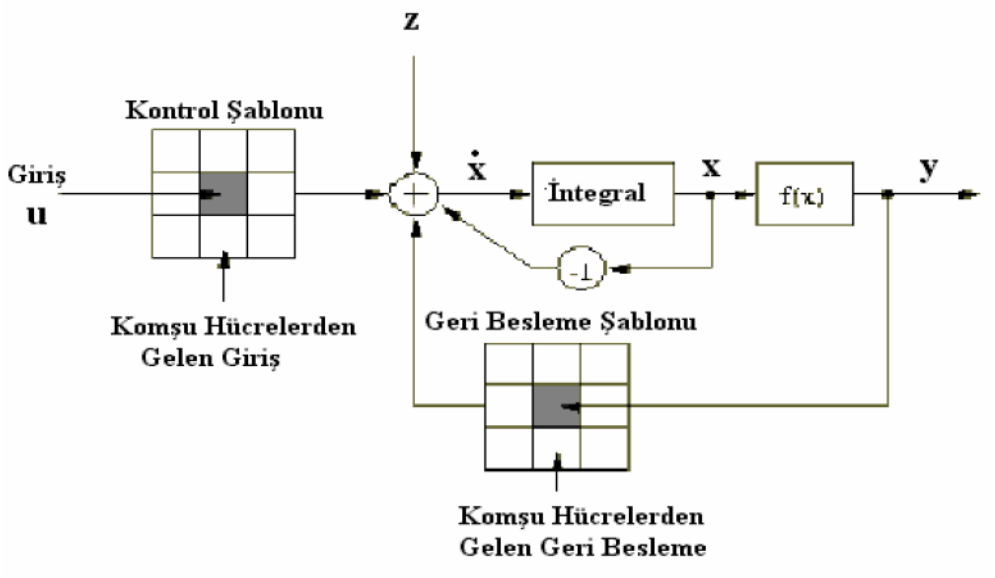
A(i,j;k,l), geri besleme operatörü;

B(i,j;k,l), giriş operatörüdür ve modele ilişkin blok diyagram Şekil 2.4'de verilmiştir.

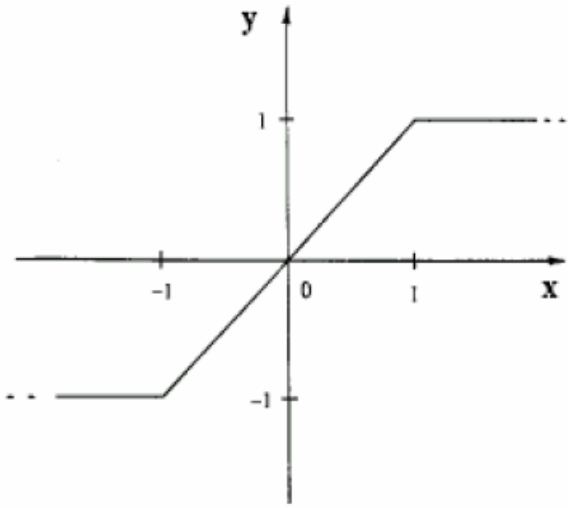
Modele ait çıkış denklemi

$$y_{ij} = f(x_{ij}) = \frac{1}{2}|x_{ij}+1| - \frac{1}{2}|x_{ij}-1| \tag{2.4}$$

olarak tanımlanıp, HYSA'ya ilişkin doğrusal olmayan fonksiyon f(x,y) Şekil 2.5'de gösterilmiştir.



Şekil 2.4: Bir hücrenin blok diyagramı [14].



Şekil 2.5: Eşik Aktivasyon Fonksiyonu [14].

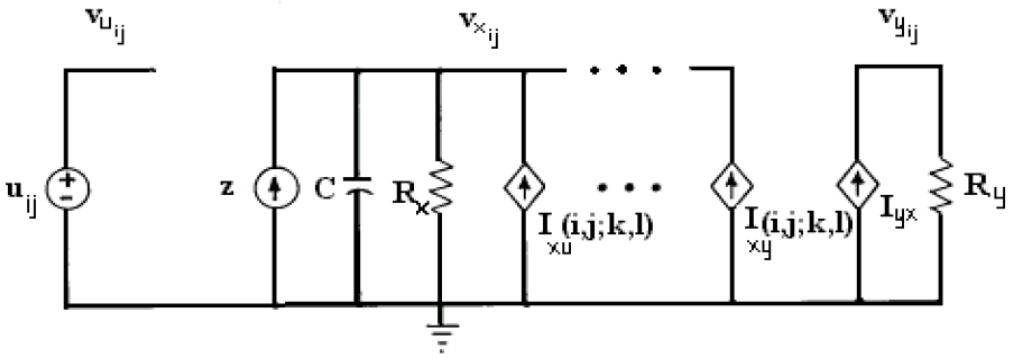
Eşik aktivasyon fonksiyonu, HYSA'ya lineer olmama özelliğini vermektedir. Bu fonksiyon, sınır hücrelerinin etki küresi içinde bulunup da mxn boyutundaki hücre dizisinin dışında kalan hücrelerin durum ve çıkış değerleri koşuludur. Dizideki tüm hücrelerin ilk koşulu, $x_{ij}(0)$ olarak verilir.

HYSA'nın bir hücresinin elektriksel devre modeli ise Şekil 2.6'da gösterilmektedir. Hücrede sırasıyla giriş, durum ve çıkışa denk düşen u, x ve y düğümleri bulunur. Hücrede bulunan elemanlar u_{ij}, sabit gerilim kaynağı; Z, sabit akım kaynağı; C, lineer kapasite; R_x ve R_y, lineer dirençler; m, komşu hücre sayısına eşit olmak üzere en fazla 2m adet $I_{xu}(i,j;k,l)$ ve $I_{xy}(i,j;k,l)$ bağımlı akım kaynağı

$$I_{\mathrm{xy}}(i,j;k,l) = A(i,j;k,l)v_{\mathrm{ykl}} \quad ,\forall C(k,l) \in S_r(i,j) \tag{2.5}$$

$$I_{\mathrm{xu}}(i,j;k,l) = A(i,j;k,l)v_{\mathrm{ukl}} \quad ,\forall C(k,l) \in S_r(i,j) \tag{2.6}$$

ifadeleri ile tanımlanır ve son olarak I_{yx} lineer olmayan gerilim kontrollü akım kaynağıdır. Komşu hücrelerle bağlantı, kontrol girişi v_{ukl} ve geri besleme gerilimi v_{ykl}'nin ağırlıklı toplamları üzerine kurulmuştur.



Şekil 2.6: Bir HYSA hücresinin devre modeli [1].

Hücredeki tek lineer olmayan eleman ise I_{yx} parça-lineer gerilim kontrollü akım kaynağıdır ve

$$I_{\mathrm{yx}} = \frac{1}{R_y} f(v_{\mathrm{xij}}) \tag{2.7}$$

ifadesi ile tanımlanır. Buradaki f(.) fonksiyonu (2.4)'de verilen çıkış fonksiyonudur. HYSA'nın VLSI gerçeklemelerinde f(.) fonksiyonunun keskin karakteristiğini elde etmek mümkün olmadığından daha yumuşak ve sürekli bir fonksiyon olan Sigmoid fonksiyonu kullanılır.

2.3 Dalga Üreten HYSA

Bu bölümde, tez çalışmasının motivasyonu olan hücresel yapay sinir ağları alanında yapılmış çalışmalar özetlenmiştir. Altbölüm 2.3.1 'de, uygulamada kullanılacak hücresel yapay sinir ağının modellenmesi anlatılmaktadır. Altbölüm 2.3.2 'de hücre modelinin, Altbölüm 2.3.3 'de ağa ait bir hücrenin, Altbölüm 2.3.4 'de ise bu hücreler ile kurulan 4x4'lük ağın benzetimi gösterilmektedir. Bu benzetimler ağın sayısal

olarak gerçeklenmesine temel teşkil etmektedir. Elde edilen benzetim sonuçları kullanılarak hedeflenen sayısal gerçekleme platformu Bölüm 3 'de tanıtılmaktadır.

HYSA hücresi modeli olarak Yalçın'ın 2008'de önerdiği kare dalga osilatörü kullanılmıştır [3]. Literatürde relaksasyon osilatörü adı verilen bu hücre modeli kullanılarak hedeflenen sayısal gerçekleme platformu yolunda çalışmaya temel teşkil edecek benzetimler önceki tez çalışması [14] üzerinden anlatılmaktadır.

2.3.1 Hücre modeli

Yalçın, ACE16k tümdevresi ile yaptığı çalışma sonucunda bahsedilen osilasyon davranışını sergileyen hücre modelinin (2.8)'deki gibi olduğunu bulmuştur ve modelde doğrusallığı bozan g(x) fonksiyonunu saptamıştır. Bu fonksiyon (2.9)'da gösterilmektedir.

$$\begin{aligned} \dot{x} &= \alpha x + \beta y - g(x) \\ \dot{y} &= \varepsilon(x - y) \end{aligned} \tag{2.8}$$

$$g(x) = \begin{cases} m(x+1) & ;x < -1 \\ 0 & ;\ |x| \leq 1 \\ m(x-1) & ;x > 1 \end{cases} \tag{2.9}$$

Hedeflenen sayısal devre gerçeklemesi ve bilgisayar benzetimlerinin yapılabilmesi için verilen model denklemlerinde birkaç değişiklik yapılmakta ve denklemler zamanda ayrık hale getirilerek çalışmada kullanılan hücrenin modeli son olarak (2.10)'da verilmektedir. Bu ifade, (2.8)'deki diferansiyel denklemin ileri Euler integrasyonu metodu ile ayrıklaştırılmasından elde edilmiştir. Denklemdeki T, integrasyon adımı olmakla beraber doğrusallığı bozan g(x) fonksiyonu (2.11)'de verilmektedir.

$$\begin{aligned} x_{\mathrm{i,j}}[k+1] &= x_{\mathrm{i,j}}[k] + T[\alpha x_{\mathrm{i,j}}[k] + \beta y_{\mathrm{i,j}}[k] + g(x_{\mathrm{i,j}}[k]) + I_{\mathrm{i,j}}[k] + u_{\mathrm{i,j}}] \\ y_{\mathrm{i,j}}[k+1] &= y_{\mathrm{i,j}}[k] + T[\varepsilon x_{\mathrm{i,j}}[k] + \sigma y_{\mathrm{i,j}}[k]] \end{aligned} \tag{2.10}$$

$$g(x_{\mathrm{i,j}}[k]) = \begin{cases} m.(x_{\mathrm{i,j}}[k] - \lambda) & ;\ x_{\mathrm{i,j}}[k] > \lambda \\ 0 & ;\ |x_{\mathrm{i,j}}[k]| \leq \lambda \\ m.(x_{\mathrm{i,j}}[k] + \lambda) & ;\ x_{\mathrm{i,j}}[k] < -\lambda \end{cases} \tag{2.11}$$

Burada λ, g(x) fonksiyonunun doğrusal olduğu aralıkları belirlemektedir. I(.) komşuluk etki fonksiyonu ise, i. satır j. sütundaki hücrenin komşu hücreleri için kullanılan sinaptik kural ile tanımlanan ve bu hücreye ait komşularının x durum değişkenlerinin ağırlıklandırılmış toplamını ifade etmektedir. Belirtildiği üzere komşuluk etkisi bir hücreye ait sağ(doğu), üst(kuzey), sol(batı) ve alt(güney) komşularının o hücrenin x durum değişkenine etkisi olarak kullanılmakta olup denklemi (2.12)'de verilmektedir.

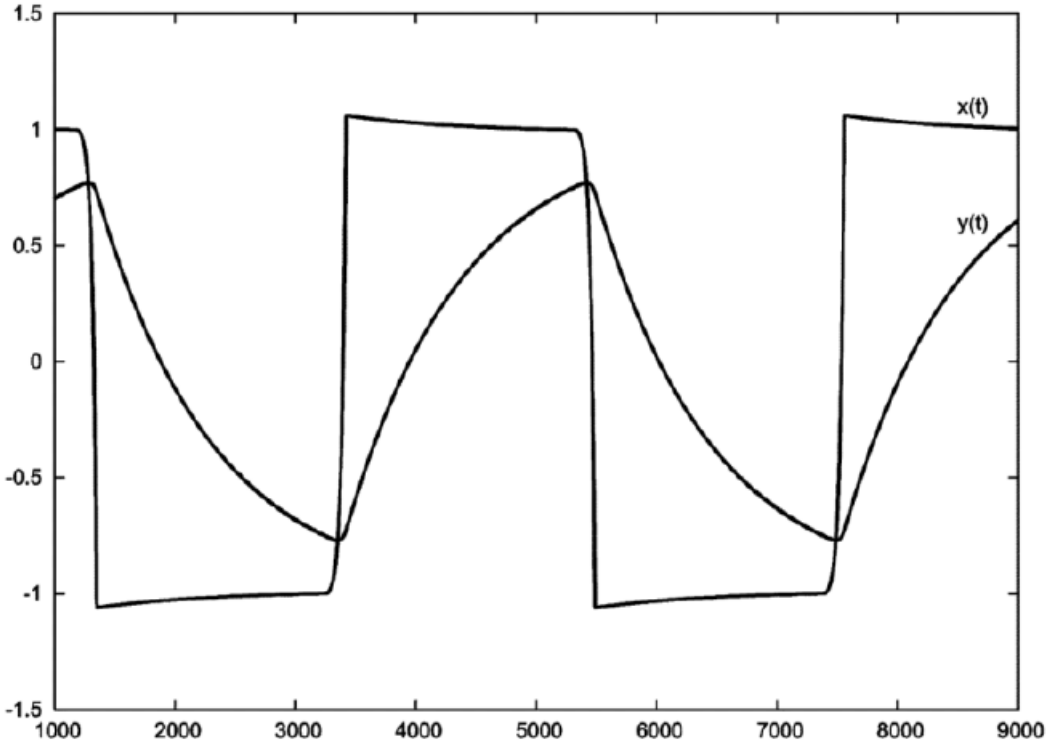
$$I_{i,j}[k] = a_{i,j+1}x_{i,j+1}[k] + a_{i\text{-}1,j}x_{i\text{-}1,j}[k] + a_{i,j\text{-}1}x_{i,j\text{-}1}[k] + a_{i+1,j}x_{i+1,j}[k] \qquad \textbf{(2.12)}$$

Modelde, sürekli giriş u ile ifade edilmektedir. Kurulan HYSA'dan beklenen davranış çeşitli aktif dalgaların üretilmesidir. Yapılan çalışmalarda bu dalgaların kaynağının u giriş işareti ile belirlenebildiği ve sonraki çalışmalarda dalga kaynağı olarak başka koşulların da oluşturulabileceği görülmektedir [14].

Yukarıdaki denklemlerle kurulan model için belirtilmesi gereken bir durum sözkonusudur. Modele göre HYSA ağının tüm hücrelerinin belirtilen dört yöndeki komşuluğunda gerçek hücrelerin olduğu varsayılmaktadır. Oysa ki, HYSA ağının kenar ve köşe hücreleri için sınır koşulları belirlenmeli ve bu hücreler o koşullara göre komşuluktan etkilenmelidir. Bu durum ilerideki benzetim ve gerçeklemelerde dikkate alınmaktadır.

2.3.2 Hücre modelinin benzetimi

Çalışmaya temel teşkil eden modelde Yalçın, tek hücreye ait x ve y durum değişkenlerinin zaman içindeki değişiminin Şekil 2.7'deki gibi olduğunu göstermektedir. HYSA'nın üreteceği dalgaların dinamiği modelde kullanılan parametrelerle alakalıdır. Bu bölümde yapılan benzetimler dalga üretiminde kullanılacak en uygun parametreleri bulmak için yapılmaktadır. Elde edilecek parametre ve sonuçlar ışığında gerçeklemeler daha sağlam bir zemin üzerine kurulacak olup bu benzetim sonuçları gerçeklemenin referans noktası olmaktadır. Hücre modeline ait matematiksel ifadenin benzetimi MATLAB ortamında yapılmaktadır. Benzetim kodları MATLAB R2009a sürümünde yazılıp çalıştırılmaktadır. Benzetim çalışmalarında sabit nokta sayı aritmetiği



Şekil 2.7: Hücrenin x ve y durumlarının zamandaki değişimi [3].

kullanılmaktadır. Altbölüm 2.3.3 'de ve Altbölüm 2.3.4 'de sırasıyla modele sahip tek hücrenin ve 4x4 hücreli ağın benzetimleri gösterilmektedir.

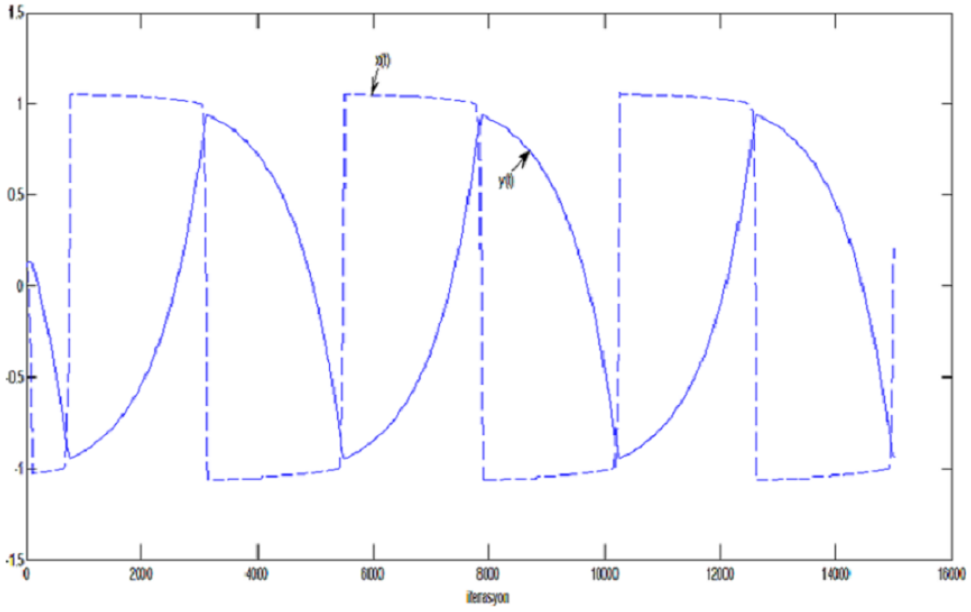
2.3.3 Tek hücrenin benzetimi

Altbölüm 2.3.1 'de tanımlanan hücre modeline sahip tek bir hücrenin benzetimi yapılarak hücreye ait x ve y değişkenlerinin zamanla değişimi gösterilmektedir. Bu benzetim sonucu elde edilecek değişim ile Şekil 2.7'deki değişimin aynı olması hedeflenmektedir. Değişkenlerin zamanla gösterdikleri değişimi gözlemlemek için önceki çalışmaya ait MATLAB kodları sabit noktalı sayı aritmetiği formatında çalıştırılmaktadır.

Model denkleminde ifade edilen α, β, ε ve σ ağırlıkları ile λ, m, T ve *iterasyon* parametreleri sırasıyla 3, -4, 0.125, 0.125, 1, -128, 0.01 ve 15000 olarak tanımlanmaktadır. Hücreye ait durum değişkenleri olan x ve y'nin değeri 0.125 seçilmiştir. Benzetimde hücrenin dört komşusuna ait değişken değerleri 0 olarak verilmiştir. Bu değer ile benzetimi yapılacak ilgili hücreye komşularının etkisi, her ne kadar komşuluk etkisi katsayısı 0.1 verilse de 0 olmaktadır. Yani hücreye dört komşuluğunun herhangi bir etkisi olmamaktadır.

```
x(1)         = 0.125;
y(1)         = 0.125;
alfa         = 3;
beta         = -4;
epsilon      = 0.125;
sigma        = 0.125;
a_d          = 0.1;
a_k          = 0.1;
a_b          = 0.1;
a_g          = 0.1;
x_d          = 0;
x_k          = 0;
x_b          = 0;
x_g          = 0;
lambda       = 1;
m            = -128;
T            = 0.01;
iterasyon    = 15000;
for k = 1:iterasyon
    if( x(k) <= - lambda )
        g = m * ( x(k) + 1 );
    elseif( x(k) >= lambda )
        g = m * ( x(k) - 1 );
    else
        g = 0;
    end
    I = a_d * x_d + a_k * x_k + a_b * x_b + a_g * x_g;
    x(k+1) = x(k) + T * ( alfa*x(k) + beta*y(k) + g + I);
    y(k+1) = y(k) + T * ( epsilon*x(k) + sigma*y(k) );
end
```

Şekil 2.8: Tek hücreye ait benzetim kodu.



Şekil 2.9: Tek hücreye ait x ve y durum değişkenlerinin zamanla değişimi .

Şekil 2.9'da verilen grafik, Şekil 2.8'deki Matlab kodunun çalıştırılması ile ve değişkenlerin aynı eksende çizdirilmesiyle elde edilmektedir. Bu değişimin Şekil 2.7'deki değişim ile örtüştüğü gözlenmektedir.

Üretilen kare dalga işaretinin frekansı model parametreleri ile değişmektedir. x ve y durum değişkenlerinin başlangıç değerleri değiştirildiğinde ise işaretin fazının kaydığı

görülmektedir. Bu sonuçlar ışığında birbirine bağlanmamış komşu hücrelerin farklı başlangıç koşullarında koşturulmasıyla, farklı fazlarda işaretler üretilebilmekte ve bu işaretlerin i, j indisli uzayda da bir uzay dalgasını oluşturacağı gözlenmektedir. Bu hücreler düzgün yerleştirildiğinde ise her bir hücrenin oluşturduğu dalga, uzayda yol alan dalgaları oluşturmaktadır [14].

2.3.4 4x4 hücreli ağın benzetimi

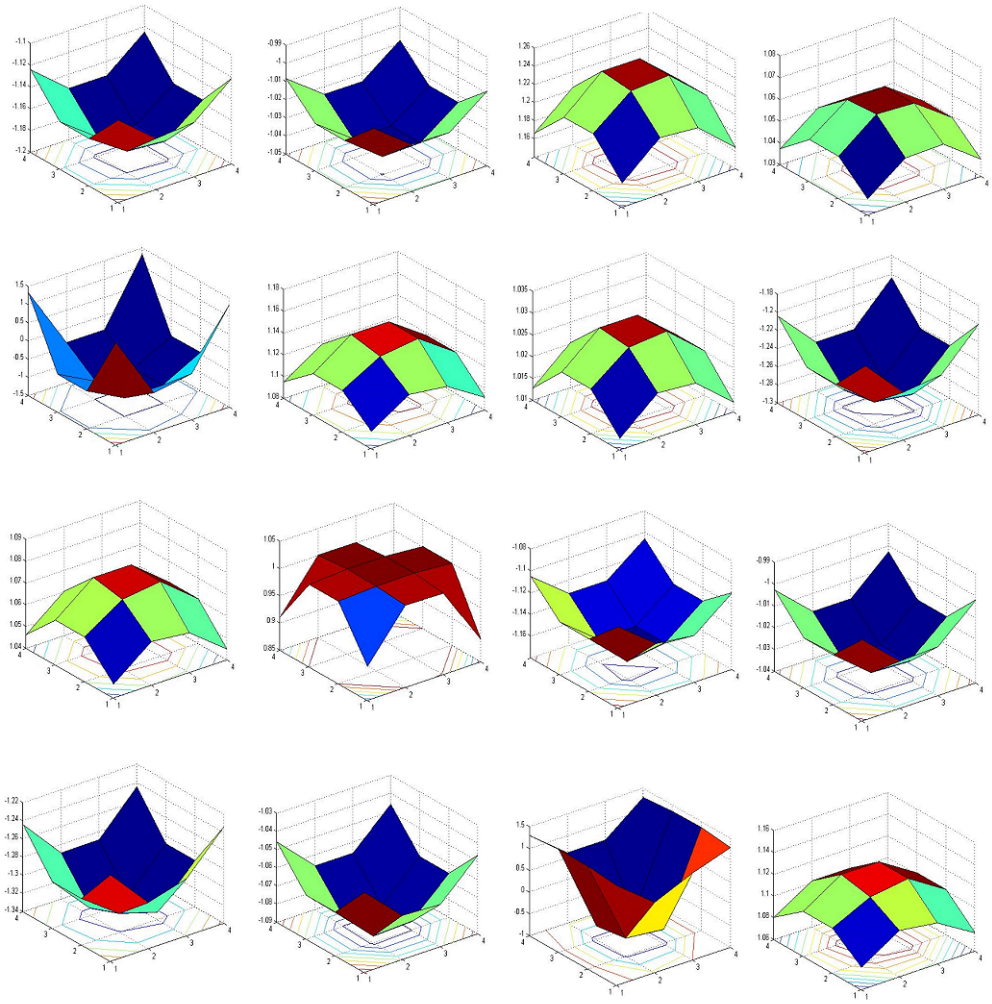
Tek hücrenin benzetiminde çıkarımı yapılan sonucu değerlendirmek için (2.10)'daki hücre modeline sahip 16 hücre düzgün yerleştirilerek oluşturulan uzayda yol alan dalga gözlenmek istenmektedir. Bu hücreler birbirine uygun komşuluklarla bağlanmaktadır. Ağın kenarlarında kalan hücrelerin komşuluk girişlerinden birer tanesi köşelerdeki hücrelerin ise ikişer girişi gerçek komşulara bağlanmamıştır. Bu komşular yerine sınır koşulu 0 olan değişken değeri verilmektedir. Bu düzende oluşturulan ağ ile komşuluk bağlantılarının ağırlıkları olan *a* parametresi kullanılmaktadır. Bu parametre değerine 0 verilmesiyle tüm hücreler birbirleriyle bağımsız hale gelmekte ve her birinin tek hücre benzetiminde olduğu gibi salınım yapması sağlanmaktadır. *a* parametresine 0 değerinden farklı bir değer verilmesiyle hücrelerin osilasyon yapması devam eder fakat aralarında farklı başlangıç koşuluyla başlatılmış gibi faz farkı oluşur. Bu faz farkının oluşması ise ağ üzerinde uzaysal dalgaların oluşmasına sebep olan en temel olaydır [14]. Şekil 2.10'da bu ağın benzetimi için MATLAB ortamında yazılan benzetim kodu gösterilmektedir.

Verilen kodda da görüldüğü üzere tüm hücreler aynı başlangıç değerleriyle iterasyona başlatılmaktadır. Şekil 2.11'de 4x4 hücreli ağdaki x durum değişkenlerinin zamanla değişimi gösterilmektedir. Grafikler, sol üst köşeden sağ alt köşeye doğru zamanda sıralı biçimde 200 iterasyon aralıklarla benzetim sonucu elde edilmektedir. Hücrelerin x durum değişkenleri yaklaşık olarak -1.5 ile +1.5 arasında değerler almakta ve hedeflenen uzay-zaman dalgalarının ağın köşe hücrelerinden yayılmaya başladığı görülmektedir. Belirtildiği üzere iki komşusu sabit bir şekilde sınır koşulu olan 0 değeri ile sürülen köşe hücreleri diğer hücrelere göre daha önce durum değiştirmektedir. Köşe hücrelerinin komşuları ise köşe hücrenin durum değiştirmesiyle eskisinden daha farklı değerler ile beslendikleri için tetiklenmekte ve

```
satir = 4; sutun = 4;
for i = 1:satir
    for j =1:sutun
        x(i,j,1) = 0.125;
        y(i,j,1) = 0.125;
    end
end
alfa        = 3;
beta        = -4;
epsilon     = 0.1;
sigma       = -0.1;
a_d         = 0.2;
a_k         = 0.2;
a_b         = 0.2;
a_g         = 0.2;
lambda      = 1;
m           = -20;
T           = 0.08;
iterasyon   = 10000;
sinir       = 0;
for k = 1:iterasyon
    for i = 1:satir
        for j =1:sutun
            if( x(i,j,k) <= - lambda )
                g = m * ( x(i,j,k) + 1 );
            elseif( x(i,j,k) >= lambda )
                g = m * ( x(i,j,k) - 1 );
            else
                g = 0;
            end
            if (j+1 <= sutun )  x_d = x(i,j+1,k); else x_d = sinir; end
            if (i-1 >= 1 )      x_k = x(i-1,j,k); else x_k = sinir; end
            if (j-1 >= 1 )      x_b = x(i,j-1,k); else x_b = sinir; end
            if (i+1 <= satir )  x_g = x(i+1,j,k); else x_g = sinir; end
            I = a_d * x_d + a_k * x_k + a_b * x_b + a_g * x_g;
            x(i,j,k+1) = x(i,j,k) + T * ( alfa*x(i,j,k) + beta*y(k) + g + I);
            y(i,j,k+1) = y(i,j,k) + T * ( epsilon*x(i,j,k) + sigma*y(i,j,k) );
        end
    end
end
```

Şekil 2.10: 4x4 hücreli ağın benzetim kodu.

durum değiştirmektedir. Burada durum değiştirme ile kastedilen, değişkenin yaklaşık -1.5 olan değerinin yaklaşık olarak +1.5 değerine hızla geçiş yapmasıdır [14].

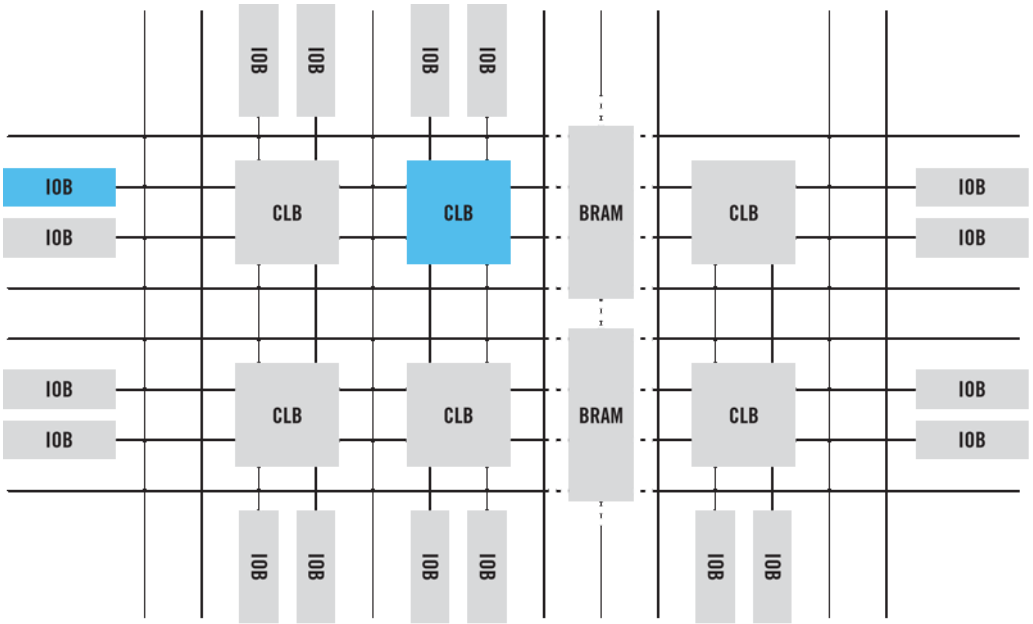


Şekil 2.11: 4x4 hücreli ağın benzetimi sonucunda x durum değişkeninin zamana göre değişimi.

3. RO-HYSA'NIN SAYISAL TASARIMI VE GERÇEKLENMESİ

3.1 Sahada Programlanabilir Kapı Dizileri (FPGA)

FPGA'lar 1980'lerin ortasında Xilinx firması tarafından icat edilmiştir. Bu firma var olan programlanabilen mantık aygıtlarını geliştirerek sahada programlanabilen ilk yongayı yapmıştır. FPGA'lar programlanabilir mantık aygıtları (PLD) ailesinin bir üyesidir. FPGA'lar kullanıcıya yaptıkları tasarım için tasarım aşamaları sırasında hatta tasarımlarını üretip sahada kullanılabilir hale getirdikten sonra dahi değiştirebilme imkanı sağlar. Bu avantajı sayesinde aygıta Sahada Programlanabilir Kapı Dizileri (FPGA) adı verilmiştir. FPGA'lar içerisinde iki boyutlu matris biçiminde dizilmiş konfigüre edilebilir mantık blokları (CLB) içermektedir. CLB'deki mantık fonksiyonları kullanıcı tarafından programlanabilir. Ayrıca FPGA içerisindeki CLB'lerin arasındaki bağlantılar da dışarıdan programlanabilir. Şekil 3.1'de FPGA içinde yer alan yapılar gösterilmektedir.

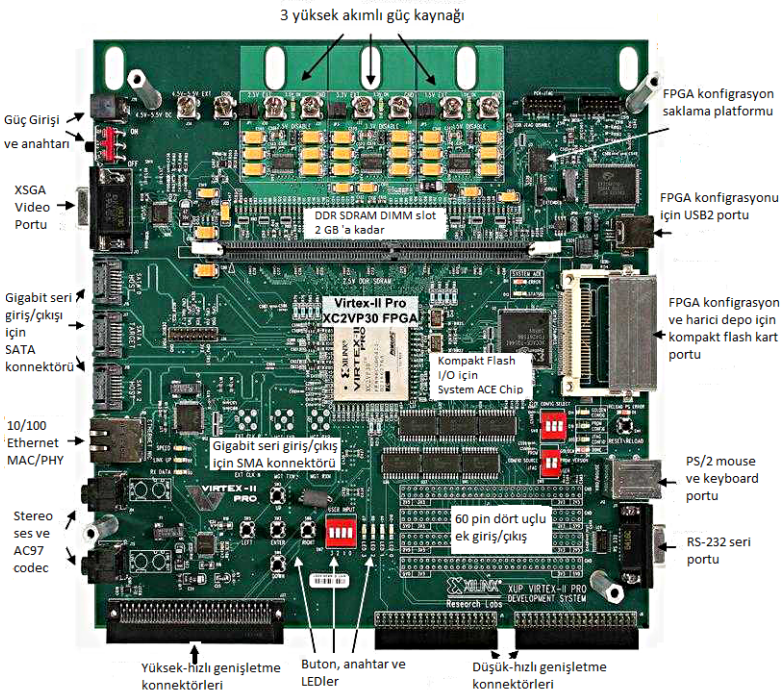


Şekil 3.1: FPGA'nın iç yapısı [18].

FPGA içerisinde CLB'ler dışında BRAM ve giriş çıkış blokları (IOB) da yer almaktadır. BRAM, FPGA üzerindeki tasarımın veri depolama ihtiyacını karşılamak

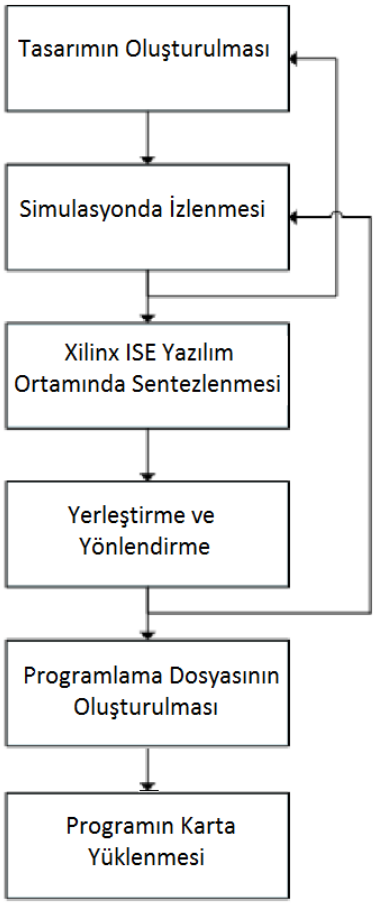
için kullanılır. IOB'ler FPGA üzerindeki tasarımın çevre birimler kullanılarak dış dünyayla olan bağlantısını sağlar. IOB'ler giriş, çıkış ya da çift yönlü olarak programlanabilirler. Bu bloklar dışında tasarımın ihtiyacına göre çarpma bloğu, saat frekans kontrol bloğu gibi farklı bloklar da yer alabilir [18]. Bu tez çalışmasında Xilinx firmasının ürettiği Virtex-II Pro XC2VP30 FF896 FPGA kartı kullanılmaktadır. Karta ait bir görünüm Şekil 3.2'de verilmektedir.

FPGA'da oluşturulan sayısal mantık devreleri tasarımı işlemi gömülü sistem programlama işlemi ile benzerlik göstermektedir. Donanımsal yapılar, davranışsal ya da yapısal olarak yüksek seviyeli donanım tanımlama dilleri ile yazılırlar. Bu diller VHDL ya da Verilog dilidir. Tez çalışması süresince tasarımda Verilog dili kullanılmaktadır. Tasarım oluşturulma sürecinde iken, probleme ilişkin kullanılacak



Şekil 3.2: Xilinx University Program Virtex-II Pro Geliştirme Kartı [19].

bloklar Verilog donanım tanımlama dili ile yazılır ve simulasyonda izlenerek sentezleme işlemine geçilir. Yerleştirme ve yönlendirme işlemlerinin ardından oluşturulan programlama dosyası karta yüklenir. Tasarımın kullanılmasına kadar geçen sürede geride bırakılan süreçler Şekil 3.3'de resmedilmektedir.



Şekil 3.3: Tasarımın akış yolu.

3.2 RO-HYSA'nın Sayısal Tasarımı

Hücresel sinir ağları (CNN), çok boyutlu ortamlar üzerinde işlem yapma yeteneği olan bir yapı olarak ortaya atılmıştır ve iki boyutlu analog ve sayısal gerçeklemeleri görüntü işlemede kullanılabilmektedir. CNN gerçeklemelerinin en büyük avantajı, aynı yapı üzerinde birçok farklı algoritmanın gerçeklenebilmesidir.

Yeniçeri ve Yalçın, 2008'de yaptığı çalışmada [4] yeni basit bir programlanabilen oto-dalga üreteci ağını bir FPGA üzerinde sayısal olarak gerçeklemiştir. Bu ağ iki boyutlu relaksasyon osilatörü tabanlı CNN hücreleri içermektedir. Tasarım, 25,600 nöronun gerçek zamanlı olarak ve Hücresel Nöral İşlem Ağı (CNPN) mimarisini kullanarak benzetimini yapmaktadır. Altbölüm 2.3.1 'de anlatılan ve denklemleri (2.10), (2.11) ve (2.12)'de verilen hücre modeli ile sayısal hesaplamalar için kayan noktalı sayı aritmetiği kullanılarak FPGA'da yapılan benzetimlerin sonucunda uygun parametreler tespit edilip oto-dalga üretimi dinamik olarak VGA monitör aracılığıyla gözlenmiştir.

FPGA üzerinde gerçeklenen CNN uygulamalarından biri de Yıldız'ın 2012'de yaptığı doktora tez çalışmasıdır [20]. Bu çalışmada full HD 1080p@60 video görüntülerini işleyebilen gelişmiş bir gerçek zamanlı sayısal CNN mimarisi önerilmiş, VHDL dilinde kodlanmış ve iki farklı FPGA üzerinde gerçeklenmiştir. Çalışmadaki aritmetik devre işlemleri için kullanılan sabit noktalı sayı aritmetiği sayesinde kaynak kullanımı minimum seviyede tutulmaktadır.

Hücre modeli Altbölüm 2.3.1 'de tanıtılan ve tek hücresinin benzetimi Altbölüm 2.3.3 'de yapılan RO-HYSA, 128x128 hücreli ve programlanabilir olarak bu bölümde tanıtılıp gerçeklenecektir. Ayrıca RO-HYSA'nın 4x4 hücreli olarak benzetimi Altbölüm 2.3.4 'de anlatılmıştır. Bu tasarım 128x128 hücreli programlanabilir RO-HYSA'ya zemin teşkil etmektedir. Gerçeklenecek olan 128x128 hücreli programlanabilir RO-HYSA, 4x4 hücre birimlerinin çoğaltılmasıyla yeni bir tasarıma sahip olacaktır. Bu tasarımda, tüm ağ bu alt dilimlere ayrılmış şekilde sırası ile öykünlenecektir. Hücre sayısının artması öykünleme süresini artıracaktır fakat zamanda sıralı iş yükü ortadan kaldırılacağı için kaynak kullanımı ihtiyacını minimum seviyede tutacaktır. RO-HYSA devresinde kullanılan parametreler ve başlangıç koşulları tasarımın FPGA'ya gönderilmesinden önce ayarlanabilmekte ve çalışması esnasında değiştirilebilmektedir [14].

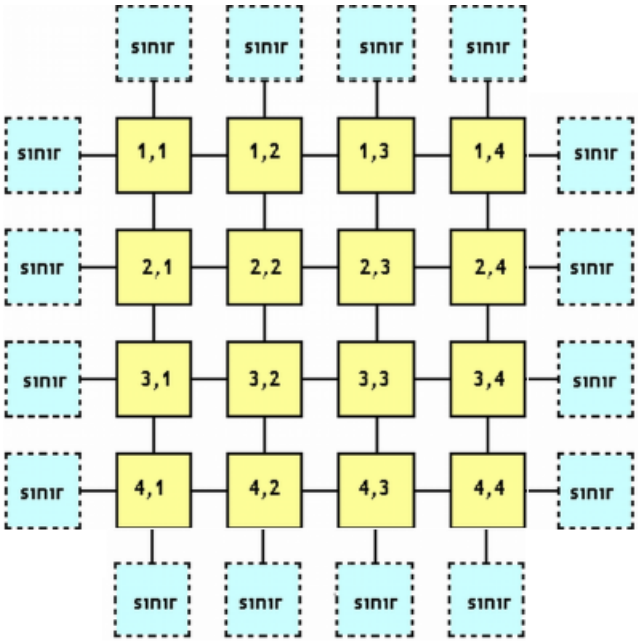
3.3 128x128 Hücreli Programlanabilir RO-HYSA'nın Tasarımı ve Gerçeklenmesi

Bu bölümde, Altbölüm 2.3.4 'de benzetimi yapılan 4x4 hücreli RO-HYSA kullanılarak 128x128 hücreli programlanabilir RO-HYSA gerçeklenmesi anlatılmaktadır. Şekil 3.4'deki 4x4'lük ağdaki her bir hücre dört ana yöndeki (doğu, kuzey, batı, güney) komşusu ile bağlıdır. Ağın sınır kısmındaki hücrelerin ağ boyutunun dışına uzanan bağlantıları 'sınır' olarak adlandırılımış sanal komşulara bağlanmış, bu komşulardan her zaman aynı ilgili değişken değeri ile beslenmiştir. Ağdaki hücrelerden her biri hangi satırda bulunduğunu belirten i ve hangi sütunda bulunduğunu belirten j indisleri ile etiketlenmiştir. Ağdaki komşu hücreler, k anındaki x durum değişken değerlerini birbirlerine aktarmaktadır. Hücrenin y durum değişkeninin ise ne komşu hücrelere ne de ağın dışına aktarılması gerekmektedir. Bu hücre yapısı Şekil 3.5'deki gibi kullanılarak 128x128 hücreli RO-HYSA oluşturulur. Bu tasarımda dalgaya kaynaklık edecek hücreler isteğe göre seçilebilecektir. Bunun yanısıra ağ üzerindeki

aktif hücreler (2.10)'daki hücre modelini gerçekleştirirken işaretlenmiş pasif hücreler ise (3.1)'de verilen pasif modeli gerçekleştirirler. x değişkenine ait bu durum denklemi pasif hücrenin sabit değerde kaldığını ve böylece dalga iletimi yapmadığını göstermektedir. Aktif hücrelerden seçilen birine uygulanan salınım aralığından daha yüksek bir u girişi, hücrenin dalga kaynağı olarak davranmasını sağlamaktadır.

$$x_{\mathrm{i,j}}[k+1] = x_{\mathrm{sabit}}$$

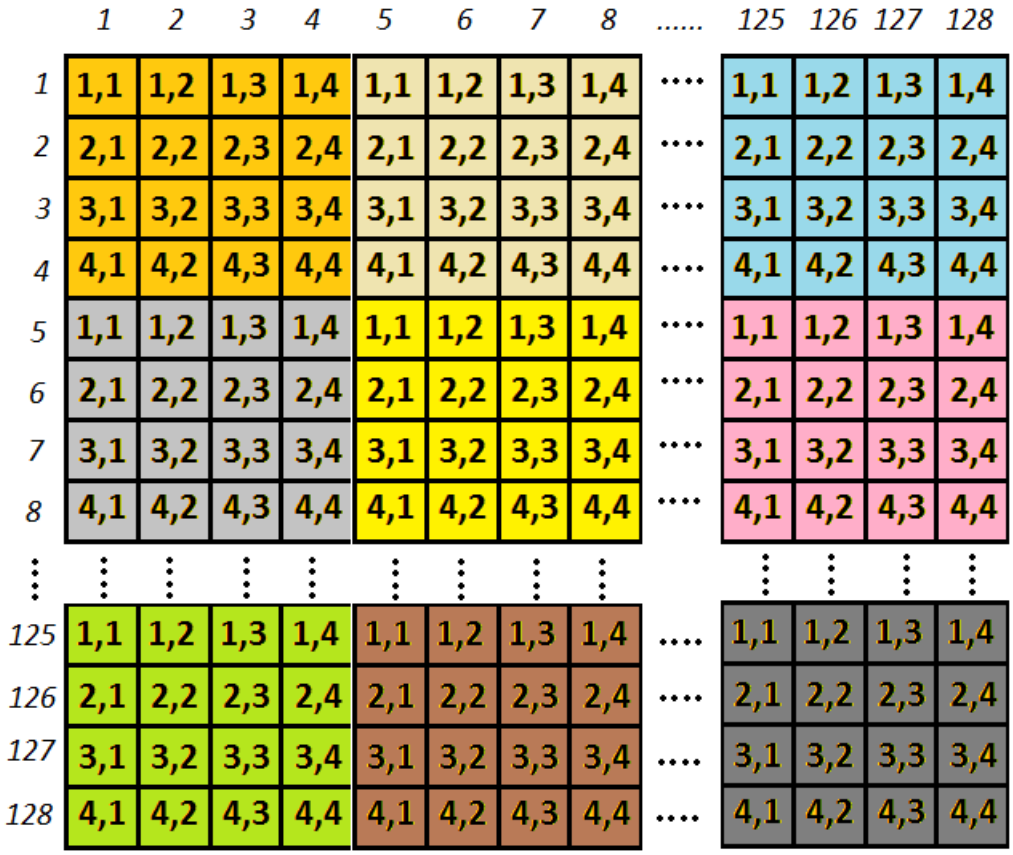
$$y_{\mathrm{i,j}}[k+1] = y[k] \tag{3.1}$$



Şekil 3.4: 4x4 HYSA yapısı.

3.3.1 Hücre çekirdeğinin yapısı

Tasarımda kullanılan hücre çekirdeği $x_{\mathrm{i,j+1}}$, $x_{\mathrm{i-1,j}}$, $x_{\mathrm{i,j-1}}$, $x_{\mathrm{i+1,j}}$ ve $x_{\mathrm{i,j}}$ ile $y_{\mathrm{i,j}}$ değişkenlerinin tutulmasından sorumlu değildir. Çekirdeğin bu girişleri ait olduğu hücredeki tutucular ya da komşuluğu olan hücreler içindeki tutucular tarafından sürülür. Hücre çekirdeği kontrol bloğu, toplayıcı devresi ve çarpıcı devresinden oluşmaktadır. Kontrol bloğu, çekirdeğe gelen başlama işareti ile çalışmaya başlayan bir durum makinası barındırır. Bu durum makinası ile çözülen denklem takımının gerektirdiği sıraya uygun şekilde toplama ve çarpma devreleri ile birlikte seçiciler kontrol edilip ayrık denklem takımındaki aritmetik basamaklar zaman içinde sıralı



Şekil 3.5: 128x128 HYSA yapısı.

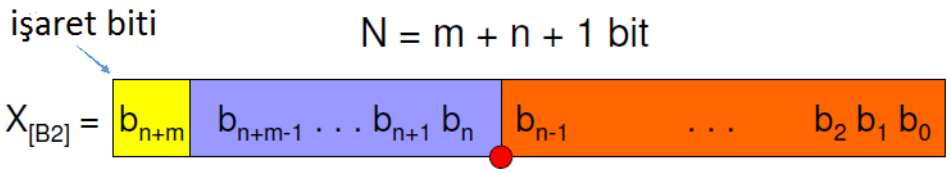
bir şekilde çekirdekte gerçekleştirilir. Denklem takımının sonuçları ise $x_{i,j}[k+1]$ ve $y_{i,j}[k+1]$ tutucularına gönderilir. Aritmetik işlemler sürerken ilk önce x'in k+1 için değeri daha sonra y'nin k+1 için değeri hesaplanır.

Sayısal gerçeklemelerin tamamında toplama ve çarpma işlemlerini gerçekleştiren aritmetik işlemciler tasarlanmış ve kullanılmıştır. Aritmetik işlemciler, önceki çalışmada [14] kullanılan 16 bit genişlikte yarı duyarlı kayan noktalı sayılar yerine 16 bit genişlikte sabit noktalı sayılar üzerinden işlem yapmaktadır. Gerçeklemedeki tüm değişkenler bu sayı formatında saklanır ve işlenir. Ayrıca değişken tutucuları ve veriyolları 16 bit genişliğindedir.

Aritmetik işlemlerde hücrelerin değişken değerleri için tam sayıların yanısıra ondalıklı sayılar da kullanılmaktadır. Tasarımda, ondalıklı sayıları ondalıklı ikilik tabanda sayılar olarak gösterebilmek için sabit noktalı sayı formatı kullanılmaktadır. Hücreye ait x ve y değerlerinin hesaplanması için kullanılan ağın başlangıç koşulları da ondalıklı sayı olabilir. Bu nedenle, ondalıklı sayı olarak gösterilebilecek ve ondalıklı ikilik

tabanda sayılara tekrarlı bir şekilde 0.5 değeri ile çarpılarak dönüştürülebilecek bir ondalıklı sayı formatı tasarlanmaktadır.

Sabit noktalı sayı formatı Şekil 3.6'da görüldüğü gibi *Qm.n* ile sembolize edilir ve bu gösterimde ondalık sayı bitleri ve tam sayı bitleri belirtilmektedir. Sayı formatındaki *m* sayısı tam sayı bit sayısını belirtirken, *n* ondalık kısmına ait bit sayısını temsil eder. Durum değişkenleri, başlangıç koşul değerleri ve sistem parametre değerleri 16 bit genişlikte olup *Q*7.9 sabit noktalı sayı formatına sahiptir. Kullanılacak ve hesaplanacak sayılar işaretli sayılar olacağından, sayının işaretini belirtmek üzere 1 bit tanımlanır. Bu nedenle sayı formatı, *Q*6.9 sabit noktalı sayı formatına dönüşmektedir [21].



Şekil 3.6: Qm.n sabit noktalı sayı formatı gösterimi [21].

Bu sayı formatı kullanılarak gösterimi yapılan sayılar (3.2)'de verilen eşitlik yardımıyla onluk tabana kolaylıkla çevrilebilir. $x_{[\mathrm{B10}]}$, onluk tabanda ondalıklı sayıyı temsil etmektedir.

$$x_{[\mathrm{B10}]} = \frac{1}{2^n}[-2^{N-1} b_{\mathrm{N\text{-}1}} + \sum_{i=0}^{N-2} 2^i b_i] \tag{3.2}$$

$x_{\min} = -2^m$, $x_{\max} = 2^m - 2^{-n}$, and Adım $= 2^{-n}$ olarak düşünüleceğinden, *Q*6.9 işaretli sabit noktalı sayı formatı bize $N = 16bit$, Aralık $= -64\,ile\,63.99805$, Hassasiyet $= 1.95310^{-3}$ değerlerini vermektedir [22].

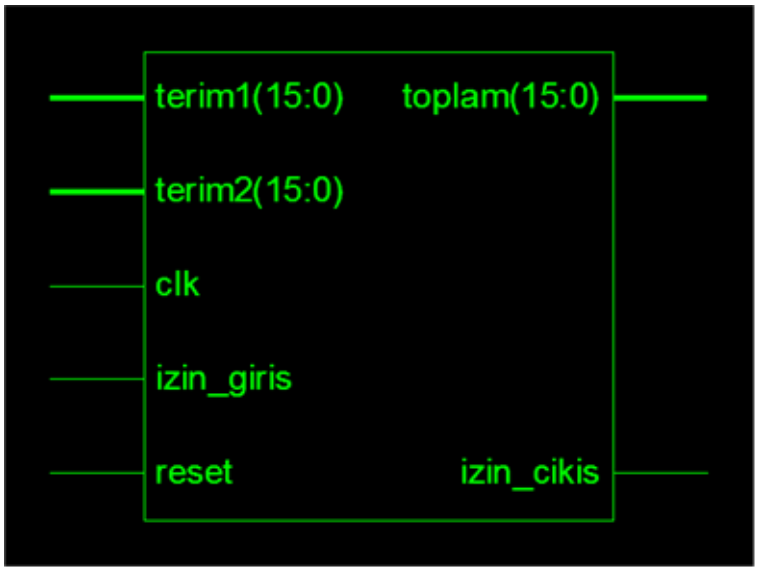
Çizelge 3.1'de *Q*6.9 sayı formatı kullanılarak elde edilebilecek işaretli sayı örnekleri verilmektedir. Sayı formatı tamamen tasarımcıya özgü ayarlanabilmektedir. Sayının tam kısmına ait bit sayısı değiştirilerek daha küçük hassasiyete sahip sayı formatı oluştulabilir. Fakat bu çalışmada kullanılacak parametre değerlerinden ötürü en küçük hassasiyet değerinin 1.95310^{-3} olduğu öngörülmektedir.

Çizelge 3.1: 16 bit $Q6.9$ sabit noktalı formatta örnek sayılar.

Açıklama	Formata Göre Gösterimi	Değer
Sıfır	0 000000 000000000	0
Gösterilebilen en küçük sayı	1 000000 000000000	-64
Gösterilebilen en büyük sayı	0 111111 111111111	63.998046875
1'den küçük bir sayı	0 000000 101010010	0.66
1'den büyük bir sayı	0 111011 110011010	59.8
Bir	0 000001 000000000	1

3.3.2 Toplama devresinin yapısı

Hücre çekirdeğinin aritmetik işlem birimlerinden biri olan toplama devresi, 16 bitlik sabit nokta biçimli iki sayıyı 2 saat çevriminde toplamaktadır. Devre girişine verilen $terim1$ ve $terim2$ değerleri kayan noktalı sayıların toplanmasında olduğu gibi işaret biti ayrımına sokulmazlar. Sayıları oluşturan tüm bit dizileri üzerinde noktalar her zaman aynı konumdadır. Bu nedenle kayan nokta aritmetiğinde uygulanan kaydırma işlemleri sabit noktalı iki sayının toplanması işleminde uygulanmaz. Bu avantajlarından ötürü kullanılan toplama işlemi, kayan noktalı sayılardaki aritmetik toplama işlemine kıyasla daha kolaydır. Şekil 3.7'de basit bir blok gösterimi verilen toplama devresinin 5 girişi 2 çıkışı bulunmaktadır.



Şekil 3.7: Tasarlanan toplama devresinin blok gösterimi.

$terim1$ ve $terim2$ toplanacak iki sayıyı temsil etmektedir. Bu iki sayı 16 bit sabit nokta biçiminde olup hücre çekirdeği tarafından toplama devresine gönderilmektedir. Tasarımdaki tüm devrelerde olduğu gibi toplama devresine de *clk* ve *reset* sinyal

bilgileri iletilmektedir. Bu sinyallerle birlikte toplama devresine gönderilen iki sayının toplanması işlemine başlanması gerektiğini belirten *izin_giris* sinyali ve bu iki sayının toplanıp çekirdeğe gönderilmeye hazır olduğunu belirten *izin_cikis* sinyali, toplama devresini kontrol edilebilir ve izlenebilir kılmaktadır.

HYSA hücresi matematiksel ifadeyi çözümlerken toplama işlemi için bu devreyi kullanmaktadır. Hücre çekirdeği içerisindeki kontrol bloğu matematiksel denklemi çözümlerken toplama işlemi gerektiğinde, öncelikle toplama devresinin girişleri olan 16 bit sabit noktalı $terim1$ ve $terim2$ sayılarını ve bir saat çevrimi sonra ise kendisi için $topla$ fakat toplama devresi için *izin_giris* sinyali anlamına gelen lojik 1 sinyalini toplama devresine gönderir. Bu sinyal ile birlikte devre iki sayıyı toplar ve $toplam$ adında yine 16 bit genişlikte sabit noktalı sayıyı hücreye döndürür. Toplam devresi, $toplam$ değeri ile birlikte hücreye $toplandi$ sinyaline karşılık düşen *izin_cikis* sinyalini lojik 1 yapar. Böylece toplama devresi, kendisine gönderilen iki sayıyı toplamış ve daha sonraki toplama işlemi için gelecek iki sayıyı ve *izin_giris* sinyalini hazırda bekler.

Aynı işaretli iki sayıyı toplarken ya da bu iki sayı çıkarma işlemine tabi tutulduğunda sonuç, beklenen bit genişliğine sığmayablir. Aritmetik taşma olarak belirtilen bu durum, sabit noktalı sayı gösterimi için tehlike arz etmektedir. Tasarımda bu durumla karşılaşmamak için sayının tam kısmına ait bit sayısı büyük tutulmuş olup bu nedenden dolayı ikili sayıların artimetik işlem uygulamaları için önemli bir unsur olan hassasiyet değeri istenilen değerden büyük kullanılmaktadır.

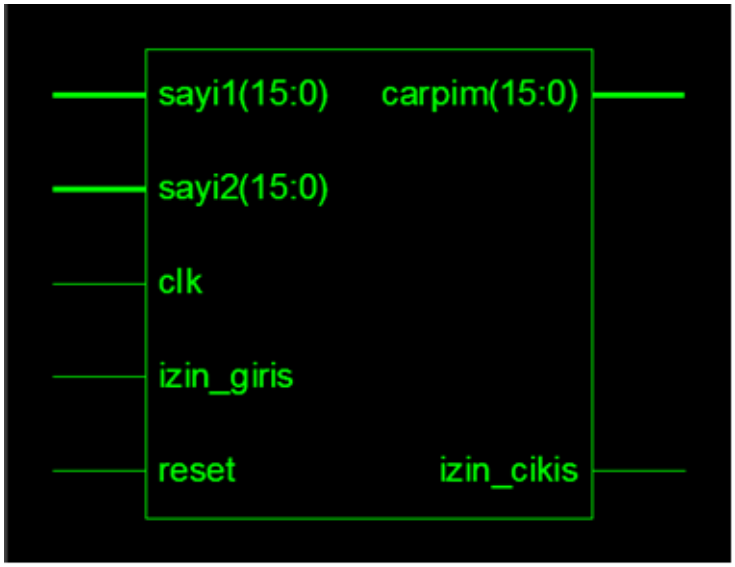
Çizelge 3.2: Toplama devresinin kaynak kullanımı ve karşılaştırılması.

Aritmetik Devreler	Dilim Sayısı	Flip-Flop Sayısı	LUT Sayısı	Maksimum Gecikme
Sabit noktalı Toplama Devresi	9	1	18	2
Kayan noktalı Toplama Devresi [14]	126	128	202	20

Çizelge 3.2'den de görüldüğü üzere tasarlanan toplama devresi, önceki tasarıma kıyasla daha az kaynak kullanmaktadır. Devre önceki tasarıma göre daha hızlı cevap vermekte olup iki sabit noktalı sayıyı 2 saat çevrim süresinde toplayarak maksimum gecikme anlamında başarımı %90 ve kaynak kullanım başarımı ise %95 olarak elde edilir.

3.3.3 Çarpma devresinin yapısı

Hücre çekirdeğinin aritmetik işlem birimlerinden bir diğeri ise çarpma devresi olup, 16 bitlik sabit nokta biçimli iki sayıyı 2 saat çevriminde çarpmaktadır. Şekil 3.8'de basit bir blok gösterimi verilen çarpma devresinin de 5 girişi 2 çıkışı bulunmaktadır.



Şekil 3.8: Tasarlanan çarpma devresinin blok gösterimi.

*sayi*1 ve *sayi*2 çarpılacak iki sayıyı temsil etmektedir. Bu iki sayı 16 bit sabit nokta biçiminde olup hücre çekirdeği tarafından çarpma devresine gönderilmektedir. Toplama devresinde olduğu gibi çarpma devresine de *clk* ve *reset* sinyal bilgileri iletilmektedir. Bu sinyallerle birlikte çarpma devresine gönderilen iki sayının çarpılması işlemine başlanması gerektiğini belirten *izin_giris* sinyali ve bu iki sayının çarpılıp çekirdeğe gönderilmeye hazır olduğunu belirten *izin_cikis* sinyali, çarpma devresini kontrol edilebilir ve izlenebilir kılmaktadır.

Çarpma işlemi için öncelikle çarpma devresinin girişleri olan 16 bit sabit noktalı *sayi*1 ve *sayi*2 sayılarını ve bir saat çevrimi sonra ise kendisi için *carp* fakat çarpma devresi için *izin_giris* sinyali anlamına gelen lojik 1 sinyalini çarpma devresine gönderir. Bu sinyal ile birlikte devre iki sayıyı çarpar ve *carpim* adında yine 16 bit genişlikte sabit noktalı sayıyı hücreye döndürür. Çarpma devresi, *carpim* değeri ile birlikte hücreye *carpildi* sinyaline karşılık düşen *izin_cikis* sinyalini lojik 1 yapar. Böylece çarpma devresi, kendisine gönderilen iki sayıyı çarpmış ve daha sonraki çarpma işlemi için gelecek iki sayıyı ve *izin_giris* sinyalini hazırda bekler.

Çizelge 3.3: Çarpma devresinin kaynak kullanımı ve karşılaştırılması.

Aritmetik Devreler	Dilim Sayısı	Flip-Flop Sayısı	LUT Sayısı	Maksimum Gecikme
Sabit noktalı Çarpma Devresi	9	1	17	2
Kayan noktalı Çarpma Devresi [14]	35	20	61	2

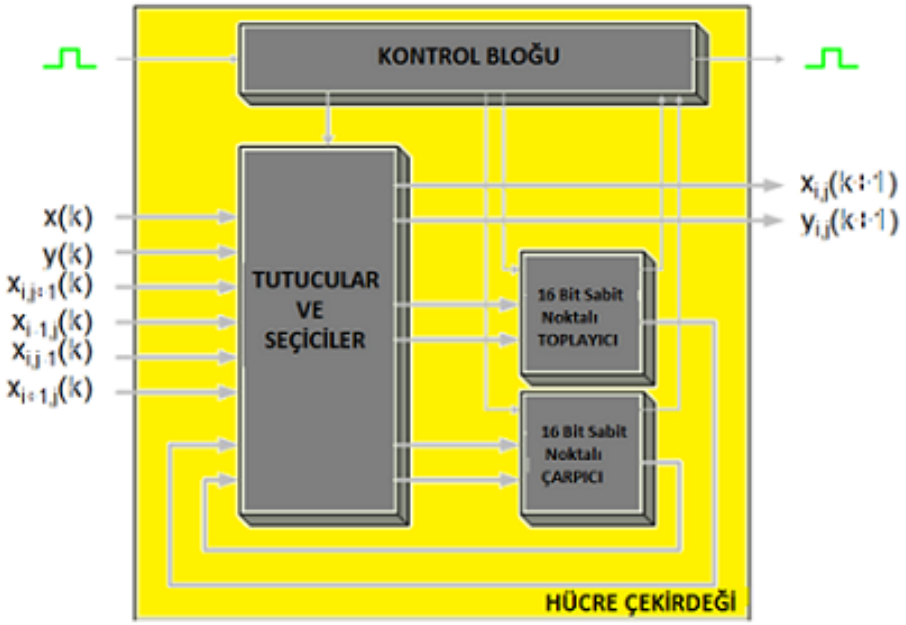
Çizelge 3.3'den de görüldüğü üzere sabit noktalı sayılar için tasarlanan çarpma devresi, kayan noktalı sayılar için tasarlanan çarpma devresinden daha az kaynak kullanmaktadır. Fakat devre önceki tasarımla aynı hızda cevap vermekte olup iki sabit noktalı sayıyı 2 saat çevrim süresinde çarpmaktadır. Tasarlanan çarpma devresinin kaynak kullanım başarımı ise %80 olarak elde edilir.

3.3.4 Yeni NPE modülü

HYSA devresinin en önemli birimlerinden biri olan Nöral İşlem Elemanı (Neural Processing Element, NPE), hücrenin işlevini yerine getiren modüldür. NPE, üzerinde işlemler için gerekli değerlerin tutucusunu barındırır fakat herhangi bir hücreye ait değerleri sürekli olarak tutmaz. NPE'lerin tasarımı, kendilerinden oluşturulacak bir matris ile paralel işlem yeteneğini devreye kazandıracak şekilde yapılmıştır. 128x128 hücreli ağ için 4x4 paralel işlem üniteleri bu NPE'ler ile sağlanmıştır.

NPE iterasyona başlamadan önce değişken girişlerine ve katsayı girişlerine uygun işaretler verilmelidir. NPE, Altbölüm 3.3.2 ve Altbölüm 3.3.3 'de tasarımı anlatılan toplayıcı ve çarpıcı alt modüllerini içerir. Şekil 3.9'da ise NPE modülünü oluşturan alt modüllerin blok gösterimi verilmektedir.

Devre, içerdiği her bir NPE'yi ilgili nöron verisi ile ilişkilendirecek şekilde üst modüller halinde tasarlanmıştır. Bu üst modüller NPE'nin değişken (x_sağ, x_ust, x_sol, x_alt, xk, yk, u), ağırlık (alfa_sağ, alfa_ust, alfa_sol, alfa_alt, a, b, c, d) ve diğer parametre (limit, m, T, pasif_değeri) girişlerine uygun işaretleri verir. Bu işaretlerin girişlere uygulanmasının ardından NPE'nin x_izin_giris'ine lojik 1 darbesi uygulandığında NPE içinde iterasyonu yürüten durum makinası çalışır. NPE, önce x[k+1] değerini hesaplar ve çıkışa verir. Bu çıkışı, üst modüllere x_izin_cikis sinyalini lojik 1 yaparak bildirir. Daha sonra ise üst devre çıkıştaki x[k+1] değerini okur ve y_izin_giris'i lojik 1 yaparak NPE'nin y[k+1] değerini hesaplaması için işlemlere devam eder. y[k+1] değerinin hesaplanmasının ardından NPE, y_izin_cikis'a lojik 1



Şekil 3.9: NPE'nin blok gösterimi.

uygulayarak üst modüllerin çıkıştan bu değeri okumasını sağlar. NPE modülünün giriş çıkışları Çizelge 3.4'de listelenmiştir.

Çizelge 3.4: NPE modülünün [14] giriş çıkışları.

İşaret Adı	Bit Genişliği	G/Ç	Açıklama
clk	1	giriş	NPE'yi ve alt modüllerini yükselen kenarında tetikleyen saat işareti
reset	1	giriş	Asenkron aktif 0 reset işareti
x_izin_giris	1	giriş	x[k+1]'in hesaplanmasını başlatan 1 saat çevrimi süreli lojik 1 darbesi uygulanan kontrol işareti
x_izin_cikis	1	çıkış	x[k+1]'in hesaplandığını 1 saat çevrimi süreli lojik 1 darbesi ile bildiren kontrol işareti
y_izin_giris	1	giriş	y[k+1]'in hesaplanmasını başlatan 1 saat çevrimi süreli lojik 1 darbesi uygulanan kontrol işareti
y_izin_cikis	1	çıkış	y[k+1]'in hesaplandığını 1 saat çevrimi süreli lojik 1 darbesi ile bildiren kontrol işareti
x_sag	16	giriş	İlgili nöronun sağ (doğu) komşusuna ait x[k] değişken değeri
x_ust	16	giriş	İlgili nöronun üst (kuzey) komşusuna ait x[k] değişken değeri
x_sol	16	giriş	İlgili nöronun sol (batı) komşusuna ait x[k] değişken değeri
x_alt	16	giriş	İlgili nöronun alt (güney) komşusuna ait x[k] değişken değeri
xk	16	giriş	İlgili nörona ait x[k] değişken değeri
yk	16	giriş	İlgili nörona ait y[k] değişken değeri
u	16	giriş	İlgili nörona ait u değişken değeri
cikis	16	çıkış	Hesaplanan değer, x[k+1] veya y[k+1]'dir
alfa_sag	16	giriş	Sağ (doğu) komşu nöron için kuplaj ağırlığı
alfa_ust	16	giriş	Üst (kuzey) komşu nöron için kuplaj ağırlığı
alfa_sol	16	giriş	Sol (batı) komşu nöron için kuplaj ağırlığı
alfa_alt	16	giriş	Alt (güney) komşu nöron için kuplaj ağırlığı
limit	16	giriş	g fonksiyonunun hesabında kullanılan x_{limit} değeri
m	16	giriş	g fonksiyonunun hesabında kullanılan katsayı değeri
a	16	giriş	x[k+1] için x[k]'nın ağırlığı
b	16	giriş	x[k+1] için y[k]'nın ağırlığı
c	16	giriş	y[k+1] için x[k]'nın ağırlığı
d	16	giriş	yk+1] için y[k]'nın ağırlığı
T	16	giriş	Denklemdeki zaman farkı ifadesi
pasif_değeri	16	giriş	x_{pasif} değeri

Tasarımda 16 adet NPE bulunmakta ve tüm NPE'ler paralel çalışmaktadır. Bu paralel yapıya ait detaylı bilgi CNPN modülünde açıklanmaktadır. NPE tasarımı önceki çalışma [14] ile aynı yapıdadır fakat içinde barındırdığı toplama ve çarpma devreleri dolayısıyla farklılık arz etmektedir. NPE modülünün kaynak kullanımı

ve maksimum gecikme bakımından önceki çalışmaya göre karşılaştırılması Çizelge 3.5'de gösterilmektedir. Elde edilen bu değerler, kullanılan Xilinx XC2VP30 FPGA'sı içindir. NPE tasarımlarından Eski Hücre Modülü ve NPE v1.3o Modülü kayan noktalı sayı aritmetiği kullanılarak tasarlanırken, Yeni NPE Modülü bu çalışmaya özgü olarak Q6.9 işaretli sabit noktalı sayı aritmetiği kullanılarak tasarlanmıştır. Yeni NPE modülünün, eski ve NPE v1.3o modüllerine göre daha az alan kapladığı görülmektedir.

Çizelge 3.5: NPE tasarımlarının karşılaştırılması.

	Eski Hücre Modülü [14]	NPE v1.3o modülü [14]	Yeni NPE modülü
Lojik Dilim Sayısı	417	510	395
FF Sayısı	387	624	416
LUT Sayısı	688	918	705
MULT18x18 Sayısı	1	1	1
En Uzun İterasyon Süresi	258	271	50

Çizelge 3.6: 128x128 hücreli RO-HYSA'nın NPE devresinde bir iterasyonda gerçekleşen işlemler.

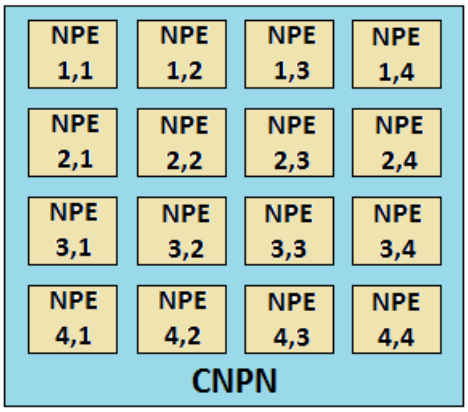
İşlem Sırası	Gerçekleştirilen İşlem	İşlem Sırası	Gerçekleştirilen İşlem
1	**eğer** $x_{\mathrm{i,j}}[k] = x_{\mathrm{pasif}}$ **ise**		**son**
2	$x_{\mathrm{i,j}}[k+1] = x_{\mathrm{pasif}}$	19	**eğer** $x_{\mathrm{limit}} \geq x_{\mathrm{i,j}}[k] > -x_{\mathrm{limit}}$ **ise**
	son	20	$toplam_1$=0
3	**eğer** $x_{\mathrm{i,j}}[k] \neq x_{\mathrm{pasif}}$ **ise**		**son**
4	$carpim_1 = a_{\mathrm{dogu}} \cdot x_{\mathrm{i,j+1}}[k]$	21	$carpim_1 = \mathrm{m} \cdot toplam_1$
5	$carpim_2 = a_{\mathrm{bati}} \cdot x_{\mathrm{i,j-1}}[k]$	22	$toplam_2 = carpim_1 + toplam_2$
6	$toplam_1 = carpim_1 + carpim_2$	23	$carpim_1 = \mathrm{T} \cdot toplam_2$
7	$carpim_1 = a_{\mathrm{kuzey}} \cdot x_{\mathrm{i-1,j}}[k]$	24	$toplam_2 = carpim_1 + x_{\mathrm{i,j}}[k]$
8	$carpim_2 = a_{\mathrm{guney}} \cdot x_{\mathrm{i+1,j}}[k]$	25	$x_{\mathrm{i,j}}[k+1] = toplam_2 + \mathrm{u}$
9	$toplam_2 = carpim_1 + carpim_2$		**son**
10	$toplam_2 = toplam_1 + toplam_2$	26	**eğer** $x_{\mathrm{i,j}}[k] = x_{\mathrm{pasif}}$ **ise**
11	$carpim_1 = \mathrm{a} \cdot x_{\mathrm{i,j}}[k]$	27	$y_{\mathrm{i,j}}[k+1] = y_{\mathrm{i,j}}[k]$
12	$toplam_2 = carpim_1 + toplam_2$		**son**
13	$carpim_1 = \mathrm{b} \cdot y_{\mathrm{i,j}}[k]$	28	**eğer** $x_{\mathrm{i,j}}[k] \neq x_{\mathrm{pasif}}$ **ise**
14	$toplam_2 = carpim_1 + toplam_2$	29	$carpim_1 = \mathrm{c} \cdot x_{\mathrm{i,j}}[k]$
15	**eğer** $x_{\mathrm{i,j}}[k] > x_{\mathrm{limit}}$ **ise**	30	$carpim_2 = \mathrm{b} \cdot y_{\mathrm{i,j}}[k]$
16	$toplam_1 = x_{\mathrm{i,j}}[k]$ - x_{limit}	31	$toplam_1 = carpim_1 + carpim_2$
	son	32	$carpim_1 = \mathrm{T} \cdot toplam_1$
17	**eğer** $x_{\mathrm{i,j}}[k] < -x_{\mathrm{limit}}$ **ise**	33	$y_{\mathrm{i,j}}[k+1] = y_{\mathrm{i,j}}[k] + carpim_1$
18	$toplam_1 = x_{\mathrm{i,j}}[k] + x_{\mathrm{limit}}$		**son**

Çizelgedeki LUT, FPGA üzerinde kombinezonsal kapı elemanlarının gerçeklendiği bellek tabanlı 'look-up table'dır. FF ise flip-flop anlamındadır. MULT18x18, kullanılan FPGA'nın içinde hazır bulunan 18 bitlik çarpıcı donanım elemanıdır. Yeni NPE modülü bir önceki tasarım olan NPE v1.3o modülüne göre %23 daha az sayıda LUT, %33 daha az sayıda FF ve %23 daha az sayıda lojik dilim kullanmaktadır.

Sabit noktalı sayılar kullanılarak NPE içerisinde (2.10)'daki denklem takımını çözen algoritma Çizelge 3.6'da verilmiştir. NPE'nin algoritmayı bir kez koşması ile ilgili nöronun bir iterasyonu gerçeklenmiş olur.

3.3.5 CNPN modülü

Tasarımda 16 adet NPE birleştirilerek 4x4 boyutlu bir hücresel nöral işlem ağı (Cellular Neural Processing Network, CNPN) kurulmuştur. Şekil 3.10'da CNPN'ye ait blok diyagram ve Şekil 3.11'de ise CNPN'nin hiyerarşik yapısı gösterilmektedir.



Şekil 3.10: CNPN'nin blok gösterimi.

Tasarımda CNPN, iki önemli görevi üstlenmektedir. Bunlardan ilki, girişlerine uygulanan değişken, ağırlık ve parametre işaretlerini barındırdığı NPE'lere dağıtmak ve NPE çıkışlarını üst modüllere aktarmaktır. Bunu yapabilmek için NPE'ler uygun iç ve dış bağlantılar ile CNPN modülü içinde birleştirilmiştir. CNPN modülünde tüm NPE'ler için ortak olan ağırlık (alfa_sağ, alfa_ust, alfa_sol, alfa_alt, a, b, c, d) ve parametre (limit, m, T, pasif_değeri) girişleri ile her hücreye özel ulaştırılan değişken (x_sağ, x_ust, x_sol, x_alt, xk, yk, u) girişleri mevcuttur.

CNPN'nin ikinci önemli görevi ise, NPE'lerin iterasyona başlatılması ve iterasyonun sonlandırılmasıdır. Üst modüllerden gelen iterasyona başlama sinyali, CNPN bağlantıları sayesinde tüm NPE'lere ulaştırılır ve NPE'ler paralel bir şekilde iterasyona başlatılır. Sabit noktalı sayı aritmetiği kullanılması dolayısıyla tüm NPE'ler aynı anda iterasyona başlar ve aynı anda iterasyonu bitirirler. Kayan noktalı sayı aritmetiğinde ortaya çıkan iterasyon süresi belirsizliği bu sayede ortadan kaldırılmış ve yeni iterasyon için tüm NPE'ler aynı anda hazır bekleyerek çıkışlar üst modüllere aktarılır. CNPN modülünde her bir NPE için 2 adet, toplamda 32 adet bayrak

Şekil 3.11: CNPN'nin hiyerarşik yapısı.

(flipflop) bulunmaktadır. Bu bayraklar iterasyon esnasında hesaplanan x[k+1] değeri ve iterasyon sonunda elde edilen y[k+1] değerlerinin işaretçisidir. NPE'ler ilk olarak x[k+1] değerini daha sonra y[k+1] değerini hesaplarlar. Tüm NPE'lerin önce x[k+1] daha sonra y[k+1] değerlerinin hesaplandığına dair bayrakları kaldırmalarının ardından yeni iterasyon başlatılır ve bu bayraklar indirilir. Böylece ardışıl bir şekilde tüm ağ hücreleri için aynı işlemler tekrarlanır [14].

Çizelge 3.7'de kayan noktalı sayı aritmetiği kullanılacak şekilde tasarlanan CNPN v1.2 modülü ile sabit noktalı sayı aritmetiği kullanılacak biçimde tasarlanan Yeni CNPN modülü kaynak kullanımı bakımından kıyaslanmaktadır.

Çizelge 3.7: CNPN v1.2 modülü ile Yeni CNPN modülünün karşılaştırılması.

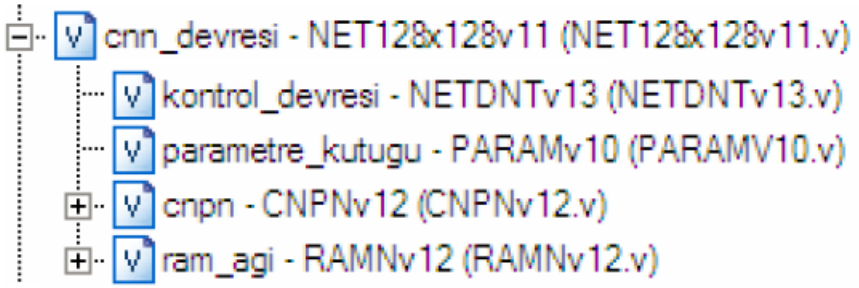
	CNPN v1.2 modülü [14]	Yeni CNPN modülü
Lojik Dilim Sayısı	8192	6041
FF Sayısı	10023	6952
LUT Sayısı	14745	10552
MULT18x18 Sayısı	25	16

Tablodan da görüldüğü üzere yeni tasarım CNPN v1.2 modülüne kıyasla daha az kaynak kullanmaktadır. Yeni tasarımda %26 oranında daha az lojik dilim, %31 oranında daha az FF, %28 oranında daha az sayıda LUT ve % 36 oranında daha az sayıda MULT18x18 çarpıcı donanım elemanı kullanılmaktadır. Bu iyileştirmeler

gerçek anlamda NPE içerisinde bulunan toplayıcı ve çarpıcı devrelerin daha az sayıda kaynak kullanması ile elde edilmektedir. Kaynak kullanımının bu oranlarda azalmasının en önemli sebebi aritmetik devrelerde kullanılan sabit noktalı sayı aritmetiğidir.

3.3.6 Dalga bilgisayarı çekirdeği

Dalga Bilgisayarı Çekirdeği adı verilen HYSA öykünleyici sayısal devre, Kontrol Devresi, Parametre Kütüğü, CNPN Devresi ve Ram Ağını kapsamaktadır. Dalga Bilgisayarı Çekirdeği, 128x128 hücreli programlanabilir RO-HYSA öykünleme işinin temel bloğudur ve kısaca Çekirdek olarak bahsedilecektir [14]. Şekil 3.12'de Çekirdeğin hiyerarşik yapısı ve Çizelge 3.8'de önceki tasarıma kıyasla başarımı ve kaynak tüketim değerleri gösterilmektedir.



Şekil 3.12: Çekirdeğin hiyerarşik yapısı.

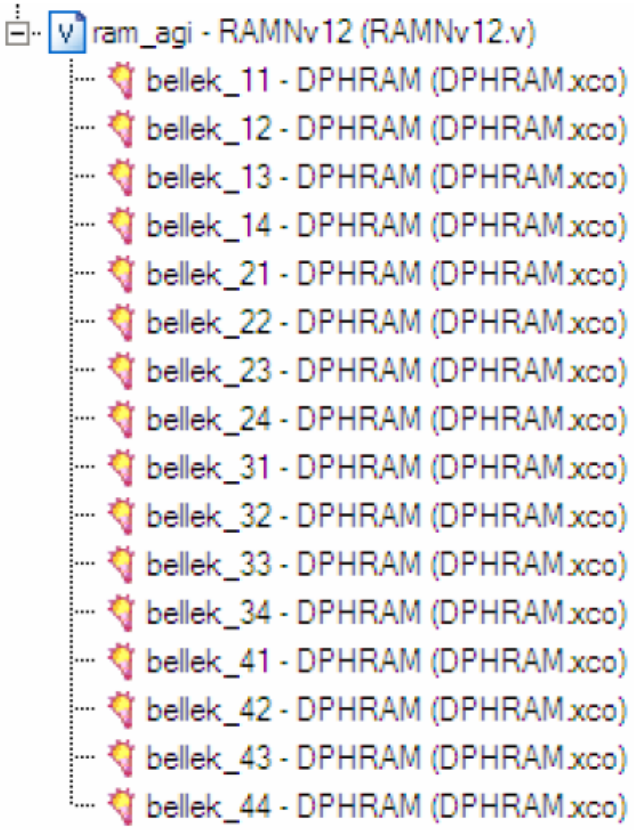
Yeni Çekirdek tasarımı önceki tasarıma göre daha az kaynak kullanmaktadır. Bu tasarım önceki çekirdek tasarımına kıyasla %28 oranında daha az sayıda lojik dilim, %20 oranında daha az sayıda FF, %29 oranında daha az sayıda LUT ve %36 oranında daha az sayıda MULT18x18 çarpıcı donanım elemanı tüketmektedir. Kullanılan Block RAM sayısı her iki tasarımda da 80'dir.

Çizelge 3.8: HYSA öykünleyici Çekirdeğin önceki tasarım ile karşılaştırılması.

	Önceki Çekirdek Tasarımı [14]	Yeni Çekirdek devresi
Lojik Dilim Sayısı	10731	7725
FF Sayısı	12296	9872
LUT Sayısı	17825	12685
Block RAM Sayısı	80	80
MULT18x18 Sayısı	25	16

128x128 boyutlu ağ için çekirdek toplam 16,384 nöronu öykünleyebilmektedir. Her bir nörona ait x[k], x[k+1], y[k], y[k+1] ve u değişkenleri bellekte tutulur. Belirtilen her

bir değişken 16 bit genişliğinde olup ağın toplam bellek ihtiyacı 160 KB'dır. Bu bellek ihtiyacı, FPGA içerisinde bulunan toplam kapasitesi 272 KB olan BlockRAM'lerden karşılanır. Bu bellek elemanları, Bellek Dizisi modülü içerisinde bir araya getirilmiştir. Bu modül, CNPN modülündeki gibi birbirine paralel dizilmiş 16 adet RAM bloğundan ve bu blokları kontrol eden bir kısımdan oluşmaktadır. Bu RAM komponenti dual-port hücresel RAM (DPHRAM) olarak tanımlanmış olup kapasitesi 16 bit uzunluklu 5120 kelimedir. A portundan okuma ve yazma yapılabilirken, B portundan sadece okuma yapılabilir. Adres veri yolu genişliği ise 13 bittir. Bellek dizisindeki her bir DPHRAM, CNPN modülündeki her bir NPE ile eşleşmektedir. Şekil 3.13'de bellek dizisi modülünün hiyerarşik yapısı gösterilmektedir.



Şekil 3.13: Bellek Dizisi Modülünün hiyerarşik yapısı.

RAM ağı modülünün çalışması paralel modülü olan Kontrol devresinin kontrolündedir. Kontrol devresi, RAM ağından HYSA'nın 4x4'lük belirli bir bölümüne ait x, y, u değişken setini okumasını ve çıkış portuna yazmasını, giriş portlarındaki verileri HYSA'nın 4x4'lük belirli bir bölümüne ait x, y veya u değişkeni olarak kaydetmesini isteyebilir. Bu veri akışı DPHRAM komponentlerinin A portunda gerçekleşmektedir.

Kontrol devresi, RAM ağının tuttuğu verilerin denetleyici bilgisayar tarafından okunmasını ve güncellenmesini de sağlar. Bunun için ileride anlatılacak olan haberleşme devresi de kontrol devresi tarafından kullanılır. Benzer şekilde kontrol devresi belirli bir adres için aldığı 16 kelimelik veriyi haberleşme devresi üzerinden denetleyici bilgisayara gönderir, ya da bilgisayardan gelen 16 kelimelik veriyi ilgili adres gözlerine yazar.

Tasarımda iterasyon sonucunda hesaplanan değişken değerleri izlenebilmektedir. Bu özellik için DPHRAM'lerin B portu kullanılır. B portunun kontrolü ve çıkış veriyolları VGA sürücü devreye bağlıdır. Bu sayede değişken değerleri VGA ekranda iterasyon sürerken gerçek zamanlı olarak okunabilmekte ve ağ görüntüsü oluşturulmaktadır. B portu adresinin değişken bitleri ise FPGA kartı üzerindeki iki adet sürgü anahtar ile girilir. Bu sayede kullanıcı kart üzerindeki anahtarı açıp kapayarak, sistem monitöründe görmek istediği ağ değişkeninin adresinin oluşmasını sağlar.

Çekirdeğin bir diğer alt modülü, tüm ağın kullandığı parametreleri, ilgili değişken değerlerini ve devrenin çalışması ile ilgili bilgileri saklayan Parametre Kütüğü'dür. Bu modülün RAM ağından farklı olmasını sağlayan özellik, sakladığı verilerin tüm ağ için ortak olmasıdır. Yapısal farklılığı ise, verileri RAM üzerinde değil flipfloplardan oluşturulan tutuculardа saklamasıdır. Sakladığı veriler için tasarlanan veriyolları tüm işlem bloklarına paralel bağlanmaktadır. Çizelge 3.9'da Parametre kütüğünde saklanan parametreler tanıtılmaktadır.

Parametre kütüğü parametre değerlerini çıkışlarından sürekli servis etmektedir. Paralel çalıştığı kontrol devresi, parametre kütüğündeki parametre değerlerini 16 bit genişlikli başka giriş ve çıkış portlarından okuyabilir ya da değiştirebilir. Bu işlem sayesinde denetleyici bilgisayar ile parametrelere erişilebilir. Ayrıca kontrol devresi tüm ağın iterasyonu koşmasının ardından parametre kütüğündeki koşulacak ve iterasyon değerlerini sırasıyla azaltıp artırır. Ağın çalışacak kısmının sol, sağ, üst ve alt marjinleri, kontrol devresi tarafından her iterasyona başlandığında okunur ve güncel değerleri kullanılır.

Kontrol devresi, sadece Çekirdeği değil, tüm tasarımı kontrol etmektedir. CNPN'nin iterasyonu gerçekleştirmesi, RAM ağının yeni değişkenleri okuması ve hesaplanan değerlerin yazılması, parametre kütüğündeki değerlerin okunması ve yazılması, haber-

Çizelge 3.9: Parametre Kütüğünde saklanan parametreler.

Parametre Adı	Bit Genişliği	Açıklama
alfa_sag	16 bit	Nöronların doğu kuplaj ağırlığıdır
alfa_ust	16 bit	Nöronların kuzey kuplaj ağırlığıdır
alfa_sol	16 bit	Nöronların batı kuplaj ağırlığıdır
alfa_alt	16 bit	Nöronların güney kuplaj ağırlığıdır
limit	16 bit	g fonksiyonunun hesabında kullanılan x_{limit} değeri
m	16 bit	g fonksiyonunun hesabında kullanılan katsayı değeri
a	16 bit	x[k+1] için x[k]'nın ağırlığı
b	16 bit	x[k+1] için y[k]'nın ağırlığı
c	16 bit	y[k+1] için x[k]'nın ağırlığı
d	16 bit	y[k+1] için y[k]'nın ağırlığı
T	16 bit	Denklemdeki zaman farkı ifadesi
pasif_değeri	16 bit	x_{pasif} değeri
solsag	10 bit	Ağın çalıştırılacak kısmının sol ve sağ marjinini belirtir
ustalt	10 bit	Ağın çalıştırılacak kısmının üst ve alt marjinini belirtir
kosulacak	16 bit	Ağın kaç iterasyon koşacağını belirtir
iterasyon	32 bit	Ağın kaç iterasyon koştuğunu belirtir

leşme devresinin çalıştırılması ve VGA sürücünün çalıştırılarak iterasyon esnasında istenilen değişken değerlerinin okunabilmesi kontrol devresi ile sağlanmaktadır [14]. Kontrol devresine ait bağlantılar Çizelge 3.10'da listelenmektedir.

Tasarımda kullanılan kontrol devresinin iki modu bulunmaktadır. İlk mod, denetleyici bilgisayar ile haberleşme halinde olduğu 'bağlantı' modu, diğeri ise iterasyonları gerçekleştirmek için çalıştığı 'operasyon' modudur. Devre, ilk enerjilendirildiği anda bağlantı modunda çalışmaya başlar. Bu moddayken, denetleyici bilgisayar sistem üzerindeki parametreler, değişkenler ile bellekteki değişken değerlerini okuyabilir hatta değiştirebilir. Bunlar dışında ise sistemi iterasyonu gerçekleştirmek için operasyon moduna geçirebilir. Operasyon modunda ise Çekirdek, parametre kütüğü içine yazılan 'kosulacak' parametresinin adedince iterasyonu gerçekleştirmeye çalışır. Bu değer sıfırlandığında ise istenilen sayıda iterasyon yapılmış ve artık tüm hesaplamalar ve bellek işlemleri bitip sistem operasyon modunda beklemeye başlar. Bu bekleme durumu, çekirdeğin çevre birimlerinden gelecek 'baglan' komutu alınana kadar devam eder. Sistem denetleyici bilgisayara bağlandıktan sonra kontrol devresi yardımı ile istenilen işlem yerine getirilir ve sistem tekrar operasyon moduna geçirilir. Kontrol devresi önceki çalışmada [14] olduğu haliyle bu çalışmada kullanılmış olup devrenin FPGA üzerinde kapladığı alan Çizelge 3.11'de verilmiştir.

Çizelge 3.10: Kontrol devresi [14] bağlantılarının tanıtılması.

Bağlantı Adı	Bit Genişliği	Giriş / Çıkış	İşaretin Açıklaması
x_hesapla	1	çıkış	Tüm NPE'lerin x[k+1] değerinin hesaplanmasını başlatan işaret
x_hesaplaniyor	1	giriş	NPE'lerin x[k+1] değeri hesaplamakla meşgul olduğunu belirten işaret
y_hesapla	1	çıkış	Tüm NPE'lerin y[k+1] değerinin hesaplanmasını başlatan işaret
y_hesaplaniyor	1	giriş	NPE'lerin y[k+1] değeri hesaplamakla meşgul olduğunu belirten işaret
degisken	2	çıkış	A portlarının ortak adresinin değişken bitleri
k	1	çıkış	A portlarının ortak adresinin sayfa biti
satirsutun	10	çıkış	A portlarının ortak adresinin satır ve sütun belirten bitleri
oku	1	çıkış	RAM ağının okumasını başlatan işaret
okunuyor	1	giriş	RAM ağının okuma işlemi yürüttüğünü belirten işaret
yaz	1	çıkış	RAM ağının yazmasını başlatan işaret
yaziliyor	1	giriş	RAM ağının yazma işlemi yürüttüğünü belirten işaret
Aen	1	çıkış	A portunun izin işareti
b_yaz	1	çıkış	RAM ağında bir parçaya yapılan işlemin yazma işi olduğunu belirten işaret
b_isle	1	çıkış	RAM ağında bir parçanın işlenmesini belirten işaret
b_isleniyor	1	giriş	RAM ağında bir parçanın işlenmekte olduğunu belirten işaret
Ben	1	çıkış	B portunun izin işareti
vga_degisken	2	çıkış	Sistem monitöründe gerçek zamanlı izlenecek değişkeni belirten işaret
b_giris	256	çıkış	İşlenecek RAM ağı parçasına yazılacak veri
b_cikis	256	giriş	İşlenen RAM ağı parçasından okunan veri
arttir	1	çıkış	İterasyon değerini bir artırıp, koşulacak değerini bir azaltan işaret
p_adres	5	çıkış	Erişilecek parametrenin adresi
p_yaz	1	çıkış	Erişimin bir yazma işlemi olacağını belirten işaret
p_isle	1	çıkış	Bir parametrenin işlenmesini başlatan işaret
p_isleniyor	1	giriş	Bir parametrenin işlenmekte olduğunu belirten işaret
p_giris	16	çıkış	İşlenecek parametreye yazılacak veri
p_cikis	16	giriş	İşlenen parametreden okunan veri
solsag	10	giriş	Ağın çalıştırılan bölgesinin sol–sağ marjinleri
ustalt	10	giriş	Ağın çalıştırılan bölgesinin üst–alt marjinleri
kosulacak	16	giriş	Ağın koşması gereken iterasyon sayısı
gidecek_hazir	1	çıkış	Haberleşme hattından bir baytlık veri gönderilmesini başlatan işaret
gonderiliyor	1	giriş	Bir baytlık verinin gönderiliyor olduğunu belirten işaret
gelen_var	1	giriş	Haberleşme hattından bir baytlık veri geldiğini belirten işaret
alimda_hata	1	giriş	Hattan veri alımı esnasında hata algılandığını belirten işaret
gelen_alindi	1	çıkış	Haberleşme hattından gelen bir baytlık verinin alındığını belirten işaret
giden_bayt	8	çıkış	Haberleşme hattından gönderilecek bir baytlık veri
gelen_bayt	8	giriş	Haberleşme hattından gelen bir baytlık veri

Çizelge 3.11: Kontrol Devresinin kaynak kullanımı.

	Kontrol Devresi
Lojik Dilim Sayısı	424
FF Sayısı	328
LUT Sayısı	837

3.3.7 Çekirdeğin çevre birimleri

Çalışmada tasarlanan 128x128 hücreli programlanabilir RO-HYSA'nın çevre birimlerinden biri, devre çalışırken belirlenen veriyolları ve tutucuların değerleri ve tasarlanan modüllerin giriş çıkış sinyallerinin izlenebilirliği için kullanılan Chipscope aracıdır. Devre tasarımına dahili lojik izleyici modüller eklenerek veriyolu ve tutucuların değerleri bilgisayara programlama kablosu ile aktarılmaktadır.

Tasarım, RS-232 portu üzerinden harici bir donanım olan bilgisayar ile haberleşebilir. 16,384 hücreli ağa ait x, y ve u değişkenleri, tüm ağırlıklar ve kullanılan

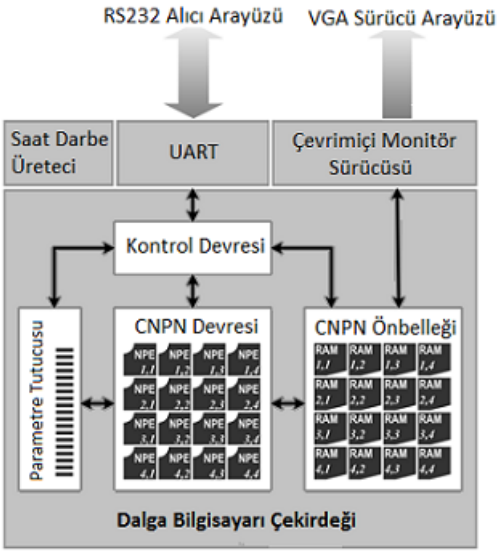
parametreler bu haberleşme kanalı üzerinden tasarlanan RO-HYSA'ya yazılabilir ya da RO-HYSA'dan okunabilir. Bu özellik sayesinde devre, kullanıcıya farklı parametreler ile farklı uzaylar oluşturarak çalışma ve çalışmasının sonuçlarını saklayabilme yeteneği sağlamaktadır. Ağa ait x, y ve u değişkenlerinin izlenebilmesi için tasarıma VGA devresi eklenmiş ve kart üzerindeki anahtarlar ile bu değişkenlerin gözlenebilirliği sağlanmıştır. Kullanılan haberleşme devresi, RS-232 protokolünü yürüten bir UART'dır. Bayt almak ve bayt göndermek için alıcı ve verici adında iki devre kullanılmaktadır. Haberleşme devresi de HYSA devresi gibi Kontrol Devresi tarafından kontrol edilir. Çalışmada kullanılan haberleşme ve VGA modülleri, Kontrol Devresi gibi önceki çalışmaya [14] aittir ve çalışmaya temel teşkil eden sayı aritmetiği değişiminden etkilenmemekte olup bu modüllerin kaynak tüketimi Çizelge 3.12'de verilmektedir.

Çizelge 3.12: Kullanılan Haberleşme ve VGA sürücü devrelerinin kaynak kullanımı.

	Haberleşme devresi	VGA sürücü devresi
Lojik Dilim Sayısı	72	127
FF Sayısı	81	50
LUT Sayısı	129	243

Kapı, flipflop ve veriyolu sayısının çok olması dolayısıyla tasarlanan devrenin gerçeklenmesi güçleşmekte ve bazı veriyollarının veri iletim sürelerinin uzun olması dolayısıyla, FPGA tümdevresine kart üzerindeki osilatörden sağlanan 100 MHz'lik ana saat işaretinin frekansı, mclk_ureteci isimli modülle 4'e bölünmektedir. Böylece 128x128 RO-HYSA devresinin tüm alt modüllerinin saat girişleri, 25 MHz'lik saat işareti ile sürülmektedir. Periyodun 40 ns yapılması ile tüm veriyolları gecikme süreleri önemsiz hale gelmiş ve devrenin aksamadan çalışması garanti edilmiştir. Kısaca tanıtılan bu çevre birimler ile birlikte RO-HYSA'nın blok gösterimi Şekil 3.14'de verilmekte olup AWG (Auto-wave Generator, Oto-dalga Üreteci) adında kısaltılmış olarak anılan ve sabit noktalı sayı aritmetiği kullanılarak yenilenen tasarımın önceki çalışmaya [14] karşın başarımı ve kaynak kullanımı Çizelge 3.13'de verilmektedir.

Yenilenmiş AWG tasarımı sayesinde 128x128 hücreli programlanabilir RO-HYSA devresi önceki çalışmaya göre %19 oranında daha az sayıda lojik dilim, %21 oranında daha az sayıda FF, %21 oranında daha az sayıda LUT ve %36 oranında daha az sayıda MULT18x18 kullanmaktadır. Kullanılan Block RAM sayısı, önceki çalışmayla



Şekil 3.14: AWG tasarımının blok gösterimi.

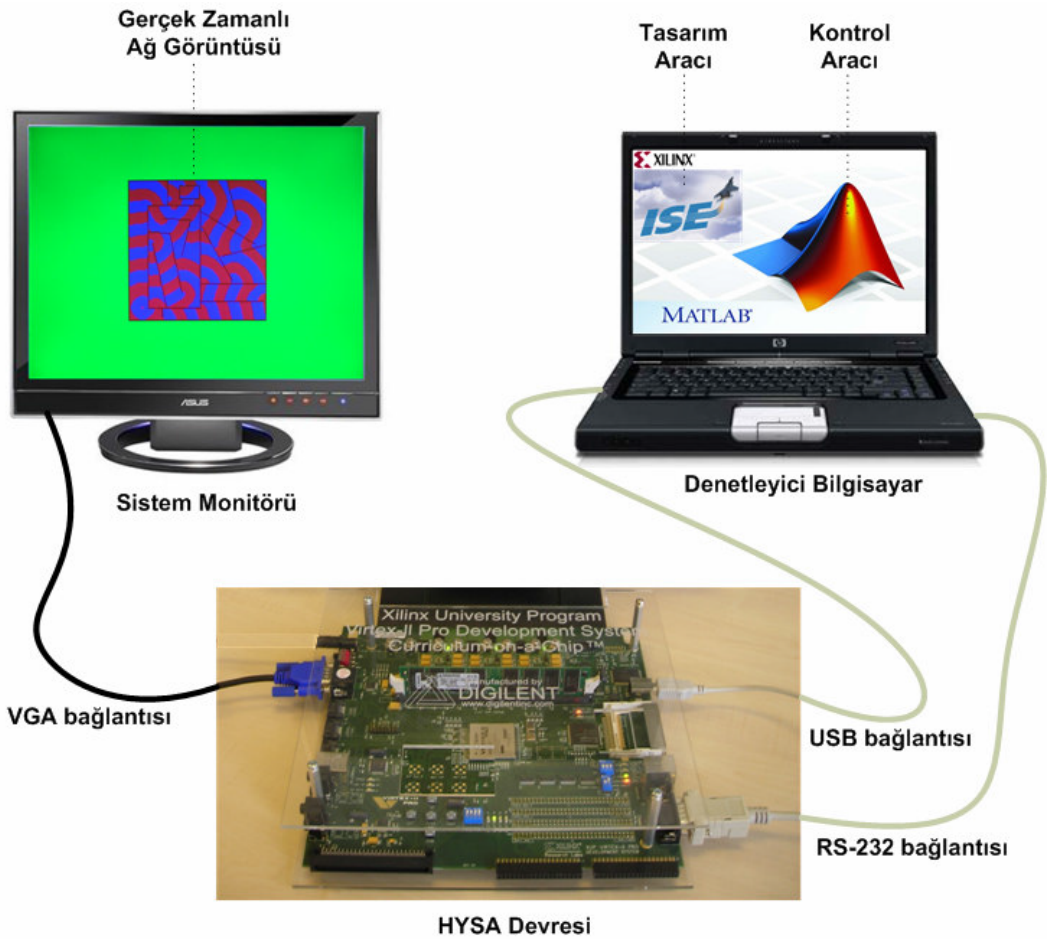
Çizelge 3.13: 128x128 programlanabilir RO-HYSA'nın kaynak tüketimi bakımından karşılaştırılması.

	AWG devresi [14]	Yeni AWG devresi
Lojik Dilim Sayısı	10944	8825
FF Sayısı	12444	9856
LUT Sayısı	18004	14200
Block RAM Sayısı	80	80
MULT18x18 Sayısı	25	16

aynı RAM ağı kullanılması dolayısı ile değişmemiştir. Tasarımın FPGA tümdevresi üzerinde bu oranlarda daha az yer kaplaması sayesinde daha çok hücreye sahip olan bir ağ gerçeklenebilir. Gelinen son noktada, Xilinx XC2VP30 FPGA'sı üzerinde harici bir bellek ihtiyacı olmadan 128x128 programlanabilir RO-HYSA, daha az alan kullanılarak gerçeklenmektedir.

4. 128x128 HÜCRELİ PROGRAMLANABİLİR RO-HYSA'NIN BİLGİSAYAR YARDIMI İLE KULLANILMASI

Bu bölümde Altbölüm 3.3 'de tasarımı ve gerçeklenmesi anlatılan 128x128 hücreli programlanabilir RO-HYSA kullanılarak aktif dalga yayan ve tamamen kontrol edilebilen bir gerçek zamanlı çalışma anlatılmaktadır. Şekil 4.1'de, uygulamada kullanılan FPGA'yı barındıran devre kartı, HYSA'nın gerçek zamanlı görüntüsünü izlemek için kullanılan VGA monitörü, gerçeklenen devrenin FPGA üzerinde konfigüre edilmesi için tasarımda anlatıldığı gibi kullanılan Xilinx ISE aracı ve denetleyici yazılım olarak kullanılan MATLAB programlarını içeren denetleyici bilgisayar gösterilmektedir.



Şekil 4.1: Karşılaştırma amaçlı kurulan ve önceki çalışmada [14] da kullanılan test platformunun yapısı .

Denetleyici bilgisayarın işlevlerinden biri, programlanan tasarımın FPGA'ya aktarılmasıdır. Tasarım Xilinx ISE yazılımı ile gerçekleştirildikten sonra, Xilix Impact yazılımı ile 'Xilinx University Program Virtex-II Pro Development System' kartındaki XC2VP30 FPGA USB kablo aracılığıyla programlanır. Daha sonra denetleyici bilgisayardaki MATLAB yazılımı kullanılarak, tasarıma ait parametre okuma-yazma, değişken okuma-yazma, tüm ağ verisini yükleme ya da ağı kaydetme işlemlerine ait bilgilerin aktarımı, bilgisayar ile devre kartı arasındaki RS-232 hattı üzerinden gerçekleşmektedir. Bu bilgileri oluşturmak için Matlab fonksiyonları yazılmıştır. Bu fonksiyonlar sayesinde istenen başlangıç değerleri yüklenebilmekte, istendiği kadar iterasyon yaptırılmakta, iterasyonların tamamlanması ile sistemin o anki değerleri bilgisayara aktarılabilmektedir. Sitemin bu özelliğinden faydalanılarak, farklı parametre değerleri için sistemin dinamiğinin nasıl etkilendiğini gözlemleyebilmekteyiz [14].

Tasarımdaki HYSA'nın kullandığı parametreler, devre işleyişi ile ilgili değişkenler, haberleşme kanalı gibi global değişkenlerin sistemin çalışmaya başlamasından önce hazır olması gerekmektedir. Matlab fonksiyonlarından 'set_global' fonksiyonu, HYSA devresi ile kullanıcı bilgisayarı arasındaki haberleşme kanalını uygun bir COM objesi oluşturarak hazırlar ve tüm gerekli değişkenleri global olarak atar. Bu fonksiyonun içinde ayrıca sayı dönüşümlerinde kullanılacak kuantalayıcılar da hazırlanır. Dolayısıyla bu fonksiyon sistem çalıştırılırken ilk çalıştırılmalıdır. 'connect' fonksiyonu ise denetleyici bilgisayarın devreye bağlanmasını sağlar. İkinci olarak çalıştırılması gereken bu fonksiyon, sistemi bilgisayar ile haberleşme moduna alır [14].

'initialize_network' fonksiyonu ise x, y ve u değişkenleri için hazırlanan bitmap dosyalarını barındırır ve üçüncü olarak çalıştırılıp hem bu değişkenlerin değerlerini hem de parametre, iterasyon ve sistemin çalışmasında kullanılacak diğer değişkenlerin değerlerini FPGA'ya yüklememizi sağlar. HSYA'nın çalıştırılmasının ardından 'run' fonksiyonu çağrılır ve devre operasyon moduna geçirilir. Böylelikle öykünleme işlemi başlamış olur. Koşulacak iterasyon parametresindeki iterasyon sayısı sıfırlandığında devre RO-HYSA'yı durdurur [14].

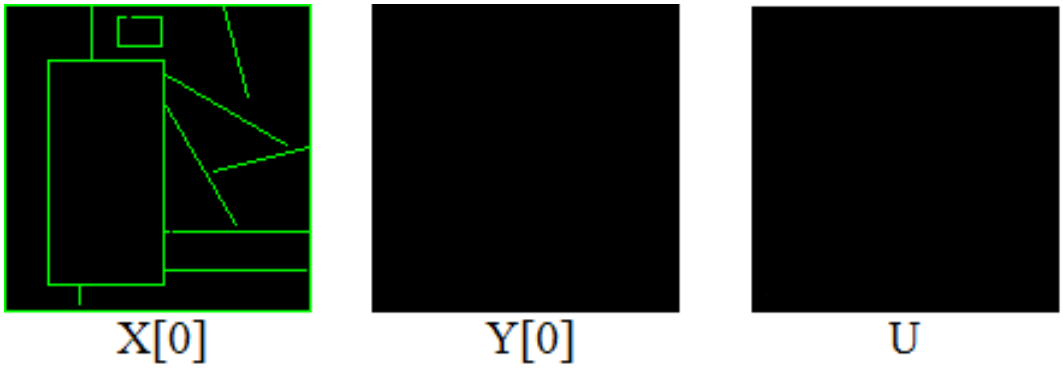
Anlatılan bu fonksiyonların çalıştırılması ile yapılan öykünlemeler sonucu 128x128 hücreli RO-HYSA'dan elde edilen aktif ve uzay-zaman dalga çıktıları Altbölüm 4.1 'de yer almaktadır.

4.1 Gerçekleme Sonucu Elde Edilen Aktif ve Uzay-Zaman Dalga Çıktıları

Gerçeklenen 128x128 hücreli programlanabilir RO-HYSA devresinin programlanabilme ve denetlenebilme yeteneği kullanılarak parametre değerlerindeki değişimlerle çeşitli dalga formları elde edilmektedir. Bu dalga formları çeşitli oto dalga [4], yürüyen dalga [7] ve dama desenli dalgalar [5] olup altbölümlerde yer almaktadır.

4.1.1 Oto Dalga

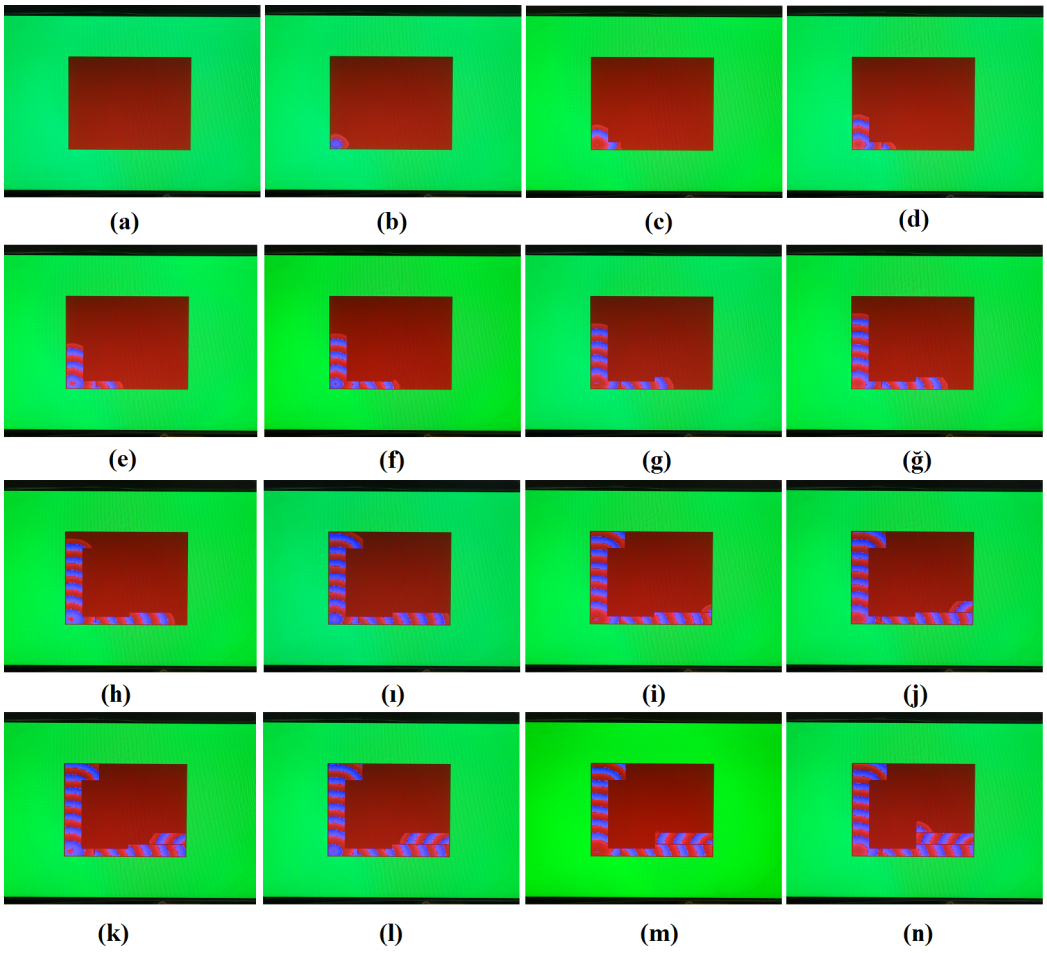
HYSA devresi FPGA üzerine kurulduktan sonra 128x128 hücreli ağa Şekil 4.2'deki X[0], Y[0] ve U başlangıç değerleri yüklenmiştir. Şekildeki siyah renk sayısal değer olarak 0'ı, yeşil renk pasif hücreleri ifade etmektedir. U matrisinin sol alt köşesinde değeri 10 olan bir hücre bulunmaktadır ve sayı formatı gereği resimde görünmemektedir. Bu başlangıç değerleriyle birlikte Matlab fonksiyonlarındaki parametreler Çizelge 4.1'de belirtilen değerlere [14] ayarlanıp ağa yüklenmiştir. HYSA bu parametre ve başlangıç koşullarını kullanarak öykünleme işlemine başlamıştır ve her 100 iterasyon sonucunda Şekil 4.3 ve Şekil 4.4'de verilen x değişkenine ait görüntüler yani X matrisi elde edilmiştir.



Şekil 4.2: Başlangıç koşulları.

Çizelge 4.1: Oto dalga parametre ve değerleri.

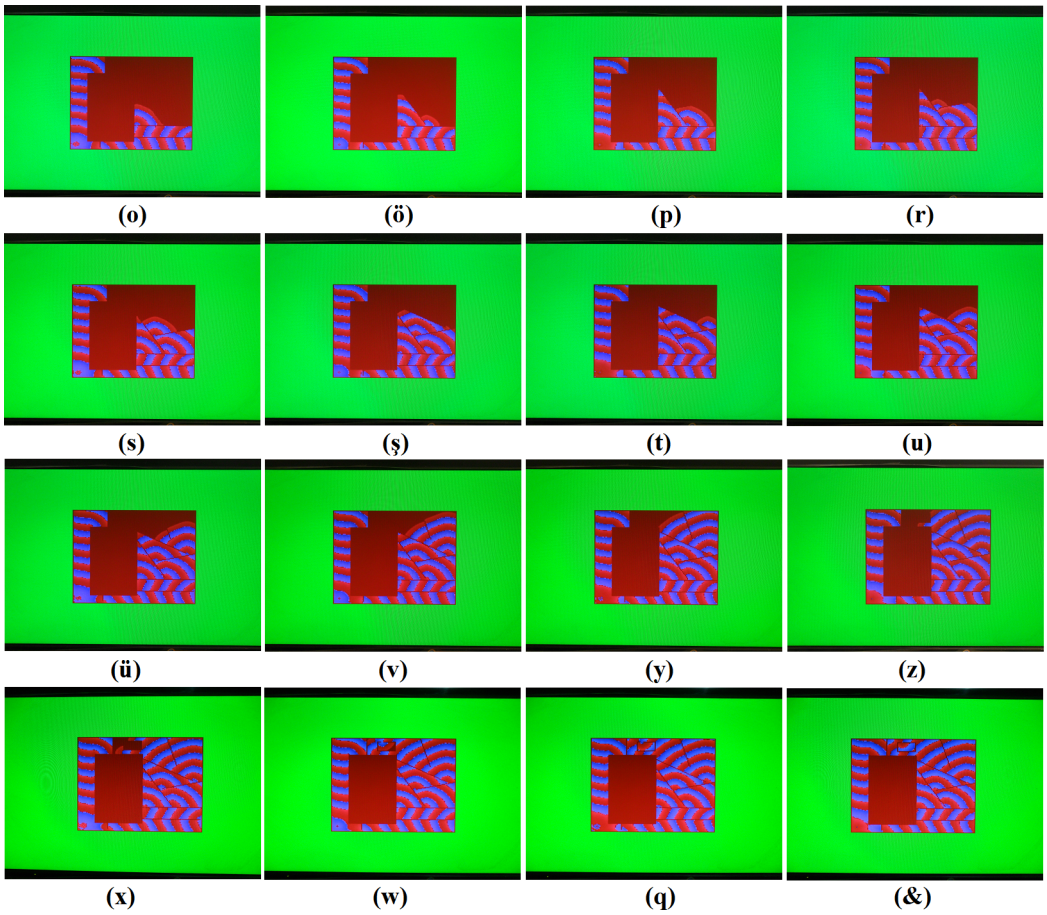
Parametre	Değeri
α	3
β	-4
ε	0.15
σ	-0.1
m	-8
λ	1.1
T	0.15
x_{sabit}	0
a	0.09
iterasyon	değişken



Şekil 4.3: Oto dalga yayan ağın her 100 iterasyonda monitörden çekilen resimleri (1).

4.1.2 Yürüyen Dalga

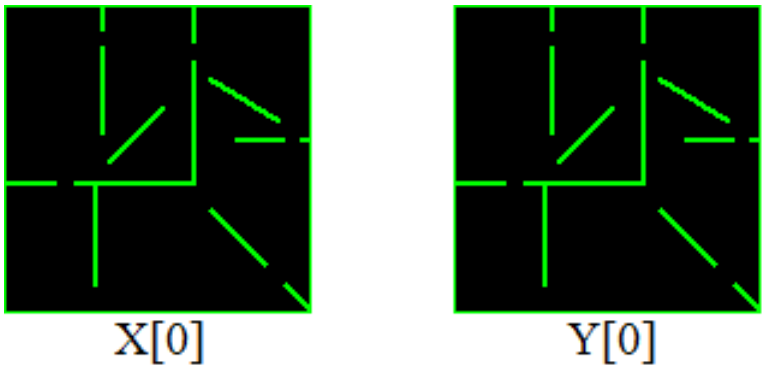
128x128 hücreli ağa Şekil 4.5'deki başlangıç koşulları ve parametrelere Çizelge 4.2'deki değerler yüklendiğinde x değişkenine ait görüntü yürüyen dalga oluşumunu göstermektedir. Bu dalga, yol bulma uygulamalarına kolaylık sağlamaktadır [7].



Şekil 4.4: Oto dalga yayan ağın her 100 iterasyonda monitörden çekilen resimleri (2).

Şekildeki siyah renk sayısal değer olarak 0'ı, yeşil renk pasif hücreleri ifade etmektedir. Bu şekilde de bir önceki ağ başlangıç koşulunda olduğu gibi görünmeyen, sıfır ya da pasif değerden farklı bir değere sahip olan bir hücre bulunmaktadır. Dalganın yayılımına bakacak olursak yaklaşık olarak dalgaya kaynaklık eden bu hücrenin orta kısımda olduğu gözlenmektedir ve bu hücrenin değerinin 1 olduğu bilinmektedir.

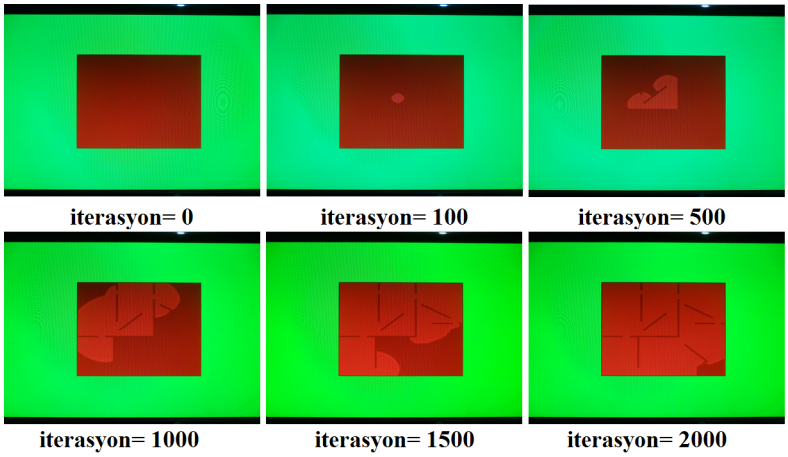
HYSA bu parametre ve başlangıç koşullarını kullanarak öykünleme işlemine başlamıştır. Şekil 4.6 ve Şekil 4.7'de görüldüğü üzere farklı a komşuluk etki değeri ve $iterasyon$ değerine göre x değişkenine ait görüntüler yani X matrisi elde edilmiştir. Dalga çıktılarından görüldüğü üzere komşuluk etki değeri arttıkça daha az sayıda iterasyon ile tüm ağ öykünlenebilmektedir.



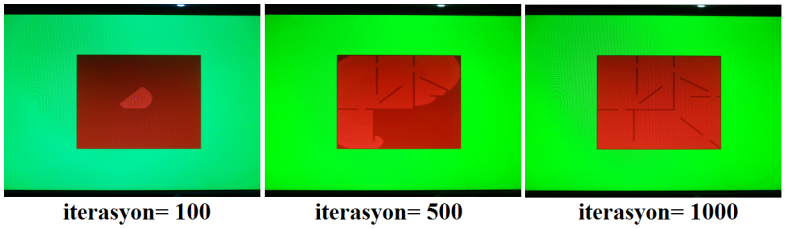
Şekil 4.5: Başlangıç koşulları.

Çizelge 4.2: Yürüyen dalga parametre ve değerleri.

Parametre	Değeri
α	3
β	-4
ε	0.1
σ	-0.1
m	-20
λ	1
T	0.08
x_{sabit}	0
a	değişken
iterasyon	değişken



Şekil 4.6: a= 0,09 ve farklı iterasyon değerlerinde yürüyen dalga çıktıları (1) .



Şekil 4.7: a= 0,5 ve farklı iterasyon değerlerinde yürüyen dalga çıktıları (2).

4.1.3 Gözlemlenen Çeşitli Dalgalar

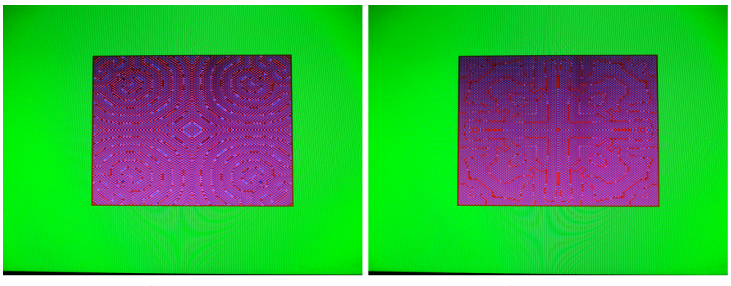
128x128 hücreli ağa Şekil 4.8'deki başlangıç koşulları ve parametrelere Çizelge 4.3 ve Çizelge 4.4'deki değerler [14] yüklendiğinde x değişkenine ait görüntüye bakarak Şekil 4.9'da *m* ve *iterasyon* değerlerine bağlı olarak, Şekil 4.10 ve Şekil 4.11'de ise *a* komşuluk etki değerine bağlı olarak çeşitli formlarda dalga oluştuğu gözlenmektedir.

U ve Y[O] matrislerinin tüm elemanları 0 değerindedir. X[0] matrisinde ise sınır hücreler yeşil renkte olup pasif olarak değerlendirilmekte ve bu çerçeve içinde kalan aktif hücreler 0 sayısal değeriyle yüklenmiştir.

Şekil 4.8: Başlangıç koşulları.

Çizelge 4.3: HYSA parametre ve değerleri.

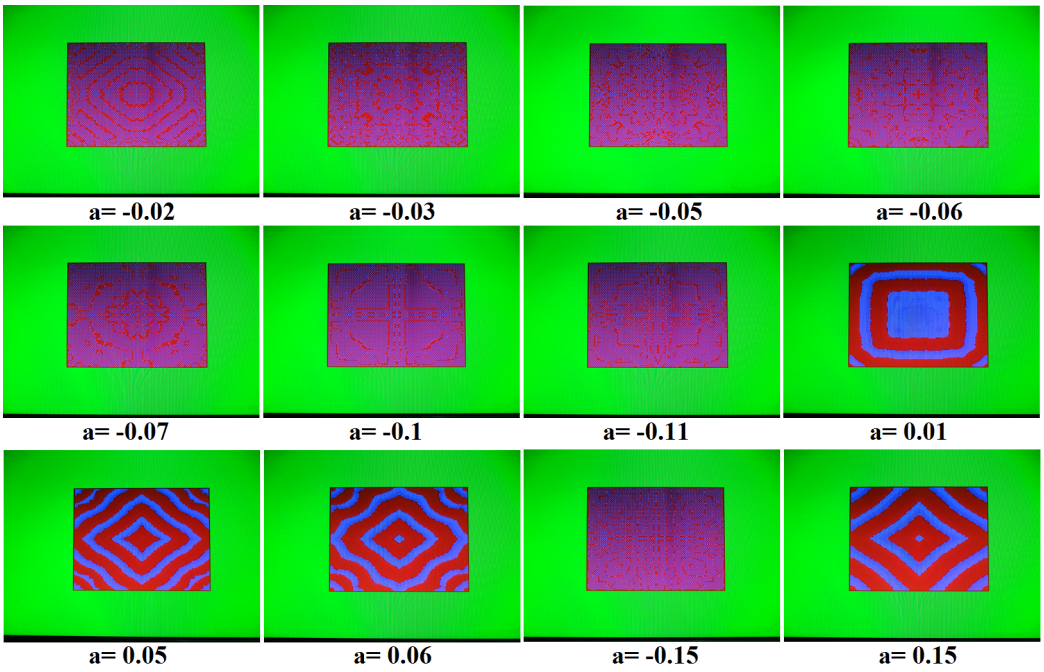
Parametre	Değeri
α	3
β	-4
ε	0.1
σ	-0.1
m	değişken
λ	1.05
T	0.08
x_{sabit}	-1
a	-0.1
iterasyon	değişken



Şekil 4.9: Elde edilen dalgalar ve desenler.

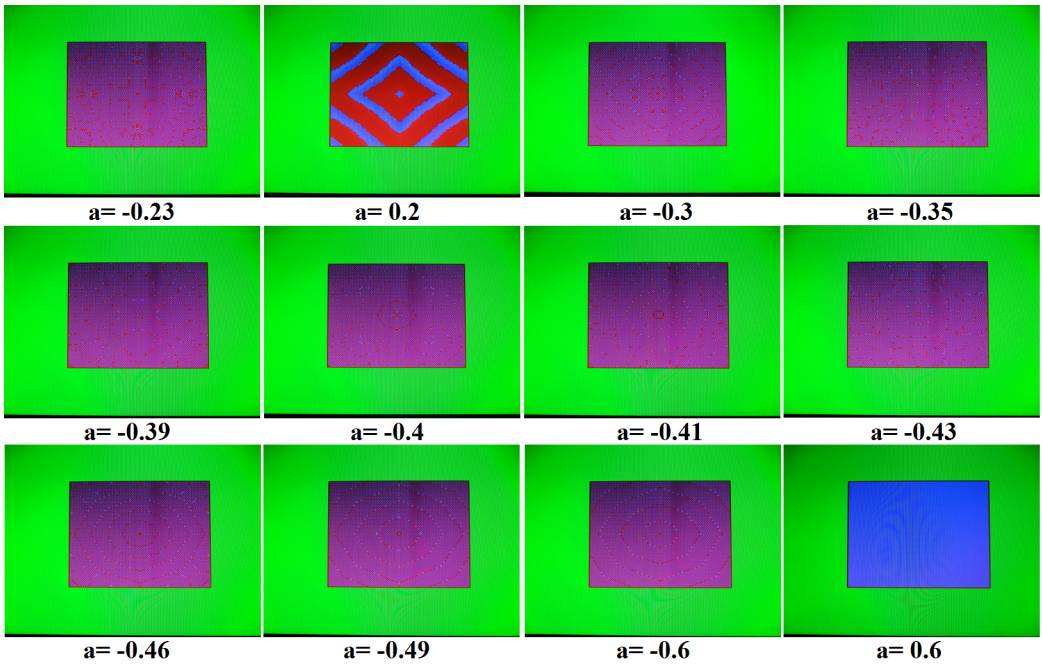
Çizelge 4.4: HYSA parametre ve değerleri.

Parametre	Değeri
α	3
β	-4
ε	0.1
σ	-0.1
m	-20
λ	1.05
T	0.08
x_{sabit}	-1
a	değişken
iterasyon	30000

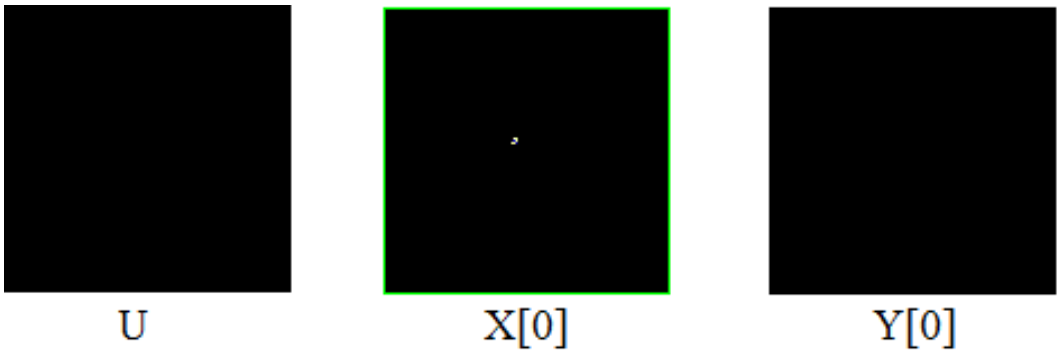


Şekil 4.10: Elde edilen dalgalar ve desenler.

Şekil 4.9, Şekil 4.10 ve Şekil 4.11'de HYSA'nın dört köşesindeki hücre dalga kaynağı olarak davranmaktadır. Böylece üretilen desenlerin çift simetri eksenine sahip olduğu görülmektedir. Ağa, Şekil 4.12'deki başlangıç koşulları ve Çizelge 4.5'deki parametre değerleri [14] yüklendiğinde farklı komşuluk etki değeri ve iterasyon sayısında çeşitli dalga desenleri gözlemlenmektedir. Bu başlangıç koşullarının Şekil 4.8'deki başlangıç koşullarından tek farkı X[0] matrisidir. Bu matrisin yaklaşık orta bölgesindeki dokuz hücreye çok küçük başlangıç değerleri atanmıştır. x_{sabit} değeri ise 0 olarak işlem görmektedir.Böylece ortadan başlayan dalgalar pasif hücreye temas edinceye kadar pasif hücrelerde kaynak etkisi gözlemlenmektedir.



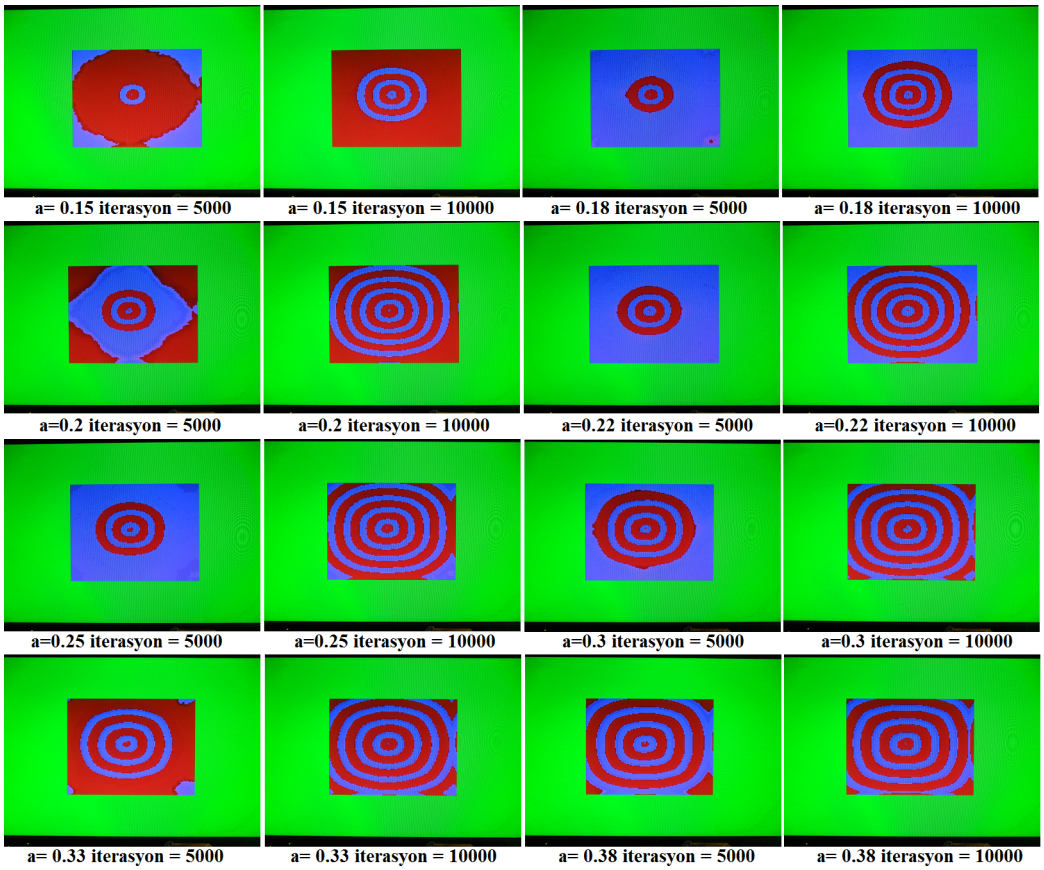
Şekil 4.11: Elde edilen dalgalar ve desenler.



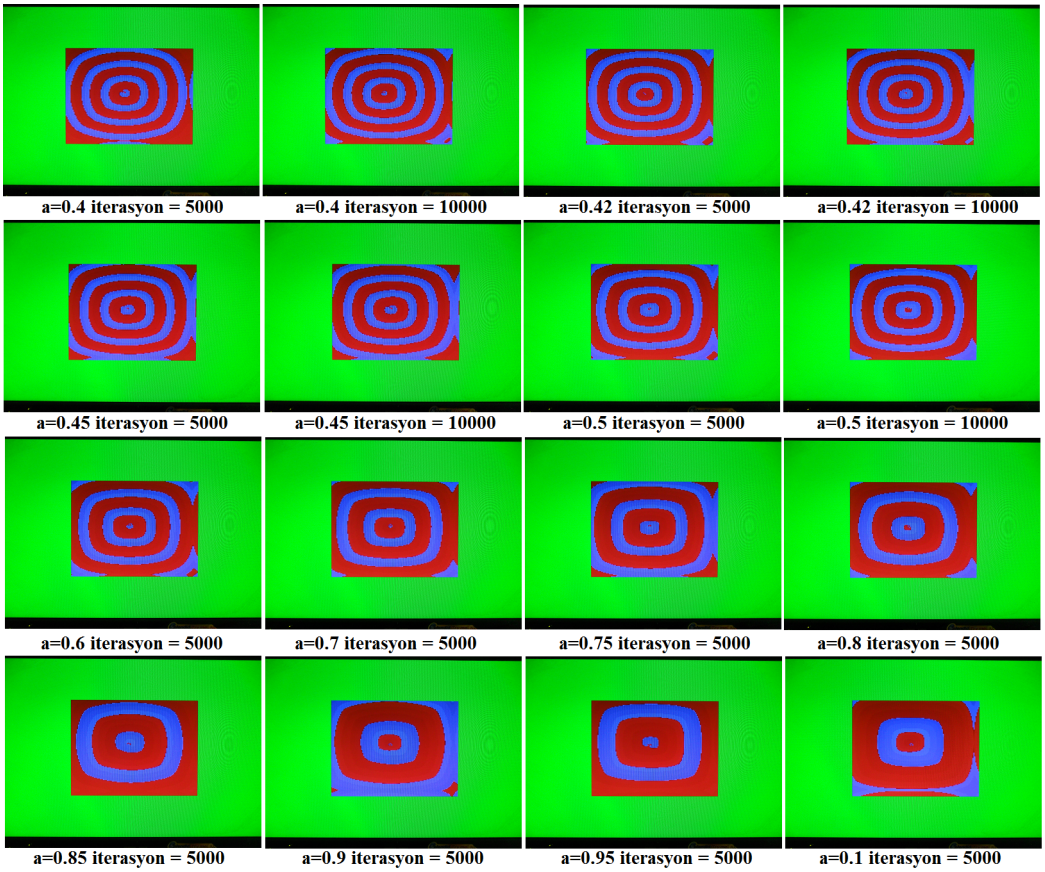
Şekil 4.12: Başlangıç koşulları.

Çizelge 4.5: HYSA parametre ve değerleri.

Parametre	Değeri
α	0.5
β	-4
ε	0.1
σ	-0.1
m	-20
λ	1.05
T	0.08
x_{sabit}	0
a	değişken
iterasyon	değişken



Şekil 4.13: Elde edilen dalgalar ve desenler.



Şekil 4.14: Elde edilen dalgalar ve desenler.

5. SONUÇ VE ÖNERİLER

Bu çalışmada 128x128 hücreli programlanabilir RO-HYSA tasarlanmış ve tasarım Xilinx Virtex II Pro Development System devre kartındaki XC2VP30 FPGA'sında gerçeklenmiştir. Ağın FPGA'da gerçeklenebilmesi için gerekli çevre birimler tasarlanıp kullanılmıştır. Ağ tasarımında kullanılan ve 128x128 hücreli ağa temel teşkil eden 4x4 hücreli ağın benzetimi MATLAB platformunda başarılı bir şekilde gerçekleştirilmiştir. Tasarımda kullanılan hücre modeli Relaksasyon Osilatörü temelli Hücresel Yapay Sinir Ağı (RO-HYSA) modelidir ve tek hücreye ait benzetim sonuçları ile Yalçın'ın elde ettiği benzetim örtüşmektedir.

MATLAB platformundaki benzetim çalışmalarında ve tasarım aşamalarında sabit noktalı sayı aritmetiği kullanılmıştır. Bu sayı aritmetiği, hücre modeline ilişkin durum değişken değerleri ve parametre değerleri düşünülerek tasarlanmış ve hücrenin matematiksel modelini gerçekleyecek devre içerisine gömülmüştür. Tasarım, denetleyici bilgisayar ile haberleştirilip istenilen parametre değerlerinde ve başlangıç koşullarında öykünleme işlemine başlatılmıştır. Denetleyici bilgisayarda çalıştırılan MATLAB kodları ile tasarıma ait değerler değiştirilebilmektedir. Ayrıca uzay-zaman dalgalarının oluşumu ve yayılımı devre kartına bağlanan VGA monitörde izlenmiştir.

Yapılan sayısal tasarımlar senkron ardışıl devrelerdir. Aritmetik devrelerin tasarımı asenkron ardışıl devre olarak denenebilir. Böylece öykünme hızında artış sağlanabilir.

Yarı duyarlı kayan nokta sayı aritmetiği yerine sabit noktalı sayı aritmetiği kullanılarak RO-HYSA devresinin kullandığı lojik dilim sayısı, flipflop sayısı, MULT18x18 çarpıcı eleman sayısı ve LUT sayısı Bölüm 4 'de çizelgelerde belirtildiği oranlarda azaltılmıştır. Bu durum değerlendirilerek daha büyük boyutlu RO-HYSA ağı öykünlenebilir.

Kayan nokta aritmetiğinin uzun toplama işlemi süresinden kurtularak NPE devresi daha hızlı işlem yapabilir hale gelmiştir. Bu devrede en uzun iterasyonun gerçekleşme

süresi, kayan nokta aritmetiği ile kurulan NPE devresine göre %75 oranında azaltılmıştır.

Çalışmanın sonucunda elde edilen RO-HYSA üzerinde yayılan aktif dalgalar kullanılarak en kısa yolu bulma problemi için çözüm, FPGA üzerinde kullanılan alan ve devrenin ağı öykünleme hızı bakımından iyileştirilmiştir. Önceki çalışma için kurulan test düzeneğinde kullanılan FPGA'ya bu çalışmada anlatılan devre tasarımı gömülerek çözümün tutarlılığı denenebilir.

Gerçeklenen tasarıma ait parametreler, tasarım FPGA 'ya yüklendikten sonra da değiştirilebilir. Var olan tasarım ile böyle bir duruma karşı ağın göstereceği davranış gözlenememektedir. Aynı şekilde ağ, uygulanan başlangıç koşulları, durum değişkenleri ve parametre değerlerine göre iterasyonlarını gerçekleştirirken herhangi bir anda denetleyici bilgisayar tarafından ağa yeni bir başlangıç koşulu uygulanabilir. Bu durumda ağın dinamik davranışını gözlemek için tasarım daha esnek hale getirilmelidir ve anlık değişimlere vereceği cevaptan yola çıkarak farklı dinamik davranışlar ileri çalışmalara zemin hazırlamaktadır.

KAYNAKLAR

[1] **Chua, L.O. ve Yang, L.** (1988). Cellular neural networks: theory, *Circuits and Systems, IEEE Transactions on*, ***35***(10), 1257–1272.

[2] **Chua, L.O. ve Yang, L.** (1988). Cellular neural networks: applications, *Circuits and Systems, IEEE Transactions on*, ***35***(10), 1273–1290.

[3] **Yalcin, M.E.** (2008). A Simple Programmable Autowave Generator Network for Wave Computing Applications, *Circuits and Systems II: Express Briefs, IEEE Transactions on*, ***55***(11), 1173–1177.

[4] **Yeniceri, R. ve Yalcin, M.E.** (2008). An implementation of 2D locally coupled relaxation oscillators on an FPGA for real-time autowave generation, *Cellular Neural Networks and Their Applications, 2008. CNNA 2008. 11th International Workshop on*, s.29–33.

[5] **Yeniceri, R. ve Yalcin, M.E.** (2008). A programmable hardware for exploring spatiotemporal waves in real-time, *Cellular Neural Networks and Their Applications, 2008. CNNA 2008. 11th International Workshop on*, s. 7.

[6] **Yeniceri, R. ve Yalcin, M.E.** (2009). An emulated digital wave computer core implementation, *Circuit Theory and Design, 2009. ECCTD 2009. European Conference on*, s.831–834.

[7] **Yeniceri, R. ve Yalcin, M.E.** (2009). Path planning on cellular nonlinear network using active wave computing technique, *Bioengineered and Bioinspired Systems IV*, ***7365***(1), 736508.

[8] **Arena, P.**, **Basile, A.**, **Fortuna, L. ve Frasca, M.** (2004). CNN wave based computation for robot navigation planning, *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, cilt 5, s.–500.

[9] **Ito, K.**, **Hiratsuka, M.**, **Aoki, T. ve Higuchi, T.** (2006). A Shortest Path Search Algorithm Using an Excitable Digital Reaction-Diffusion System, *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, ***E89-A***(3), 735–743.

[10] **Roska, T. ve Chua, L.O.** (1993). The CNN universal machine: an analogic array computer, *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, ***40***(3), 163–173.

[11] **Arena, P.**, **Fortuna, L.**, **Frasca, M.**, **Vagliasindi, G. ve Basile, A.** (2005). CNN wave based computation for robot navigation on ACE16K, *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, IEEE, s.5818–5821.

[12] **Nagy, Z. ve Szolgay, P.** (2003). Configurable multilayer CNN-UM emulator on FPGA, *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, ***50***(6), 774–778.

[13] **Kayaer, K. ve Tavsanoglu, V.** (2008). A new approach to emulate CNN on FPGAs for real time video processing, *Cellular Neural Networks and Their Applications, 2008. CNNA 2008. 11th International Workshop on*, s.23–28.

[14] **Yeniçeri, R.** (2009). Yol bulma uygulamaları için bir hücresel yapay sinir ağının sayısal tasarımı ve gerçeklenmesi, *İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi*.

[15] **Chua, L.O. ve Roska, T.** (1993). The CNN paradigm, *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, ***40***(3), 147–156.

[16] **Roska, T. ve Rodriguez-Vazquez, A.** (2002). Toward visual microprocessors, *Proceedings of the IEEE*, ***90***(7), 1244–1257.

[17] **Tükel, M.** (2009). VHDL ile Hücresel Yapay Sinir Ağı Gerçeklemesi, *İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi*.

[18] **Url-1**, `http://www.xilinx.com/fpga/index.htm`, alındığı tarih: 13.11.2014.

[19] **Xilinx**, `http://www.xilinx.com/products/boards-and-kits/XUPV2P-image.htm`, alındığı tarih: 13.11.2014.

[20] **Yıldız, N.** (2012). Design Of A Cellular Neural Network Emulator And Its Implementation On An FPGA Device, *Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Doktora Tezi*.

[21] **Mccreath, E.**, Fixed and Floating Point Numbers, `http://cs.anu.edu.au/courses/COMP2300/notes/07floatingnumbers.slides-4.pdf`, alındığı tarih: 13.11.2014.

[22] **Biccari, G.**, Fixed Point Representation (Qm.n format), `http://www.biccari.com/docs/Fixed_point_tutorial.pdf`, alındığı tarih: 13.11.2014.

ÖZGEÇMİŞ

Ad Soyad: Barış KARAKAYA

Doğum Yeri ve Tarihi: ELAZIĞ, 1990

E-Posta: karakayab@itu.edu.tr

Lisans: Fırat Üniversitesi Elektrik-Elektronik Mühendisliği, Haziran 2012
Lisans: Coventry University Electrical Engineering Programme
Coventry, United Kingdom, 2010-2011, ERASMUS

Mesleki Deneyim: Fırat Üniversitesi Elektrik-Elektronik Mühendisliği
Araştırma Görevlisi 2012 -...

Yayın Listesi:

- **Karakaya B.**, Yeniçeri R. and Yalçın M. E., 2015: Wave Computer Core Using Fixed-point Arithmetic(Gönderildi.), *ISCAS 2015 International Symposium on Circuits And Systems*, May 24-27, 2015 Lisbon, PORTUGAL.