İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

YOL BULMA UYGULAMALARI İÇİN BİR HÜCRESEL YAPAY SİNİR AĞININ SAYISAL TASARIMI VE GERÇEKLENMESİ

YÜKSEK LİSANS TEZİ

Ramazan YENİÇERİ

Anabilim Dalı : Elektronik ve Haberleşme Mühen.

Programı : Elektronik Mühendisliği

Tez Danışmanı: Doç. Dr. Müştak Erhan YALÇIN

HAZİRAN 2009

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

YOL BULMA UYGULAMALARI İÇİN BİR HÜCRESEL YAPAY SİNİR AĞININ SAYISAL TASARIMI VE GERÇEKLENMESİ

YÜKSEK LİSANS TEZİ
Ramazan YENİÇERİ
(504071217)

Tezin Enstitüye Verildiği Tarih : 4 Mayıs 2009
Tezin Savunulduğu Tarih : 28 Mayıs 2009

Tez Danışmanı : Doç. Dr. Müştak Erhan YALÇIN (İTÜ)
Diğer Jüri Üyeleri : Prof. Dr. Bilge GÜNSEL (İTÜ)
Prof. Dr. Vedat TAVŞANOĞLU (YTÜ)

HAZİRAN 2009

Nişanlıma ve aileme,

ÖNSÖZ

Sözün en başında, bu tez çalışması boyunca harcanan tüm emeğin anlamlı olduğunun bilincine varmamı sağlayan, bilgi ve birikimi ile her an desteğini hissettiğim, çok sevdiğim ve saydığım değerli hocam, tez danışmanım Doç. Dr. Müştak Erhan Yalçın'a teşekkür ederim.

Ailem ve canım nişanlımın manevi destekleri ile birlik aştığımız zorlukların, her gelen günün daha güzel olmasını sağladığının farkındayım. Onlara, yürekten sevgi ve teşekkürlerimi sunarım.

Çalışmalarım esnasında aydınlatıcı fikirlerini esirgemeyen dostum Yük. Müh. Emre Koyuncu'ya, emeği ile katkılarından dolayı Volkan Kılıç arkadaşıma; yüksek lisans eğitimim boyunca maddi olarak sıkıntı çekmememi ve çalışmama yoğunlaşabilmemi sağlayan bursu için TÜBİTAK-BİDEB 2228 programına; yine sağladığı teknik malzeme ve çalışma imkânı için 105E103 kodlu projemizin destekçisi TÜBİTAK'a teşekkürlerimi bir borç bilirim.

Önsözün sonunda, en başından beri hep yanımda olan, yol gösteren, akademik hayatı bana tanıtan ve sevdiren çok kıymetli büyüğüm Sayın Yrd. Doç. Dr. Muharrem Ök'e sonsuz sevgi, saygı ve teşekkürlerimi sunarım.

Nisan 2009

Elektronik Müh.
Ramazan Yeniçeri

İÇİNDEKİLER

Sayfa

KISALTMALAR

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

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

YOL BULMA UYGULAMALARI İÇİN BİR HÜCRESEL YAPAY SİNİR AĞININ SAYISAL TASARIMI VE GERÇEKLENMESİ

ÖZET

Bu tez çalışmasında en kısa yol bulma problemine çözüm üretebilen bir hücresel yapay sinir ağının tasarımı ve gerçeklenmesi yapılmıştır. İlk olarak hücresel yapay sinir ağları incelenmiştir. Relaksasyon osilatör temelli bir hücresel yapay sinir ağı (RO-HYSA) modeli tercih edilerek bu model üzerinden çalışma sürdürülmüştür. RO-HYSA'nın bilgisayar benzetimi yapılmış ve modelin çalışması incelenmiştir. Bu modelin devre olarak gerçeklenmesi için sayısal tasarımlar yapılmıştır. Çalışma süresince, her biri öncekinin gelişmişi olan üç sayısal tasarım tamamlanmış ve her biri gerçeklenmiştir. En son gerçekleme uygulama geliştirmesine uygundur. Bu gerçekleme ile RO-HYSA'dan beklenen dalga formları araştırılmış ve gözlenmiştir. Ardından yol bulma problemi için iki algoritma geliştirilmiş ve gerçeklenen RO-HYSA ile bu algoritmalar çalıştırılmıştır. En son olarak, hazırlanan bir test platformunda, bir mobil robotun hedefe en kısa yoldan gitmesi, gerçeklenen RO-HYSA devresi ve geliştirilen algoritmalar ile sağlanmıştır.

DIGITAL DESIGN OF A CELLULAR NEURAL NETWORK FOR PATH FINDING APPLICATIONS

SUMMARY

In this thesis, a cellular neural network, that solves the shortest path problem, is digitally designed and implemented. First, the cellular neural network paradigm is examined. Than a relaxation oscillator based cellular neural network model is chosen and the work is continued with this model. The computer simulation of the RO-CNN is done and analyzed. Digital circuits are designed to implement this model. During the circuit design process, three digital RO-CNN circuits, that each one is the improved version of the previous one, are designed and all circuits are implemented. The third version circuit is found to be suitable for further applications. Using this circuit, spatiotemporal waves and patterns are investigated and observed. Then, two algorithms to find the shorthest paths are developed and these algorithms are run on the implemented RO-CNN circuit. Finally, a test platform is set up and a mobile robot is guided to the target by the shortest way found by the algorithms run on the implemented RO-CNN circuit.

1. GİRİŞ

1988'den bu yana Hücresel Yapay Sinir Ağları (HYSA) ile onlarca mühendislik problemine çözüm aranmış ve bu süreçte ilk ortaya atıldığı günden bu yana HYSA'lar üzerinde birçok geliştirme yapılmıştır. Chua ve Yang'ın 1988'de ortaya attığı model [1] ile ilk yapılan uygulama görüntü işleme alanında olmuş [2], ilerleyen zamanda insan retinasının önerilen HYSA modeline benzediği saptanmış ve HYSA'lar biyolojik temel kazanmıştır.

Bu çalışmada Yalçın'ın ortaya koyduğu Relaksasyon Osilatörü temelli Hücresel Yapay Sinir Ağı modeli ile çalışılmıştır [3]. RO-HYSA ile aktif dalgalar üretilmiş ve aktif dalgalar yardımıyla bir robotu hedefe götürecek en kısa yolun saptanması için benzetimler, devre tasarımları, devre gerçeklemeleri, algoritma geliştirmeleri ve uygulama testleri yapılmıştır.

Aktif dalgalar ile yol bulma konusunda da literatürde yerini almış birçok yayın mevcuttur [4-11]. Bu çalışmalarda aktif dalgaların gözlendiği kimyasal ortamlar ile kurulan işlemciler, HYSA'nın analojik devre gerçeklemeleri [12] ve yeniden konfigüre edilebilir sayısal cihazlar, örneğin sahada programlanabilir kapı dizileri (FPGA) gibi donanımlar kullanılmıştır [13,14]. Bu tez çalışmasında RO-HYSA için tasarlanan sayısal devreler bir FPGA tümdevresi üzerinde gerçeklenerek gereken donanım oluşturulmuştur [15-17].

FPGA'nın kullanılması sayısal tasarımın gerçeklenmesini çok hızlandırmaktadır. FPGA'lar günümüzde uygulamaya özel tümdevre (ASIC) tasarımlarının ilk gerçeklenmesinde ve test edilmesinde çok yaygın olarak kullanılmaktadır. Sayısal tasarımların bilinen donanım tanımla dillerinden biri ile (Verilog HDL ve VHDL gibi) yapılması durumunda bu cihazlar için tasarımın sentezlenmesi, cihaz üzerine serilmesi ve yolların bağlanması otomatik olarak yapılabilmekte bu da tasarımcıya çok büyük kolaylıklar sağlamaktadır.

FPGA üzerinde gerçeklenen RO-HYSA'nın uygulamada kullanılabilir olması için, dışarıdan izlenebilir ve kontrol edilebilir olması gerekmektedir. Bu ihtiyacı

karşılamak için HYSA'nın izlenmesi ve denetleyici bir cihaz ile haberleşerek kontrol edilebilmesini saylayan bloklar tasarlanmıştır. Sonuçta, bu blokları da barındıran 128x128 hücreli bir RO-HYSA tasarımına ulaşılmış [17] ve hedeflenen uygulamaya yönelik olarak tez ilerletilmiştir.

Çalışmada, HYSA üzerinde çeşitli dalga formları ve desenleri oluştuğu gözlenmiş, bunların ağ parametreleri ve başlangıç koşulları ile ilişkisi araştırılmıştır. Oto dalga, yürüyen dalga ve spiral dalgalardan örnekler karşılaştırılmış ve yol bulma probleminde yürüyen dalgaların kullanılması tercih edilmiştir.

RO-HYSA, yürüyen dalgaları oluşturacak şekilde programlanmıştır. Bunun için çözümlenecek robot arenasının görüntüsü işlenmiş ve HYSA'ya başlangıç koşulları olarak yüklenmiştir. Görüntü işleme ile arenadaki engeller saptanmış ve ağda bunlara karşılık gelen hücreler pasif hücre haline getirilmiştir. Ayrıca robotun ve hedefin koordinatları da bulunmuş, hedefe karşılık gelen hücre dalga kaynağı olacak şekilde başlangıçta koşullandırılmıştır. Bu şekilde yayılan yürüyen dalgaların probleme çözüm olabilecek bir akış sergilediği görülmüştür. Yürüyen dalgaların yayılışına dayalı iki algoritma geliştirilmiştir. Bu algoritmaların ürettiği çözüm kurulan bir test platformunda test edilmek üzere çalışmalar sürdürülmüştür [18].

2. HÜCRESEL YAPAY SİNİR AĞLARI

Hücresel yapay sinir ağı, Leon O. Chua ve Lin Yang'ın 1988 yılında yeni bir yapay sinir ağı mimarisi ortaya koydukları makalelerinde [1,2] ortaya çıkmıştır. Sinir sistemimizdeki nöronların diğer nöronlarla yerel olarak bağlı olması HYSA yapısının dayandığı en önemli özelliktir. Hopfield sinir ağlarındaki bütün hücrelerin birbirleriyle doğrudan bağlılığına karşın HYSA'da her bir hücre en yakın komşuluğundaki hücreler ile doğrudan, diğer hücrelerle dolaylı olarak bağlıdır.

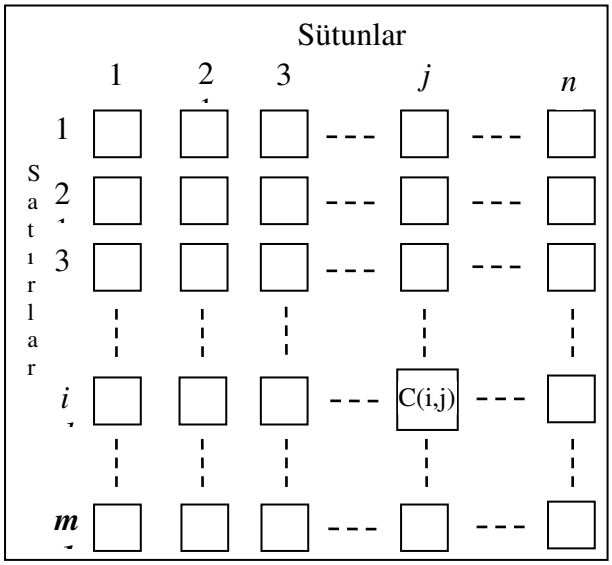
Yerel bağlantı sonucu; hem devrenin kapladığı alan azalmakta hem de verilerin devre üzerindeki taşınım yolları kısa olduğundan toplam güç harcaması düşük olmaktadır. HYSA'nın önemli avantajlarından bir diğeri de kararlı bir sonuca yakınsama süresinin devrenin boyutundan bağımsız olması ve bir kaç nöron gecikmesi ile belirlenmesidir. 1993'de Chua ve Roska tarafından yapılan çalışma, HYSA'nın ilk çıkışından itibaren yapılan çalışmaları özetleyerek HYSA paradigmasını ortaya koymaktadır [19].

Yine 1993'de, Roska ve Chua tarafından HYSA'yı merkezi işlem birimi alan yeni bir bilgisayar mimarisi ortaya atılmıştır [12]. Bu mimari aslında analog ve sayısal işlem yapma yeteneğine sahip bir merkezi işlem birimine sahiptir. Bu nedenle hücresel yapay sinir ağı evrensel makinesi, HYSA-EM (CNN-UM) olarak adlandırılan işlemci, analog ve lojik işlem yeteneği nedeniyle bu iki terimin birleşimi olan analogic işlemci olarak nitelendirilmektedir. Mimarisi analog ve sayısal olmak üzere yerel ve genel (global) belleğe ayrıca HYSA'yı programlamaya yarayan şablonların tutulduğu bellek alanlarına sahiptir.

HYSA'nın bu özellikleri gelecekte pek çok uygulama alanı bulacağının göstergeleridir. HYSA günümüzde retina modellemesi, biyonik göz gibi birçok uygulamaya sahiptir. Roska ve Rodriguez-Vazquez'in 2002'de sundukları çalışma ile HYSA'nın görüntü işlemedeki becerisinin görsel işlemcilere doğru nasıl yolalınmasına sebep olduğunu göstermektedir [20-21].

2.1 Hücresel Sinir Ağlarının Mimarisi

Bu kısımda hücresel sinir ağı mimarisi, ayrıntılı olarak incelenmiştir [22]. Standart HYSA yapısı m x n boyutundaki dikdörtgen hücreler dizisinden oluşur. Burada m, satır sayısını n ise sütun sayısını gösterir. HYSA'nın en küçük birimine hücre adı verilir. Hücreler devrenin boyutuna göre iki boyutlu uzayda kartezyen koordinat sistemi düzeninde yerleştirilmişlerdir. C(i,j), i. satır, j. sütundaki hücre olarak ifade edilir (i=1,2,3...m, j=1,2,3...n). Şekil 2.1'de bu mimari gösterilmektedir.

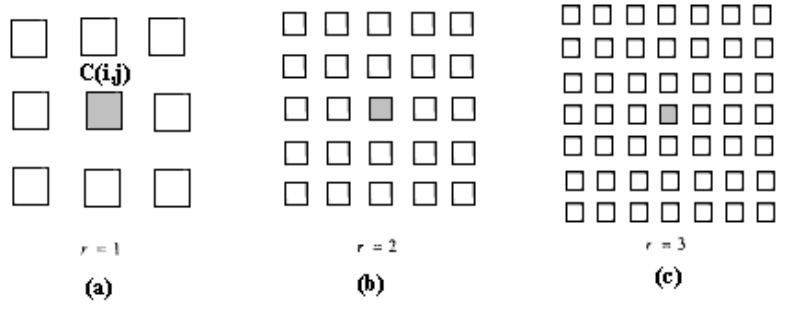


Şekil 2.1 : Standart HYSA mimarisi.

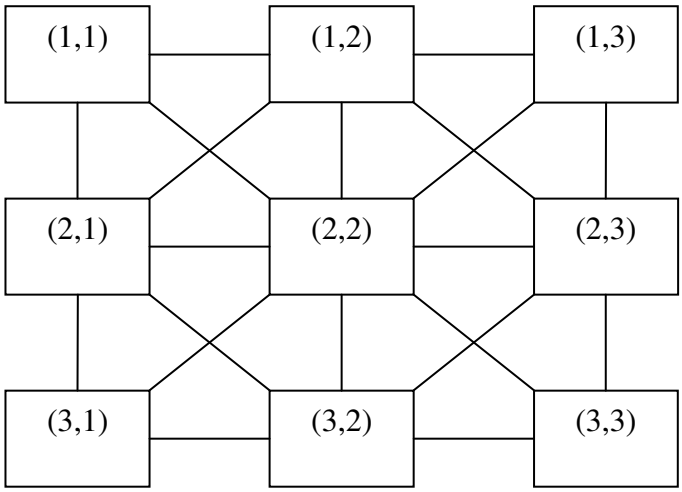
Uygulamalarda genellikle m=n alınsa da boyutların eşit olmamas da mümkündür. Örneğin 5 x 512'lik bir HYSA yapısı tarayıcı, faks ya da fotokopi makinesi için uygundur. C(i,j) hücresinin r yarıçaplı etki küresi $S_r(i, j)$ olmak üzere;

$$S_r(i,j) = \left\{ C(k,l) \middle| \max_{1\le k\le m, 1\le l\le n} \{|k-l|, |l-j|\} \le r \right\} \qquad \textbf{(2. 1)}$$

koşulunu sağlayan tüm komşu hücreler topluluğuna denir. Buradaki r pozitif bir tam sayıdır. Şekil 2.2'de komşuluklar gösterilmektedir.



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



Şekil 2.3 : Bir HYSA yapısındaki hücreler arası etkileşim.

Komşuluk r yarıçapı ile belirtilebildiği gibi Şekil 2.3'deki gibi (2r+1)x(2r+1) şeklinde de gösterilebilir. Etki küresi içerisinde $(2r+1)^2$ kadar hücre bulunmaktadır. Komşuluk sistemi simetri özelliğine sahiptir.

$$C(k,l) \in S_r(i,j) \Leftrightarrow C(i,j) \in S_r(k,l) \qquad \textbf{(2. 2)}$$

Hücresel Sinir Ağı tanımlanırken, bir r değeri seçilir (çoğu uygulamada r =1 alınır). Bu seçimden sonra her hücre sadece kendi komşuluk grubundaki hücrelerle doğrudan bağlantılı bulunur. Yani sadece komşuları ile sinaptik bağlantıları mevcuttur.

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

$$\frac{dx_{ij}}{dt} = -x_{ij} + \sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)\,y_{kl} + \sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)\,u_{kl} + z_{ij} \qquad \textbf{(2. 3)}$$

Burada;

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

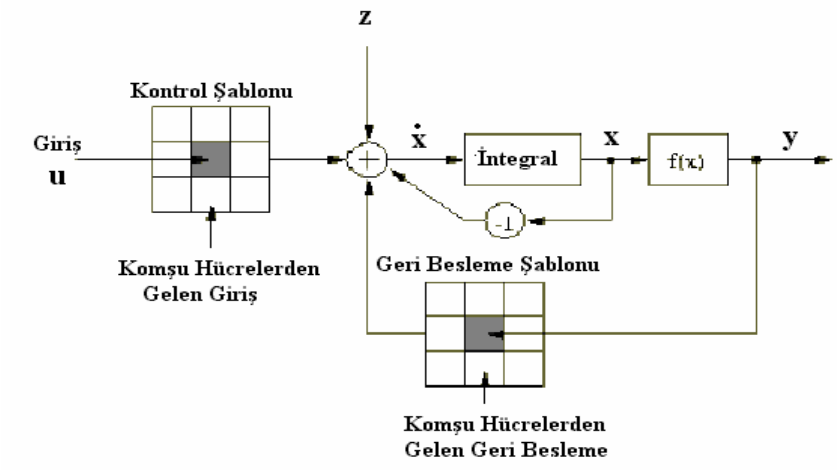
$y_{kl} \in R$, etki küresi içerisindeki hücrelerin çıkışları;

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

$z_{ij} \in R$, eşik değeri;

A(i, j ; k, l), geri Besleme operatörü;

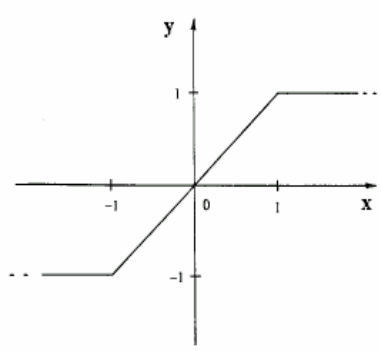
B(i, j ; k, l), giriş operatörüdür ve bu modele ilişkin blok diyagram Şekil 2.4'de verilmiştir. HYSA ya ilişkin doğrusal olmayan fonksiyon f(x,y) Şekil 2.5'de gösterilmiştir.



Şekil 2.4 : Bir hücrenin blok diyagram gösterilimi.

Bu modeldeki çıkış denklemi aşağıdaki gibidir.

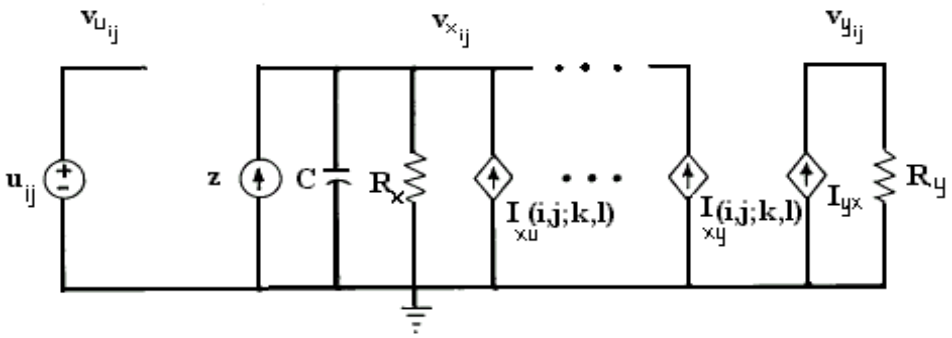
$$y_{ij} = f(x_{ij}) = \frac{1}{2}\left|x_{ij}+1\right| - \frac{1}{2}\left|x_{ij}-1\right| \qquad \textbf{(2. 4)}$$



Şekil 2.5 : Eşik Aktivasyon Fonksiyonu.

HYSA'ya lineer olmama özelliğini veren bu fonksiyondur. Sınır hücrelerinin etki küresi içinde bulunup da m x n boyutundaki hücre dizisinin dışında kalan hücrelerin durum ve çıkış değerleri koşuludur. Dizideki tüm hücrelerin ilk koşulu, xij(0), i=1,2...m , j=1,2...n olarak verilir.

HYSA'nın bir hücresinin devre modeli Şekil 2.6'da görülmektedir.



Şekil 2.6 : Bir Hücrenin devre şeması [1].

Hücrede sırasıyla giriş, durum ve çıkışa karşı 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, son olarak da 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.

$$I_{xy}(i,j;k,l) = A(i,j;k,l)v_{ykl} \qquad ,\forall C(k,l) \in S_r(i,j) \tag{2. 5}$$

$$I_{xu}(i,j;k,l) = A(i,j;k,l)v_{u_{kl}} \qquad ,\forall C(k,l) \in S_r(i,j) \tag{2. 6}$$

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

$$I_{yx} = \frac{1}{R_y} f(v_{x_{ij}}) \tag{2. 7}$$

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.

3. RELAKSASYON OSİLATÖRÜ TEMELLİ HÜCRESEL YAPAY SİNİR AĞLARI

Bölüm 2. ’de yapılan çalışmanın bilimsel temelleri açıklanmış, literatür özetlenmiş, ayrıca çalışmanın motivasyonundan bahsedilmiştir. Bölüm 3.1’de hedeflenen uygulamada kullanılacak hücresel yapay sinir ağının modellenmesi gösterilmektedir. Modellemenin ardından önce ağa ait bir hücrenin benzetimi Altbölüm 3.2.1 ’de; sonra 5x5 hücrelik küçük bir HYSA’nın benzetimi Altbölüm 3.2.2 ’de gerçekleştirilmiş ve sonuçları sunulmuştur. Benzetimler ağın sayısal devre olarak gerçeklenmesinde önemli bir kaynak olmuştur. Tasarım sürecinde elde edilen benzetim sonuçları referans alınmıştır.

3.1 Hücre Modeli

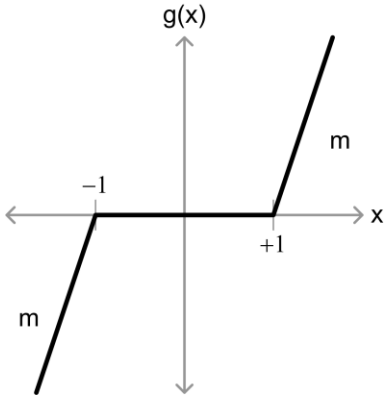
Sinir hücresi modeli olarak Bölüm 2. ’de de belirtildiği gibi Yalçın’ın 2008’de önerdiği kare dalga osilatörü kullanılmıştır [3]. Çalışmada kullanılan bu kare dalge osilatörlerine literatürde relaksasyon osilatörü denilmektedir. Yalçın, bir HYSA gerçeklemesi olan ACE16k tümdevresi ile yaptığı çalışmalar sonucunda [23] bahsedilen osilasyon davranışını sergileyecek hücresel modelin (3. 8)’deki gibi olduğunu bulmuştur.l

$$\begin{aligned} \dot{x} &= \alpha x + \beta y - g(x) \\ \dot{y} &= \varepsilon(x - y) \end{aligned} \qquad \textbf{(3. 8)}$$

Denklemdeki x ve y durum değişkenleri iken α, β ve ε reel değerli parametrelerdir. Ayrıca modelde doğrusal olmayan davranışın kaynağı olan g(x) fonksiyonu da saptanmış ve (3. 9)’daki ifade verilmiştir.

$$g(x) = \begin{cases} m(x+1) & ; \quad x < -1 \\ 0 & ; \quad |x| \le 1 \\ m(x-1) & ; \quad x > 1 \end{cases} \qquad \textbf{(3. 9)}$$

Şekil 3.1’de de g fonksiyonunun x’e göre grafiksel ifadesi gösterilmiştir.



Şekil 3.1 : Doğrusallığı bozan g(x) fonksiyonu.

Bu tek başına salınan hücrenin x durum değişkeni, zaman içerisinde kare dalga işaretinia oluşturmaktadır. Bu osilatörlerden oluşturulacak HSYA'nın hücresel modeli için yine [3]'den yararlanılarak model denklemlerine ulaşılmıştır. Yalçın'ın önerdiği modelde küçük değişikliklere gidilmiştir. Yapılan bu değişikliklerden biri g fonksiyonunun eğiminin negatif yapılarak g'nin x durumuna ait diferansiyel denklemde toplanan terim haline getirilmesidir. Diğeri de y durumuna ait diferansiyel denklemde y'nin ağırlığının x'nin ağırlığından bağımsız hale getirilmesi, yani –ε yerine σ ağırlık parametresinin kullanılması olmuştur.

Gerek sayısal devre gerçeklemelerinin gerekse bilgisayar benzetimlerinin yapılabilmesi için [3]'de verilen model denklemleri yukarıdaki değişikliklerde yapıldıktan sonra zamanda ayrık hale getirilmiştir. Böylelikle bu çalışmada kullanılacak hücreye ait model (3. 10)'daki gibi olmuştur.

$$\begin{aligned} x_{i,j}[k+1] &= x_{i,j}[k] + T\left(\alpha x_{i,j}[k] + \beta y_{i,j}[k] + g\left(x_{i,j}[k]\right) + I_{i,j}[k] + u_{i,j}\right) \\ y_{i,j}[k+1] &= y_{i,j}[k] + T\left(\varepsilon x_{i,j}[k] + \sigma y_{i,j}[k]\right) \end{aligned} \qquad \textbf{(3. 10)}$$

Bu ifade diferansiyel denklemin ileri Euler integrasyonu metodu ile ayrıklaştırılmasından elde edilmiştir. Buradaki T, integrasyon adımıdır. Doğrusallığı bozan g(x) fonksiyonu da (3. 11)'deki verilmiştir.

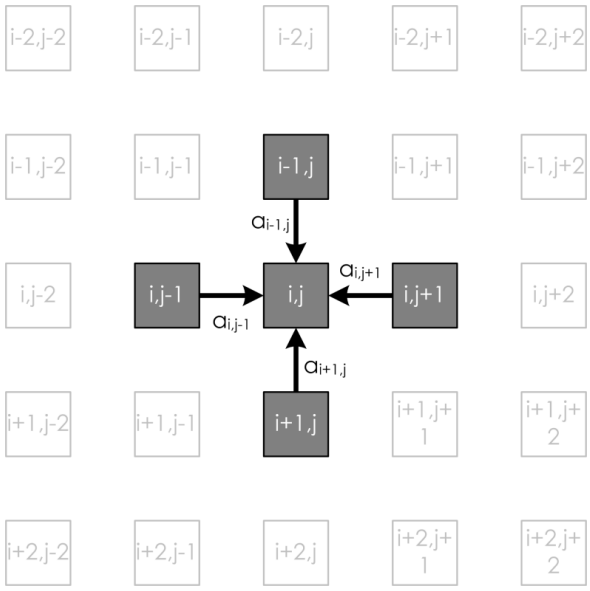
$$g\left(x_{i,j}[k]\right) = \begin{cases} m \cdot (x_{i,j}[k] + \lambda) & ; \quad x < -\lambda \\ 0 & ; \quad |x| \le \lambda \\ m \cdot (x_{i,j}[k] - \lambda) & ; \quad x > \lambda \end{cases} \qquad \textbf{(3. 11)}$$

Şekil 3.1'de görüldüğü gibi, g(x) fonksiyonu doğrusal olduğu aralıkları belirleyen değer (3. 11)'de parametrik hale getirilmiş ve λ ile ifade edilmiştir.

I(.) fonksiyonu ağın oluşmasını sağlamaktadır. I, her bir hücreye komşularının x durum değişkenlerinin ağırlıklandırılmış toplamını taşır. Komşuluk terimi ismiyle anılacak olan I, (3. 12) eşitliği ile tanımlanmaktadır.

$$I_{i,j}[k] = a_{i,j+1}x_{i,j+1}[k] + a_{i-1,j}x_{i-1,j}[k] + a_{i,j-1}x_{i,j-1}[k] + a_{i+1,j}x_{i+1,j}[k] \quad \textbf{(3. 12)}$$

Hücresel modelin ayrık zamanlı denklemleri incelendiğinde her bir hücreye ait durum değişkenlerinin i ve j indisleri ile belirtilmiş olduğu görülmektedir. İki boyutlu (MxN) hücresel yapay sinir ağları M satır ve N sütundan oluşmakta, hücrelerden herbirinin yeri bir sütun ve bir satır değeri ile belirtilmektedir [1]. Tez boyunca i indisi hücrenin bulunduğu satırı, j indisi de hücrenin bulunduğu sütunu belirtecektir. (3. 10)'da bu indisler kullanılmış, (3. 12)'de komşu hücreler yine bu indislerle ifade edilmiştir. İncelendiğinde, komşuluk terimi I, referans alınan hücrenin HYSA'daki doğu, kuzey, batı ve güney komşularının referans hücreye bağlamaktadır. Şekil 3.2'de bu komşuluk bağlantısı gösterilmiştir.

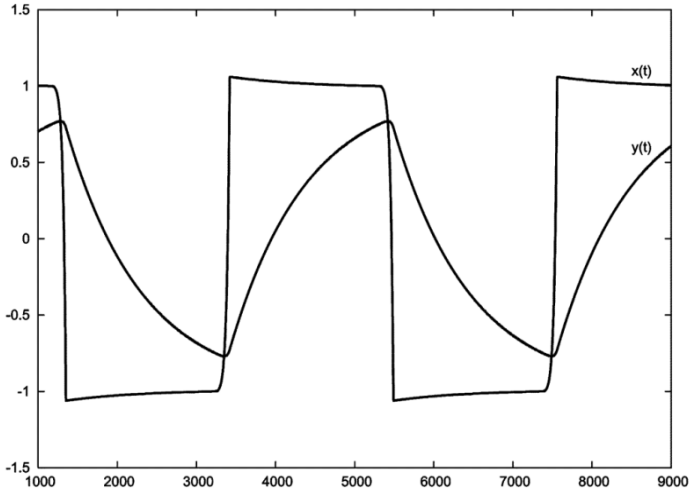


Şekil 3.2 : Ağ üzerindeki indislerin ve komşuluk terimi I'nın gösterimi.

Modelde, sürekli giriş u ile ifade bulur. Kurulan HYSA'ndan beklenen davranış çeşitli aktif dalgaların üretilmesidir. Yapılan çalışmalarda bu dalgaların kaynağının u giriş işareti ile belirlenebildiği gösterilmiştir. İlerleyen bölümlerde dalga kaynağı olarak başka koşulların oluşturabildiği de gösterilecektir. Yine de genel HYSA

modelinde bulunan giriş işareti terimi, bu çalışmada kullanılan son modelde de korunmuştur.

Yukarıdaki ifadeler ile kurulan hücresel modelin, HYSA göz önünde bulundurulduğunda bir eksiğinin bulunduğu farkedilir. Mevcut ifade HYSA'daki tüm hücrelerin dört komşuluğunda gerçek hücrelerin varolduğunu ve bu hücrelerden değerler aldığını kabul etmektedir. Fakat gerçekte HYSA'nın kenar ve köşelerinde yerleşik hücrelerin bir ya da iki (köşedekiler için) komşusu bulunmamaktadır. Bu hücreler için belirlenmesi gereken sınır koşulları ileride anlatılacak benzetim ve gerçeklemelerde ayrıca açıklanmıştır. Özellikle devre gerçeklemelerinde sınır koşulları için farklı çözümler uygulanmıştır. Örneğin Altbölüm 3.2.1 'de tek hücrenin benzetimi yapılırken, bu hücrenin tüm komşuları için sınır koşulları kullanılmıştır. Temel alınan modelinde Yalçın tek hücreye ait x ve y durum değişkenlerinin zaman içinde Şekil 3.3'deki gibi değiştiğini göstermektedir.



Şekil 3.3 : Hücrenin x ve y durumlarının zamanda değişimi.

3.2 Hücre Modelinin Benzetimleri

HYSA'nın üreteceği dalgaların dinamiği kullanılan modelle ve model parametreleriyle doğrudan ilişkilidir. Matematiksel temelin oluşturulmasından sonra çalışmaya benzetimler ile devam etmenin bir sebebi kaynaklarda verilen parametrelerin test edilmesi ve uygun parametre aralıklarının araştırılabilmesidir. Elde edilen benzetim sonuçları sayısal gerçeklemeler esnasında yapılan devre benzetimleri için referans kabul edilmiştir.

Hücreye ait matematiksel modelin benzetimi MATLAB ortamında yapılmıştır. Tez boyunca R2007a sürümü kullanılmıştır. Benzetimlerde çift duyarlı kayan nokta aritmetiği ile işlemler gerçekleştirilmiştir. Tezde tek hücreye ait benzetim sonuçları ve 5x5 hücrelik ağa ait benzetim sonuçları sırasıyla Altbölüm 3.2.1 ve Altbölüm 3.2.2 'de verilmiştir.

3.2.1 Tek hücrenin benzetimi

Yapılan tek hücre benzetimi ile x ve y'nin zamanla değişiminin, Şekil 3.3'de gösterilen davranışı sergilenmesi hedeflenmiştir. Tek hücrenin benzetiminde dört komşudan gelen bağlantı değerleri 0 yapılmıştır. Dolayısı ile, her ne kadar 0.1 olarak verilse de, komşuluk ağırlıklarının hücre dinamiğinde etkisi yoktur. Hücreye ait durum değişkenlerinin başlangıç değeri 0.125 verilmiştir. Şekil 3.4'de verilen benzetim kodunda görüldüğü gibi α, β, ε, σ ağırlıkları ile λ, m ve T parametleri belirlenmiş ve 15000 iterasyon adımı koşulmuştur.

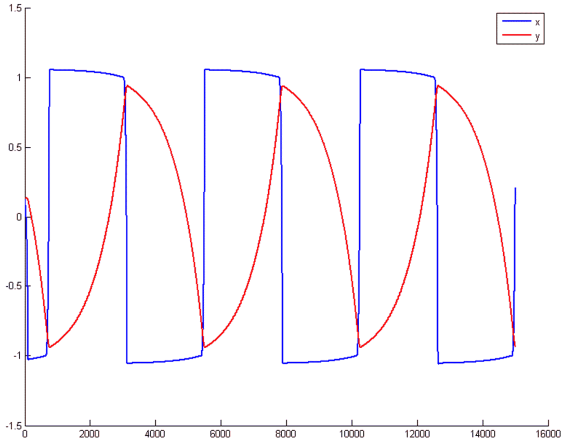
```
x(1)        = 0.125;
y(1)        = 0.125;
alfa        = 3;
beta        = -4;
epsilon     = 0.125;
sigma       = 0.125;
a_d         = 0.1;
a_k         = 0.1;
a_b         = 0.1;
a_g         = 0.1;
x_d         = 0;
x_k         = 0;
x_b         = 0;
x_g         = 0;
lambda      = 1;
m           = -128;
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 3.4 : Tek hücrenin benzetim kodu.

Verilen kod ile yapılan benzetim sonucunda x ve y durumlarının zamanla değişimi Şekil 3.5'de verilen grafiğe çizdirilmiştir.

Şekil 3.3 ve Şekil 3.5'deki grafikler beklendiği gibi davranışların örtüştüğünü göstermektedir. Üretilen kare dalga işaretinin frekansı model parametreleri ile

değişmektedir. Benzetim, sadece x ve y durum değişkenlerinin başlangıç değerleri değiştirilerek koşturulduğunda işaretin fazının kaydığı görülmüştür. Model, birbirine bağlanmamış komşu hücrelerin farklı başlangıç koşulları ile koşturulduğunda , farklı fazlarda işaretler üreteceklerini ve bu işaretlerin i,j indisli uzayda da bir uzay dalgasını oluşturabileceğini göstermektedir. Hücre bazında zamanla değişen bir dalga, düzgün yerleştirilmiş hücreler uzayında, uzayda yol alan dalgaları oluşturmaktadır.



Şekil 3.5 : Tek hücrenin benzetim sonucunda elde edilen x ve y değişkenleri.

3.2.2 5 x 5 hücreli ağın benzetimi

Yukarıda verilen tek hücrenin benzetim kodu 5x5 hücre grubu için tekrar ele alınmış ve bir ağ oluştrulmuştur. Hücreler birbirine uygun komşuluklarda bağlanmıştır. Ağ oluşturulması ile sınır koşullarının belirlenmesine gerek duyulmuştur. Ağın kenarlarında bulunan hücrelerin komşularına bağlanması gereken birer girişleri, köşelerindeki hücrelerin de ikişer girişleri gerçek komşulara bağlanmamıştır. Sınır koşulu, bu hücrelerin eksik komşularına ait girişlerine hangi sabit değerin sürekli uygulanacağını belirtir. Benzetimde sınır koşulu 0 seçilmiştir.

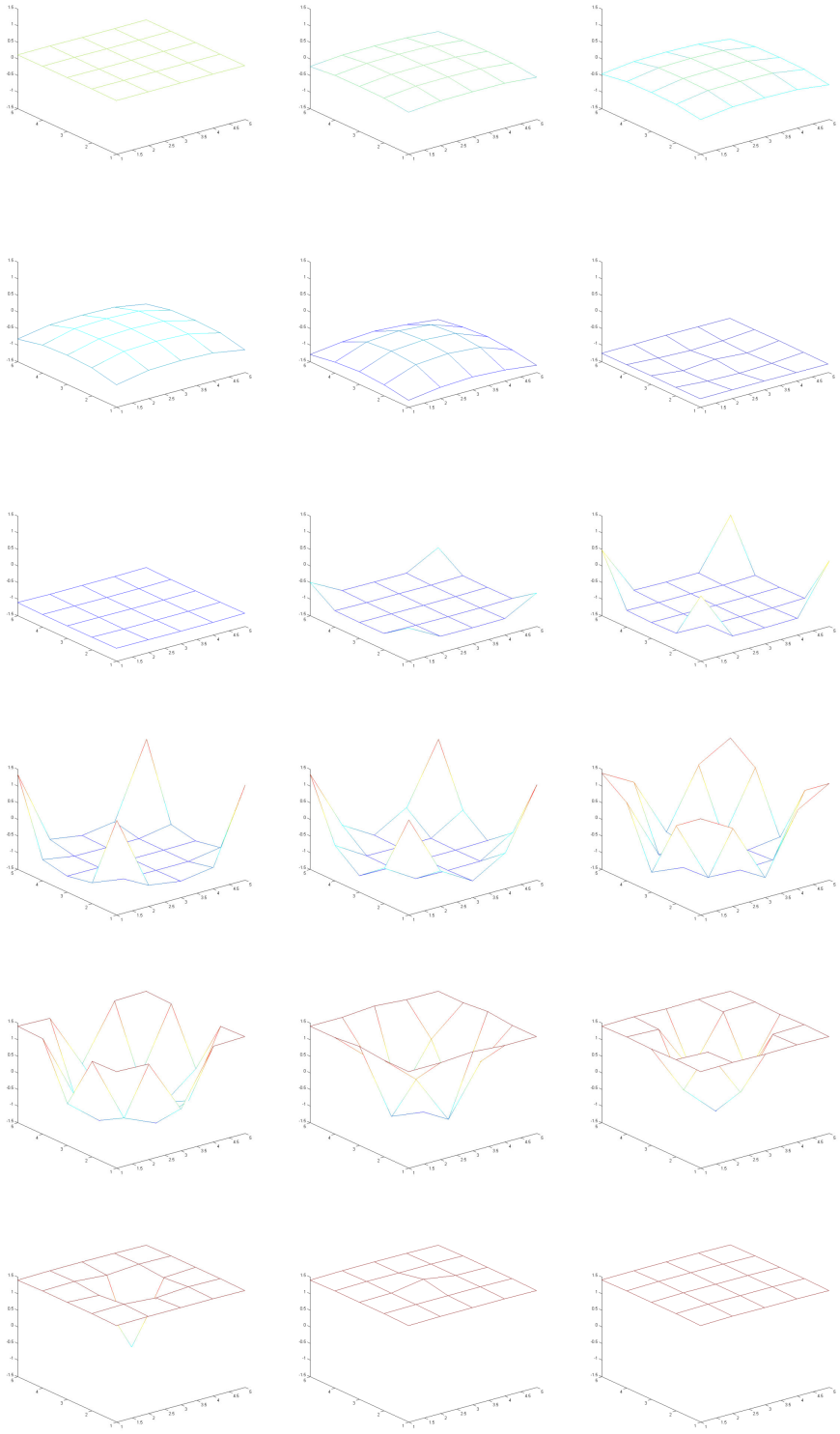
Ağ oluşumuyla komşuluk bağlantılarının ağırlıkları olan a parametreleri de önem kazanmıştır. Bu ağırlıklara 0 değeri verilirse tüm hücreler birbiri ile bağımsız hale gelir ve her biri tek hücre benzetimindeki gibi salınır. Ağırlıkların sıfırdan farklı bir değer alması ile hücreler osilasyona devam eder fakat aralarında tıpkı farklı başlangıç koşulları ile başlatılmış gibi faz farkı oluşur. Faz farkının oluşması ağ üzerinde uzaysal dalgaların oluşmasına sebep olan temel olaydır. Aşağıda bu ağın benzetimi için yazılan kod verilmiştir.

```
satir = 5; sutun = 5;
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 3.6 : Ağın benzetim kodu.

Şekil 3.7'de, ağdaki hücrelerin x durum değişkenlerinin aldığı değerler grafiklerde sunulmuştur. Grafikler, sağ üst köşeden sol alt köşeye doğru zaman içinde sıralı olarak elde edilmiştir.

Tüm hücreler aynı başlangıç durum değerleri ile benzetime başlatılmıştır. Hücrelerin x durumları yaklaşık olarak -1.5 ile 1.5 arasında değişmektedir. Hedeflenen uzay-zaman dalgalarının ağın köşe hücrelerinden yayılmaya başladığı görülmektedir. Köşe hücreleri diğerlerinden ayıran fark, iki komşusunun sınır koşuluna bağlı olmasıdır. Sınır koşulu, bu bağlantıların sürekli 0 ile sürülmesini gerektirmektedir. İki komşusu sabit olan köşe hücreleri diğerlerine göre daha önce durum değiştirmektedir. Köşe hücrelerin komşuları, köşe hücrenin durum değiştirmesiyle eskisinden daha farklı değerler ile beslendikleri için tetiklenmekte ve durum değiştirmektedir. Durum değiştirme ile kastedilen -1.5 yaklaşıklığında olan x durumunun +1.5 yaklaşığındaki bir değere hızla geçiş yapmasıdır.



Şekil 3.7 : 5x5 ağın benzetimi sonucunda x durum değişkenlerinin aldığı değerler.

4. RO-HYSA'NIN SAYISAL TASARIMI VE GERÇEKLENMESİ

RO-HYSA'nın hedeflenen tüm işlevlerini yerine getiren halinden önce, ön devre olarak iki sürümü tasarlanmış ve gerçeklenmiştir. Tek bir devre gerçeklemek yerine tasarımı üç basamaklı olarak tamamlamakla daha verimli bir devre yapısının oluşturulduğu söylenebilir. Bu sürümlerden ilki Altbölüm 3.2.2 'de sunulan benzetim sonuçlarını gerçekleştirmektedir. İkinci sürüm devresi ise çekirdeğinde ilk sürüm devresini barındırmakta ve dahili belleğinde 5x5'den daha büyük boyutlu bir ağın değişkenlerini saklamaktadır. Böylelikle en fazla 160 x 160 hücreli bir ağın öykünmesini gerçekleştirebilir olmuştur. İlk devre kullanılan FPGA'nın büyük kısmını kaplamaktadır. Bu yüzden ikinci devrede bellek elemanları kullanılarak öykünlenecek ağ dilimlere ayrılmış ve 5 x 5 boyutlu çekirdekte bu ağ dilimleri sırası ile öykünlenmiştir. İkinci devrede zamanda sıralı iş yükleri ile gerçeklemenin daha fazla sayıda sayısal devre elemanına olan ihtiyacı kaldırılmış, fakat öykünleme süresinin uzamasına sebep olunmuştur. Her iki RO-HYSA devresinde kullanılan parametreler ve başlangıç koşulları tasarımın FPGA'ya gönderilmesinden önce ayarlanmakta ve devrenin çalışması esnasında değiştirilemektedir. Ayrıca iki devre de ağın durumunun VGA monitörden izlenmesini sağlayacak şekilde tasarlanmıştır.

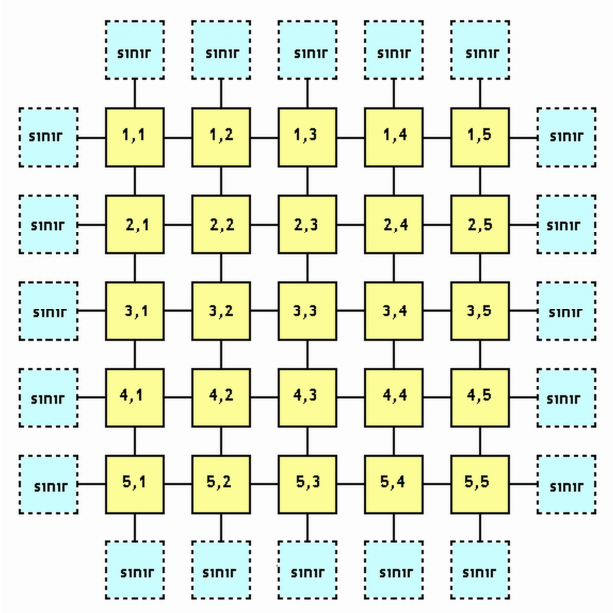
Bu iki gerçeklemeden sonra başlanan son tasarımda devreye yeni işlevler kazandırılmıştır. Üçüncü devre istendiği anda başlangıç koşulları yüklenebilecek ve parametreleri değiştirilebilecek şekilde tasarlanmıştır. Bu etkileşimli çalışma sayesinde istenen uzay-zaman dalgalarının araştırılması yapılabilir ve devre bu dalgalarla yapılacak uygulamalarda kullanılabilir olmuştur. Bu devrede ağın sınır koşullarının belirlenmesinde de farklı bir yöntem kullanılmıştır. Ek işlevler bu devrenin kapladığı kapı dizisi büyüklüğünü ve bellek ihtiyacını arttırmıştır. Bu sebeple öykünlenen ağ büyüklüğü 128 x 128 boyutuna düşürülmek zorunda kalmıştır.

RO-HYSA için tasarlanan üç devre sırasıyla Bölüm 4.1, Bölüm 4.2 ve Bölüm 4.3'de anlatılmaktadır.

4.1 5 x 5 Hücreli RO-HSYA'nın Tasarımı ve Gerçeklenmesi

Bu bölümde, Altbölüm 3.2.2 'de benzetimi yapılan 5x5 hücrelik HYSA'nın gerçeklenmesi anlatılmaktadır. 5x5 hücreli HYSA yapısı, FPGA üzerinde Verilog ile tanımlanan sayısal devre ile gerçeklenmiştir. Gerçeklemede Şekil 4.1'deki yapı kurulmuştur. Ağdaki her hücre dört ana yöndeki (doğu, kuzey, batı, güney) komşusu ile bağlıdır. Ağın sınır kesimindeki hücrelerinin sınırın dışına uzanan bağlantıları Şekil 4.1'de 'sınır' ile adlandırılan sanal komşulara bağlanmış, bu komşulardan her zaman aynı ilgili değişken değeri ile beslenmiştir.

5x5 HYSA'in gerçeklenmesi, yukarıda anlatılan devre parçalarının Verilog HDL dili ile kodlanması, ModelSim yazılımı ile benzetiminin yapılması, Xilinx ISE yazılımı ile sentezlenmesi, gerçeklenmesi ve Xilinx Impact yazılımı ile 'Xilinx University Program Virtex-II Pro Development System' kartındaki XC2VP30 FPGA'nın konfigüre edilmesi ile yapılmıştır.



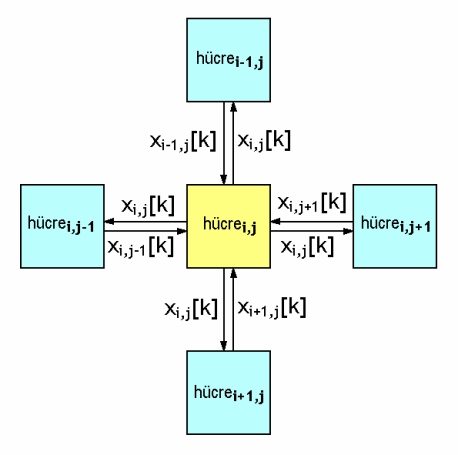
Şekil 4.1 : 5x5 HYSA yapısı.

Ağdaki hücrelerden her biri hangi satırda bulunduğunu belirten i ve hangi sütunda bulunduğunu belirten j indisleri ile etiketlenmiştir. Herhangi bir hücre referans kabul edildiğinde alacağı indisler ve komşularının alacağı indisler Şekil 3.2'de gösterilmiştir.

Her hücre Bölüm 3.1'de verilen modele uygundur. Sadece sürekli giriş işareti, u, modelden atılmıştır. HYSA'in gerçeklenebilmesi için bu denklem takımı

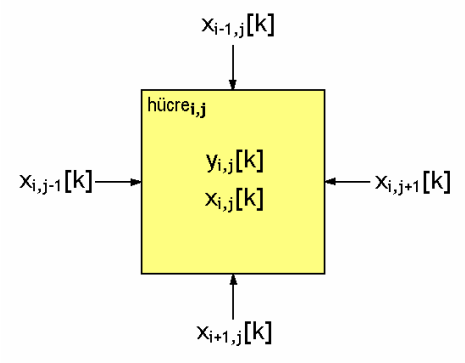
ayrıklaştırılmıştır. Denklem (3. 10), (3. 11) ve (3. 12)'deki ağırlıklar (α, β, ε, σ, m, a) ve iterasyon adımı (T) tüm hücreler için ortaktır ve devre gerçeklemesinden sonra değiştirilemezler. Ancak devrenin gerçeklenmesinden önce değiştirmek mümkündür.

Ağdaki komşu hücreler, k anındaki x durum değişken değerlerini birbirlerine aktarmaktadır. Hücrenin y durum değişkeninin ise ne komşu hücrelere ne de ağın dışına aktarılması gerekmemektedir. Ağdaki i,j indisli bir hücre göz önünde bulundurularak Şekil 4.2'de gösterilen,dört komşusuna durum değişkenini çıkış olarak veren ve her bir komşunun durum değişkenlerini de giriş olarak alan bir sayısal devre ortaya çıkmıştır.



Şekil 4.2 : Komşu hücreler arası bağlantılar.

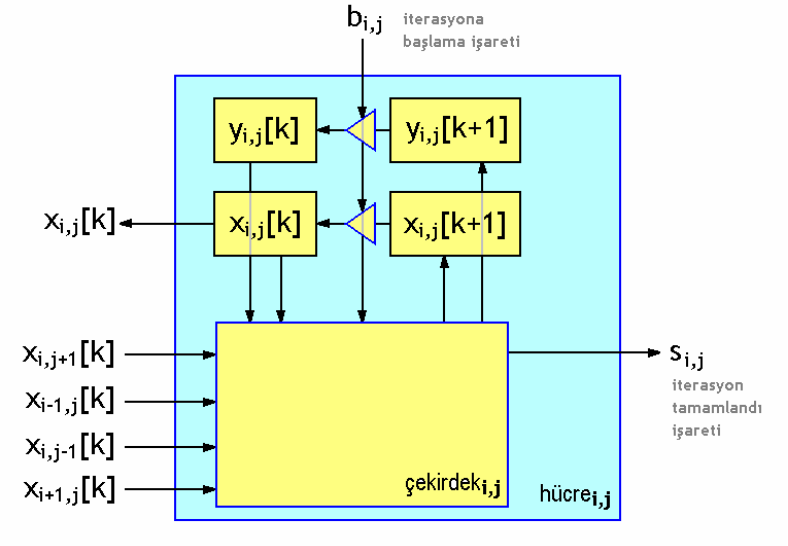
5x5 hücrelik HYSA için öncelikle tek bir hücre gerçeklenmiştir. Bir hücre, k anındaki durum değişkenlerini tutucularında saklamaktadır. Durum değişkeni tutucusu bir çıkış veriyolunu sürmektedir. Hücre dört ayrı giriş veriyolu ile komşularının durum değişkenlerinin sürdüğü çıkış veriyollarına bağlanacak şekilde tasarlanmıştır. k anında, denklem takımının k+1 anı için çözümüne bir iterasyon ile ulaşılır. Hücre bir iterasyona boyunca yaptığı işlemler esnasında komşularından durum değerlerini okumaktadır. Şekil 4.3'de tek hücrenin iterasyon esnasında kullandığı girişler ve tutucuları basitçe gösterilmektedir.



Şekil 4.3 : Bir hücrenin tutucularında sakladığı değişkenler.

Parametreler ve iterasyon adımı ise devrenin gerçeklenmesinden sonra değiştirilmediğinden sabit kablo bağlantıları olarak gerçeklenmiş ve bu sayede FPGA üzerindeki sabit miktardaki flipflop kaynağı daha az tüketilmiştir. Bunların değerleri HYSA devresinin FPGA üzerinde her yeniden kuruluşunda değiştirilebilmektedir.

Bir hücre için iterasyona başlama diğer hücrelerden bağımsız gerçekleşmemektedir. Hücrelerin gerçek iterasyon süresi değişkendir. Bu değişkenliğin sebebi ilerleyen kısımlarda detaylı olarak açıklanmaktadır. Hücrelerin değişen iterasyon süreleri sebebiyle, üstte tüm hücreleri alt devre olarak barındıran HYSA devresinin, hücrelerin iterasyonlarını bitirip bitirmediklerini izlemesi gerekmektedir. Tüm hücrelerin iterasyonlarını tamamlamasından sonra hücreleri yeniden iterasyona başlatma görevi de yine üstteki HYSA devresinindir. Bu yüzden bir hücrenin iterasyona başlama işaret girişi bir de iterasyonu tamamlama işaret çıkışı bulunmaktadır. Şekil 4.4'de hücrenin yapısı bu işaretlerle birlikte gösterilmiştir.



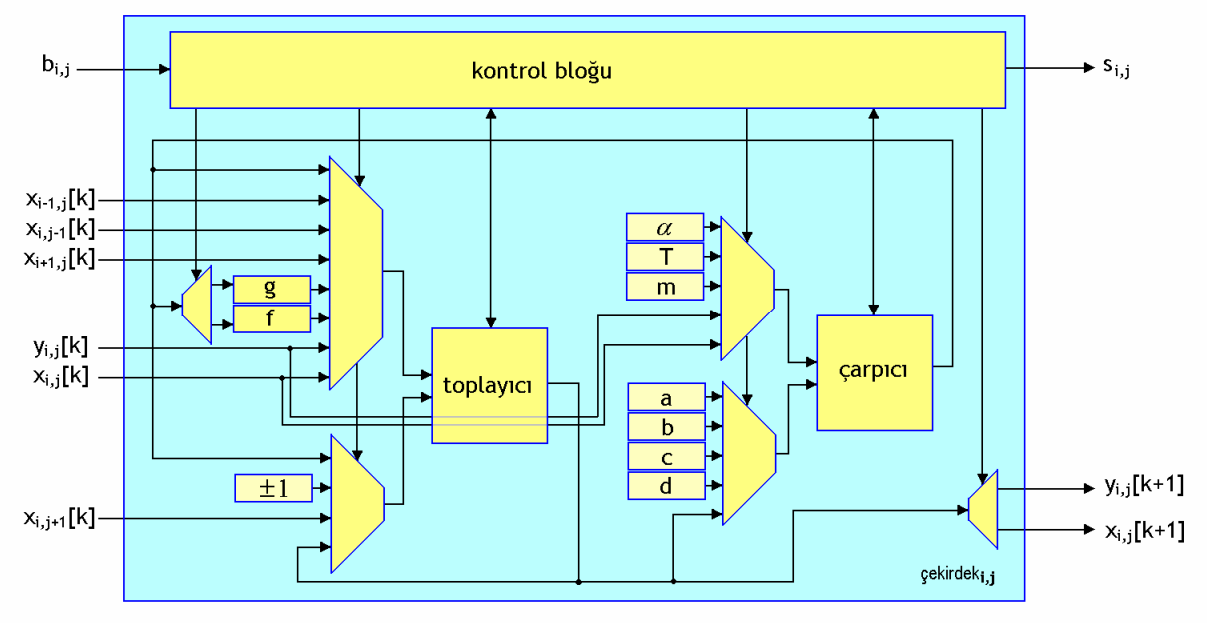
Şekil 4.4 : Hücrenin yapısı.

İterasyona başlama işareti k kadar sürenin geçtiği anlamına da gelmektedir. k kadar süre geçmesi, hücrenin bir önceki iterasyonla hesapladığı durum değişken değerlerinin artık k+1 anına ait tutuculardan k anına ait tutuculara aktarılmasını gerektirmektedir. Şekil 4.4'de görüldüğü gibi tutucu transfer sürücüsü iterasyonu başlatan işaretin kontrolündedir. Bu başlatma işaretinin tüm hücrelere aynı anda uygulanması ile hücrelerin iterasyon içerisinde komşularından önceki ya da sonraki zamana ait hatalı değerler almasının önüne geçilmiştir. Başlama işareti bir saat çevrimi süren lojik 1 formunda uygulanmaktadır.

Hücrenin iterasyonunu tamamladığının işareti ise hücre çekirdeğinin kontrolündedir. Çekirdek başlama işareti ile aktif olan ve ilgili değişkenler üzerinde gerekli aritmetik işlemleri yerine getiren yapıdır. Aritmetik işlemlerin sonlanması ile iterasyon tamamlandı işaretini hücrenin dışına gönderir. Bu işaret de giriş işareti gibi bir saat çevrimi süren bir darbe formundadır. Tamamlandı işareti bir üst yapı olan HYSA devresince sürekli izlenmektedir. Başlatma ve tamamlama işaretleri HYSA devresi içerisinde detaylıca tekrar anlatılacaktır.

4.1.1 Hücre çekirdeğinin yapısı

Hücre çekirdeği $x_{i,j+1}$,$x_{i-1,j}$,$x_{i,j-1}$,$x_{i+1,j}$ ve $x_{i,j}$ ile $y_{i,j}$ değişkenlerinin tutulması ile ilgilenmez. Çekirdeğin bu girişleri ait olduğu hücredeki tutucular ya da komşusu olan hücreler içindeki tutucular tarafından sürülür. Çekirdeğin işlemler esnasında kullandığı α, β, ε, σ, m, a ağırlıkları ile T iterasyon adımı devrenin FPGA üzerinde kurulmasından sonra değişmez. Bu değerler de önceden belirtildiği gibi sabit kablo bağlantıları ile devrede varlık bulur. Şekil 4.5'de hücre çekirdeğinin yapısı verilmiştir. Bu şekildeki şemada iterasyon adımı ve parametrelerin tutucu içinde saklanıyor görünmesi devre kurulurken değiştirilebilir olduğunun hatırlanması içindir.



Şekil 4.5 : Hücre çekirdeğinin yapısı.

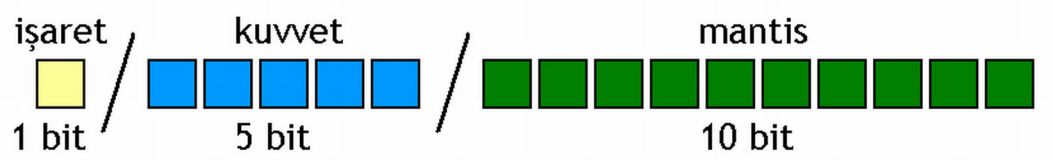
Hücre çekirdeğinin kontrol bloğu, başlama işareti ($b_{i,j}$) ile çalışmaya başlayan bir durum makinesini barındırır. Bu durum makinesi çözülen denklem takımının gerektirdiği sıraya uygun şekilde toplama ve çarpma devreleri ile seçicileri kontrol eder. Ayrık denklem takımının tüm aritmetik basamakları zaman içinde sıralı biçimde çekirdekte gerçekleştirilir. g ve f fonksiyonlarının hesaplanan sonuçları ilgili iki adet tutucuda saklanır. Denklem takımının sonuçları da Şekil 4.4'de gösterilen çekirdeğin dışındaki $x_{i,j}[k+1]$ ve $y_{i,j}[k+1]$ tutucularına gönderilir. Aritmetik işlemler sürerken önce olarak x'in k+1 için değeri ardından y'nin k+1 için değeri hesaplanır. Kontrol bloğu $x_{i,j}[k+1]$ hesaplandığında ilgili çekirdek çıkışını toplayıcının çıkışı ile sürer. Bu ve bundan sonraki devre şemalarında gösterilen kontrol bloklarının, şemalarda gösterilmeyen saat işareti ve reset işareti girişleri de mevcuttur. Tüm devreler saat işaretinin yükselen kenarında tetiklenir. Reset işareti de asenkron aktif sıfırdır. HYSA devresinin tüm alt devreleri aynı saat işareti ile sürülen senkron devrelerdir. İterasyon içerisinde yürütülen aritmetik işlemler sırası ile Çizelge 4.1'deki gibidir.

Çizelge 4.1 : Hücre çekirdeğinde bir iterasyonda gerçekleşen işlemler.

İşlem Sırası	Hücre Çekirdeğinde Gerçekleştirilen İşlem
1	$toplam = x_{i,j+1}[k] + x_{i-1,j}[k]$
2	$toplam = toplam + x_{i,j-1}[k]$
3	$toplam = toplam + x_{i+1,j}[k]$
4	$f_{i,j}[k] = \alpha \cdot toplam$
5	$x_{i,j}[k] \geq 1 \Rightarrow toplam = x_{i,j}[k] - 1$
6	$x_{i,j}[k] \leq -1 \Rightarrow toplam = x_{i,j}[k] + 1$
7	$g_{i,j}[k] = m \cdot toplam$
8	$-1 < x_{i,j}[k] < 1 \Rightarrow g = 0$
9	$çarpim_1 = a \cdot x_{i,j}[k]$
10	$çarpim_2 = b \cdot y_{i,j}[k]$
11	$toplam = çarpim_1 + çarpim_2$
12	$toplam = toplam + g_{i,j}[k]$
13	$toplam = toplam + f_{i,j}[k]$
14	$çarpim = toplam \cdot T$
15	$x_{i,j}[k+1] = çarpim + x_{i,j}[k]$
16	$çarpim_1 = c \cdot x_{i,j}[k]$
17	$çarpim_2 = d \cdot y_{i,j}[k]$
18	$toplam = çarpim_1 + çarpim_2$
19	$çarpim = toplam \cdot T$
20	$y_{i,j}[k+1] = çarpim + y_{i,j}[k]$

15. işlem sonucu x durum değişkeninin çekirdek dışındaki k+1 anı tutucusuna,20. işlem sonucu da y durum değişkeninin k+1 anı tutucusuna aktarılır. Şekil 4.4 ve Şekil 4.5'de verilen hücre ve çekirdek yapıları kolay anlaşılması için sadeleştirilmiş, detay bazı bağlantılar bu şemalarda gösterilmemiştir.

Sayısal gerçeklemelerin tamamında çarpma işlemini ve toplama işlemini gerçekleştiren aritmetik işlemciler tasarlanmış ve kullanılmıştır. Bu aritmetik işlemciler 16 bit genişlikte yarı duyarlı kayan noktalı sayılar üzerinden işlem yapabilmektedir. Gerçeklemelerdeki tüm değişkenler bu sayı formatında saklanır ve işlenir. Değişken tutucuları ve veriyolları 16 bit genişliğindedir. Devrede kullanılan sayı formatı Şekil 4.6'da gösterilmektedir.



Şekil 4.6 : Yarı duyarlı (16-bit) kayan noktalı sayı formatı.

Şekilde, i: işaret biti, e: 5 bitlik kuvvet kelimesi, m: 10 bitlik mantis kelimesi olmak üzere; gösterilen sayının ikili düzendeki değeri $(-1)^i \cdot 2^{e+bias} \cdot \{1, m\}$'dir [24]. Onluk düzendeki değeri ise ikili düzendeki değerine taban dönüştürme işlemi uygulanarak bulunur. Kullanılan bu sayı formatında bazı sayıların gösterimleri Çizelge 4.2'de verilmiştir:

Çizelge 4.2 : 16 bit kayan noktalı formatta örnek sayılar.

Açıklama	Kuvvet	Mantis	Değer
Sıfır	00000	Önemsiz	0
Gösterilebilen en küçük sayı	00001	0000000000	6.1035e-005
1'den küçük bir sayı	01010	1100101110	0.0561
Bir	01111	0000000000	1
1'den büyük bir sayı	10101	1100101110	114.8750
Gösterilebilen en büyük sayı	11110	1111111111	65504
Sonsuz	11111	Önemsiz	∞

Yapılan çalışmada genel olarak uzay zaman dalgalarının incelenmesi de amaçlandığından, hücrelerin durum değişkenleri ve parametrelerin yüksek çözünürlüklü değerlere sahip olması istenmiştir. Kayan nokta aritmetiğinde dinamik olarak değer çözünürlüğü değişmekdir. Ayrıca sabit noktalı sistemlere göre gösterilebilecek maksimum ve minimum sayı aralığı da daha geniştir. Bununla birlikte bellekte saklanacak verinin miktarı da göz önünde bulundurularak 32 bit ya da 64 bit genişlikli kayan nokta aritmetiği tercih edilmemiştir. Devre gerçeklemelerinde FPGA tümdevresi içindeki bellekler kullanılmıştır ve bu bellek miktarı da ancak 16 bit genişlikli kelimeler kullanıldığında yeteri kadar büyük bir ağı saklayabilecek kadardır. Yarı duyarlı kayan nokta sayı formatı ve aritmetiği ile ilgili daha fazla bilgi [24] kaynağından bulunabilir.

Şekil 4.5'de hücre çekirdeği içinde bir çarpıcı blok görülmektedir. Bu blok ve çekirdekteki toplayıcı bloğu, gerçeklenen HYSA devreleri için tasarlanan aritmetik işlem devreleridir. Çarpma devresi, 16 bitlik kayan nokta biçimli iki sayıyı 3 saat çevriminde çarpmaktadır. Devre girişine verilen çarpan1 ve çarpan2 değerleri işaret,

kuvvet ve mantis olarak 3'e ayrılır. Kullanılan sayı formatında kuvvetin sıfır olması, mantise bakılmaksızın, sayının sıfıra eşit olduğunu ifade etmektedir. Standart kayan noktalı sayı formlarında kuvvetin sıfır olabildiği denormalize sayılar da geçerlidir. Bu gerçeklemede ise çok küçük gerçek değerlere sahip denormalize sayılar sıfır kabul edilmiştir. Dolayısı ile girişteki iki sayıdan herhangi birinin kuvveti sıfır ise çarpım sonucu doğrudan sıfır olarak verilir.

Her iki çarpanın kuvvetlerinin sıfırdan farklı olması halinde kuvvetler toplanır, mantisler çarpılır. Kuvvetlerin toplanması ile kullanılan sayı formatında gösterilebilecek en büyük sayı olan 65504'den büyük bir çarpım elde edildiği görülüyorsa, çarpım sonucu olarak çıkışa sonsuz (0_11111_0000000000) verilir. Benzer şekilde toplanan kuvvetlerden, çarpım sonucunun gösterilebilecek en küçük sayı olan 6.1035e-005'den daha küçük değerde olduğu görülüyorsa, çarpım sonucu olarak çıkışa sıfır (0_00000_0000000000) verilir.

Çarpımın sıfıra ya da sonsuza gittiği durumların haricinde mantislerin çarpımında eğer gerekiyorsa 1 bit sağa kaydırma yapılır. Bu kaydırma büyük mantislerin çarpımının sonucunda aşağıdaki örnekteki gibi virgülün solunda iki anlamlı bit oluşması halinde gerçekleşir:

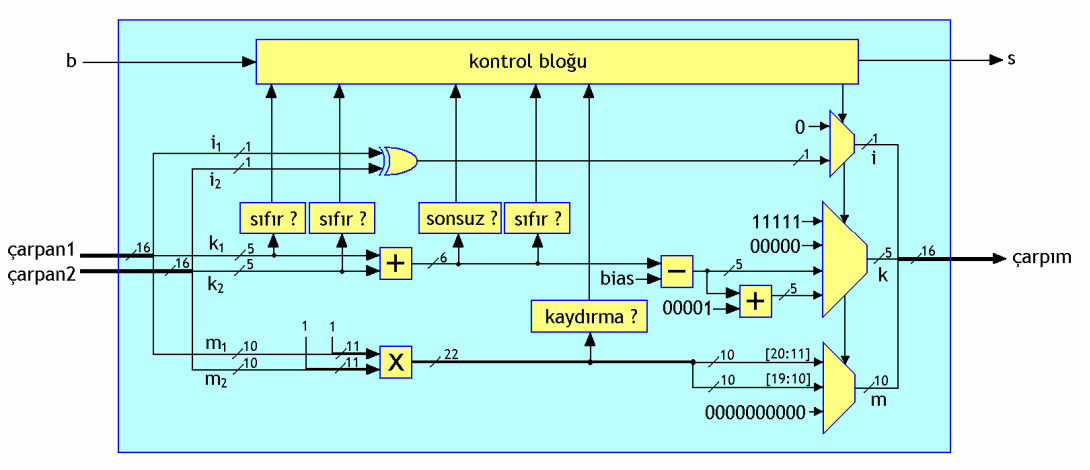
$$\begin{array}{l} 1{,}1101 \\ \underline{1{,}1100} \\ 11{,}001011 \end{array}$$

Çarpımın kuvveti çarpan kuvvetlerinin toplamından bias değerinin çıkarılması ve mantis kaydırması gerçekleşecek ise 1 eklenmesi ile elde edilir. Bias değeri her iki çarpanın kuvvetinde de bulunduğu için çıkartılır. Denklem (4. 1) bunu göstermektedir:

$$\left[(-1)^{i_1} \cdot 2^{e_1+bias} \cdot \{1, m_1)\right] \cdot \left[(-1)^{i_2} \cdot 2^{e_2+bias} \cdot \{1, m_2)\right] = (-1)^{i_1+i_2} \cdot 2^{e_1+e_2+bias+bias} \cdot \{1, m_{carpim}\} \quad \textbf{(4. 1)}$$

Aynı işaretli çarpanların işareti pozitif, farklı işaretli çarpanlarınki ise negatiftir. Kullanılan sayı formunda negatif sayıların işareti 1 ile, pozitif sayıların işareti 0 ile ifade edilir. Dolayısı ile çarpımın işareti, çarpanların işaretinin özel-veya'sı ile bulunur.

Şekil 4.7'de çarpma devresinin yapısı verilmiştir. Çarpım çıkışı; işaret, kuvvet ve mantis çıkışlarının bir veriyolunda birleştirilmesi ile oluşturulur. İşaret, kuvvet ve mantis çıkışları çıkış seçicisi ile seçilir. Bu seçiciler kontrol bloğunun kontrolündedir. İkili düzende toplama ve çarpma, kullanılan devre sentezleyici yazılım tarafından kombinezonsal olarak sentezlenmekte ve FPGA üzerinde gerçeklenmektedir.



Şekil 4.7 : Çarpma devresinin yapısı.

Toplama devresi de 16 bitlik iki kayan noktalı sayıyı toplamaktadır. Kayan noktalı sayılarda toplama işlemi çarpmaya göre daha karmaşıktır. Bu karmaşıklık kayan noktalı sayılarda her iki sayıyı oluşturan bit dizileri üzerinde noktaların aynı konumda olmamalarından kaynaklanmaktadır. Bir başka deyişle problem, terimlerin kuvvetlerinin aynı olmamasıdır.

Kayan noktalı sayıları toplamak için öncelikle sayıların denormalizasyonu gerekmektedir. Denormalizasyon bir sayının kuvvetinin değiştirilerek, mantisinin kaydırılması ile sayının normal gösterimden çıkarılmasıdır. Toplama devresinde denormalizasyon ile iki sayının kuvveti birbirine eşitlenir. Aynı işlem ile mantislerin noktalarının birbirleri ile aynı hizada olması sağlanır.

Denormalizasyon sonrasında mantisler, toplamaya giren sayıların işaretlerine ve genliklerine göre ikilik tabanda toplanır ya da çıkartılır. Birbirine eşitlenen kuvvetler ve toplanan mantisler hala denormalize formdadır.

Toplamın mantisi sayıyı normal forma sokacak şekilde kaydırılır ve kuvveti kaydırma miktarı kadar değiştirilir. Bu işleme normalizasyon denir. Aşağıda normalize, denormalize gösterim ve bir toplama örneği verilmiştir:

i:1/k:5/m:10 bit formatında kayan noktalı sayı gösteriminde bias değeri 01111 iken; 1_01111_0011100000 sayısı normalize bir sayı ise değeri

$(-1)^i \cdot 2^{e+bias} \cdot \{1, m[9:0]\} = -1 \cdot 2^0 \cdot (1{,}0011100000)_2 = (-1{,}0011100000)_2$'dir.

Aynı sayı $(-1)^i \cdot 2^{e+bias} \cdot \{0, m[9:0]\}$ formunda denormalize olarak 1_10000_1001110000 şeklinde de gösterilebilir. Bu sayı yine

$(-1)^i \cdot 2^{e+bias} \cdot \{0, m[9:0]\} = -1 \cdot 2^1 \cdot (0{,}1001110000)_2 = (-1{,}0011100000)_2$ olacaktır.

Örneğin 0,125 ile 8'in normalize gösterimleri

$(0{,}125)_{10}$=$(0\_01100\_0000000000)_{norm}$ ve$(8)_{10}$=$(0_10010_0000000000)_{norm}$'dir.

Bu iki sayıyı toplamak için denormalize forma sokulmaları ve kuvvetlerinin eşitlenmesi gerekir

$(0{,}125)_{10}$=$(0\_10011\_0000001000)_{denorm}$ ve$(8)_{10}$=$(0_10011_1000000000)_{denorm}$ olur.

İki sayının toplanması için denormalize sayıların mantisleri toplanır, kuvvet aynen kalır

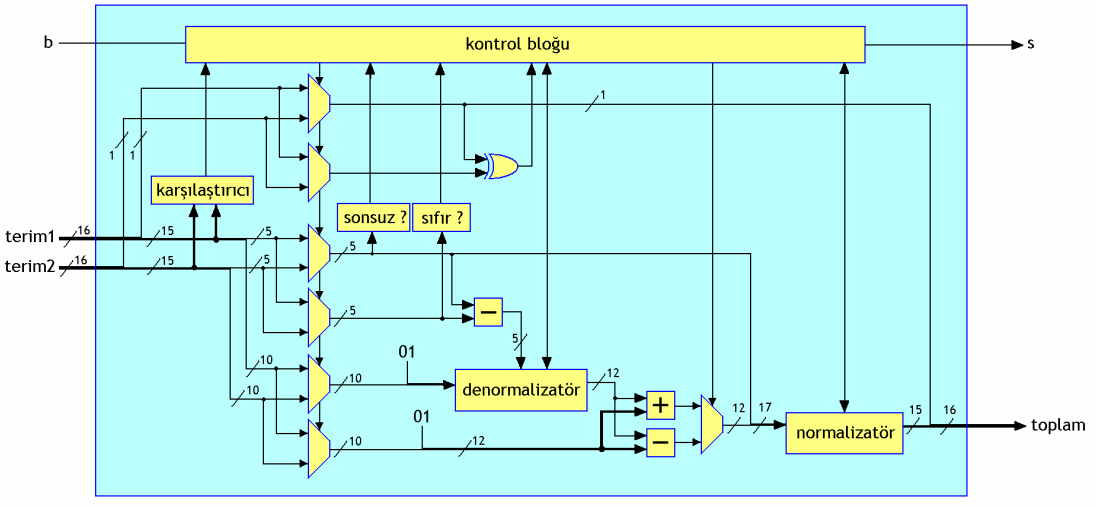
$(8{,}125)_{10}$=$(0\_10011\_1000001000)_{denorm}$

Sonuç normalize edilir

$(8{,}125)_{10}$=$(0\_10010\_0000010000)_{norm}$

Tasarlanan toplayıcı devresinde yukarıda örneği verilenden farklı bir denormalize form kullanılmıştır. Terimlerin 10 bitlik mantisleri, daha yüksek öncelikli 2 bitlik '01' ile birleştirilerek 12 bite dönüştürülür. Terimlerden büyük olanın mantisi ve kuvveti değiştirilmeden, küçük olanının kuvveti büyük olanınkine eşit hale getirilir. Bunun için küçük terimin mantisi kuvvetlerin farkı kadar sağa kaydırılır. 12 bitlik mantis formunda virgülün yeri 11. ve 10. bitlerin arasıdır. Küçük sayının mantisinin kaydırılması denormalizatör devre tarafından yapılır. Kaydırılmış mantis ile büyük sayının mantisi, iki sayının işaretleri aynı ise ikilik düzende toplanır, farklı ise çıkartılır. Aritmetik işleme giren sayıların en yüksek öncelikli biti 0 olduğu için sonucunda taşma olmaz ve sonuç yine 12 bit kalır. 12 bitlik toplam mantisi ile genliği büyük sayının 5 bitlik kuvveti 17 bitlik veriyolunda birleştirilir. Bu değer normalizatör devreye giriş olarak uygulanır. Normalizatör devre 12 bitlik mantis kısmını 12. ve 11. bitleri '01' olacak şekilde kaydırır. Kuvvet değerini de kaydırma miktarı kadar, kaydırma yönüne göre arttırır ya da azaltır. Bu sayede toplam,

hesaplanan kuvvet kelimesinin tamamı ve kaydırılan mantisin 10. ve 1. bitleri arasındaki bir dizisi ile 15 bit olarak normalize forma getirilir. Normalizatörün çıkışı,toplayıcı devrenin çıkışı olarak sürülür. Şekil 4.8'de toplama devresinin yapısı verilmiştir.



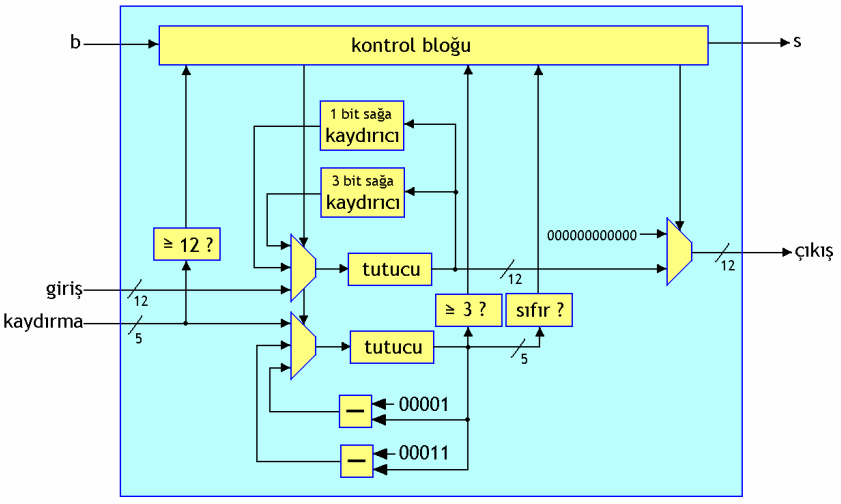
Şekil 4.8 : Toplama devresinin yapısı.

Toplama devresinin toplama süresi, denormalizatör ve normalizatör alt devrelerinde gerçekleşen kaydırma işlemlerinin, kaydırılacak sayıya göre değişmesi sebebi ile değişkendir. Hem kaydırma miktarlarının değişken olması hem de kaydırmanın bir saat çevriminde duruma göre 3 bit veya 1 bit yapılabiliyor olması toplayıcı devresinin çalışma süresini tahmin edilemez kılmaktadır. Daha önce de belirtildiği gibi, toplama işlemlerinin süresinin değişmesi ile bir hücrenin iterasyon süresi de değişmektedir. Bu yüzden hücreleri barındıran HYSA devresinde tüm hücrelerin iterasyonlarının tamamlanmaları takip edilmektedir.

Toplama devresinin alt bloğu olan denormalizatör devre girişine uygulanan sayıyı kaydırma miktarı kadar sağa kaydırır. Giriş 12 bitlik olduğu için kaydırma miktarı 11'den büyük ise kaydırma sonucu 0 olacaktır. Bu yüzden kontrol bloğu tarafından ilk olarak kaydırma girişinin 12'den büyük olup olmadığına bakılır. Kaydırma ve sayı girişleri, kaydırma ve eksiltme işlemlerini yapabilmek için tutuculardа saklanır. Kaydırma miktarı sıfırlanıncaya kadar kaydırma sürer. Kaydırma miktarı 2'den büyük ise 3 bit kaydırılır. Bu sayede denormalizatörün hız/alan faktöründe iyileştirme yapılmıştır. Her bir kaydırma adımı 1 saat çevrimi sürmektedir. Yalnız bir bit kaydırma ile üç veya bir bit kaydırmanın işlem süreleri Çizelge 4.3'de karşılaştırılmıştır. Denormalizatör devre yapısı Şekil 4.9'da verilmiştir.

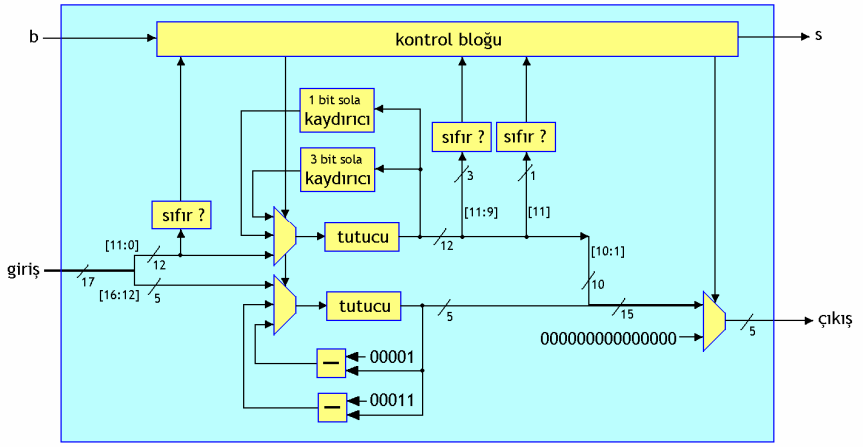
Çizelge 4.3 : Kaydırma süreleri karşılaştırması.

	0	1	2	3	4	5	6	7	8	9	10	11	12	kayma miktarı
1 bit kaydırıcı	0	1	2	3	4	5	6	7	8	9	10	11	12	saat çevrimi
1-3 bit kaydırıcı	0	1	2	1	2	3	2	3	4	3	4	5	4	saat çevrimi



Şekil 4.9 : Denormalizatör devrenin yapısı.

Normalizatör devrede yapılan ilk kontrol mantisin sıfır olup olmamasıdır. Mantis sıfır ise kuvvet de sıfırlanır ve çıkışa verilir. Aksi durumda mantis içindeki en yüksek öncelikli '1' 12. bite gelinceye dek mantis sola kaydırılır. Kaydırma denormalizatör devredeki gibi üç ya da bir bit sola kaydırma ile yapılır. Kuvvet tutucusunu değeri her sola kaydırmada azaltılır. 12 bitlik mantis tutucusunun 12. biti 1 yapıldığında, mantis tutucusunun 11. ve 2. bitleri arası,5 bitlik kuvvet tutucusu veriyolu ile birleştirilir ve çıkışa sürülür. Şekil 4.10'da normalizatör devresinin yapısı verilmiştir.

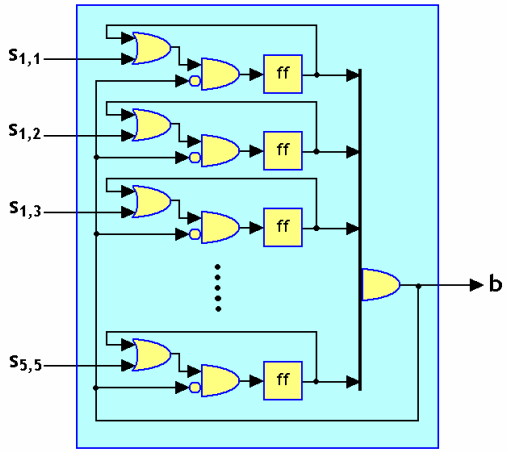


Şekil 4.10 : Normalizatör devrenin yapısı.

4.1.2 HYSA devresi

HYSA devresi 5x5 HYSA'yi oluşturan 25 hücreyi, bir adet frekans bölücü devreyi,iterasyon başlatma bloğunu ve bunlar arasındaki bağlantıları içeren devredir. Frekans bölücü devre ile FPGA kartının 100 MHz'lik saat işaretinden 12,207 KHz'lik saat işareti üretilmiştir. HYSA devresinin barındırdığı tüm hücrelere ve iterasyon başlatma bloğuna bu düşük frekanslı saat işareti verilmiştir. Çalışma frekansının düşürülmesi, HYSA devresinin çıkışlarını VGA ekranda insan gözünün yakalayabilmesi için yapılmıştır.

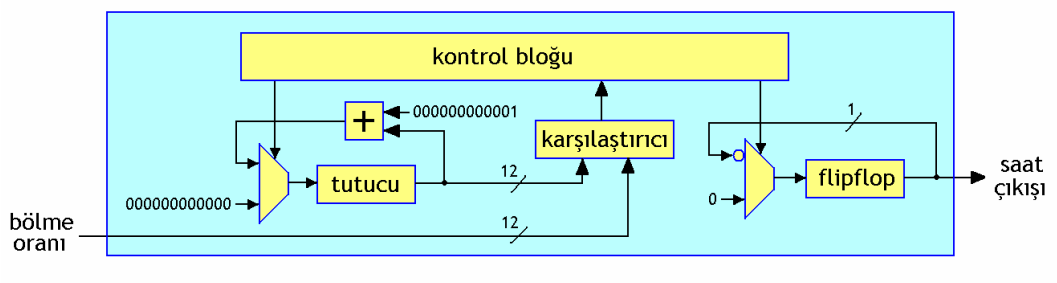
İterasyona başlatma bloğu da hücrelerin iterasyonu tamamlama çıkışları ile sürülen ardışıl bir devredir. Bu blokta her hücre için bir flipflop bulunmaktadır. Bu blok tarafından bir kere iterasyona başlama işareti verilmesini takiben iterasyonunu bitiren her hücrenin ilgili flipflop'u 1 yapılarak bu saklanır. Flipflop'ların hepsi 1 olduğunda tüm iterasyonlar tamamlanmıştır ve iterasyon başlatma işareti yeniden 1 olur. Başlama işaretinin 1 olması, yeni bir iterasyonun başladığı ve flipfloplarının tümünün 0 yapılması gerektiği anlamına gelir. Şekil 4.11'de iterasyon başlatma bloğunun devre şematiği verilmiştir.



Şekil 4.11 : İterasyon başlatma bloğunun yapısı.

Frekans bölücü devre hem HYSA devresinin yavaşlatılması hem de VGA sürücü devrede senkronizasyon için gereken saat işaretinin üretilmesinde kullanılmıştır. En basit ilke ile her giriş saat işaretinde 12 bitlik bir tutucunun değerini 1 arttırır. Bölme oranı girişi ile karşılaştırır. Eğer tutucu değeri girişe eşit ise saat çıkış flipflopunun değeri terslenir ve tutucunun değeri sıfırlanır. Reset işareti gelmesi halinde tutucu ile

saat çıkış flipflopu sıfırlanır. Frekans bölücü devre yapısı Şekil 4.12'de verilmiştir.



Şekil 4.12 : Frekans bölücü devrenin yapısı.

Diğer devre yapılarını gösteren şekillerde olduğu gibi,(giriş)saat işareti ve reset işareti devrede olmasına rağmen bu şemada da gösterilmemiştir. Bu devre yapısında çıkış ve giriş saat işaretinin frekansları arasındaki ilişki (4. 2)'deki gibidir.

$$f_{cikis} = \frac{f_{giris}}{2 \cdot (bolme\_orani + 1)} \qquad \textbf{(4. 2)}$$

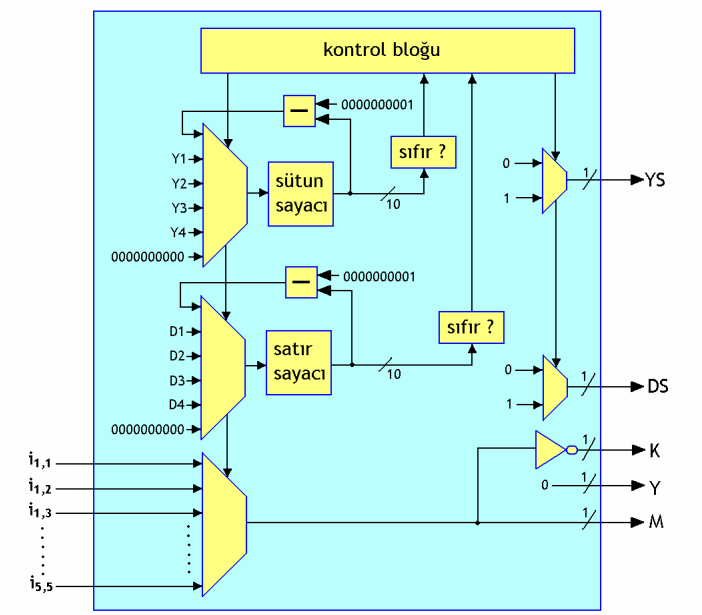
Gerçeklen HYSA'in x durum değişkenini gerçek zamanlı izleyebilmek için VGA ekran kullanılmıştır. Kullanılan FPGA kartında, VGA arayüzündeki kırmızı,yeşil ve mavi rengi belirleyen analog işaretleri üretmek için özel bir sayısaldan analoğa dönüştürücü tümdevre bulunmaktadır. Bu tümdevre ile 24 bit renk derinliğine sahip görüntüler oluşturmak mümkün olmasına rağmen, bu gerçeklemede 3 bit renk derinliği kullanılmıştır.

VGA ekranların katot ışın tüpünde, fırlatılan elektron düşey ve yatay eksende bobinler vasıtası ile saptırılır ve flor kaplı ekran yüzeyinde istenen noktaya düşürülür. VGA ekranlar soldan sağa satıların ve yukarıdan aşağıya tüm ekranın taranması ile çalışır. Katot ışını ile bir satırın ekrana çizilmesinin ardından bir sonraki satıra geçilir. Bir satırın taranması sırasında, satırın sonunda hala ışın kaynağının ilerlemeye devam ettiği arka boşluk zamanı, alt satırın başına yönlendirildiği geçme zamanı, ekranın duyarlı alanına kadar ilerlediği ön boşluk zamanı ve ekranın duyarlı bölgesini taradığı gösterme zamanı geçer. Aynı şekilde tüm ekranın bir kere taranması için katot ışınının düşey hareketlerinin gerçekleştiği süreler geçer. Bu süreler görüntü çözünürlüğü ve tazeleme oranı ilişkilidir. Bu parametreler ile sürelerin alacağı standart değerler bulunmaktadır. Bu gerçeklemede kullanılan 640 x 480 çözünürlük ve 60 Hz tazeleme oranı için, 25 Mhz olan piksel saat işaretinden ilgili zaman dilimlerinde kaçar periyot geçmesi gerektiği Çizelge 4.4'de verilmiştir.

Çizelge 4.4 : VGA zamanlama dökümü.

	Düşey tarama süreleri	Yatay tarama süreleri
Gösterme zamanı	384.000	640
Arka boşluk zamanı	1.600	96
Ön boşluk zamanı	8.000	16
Geçme zamanı	23.200	48
Toplam süre	416.800	800

Her zaman diliminde, VGA'nın düşey senkronizasyon (DS) ve yatay senkronizasyon (YS) işaretlerinin alması gereken değerleri, Şekil 4.13'de gösterildiği üzere kontrol bloğu tarafından verilmektedir. Her zaman dilimin başlangıcında, bu dilimde geçmesi gereken süre yatay ve düşey tutuculara yüklenmekte ve her bir piksel saat işaretinde tutucu içerikleri bir azaltılmaktadır.



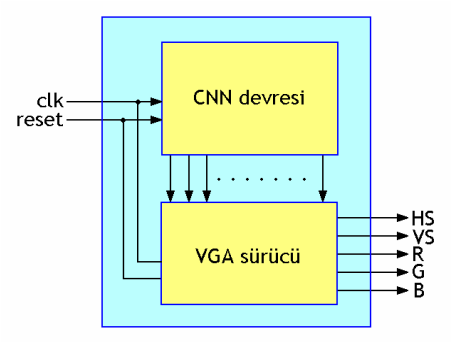
Şekil 4.13 : VGA sürücü devrenin yapısı.

VGA sürücü devresinin ihtiyaç duyduğu 25 MHz frekanslı piksel saat işareti, HYSA devresinden kullanılan frekans bölücünün aynından kullanılarak gerçeklenmiştir. 100 MHz'lik ana saat işareti 4'e bölünerek piksel saat işareti elde edilir.

Ekranın görüntü oluşturulan 640 x 480 piksellik yüzeyi, 32 x 32 piksellik 20 x 15 kareye ayrılmıştır. Bu karelere bölünen ekranda 5 x 5 karelik bir bölge, VGA sürücü tarafından görüntülenirken, HYSA devresinden aldığı girişlere göre renklendirilmektedir.

Ağdaki hücrelerin -1/+1 arasında salınmasından dolayı, her hücrenin x durum değişkeninin yalnızca işaret biti HYSA devresinin çıkışına aktarılmıştır. Dolayısıyla

HYSA devresinin 25 hücre için 25 bitlik çıkış dizisi oluşmuştur. Bu çıkış dizisi VGA sürücüsünün 25 bitlik giriş dizisine bağlıdır. HYSA devresi gerçek zamanlı olarak HYSA'i çalıştırmakta, her iterasyonda x durum değişkenlerinin işaretlerini çıkışa vermektedir. VGA sürücü sürekli olarak ekranı sürerken, ekranda hücreleri temsil eden bölgelerin renklendirmesini HYSA devresinden aldığı girişlerle oluşturmaktadır. Üst devre 100 MHz frekanslı ana saat işareti ve asenkron aktif 0 reset işaretini giriş olarak alır; VGA ekranı sürmek için gereken düşey senkronizasyon, yatay senkronizasyon, kırmızı, yeşil ve mavi işaret bitlerini çıkış olarak verir. Gerçeklenen üst devrenin yapısı Şekil 4.14'de verilmiştir.

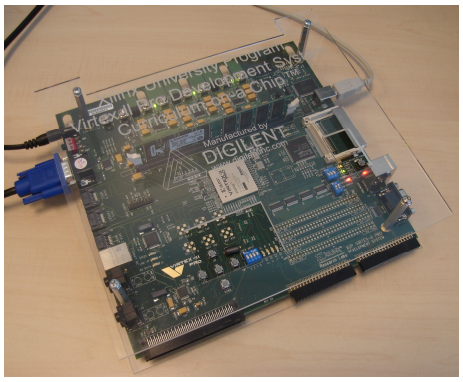


Şekil 4.14 : Gerçeklenen üst devre.

4.1.3 Devrenin çalıştırılması ve elde edilen sonuçlar

Devre gerçeklemeleri için kullanılan donanım Xilinx University Program Virtex-II Pro Development System isimli karttır. Kart üzerinde Xilinx firmasının XC2VP30-FF896 isimli Sahada Programlanabilir Kapı Dizisi (Field Programmable Gate Array, FPGA) bulunmaktadır. Sayısal HYSA devreleri bu FPGA üzerinde Verilog donanım tanımlama dili ile kodlanarak gerçeklenmiştir.

Devre, FPGA üzerinde bulunan 13696 lojik dilimin 10392'sini (%75) kaplamıştır. Bununla birlikte FPGA üzerinde 136 adet bulunan hazır 18 bitlik kombinezonsal çarpıcı bloğunun (MULT18x18) 25'ini kullanmaktadır. Kullanılan kartın fotoğrafı Şekil 4.15'de verilmiştir.



Şekil 4.15 : Xilinx University Program Virtex-II Pro Development System.

HYSA devresindeki bir hücre iterasyona başlama işaretinin ardından en geç 255 saat çevriminde tüm işlemlerini bitirmekte, iterasyonunu tamamlamakta ve tamamlandı işaretini vermektedir. HYSA devresinin en son iterasyonunu tamamlayan hücreden sonra ağı yeniden iterasyona sokması için 3 saat çevrimi geçmesi gerekmektedir. Dolayısı ile HYSA devresinin en uzun iterasyon süresi 258 saat çevrimidir. Buna karşın devrenin modelsim ile koşturulan 40000 iterasyonluk benzetimi ile HYSA devresinin ortalama iterasyon süresinin 183 saat çevrimi olduğu görülmüştür.

Hücre çekirdeğindeki toplayıcının denormalizatörü ve normalizatörünün ikisi de en geç 7 saat çevrimde işlemlerini tamamlamaktadır. Bu 7 saat çevriminin 5'i kaydırmada harcanmaktadır. Çizelge 4.3'e bakıldığında ancak 11 bit kaydırma yapılacağında 5 saat çevrimi geçmektedir. 0'dan 11'e kadar her bir kaydırma miktarının gelme olasılığını eşit ve 11'den fazla kaydırma gelme olasılığını 0-11 aralığındaki bir sayının olasılığına eşit kabul edersek bir denormalizatör veya normalizatör için gereken ortalama kaydırma süresi

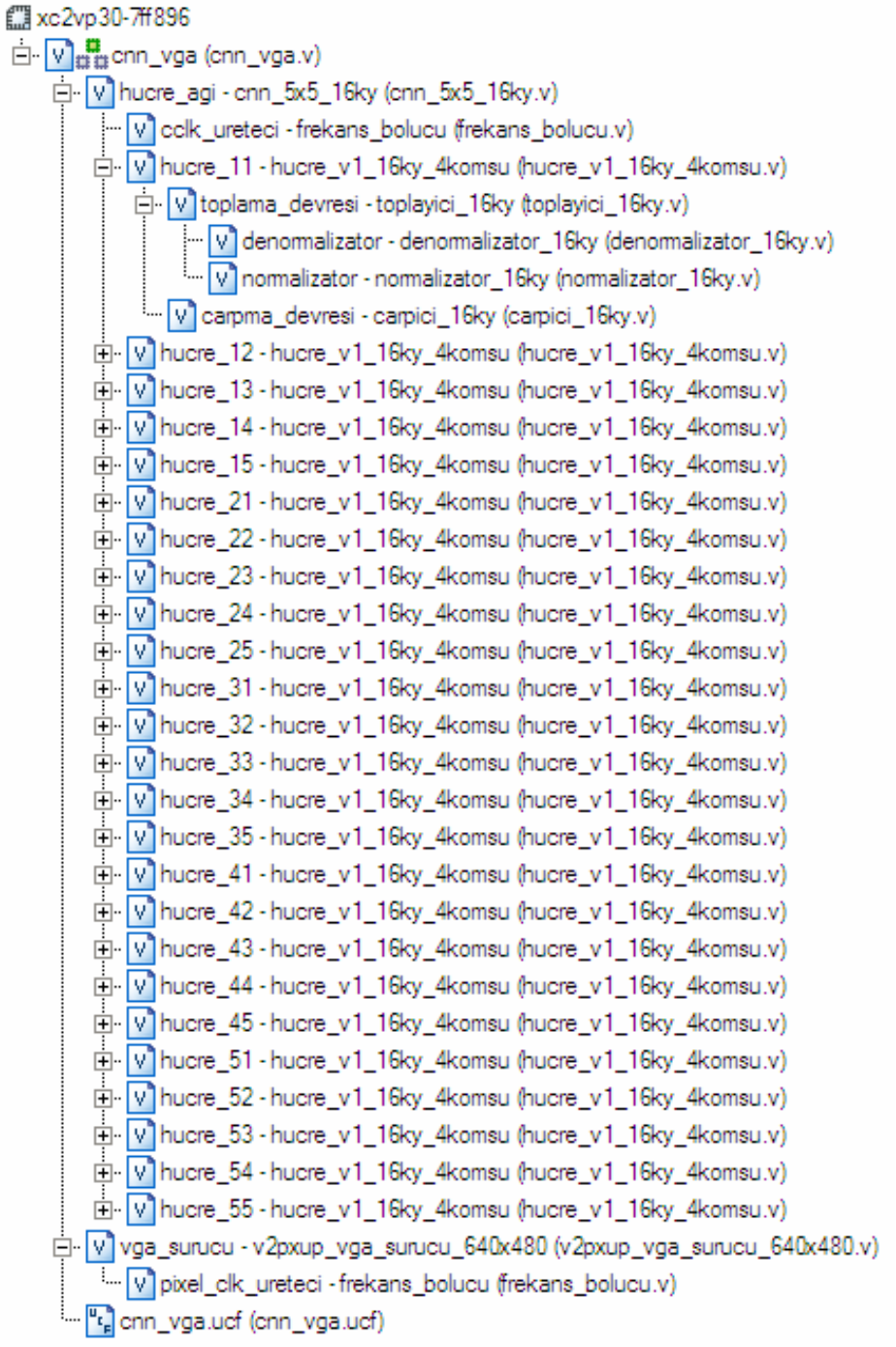
$$t_{ort} = \frac{0+1+2+1+2+3+2+3+4+3+4+5+0}{13} \cong 2{,}31 \text{ saat çevrimi olur}$$

En uzun iterasyon süresi 258 saat çevrimi iken, simülasyon sonucunda ortalama iterasyon süresinin 183 saat çevrimi çıkması denormalizatörün ve normalizatörün çalışma hızları ile ilişkilidir. Bir iterasyonda en fazla 10 denormalizasyon ve 10 normalizasyon yapılmaktadır. Bazı durumlarda bu miktarlar 9'a düşmektedir. İterasyonun en uzun süresi ile simülasyon süresi arasındaki 75 saat çevrimlik fark her bir denormalizasyon veya normalizasyonun en geç süreden 3,75 saat çevrimi daha kısa sürdüğü anlamında yorumlanmalıdır. Dolayısı ile simülasyon sonuçlarının

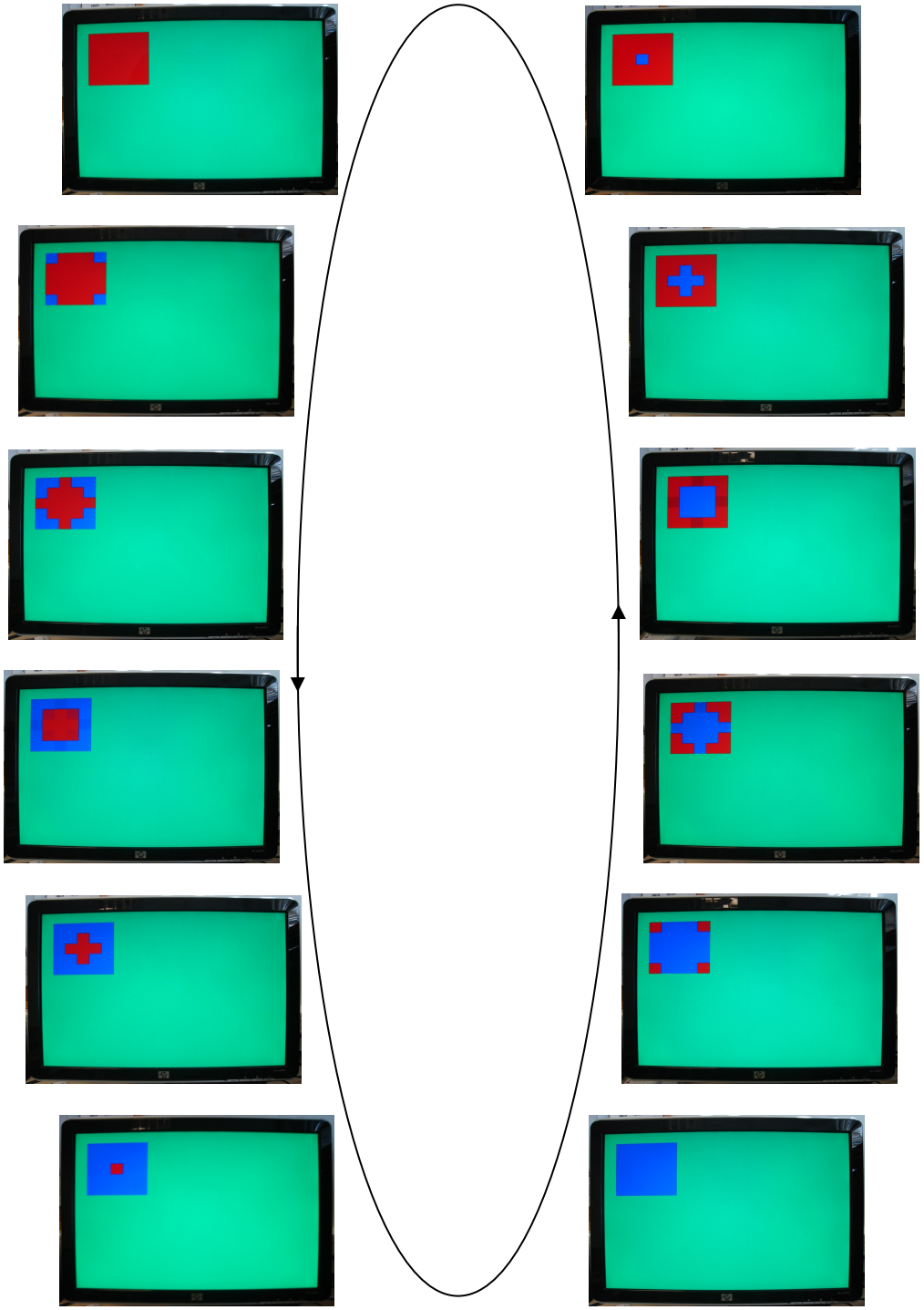
analizinden ortalama denormalizasyon ya da normalizasyonda kaydırma süresinin ortalama 1,25 saat çevrimi olduğu anlaşılmıştır.

Verilog kodları Xilinx ISE yazılımı ile sentezlenmiş ve gerçeklenmiştir. Bu işlemler sonucunda FPGA'yı konfigüre edecek bit dosyası üretilmiştir. Bu dosya içerisindeki bit dizisi Xilinx Impact ile FPGA'ya yollanmış ve FPGA konfigüre edilmiştir. Bu FGPA de HYSA'in gerçeklenmesinin son basamağıdır.

Xilinx ISE'nin sentezleme sonrası raporundan kapı seviyesi analizde en kötü yolun 4,797 ns gecikmeye sahip olduğu, dolayısıyla ana saat işareti frekansının 208,459 MHz olabileceği görülmüştür. Gerçeklenen devre ise FPGA kartı üzerindeki osilatörün ürettiği 100 MHz frekanslı saat işaretini ana saat işareti olarak kullanmıştır. HYSA devresinin yavaş çalışması için bu frekans 8192'ye bölünmüştür. Böylece HYSA devresi 12,207 KHZ saat işareti ile sürülmüş, 1 iterasyonu ortalama yaklaşık 15 ms sürmüştür. Bu hızla HYSA'daki hücrelerin x durum değişkenlerinin işaretleri VGA ekranda izlenmiştir. Ekranda x durum değişkeninin negatif değerleri mavi, pozitif değerleri kırmızı ile; hücrelere karşılık gelmeyen bölge yeşil ile gösterilmiştir. Gerçeklenen HYSA, gerçek zamanlı olarak VGA ekranda izlenmiş, fotoğraflanmış ve Şekil 4.17'de bu fotoğraflar gösterilmiştir. Şekil 4.16'da da gerçeklenen devrenin alt modüllerinin hiyerarşik ilişkisi gösterilmektedir.



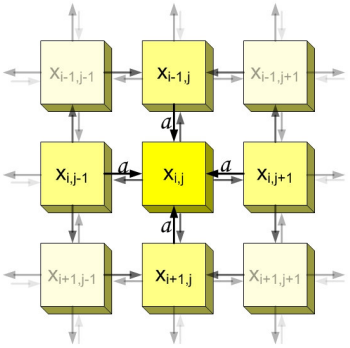
Şekil 4.16 : Tasarlanan 5x5 hücreli RO-HYSA devresinin modül ağacı.



Şekil 4.17 : Gerçeklene 5x5 hücreli RO-HYSA'nın ekrandan alınan fotoğraflar.

4.2 160 x 160 Hücreli RO-HYSA'nın Tasarımı ve Gerçeklenmesi

5 x 5 hücreli HYSA'nın FPGA'da gerçeklenmesinin ardından daha büyük boyutlu ağlar için çalışılmıştır. Hücre modelinde komşuluk ağırlıkları her yön için ayrı ayrı olmasına rağmen bu gerçeklemede, aynen bir önceki gerçeklemede olduğu gibi tek bir a parametresi ile ifade edilmiştir. Şekil 4.18'de komşuluklar ve a ağırlığı gösterilmektedir.



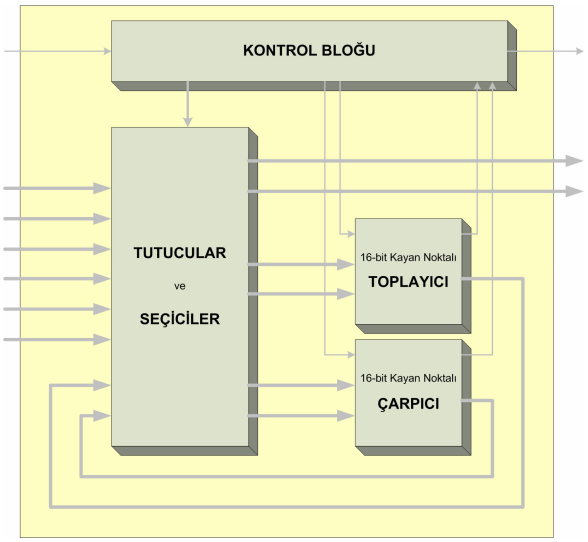
Şekil 4.18 : Tasarlanan 160x160 hücreli RO-HYSA'nın komşulukları.

Bu gerçeklemede (3. 11)'de g fonksiyonununda tanımlı λ parametresi 1 olarak, ayarlanmış, (3. 10)'daki σ parametresi de –ε olacak şekilde devre gerçeklenmiştir.

4.2.1 Hücre ve ağın yenilenmiş tasarımı: NPE ve CNPN blokları

Bu tasarımda, ilk tasarımdaki hücre yapısı temel alınmış, bellek elemanları da tasarıma alındığı için gereken değişiklikler yapılmıştır. Bu devrede bir öncekinde hücre olarak isimlendirilen devre bloğuna Nöral İşlem Elemanı (Neural Processing Element, NPE) ismi verilmiştir. Bir önceki tasarımda her bir hücre gerçekten ağ üzerinde yalnız bir hücreyi temsil etmekte iken, bu tasarımda her bir NPE ağ üzerinde ağın boyutu ile ilişkili olarak birden fazla hücreyi temsil etmekte, zamanda sıralı olarak temsil ettiği herbir hücrenin öykünmesini gerçekleştirmektedir. Şekil 4.19'da NPE modülünün basitleştirilmiş blok diyagramı verilmiştir. NPE modülü içerisinde bir toplayıcı, bir çarpıcı blok, bir tutucu ve seçici bloğu, bir de kontrol bloğu bulunmaktadır. Tüm gerçeklemelerdeki gibi bu devrenin de toplayıcı ve çarpıcı blokları 16 bit kayan nokta aritmetiği ile işlem yaparlar. İlk tasarlanan devrenin bir hücresinden, bu tasarımdaki bir NPE'ye geçerken ilk olarak aritmetik işlem bloklarında saptanan bazı hatalar giderilmiştir. Ardından ilk tasarımdaki hücrenin içerisinde dağınık olarak bulunan parametre değerlerini belirleyen kablolar, seçiciler, tutucular NPE'de tutucu ve seçici bloğu içerisinde toplanmıştır. NPE,

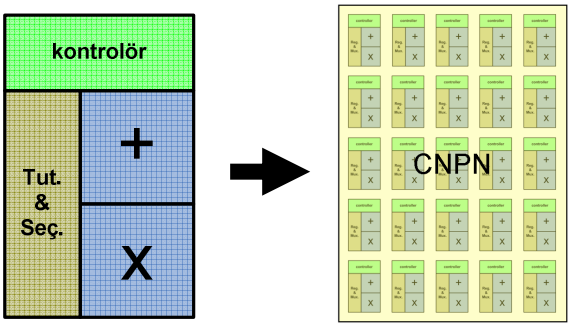
işlemlerine başlamadan önce dışarıdaki bellek elemanlarından gelen veriyi bu tutuculara saklar. Tutuculara saklanmakta olan verilerden doğru olanın işlem bloklarına gönderilmesi, kontrol bloğunun, tutucular ve seçiciler bloğundaki seçicilere doğru kontrol işaretini göndermesi ile olur. Seçilen veri işlem bloklarının giriş veriyollarında görülür ve kontrol bloğu işlem bloklarından birine çalışma komutunu yollar. Toplayıcı ya da çarpıcı, işlemini tamamlandığında çıkış veriyoluna sonucu yazar ve kontrol bloğuna tamamlandı işareti gönderir. Kontrol bloğu da tutucular ve seçiciler bloğunun uygun tutucusuna bu işlem sonucunun yazılması için gerekli kontrol işaretini seçicilere yollar.



Şekil 4.19 : 160x160 hücreli RO-HYSA için tasarlanan NPE'nin blok diyagramı.

İlk devredekine benzer yapıda, 5 x 5 adet NPE birleştirilerek bir hücresel nöral işlem ağı (Cellular Neural Processing Network, CNPN) kurulmuştur. CNPN, ilk tasarımdaki hücrelerin birleşiminden oluşan 5x5 boyutlu HYSA'ya karşılık gelmektedir. Şekil 4.20'de CNPN bloğu görselleştirilmiştir. CNPN'nin giriş ve çıkış veriyolları, barındırdığı tüm NPE'lerin giriş ve çıkış veriyollarının paralel olarak birleşiminden oluşur. Her bir NPE, durum değişkenlerini taşıyan 6 giriş 2 çıkış veriyoluna sahiptir. Birbiri ile komşulukta olan her NPE'nin ikişer girişleri ortaktır. Örneğin iki NPE'den soldakinin x durum değişken girişi sağdakinin batı komşu x durum değişkeni demek, benzer şekilde sağdakinini x durum değişkeni de soldaki için doğu komşu x durum değişkeni demektir. Bu sebeple CNPN içinde aynı veriyi taşıyan veriyolları birleştirilmiştir. Dolayısı ile CNPN'ye ait toplam 70 adet giriş ve 50 adet çıkış veriyolunun bulunduğu görülür. Bu veriyollarından herbiri 16 bit

genişliklidir ve x ya da y durum değişkenlerinden [k] ya da [k+1] zamanı için olanı taşırlar.



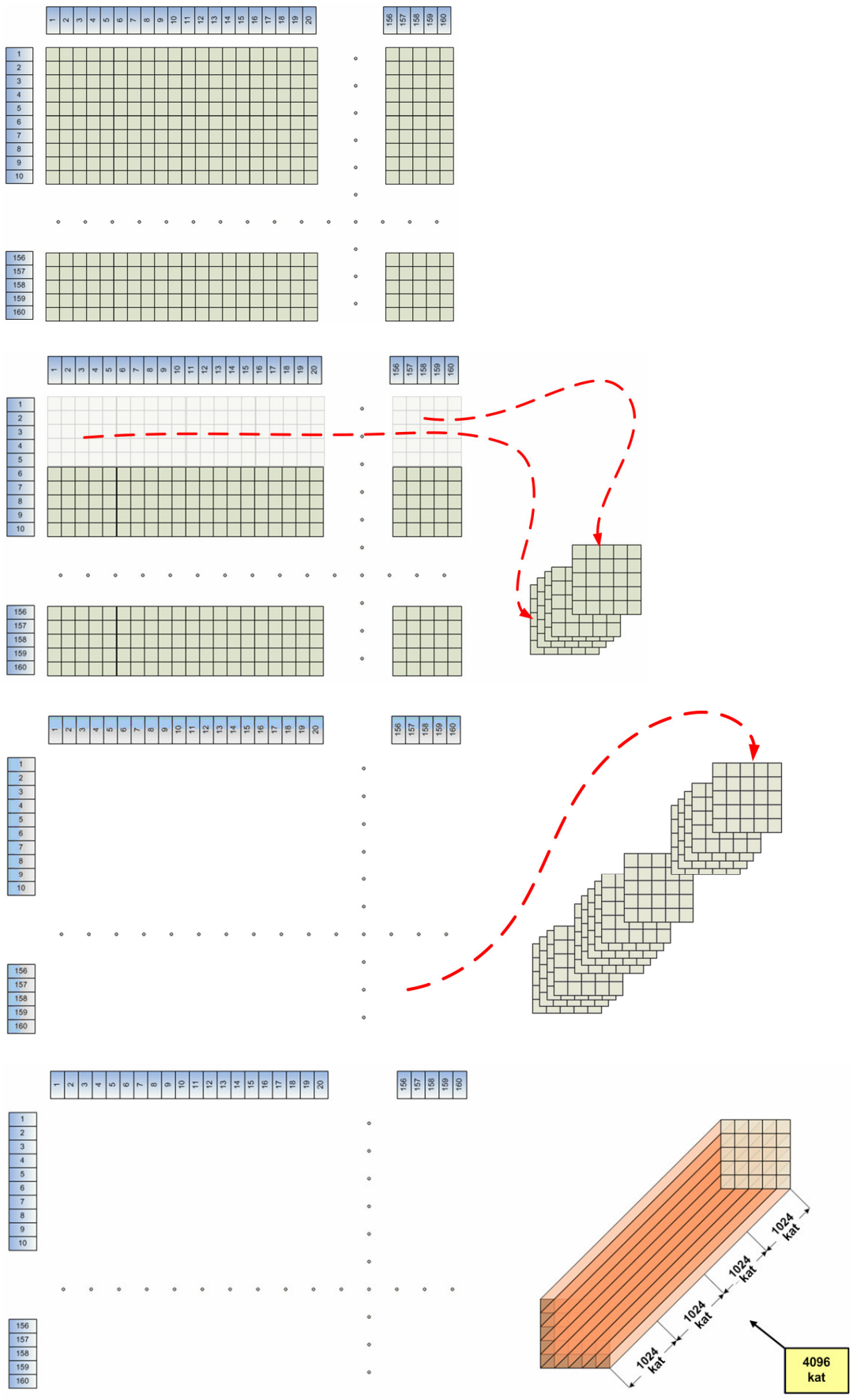
Şekil 4.20 : 25 adet NPE modülü 5x5 formunda dizilerek CNPN bloğunu oluşturur.

NPE'nin bir hücrenin öykünmesini gerçekleştirmesi yani hücreye ait işlemleri tamamlaması 149 ile 266 saat çevrimi sürmektedir. Ama yapılan ölçümler ile bu sürenin ortalama 182 saat çevrimi olduğu saptanmıştır. Değişken işlem süresinin sebebi Altbölüm 4.1.1 'de açıklanmıştır.

4.2.2 Ağ boyutunun büyütülmesi: Bellek Dizisi modülü

NPE, değişmeyen parametreleri, tutucular ve seçiciler bloğu içinde sabit kablo atamaları olarak saklamaktadır. Fakat, NPE ağ üzerindeki birden fazla hücrenin öykünmesini gerçekleştirdiği için hücrelere ait durum değişkenleri CNPN dışındaki belleklerde saklanır. CNPN'nin toplam 1920 bit genişlikli giriş çıkış veriyolu bellek elemanlarından tasarlanan bellek modülünden gereken verinin hızla okunabilmesi ve yazılabilmesi içindir.

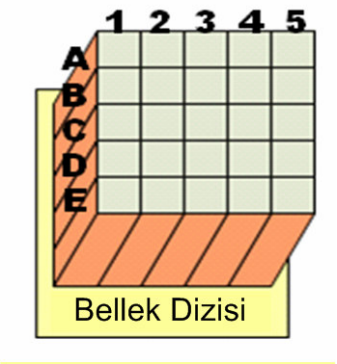
Kullanılan FPGA içerisinde dağınık olarak yerleştirilmiş, BlockRAM olarak adlandırılmış 272 KB kapasiteli bellek kaynağı mevcuttur. 160 x 160 hücrenin herbiri için bellekte x[k], y[k], x[k+1] ve y[k+1] değişkenlerinin değerleri tutulmalıdır. Her bir kelime 2 bayt olduğundan CNPN'in ihtiyaç duyduğu toplam bellek miktarı 200 KB'tır. Bu verinin nasıl bir bellek organizasyonu yapılarak saklandığını Şekil 4.21'i inceleyerek daha iyi anlamak mümkündür.



Şekil 4.21 : 160 x 160 boyutlu HYSA'nın dilimlenmesi ve dilimlerin sıralanması

160 x 160 boyutlu ağın üst satırı i=1 ve sol sütunu j=1 indisini almaktadır. Bütün ağ başlangıçtan yani sol üst köşeden itibaren her beş satırda bir ve her beş sütunda bir dilimlenmiştir. Her bir dilim 5 x 5 hücreden oluşur. 160x160'lık ağın tamamına dilim bazında bakıldığında ve yeniden indislendiğinde sol üst köşedeki başlangıç dilimi i_2=1, j_2=1 indisleri ile, sağ alt köşedeki son dilim ise i_2=32, j_2=32 indisleri ile indislenebilir. Başka deyişle ağ, herbiri CNPN boyutunda olan 32 x 32 dilime ayrılmıştır.

Dilimlemenin ardından bu dilimler sırasıyla üst üste dizilir. Sol üst köşedeki dilim dizinin en altında kalır. Bu sıra üst satırın sol başından sağına doğru ilerler ve satır sonunda bir alt dilim satırına geçer. Böylelikle en alt dilim satırına gelindiğinde yine sol baştaki dilimden devam edilir. Sağ alt köşedeki dilim de dilimler dizisinin son elemanı olarak dizinin en üstüne yerleşir. 32 x 32 dilimin toplam sayısı 1024'dür. 1024 dilim üst üste dizildikten sonra her hücre için saklanacak değişken sayısı hesaba katılır. Her hücre için 4 değişken saklanacağından, 1024 dilimli dizi 4 katı uzunluğuna çıkartılır. Bu yapı her biri 4096 katlı 25 tane binanın 5x5 formunda dizilmiş haline benzetilmelidir. İlk 1024 kat x[k]'ların barındığı, ikinci 1024 y[k]'ların, diğerleri de sırasıyla x[k+1]'lerin ve y[k+1]'lerin barındığı katlar olarak seçilir. Bu noktadan sonra herbir binanın bir bellek elemanına karşılık geldiği, her bellek elemanının 4096 kelime kapasiteli olduğu yani 12 bit adres yoluna sahip olduğu ve her kelimenin kullanılan sayı formatı sebebiyle 2 bayt olduğu sonucuna varılır. Şekil 4.22'de bellek dizisi modülü görselleştirilmiştir.



Şekil 4.22 : Bellek Dizisi Modülü.

Bu tasarımda FPGA içerisinde yukarıda açıklandığı gibi 4096 x 16 bit BlockRAM'den 25 adet gerçeklenmiştir. Bu alt devreye Bellek Dizisi Modülü ismi verilmiştir. Bellek dizisindeki 25 bellek elemanının adres yolları ortaktır. Yani bir anda tüm belleklerde aynı adresten veri okunur ya da yazılır. Belirlenen bir adresten

okunan 25 kelime CNPN'e sunulacak x[k] ya da y[k]'dır. Ayrıca, CNPN için dilime komşu dilimlerden de veri hazırlanması gerekmektedir. Bu 5 x 5 ağ diliminin sınır komşuluğundaki hücrelerin de kullanılmasındandır.

CNPN için Bellek dizisini adresleyen ve kontrol eden kısım her iki alt devrenin de dışında yer almaktadır Ana Sıralayıcı olarak isimlendirilen bu alt devre sırasıyla Çizelge 4.5'deki işlemleri gerçekleştirmektedir.

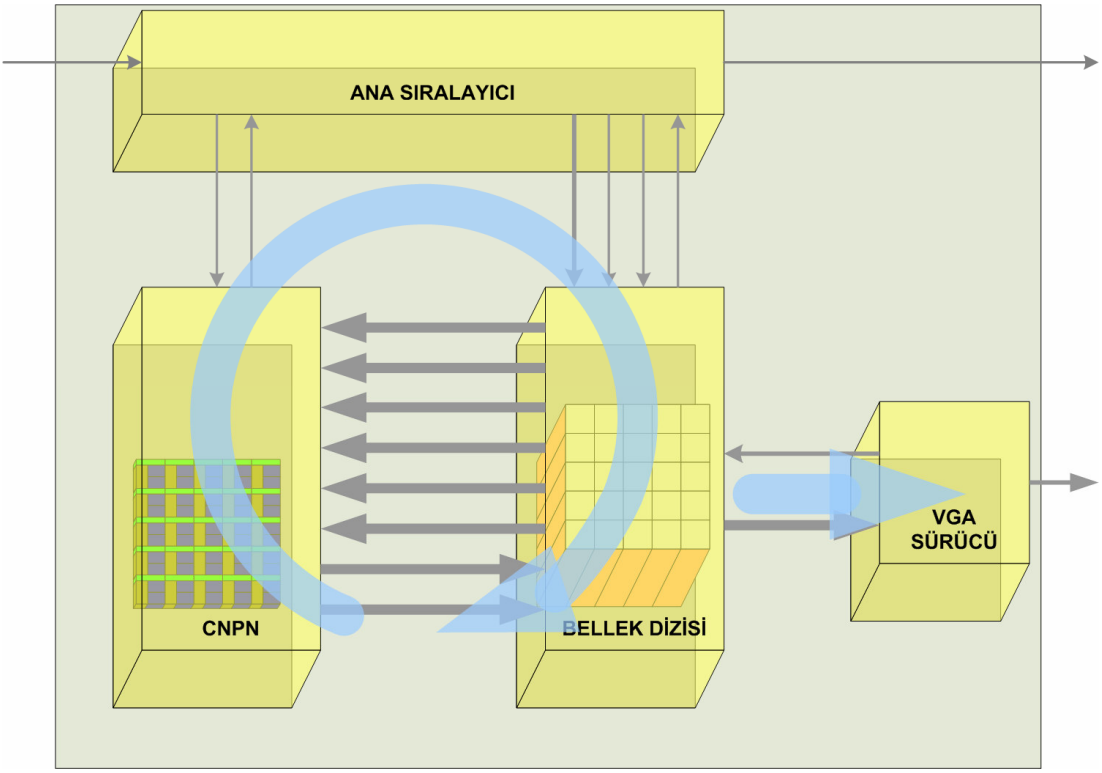
Çizelge 4.5 : Ana Sıralayıcı bloğunun işlemleri.

İşlem Sırası	İşlemler	Kontrol Edilen
1.1	Referans dilimin x[k] değişkenini adresle	Bellek Dizisi
1.2	Oku komutunu yolla	Bellek Dizisi
2.1	Referans dilimin y[k] değişkenini adresle	Bellek Dizisi
2.2	Oku komutunu yolla	Bellek Dizisi
3.1	Referans dilimin doğu komşusununx[k] değişkenini adresle	Bellek Dizisi
3.2	Oku komutunu yolla	Bellek Dizisi
4.1	Referans dilimin kuzey komşusununx[k] değişkenini adresle	Bellek Dizisi
4.2	Oku komutunu yolla	Bellek Dizisi
5.1	Referans dilimin batı komşusununx[k] değişkenini adresle	Bellek Dizisi
5.2	Oku komutunu yolla	Bellek Dizisi
6.1	Referans dilimin güney komşusununx[k] değişkenini adresle	Bellek Dizisi
6.2	Oku komutunu yolla	Bellek Dizisi
7	İşle komtunu yolla	CNPN
8	Referans dilimin x[k] değişkenini adresle	Bellek Dizisi
8.1	Yaz komutunu yolla	Bellek Dizisi
9.1	Referans dilimin y[k] değişkenini adresle	Bellek Dizisi
9.2	Yaz komutunu yolla	Bellek Dizisi

Oku ve yaz komutlarından sonra Bellek Dizisi, İşle komutundan sonra da CNPN, gerekli işlemleri tamamlayıp Ana Sıralayıcıya tamamlandı sinyali yollarlar ve Ana Sıralayıcı bir sonraki işleme geçer. Bir ağ dilimin CNPN tarafından öykünmesi için gereken veri 70 kelimedir. Bellek Dizisinin çıkış veriyolu ve CNPN'nin giriş veriyolu belirtildiği gibi 70 x 16 bit genişliğindedir. Öykünme sonunda üretilen veri ile 50 kelimedir. Benzer şekilde CNPN'nin çıkış veriyolu ve Bellek Dizisinin giriş veriyolu 50 x 16 bit genişliğindedir.

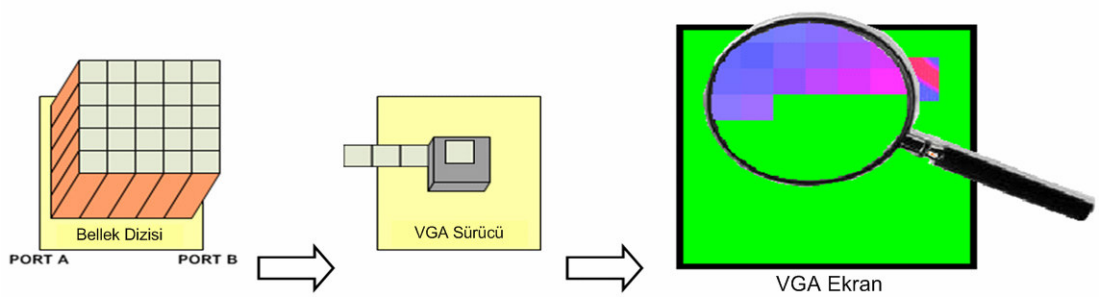
Ana sıralayıcı devre çalışmaya başladığı anda sol üstteki dilimin öykünmesini sağlar ve dilim sırasında ilerler. Sağ alttaki dilimden sonra tekrar sol üst başa döner. Sol üst başa tekrar gelişinde 160 x 160 hücreli HYSA için bir iterasyon adımı ilerlenmiş olur. Bu iterasyon boyunca x[k] ve y[k]'lar okunarak üretilen veri x[k+1] ve y[k+1] adreslerine yazılırken takip eden iterasyonda veri x[k+1] ve y[k+1] adreslerinden

okunur ve x[k], y[k] adreslerine yazılır. Takip eden ikinci iterasyon ilk baştaki iterasyon gibi olacaktır. Bu şekilde çalışmak için gereken adreslemeyi yapmak da Ana Sıralayıcı'nın görevidir. Ana Sıralayıcı, Bellek Dizisi ve CNPN arasında oluşan işlem çevrimi Şekil 4.23'de üzerine kapanan ok ile gösterilmiştir. Aynı şekilde ikinci bir iş daha gerçekleştirilmektedir. Bu da Bellek Dizisindeki verinin sürekli olarak gecikmesiz biçimde VGA monitöre basılması işidir.



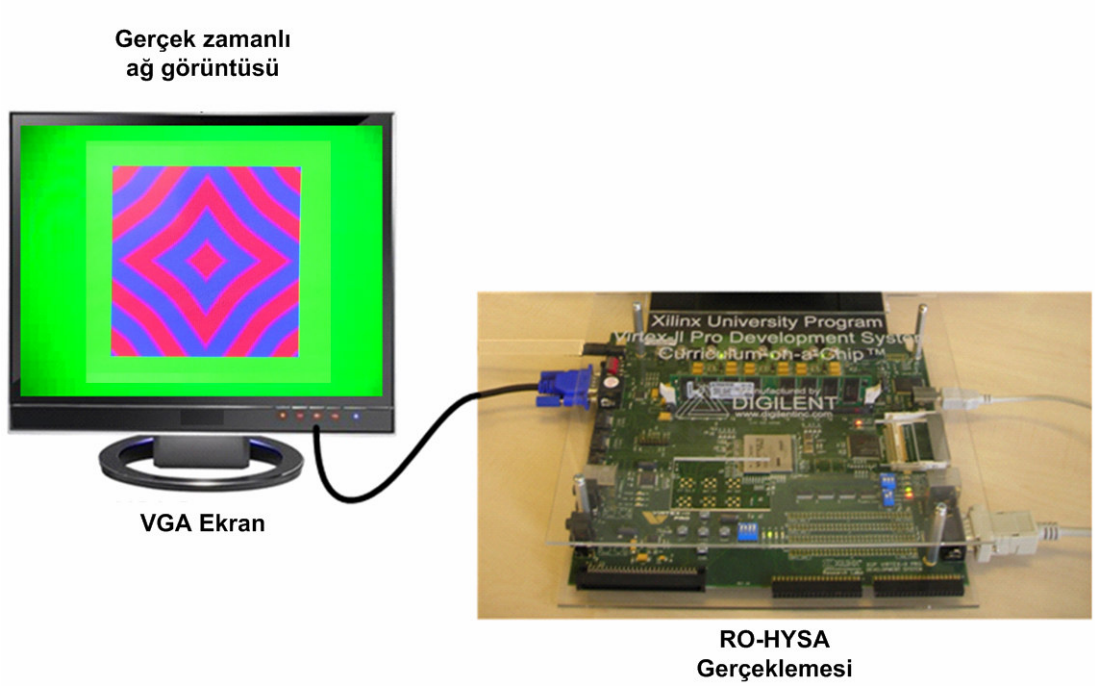
Şekil 4.23 : 160x160 hücreli RO-HYSA'nın blok diyagramı ve iş çevrimleri.

Şekil 4.23'de gösterilen VGA sürücü ilk devrede tasarlanan alt devredir. Ağ boyutunun büyümesi ile gereken değişiklikler sürücü modüle uygulanmıştır. Bu sürücü ağ görüntüsünün ekranın ortasında ve 2 katı büyütülmüş olarak gösterir. Bir hücre 2x2 piksel ile ifade edilir. Şekil 4.24'de VGA sürücünn Bellek Dizisinden veri okuyuşu görselleştirilmiştir. VGA sürücü Bellek Disizindeki bellek elemanlarının B portunu kullandığı için, ağ öykünmesi işlemlerine engel olmaz. Bu sayede CNPN ile VGA sürücü işlevlerini aynı veriyi kullanarak birbirini asla engellemeden sürdürürler.



Şekil 4.24 : 160x160 hücreli RO-HYSA'nın VGA sürücüsünün çalışması.

Şekil 4.25'de de 160x160 hücreli RO-HYSA'nın gerçeklendiği FPGA kartı ve devrenin VGA monitörü birlikte görselleştirilmiştir. VGA monitör üzerinde gösterilen oto dalga bu devre ile elde edilmiş gerçek bir fotoğraftır.



Şekil 4.25 : 160x160 hücreli RO-HYSA'nın monitör bağlantısı ve elde edilen dalga.

4.3 128 x 128 Hücreli Programlanabilir RO-HYSA'nın Tasarımı ve Gerçeklenmesi

128 x 128 boyutlu hücresel sinir ağının gerçeklenmesinde, çalışmanın bir önceki sürümü olan 160 x 160 boyutlu ağ temel alınmıştır. Önceki tasarımın çözdüğü denklem takımında değişiklikler yapılmıştır. Bunun temel sebebi, ağ üzerinde yayılan uzay-zaman dalgalarının, ağın istenen herhangi bir yerinden kaynaklanabilmesini ve ağ üzerinde bazı hücrelerin dalgaları iletmemesini sağlayabilmektir. Bu bölümde anlatılan üçüncü RO-HYSA devre tasarımında ağ boyutunun küçülmesinin sebebi, her bir hücre için 160x160 hücreli ağdakinden daha fazla miktarda verinin belekte tutulmasının gerekmesi ve bellek miktarının kısıtlı oluşudur. Bu değişiklikler ile gerçeklenen ağ beklenen işlevine yaklaşmıştır. İletmeyen hücrelerin belirli bir dizilimi ile dalganın yayılacağı farklı geometride uzaylar oluşturulabilir hale gelmiştir. Kaynak hücrenin serbestçe belirlenmesi ile bu uzayda istenen noktadan dalganın oluşması sağlanmıştır. Ağ üzerinde aktif hücreler (3. 10)'daki hücre modelini gerçekleştirirler fakat işaretlenmiş pasif hücreler (4. 3)'de verilen pasif modeli gerçekleştirilirler.

$$x_{i,j}[k+1] = x_{sabit}$$
$$y_{i,j}[k+1] = y[k] \qquad \textbf{(4. 3)}$$

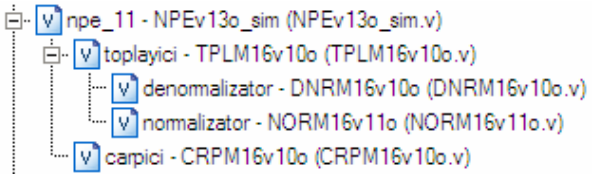
Yukarıdaki x değişkenine ait durum denklemi pasif hücrenin sabit değerde kaldığını ve böylece dalga iletimi yapmadığını göstermektedir. Aktif hücrelereden seçilen birine uygulanan salınım aralığından daha yüksek bir u girişi, hücrenin dalga kaynağı olarak davranmasını sağlamaktadır. Kaynağın etkisi u'nun genliğine bağlıdır. Tasarlanan ilk devrede, I fonksiyonuna tüm yönlerdeki komşuların ağırlıkları eşitken ikinci tasarımde (3. 10)'da verildiği gibi tüm yönler için farklı ağırlık tanımlıdır.

4.3.1 Yeni NPE ve yeni CNPN modülleri

Tasarlanan HYSA'da hücrenin işlevini yerine getiren modüle NPE (nöral işlem elemanı) denilmişi. 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.

NPE bir iterasyon için işleme başlamadan önce değişken girişlerine ve katsayı girişlerine uygun işaretler verilmelidir. NPE önceki tasarımda olduğu gibi bir toplayıcı ve bir çarpıcı alt modüle sahiptir. Çarpıcı modülün çarpan girişlerini süren iki adet 16 bit genişlikli yığın NPE devresinde bulunmaktadır. Bu yığınlara veriler iterasyonun başlangıcında paralel olarak atanır. İterasyon için işlemler sürerken yığının en üstündeki bir değer okunduğunda diğer tüm değerler birer kademe üste kaydırılır. Bu yapı bir önceki tasarıma göre NPE'nin daha az şartlı tutucu ataması ile tasarlanmasını, böylelikle flipflop ve kapı elemanı sayısının dengelenerek NPE modülünün optimize edilmesini sağlamıştır.

Optimizasyon için bir geliştirme de Verilog kodlarının yazımında yapılmıştır. Kullanılan sentezleyici yazılım, bir modül içerisinde tüm tutucu atamalarının ayrı süreçler (ayrı 'always' yapıları) ile yazılması halinde daha az kapı elemanı ile devreyi sentezleyebilmektedir. Bu özellik kullanılarak NPE modülünü tanımlayan Verilog kodu baştan tekrar yazılmıştır. Aynı değişiklik toplayıcı ve çarpıcı modüllerde de yapılmıştır. Bu modüller hakkında detaylı bilgi kendi başlıkları altında verilmektedir. NPE modülünün giriş çıkışları Çizelge 4.6'da listelenmiştir. Şekil 4.26'da NPE modülünü oluşturan alt modüllerin hiyerarşik yapısı gösterilmeltedir.



Şekil 4.26 : NPE'nin hiyerarşik yapısı

Devre, barındırdığı her bir NPE'yi ilgili nöron verisi ile besleyecek üst modüller olacak şekilde tasarlanmıştır. Bu üst modüller NPE'nin değişken (x_sag, x_ust, x_sol, x_alt, xk, yk, u), ağırlık (alfa_sag, alfa_ust, alfa_sol, alfa_alt, a, b, c, d) ve diğer parametre (limit, m, T, pasif_degeri) girişlerine uygun işaretleri verir. Bu işaretlerin girişlere uygulanmasının ardından NPE'nin x_izin_giris'ine lojik 1 darbesi uygulanır. Bu darbe ile NPE içinde iterasyonu yürüten durum makinesi çalışır. NPE önce x[k+1] değerini hesaplar ve çıkışa verir. Bunu üst modül ya da modüllere yine 1 saat çevrimlik lojik 1 darbesini x_izin_cikis çıkışından vererek bildirir. Üst devre çıkıştaki değeri okuduktan sonra y_izin_giris'ine 1 darbesi uygulayarak NPE'nin işlemlere devam etmesini sağlar. Bu sefer NPE, ilk başta verilen değişken, ağırlık ve parametreleri kullanmayı sürdürerek y[k+1] değerini hesaplar ve çıkışa verir. Yine

bunu y_izin_cikis'tan lojik 1 darbesi göndererek üst modüllere bildirir. NPE'nin giriş çıkışları Çizelge 4.6'da verilmiştir.

Çizelge 4.6 : NPE modülünün giriş çıkışları.

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

Yeni NPE'de (3. 10)'daki denklem takımını çözen algoritma Çizelge 4.7'de verilmiştir. Çizelge 4.1'deki algoritmadan farklılıklar göstermektedir. NPE'nin algoritmayı bir kez koşması ile ilgili nöronun bir iterasyonu yapılmış olur.

Çizelge 4.7 : 128x128 RO-HYSA'nın NPE'sinde bir iterasyonda gerçekleşen işlemler.

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

NPE tek başına kullanılarak da büyük boyutlu HYS ağlarının simülasyonunu yapabilirdi. Fakat bu tasarım 16 adet NPE barındırmakta ve tüm NPE'ler paralel çalışmaktadır. Böylelikle devrenin hızı 16 katına çıkartılmıştır. Bu paralel yapının detaylı bilgisi ileride yeni CNPN modülü olarak açıklanmaktadır.

NPE'nin tasarımında son olarak 1.3o sürümüne ulaşılmıştır. Önceki tasarımda karşılığı olan bir hücre ile karşılaştırılması sonucu Çizelge 4.8'deki sonuçlar elde edilmiştir. Bu değerler kullanılan Xilinx XC2VP30 FPGA'sı içindir.

Çizelge 4.8 : İlk tasarımdaki hücre ile yeni NPE'nin karşılaştırması.

	Eski Hücre Modülü	NPE v1.3o modülü
Lojik Dilim Sayısı (tamamına oranı)	417	510
FF Sayısı (tamamına oranı)	387	624
LUT Sayısı (tamamına oranı)	688	918
MULT18x18 Sayısı(tamamına oranı)	1	1
En Uzun İterasyon Süresi	258 saat çevrimi	271 saat çevrimi
En Kısa İterasyon Süresi	138 saat çevrimi	139 saat çevrimi
Ortalama İterasyon Süresi	183 saat çevrimi	177 saat çevrimi

NPE modülünün eski hücreye göre büyümesinin sebebi, gerçekleştirdiği işlemlerin karmaşıklaşmış ve çoğalmış olmasıdır. Eski hücre mimarisi ile bu işlemleri gerçekleştirilmesi için tasarıma ilk başlandığında çok daha büyük devreler ile karşılaşılmış, optimizasyonlar ile devre boyutu bu noktaya kadar düşürülebilmiştir.

Çizelgedeki LUT, FPGA üzerinde kombinesonsal kapı elemanlarının gerçeklendiği bellek tabanlı 'look-up table'dır. FF ise flip-flop anlamındadır. MULT18x18, kullanılan FPGA'nın içinde bulunan hazır 18 bitlik çarpıcı donanım elemanıdır.

NPE'ler birer çarpıcı ve toplayıcı modül içerirler. Bu modüller eski hücrede de bulunmaktadır. Modüllerin daha az alana sığmaları için her bir tutucuya ait atamaları ayrı süreçler şeklinde yazılmıştır. Ayrıca tasarım sürecinde ortaya çıkan hatalar giderilmiş, gereksiz kısımlar atılmış, performans arttırılmıştır. Çizelge 4.9 ve Çizelge 4.10'da eski aritmetik modüller ile yeni aritmetik modül sürümlerinin, CRPM16v1.0o ile TPLM16v1.0o'nun, karşılaştırmaları verilmektedir.

Çizelge 4.9 : Çarpma modüllerinin karşılaştırması.

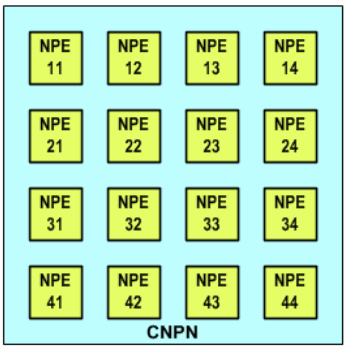
	Eski Çarpma Devresi	CRPM16 v1.0o Modülü
Lojik Dilim Sayısı (tamamına oranı)	36	32
FF Sayısı (tamamına oranı)	38	37
LUT Sayısı (tamamına oranı)	62	62
MULT18x18 Sayısı(tamamına oranı)	1	1
Çarpma Süresi	2 saat çevrimi	2 saat çevrimi

Çizelge 4.10 : Toplama modüllerinin karşılaştırması.

	Eski Toplama Devresi	TPLM16 v1.0o Modülü
Lojik Dilim Sayısı (tamamına oranı)	154	127
FF Sayısı (tamamına oranı)	147	145
LUT Sayısı (tamamına oranı)	261	229
En Uzun Toplama Süresi	21 saat çevrimi	20 saat çevrimi
En Kısa Toplama Süresi	9 saat çevrimi	8 saat çevrimi
Ortalama Toplama Süresi	13.5 saat çevrimi	11.5 saat çevrimi

Bu tasarımda NPE'lerden 16 adet kullanılarak 4 x 4 boyutlu nöral işlemci seti oluşturulmuştur. Bu 16 yeni NPE, ikinci devredeki gibi CNPN modülü olarak bir araya getirilmiştir. Şekil 4.27'de CNPN'ye ait blok diyagram ve Şekil 4.28'de hiyerarşik yapı gösterilmektedir.

CNPN'nin iki temel işlevi vardır. CNPN'nin ilk temel işlevi, girişlerine uygulanan değişken, ağırlık ve parametre işaretlerini alt modülleri olan NPE'lere dağıtmak ve NPE'lerin çıkışlarını üst modüllere aktarmaktır. Bunun için NPE'ler uygun iç ve dış bağlantılar ile CNPN içine yerleştirilmiştir. CNPN'nin tüm NPE'lere ortak ulaştırdığı ağırlık (alfa_sag, alfa_ust, alfa_sol, alfa_alt, a, b, c, d) ve parametre (limit, m, T, pasif_degeri) girişleri ile her hücreye özel ulaştırdığı değişken (x_sag, x_ust, x_sol, x_alt, xk, yk, u) girişleri mevcuttur. Bu girişler hiyerarşide üstteki ilgili modüller tarafından sürülmektedir.



Şekil 4.27 : Yeni CNPN'nin blok diyagramı

Şekil 4.28 : Yeni CNPN'nin hiyerarşik yapısı

İkinci temel işlevi ise iterasyona başlama ve iterasyonu sonlandırma ile ilgilidir. Üst modüllerden gelen iterasyon başlangıç işaretinin tüm NPE'lere eş zamanlı dağıtılması CNPN'nin iç bağlantıları ile yapılır. NPE'lerin iterasyon sürelerinde, içerdikleri TPLM16 modüllerinden kaynaklanan bir belirsizlik mevcuttur. Bu yüzden CNPN, aynı anda başlayan NPE'lerden, işlemlerini en son tamamlayan NPE'yi bekler. Bunun için NPE'lerin x[k+1] değerlerini ve y[k+1] değerlerini hesapladığını gösteren toplam 32 adet bayrak (flipflop) CNPN'de bulunmaktadır. CNPN'nin x_hesapla ve y_hesapla isimli iki adet kontrol girişi vardır. Bu girişlerden gelen bir saat çevrimlik lojik 1 darbesi ile CNPN çalıştırılır. İterasyona başlandığında CNPN içinde tüm bayraklar indirilir. Bunun için üst modülden CNPN'ye x_hesapla işareti gelmelidir. Başladıktan sonra herhangi bir NPE'den x_izin_cikis darbesi alındığında ilgili bayrak kaldırılır. Tüm x_izin_cikis bayrakları kaldırıldığında en geç NPE de x[k+1] değerini hesaplamış demektir. Bu andan sonra CNPN üst modüle giden x_hesaplanıyor kontrol çıkışını lojik 0'a çeker ve beklemeye geçer. Üst modülün CNPN'ye y_hesapla darbesi göndermesi ile tüm NPE'lere y değerlerini hesaplaması için ortak bir başlatma darbesi yollanır. Yine x bayraklarında olduğu gibi her bir NPE'den gelen y_izin_cikis darbesi ile karşılığında CNPN'de bulunan bayrak kaldırılır. Tüm y bayrakları kaldırıldığında bütün NPE'ler iterasyonlarını tamamlamıştır. CNPN bu durumda üst modüllere 4x4'lük ağ parçası için iterasyonun tamamlandığını, y_hesaplanıyor kontrol çıkışını lojik 0'a çekerek haber verir. CNPN'nin bu işlevi NPE'lerin senkronizasyonu anlamına gelmektedir ve paralel çalışma için şarttır.

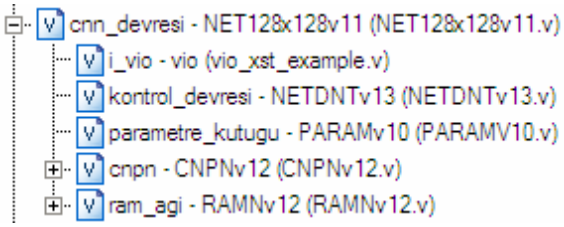
Tasarımın son halinde CNPN'nin v1.2 sürümü yazılmıştır. CNPNv12'nin işlem süresi en geç NPE'nin işlem süresinden 8 saat çevrimi uzundur. Çizelge 4.11'de CNPNv12'nin başarımı gösterilmektedir.

Çizelge 4.11 : CNPN modülünün başarımı.

	CNPN v1.2 Modülü
Lojik Dilim Sayısı (tamamına oranı)	8192
FF Sayısı (tamamına oranı)	10023
LUT Sayısı (tamamına oranı)	14745
MULT18x18 Sayısı(tamamına oranı)	25
En Uzun Toplama Süresi	278 saat çevrimi
En Kısa Toplama Süresi	147 saat çevrimi
Ortalama Toplama Süresi	185 saat çevrimi

4.3.2 Dalga bilgisayarı çekirdeği olarak organize edilen tasarım

Altbölüm 4.3.1 'de anlatılan CNPN modülünün yanında, ikinci devrede olduğu gibi bir Bellek Dizisi modülü tasarlanmış, ayrıca yeni olarak Kontrol Devresi, Parametre Kütüğü de eklenerek, hücrenin aktif ya da pasif olarak işaretlenmesine göre (3. 10)-(3. 12) ya da (4. 3) ifadesini gerçekleştiren, programlanabilir bir HYSA öykünleyici sayısal devre tasarlanmıştır. Tasarımın bu belirtilen alt bloklarından oluşan öykünleme kısmına Dalga Bilgisayarı Çekirdeği adı verilmiştir. Dalga Bilgisayarı Çekirdeği, ikinci tasarımın RO-HYSA öykünleme işinin yapan temel bloğudur ve kısaca Çekirdek olarak bahsedilecektir. Şekil 4.29'da Çekirdeğin hiyerarşik yapısı ve Çizelge 4.12'de başarım ve kaynak tüketim değerleri sunulmuştur.



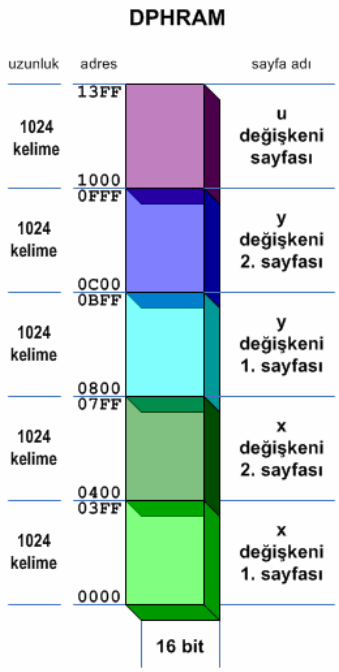
Şekil 4.29 : Çekirdeğin hiyerarşik yapısı.

Çizelge 4.12 : Çekirdeğin başarımı ve kaynak kullanımı.

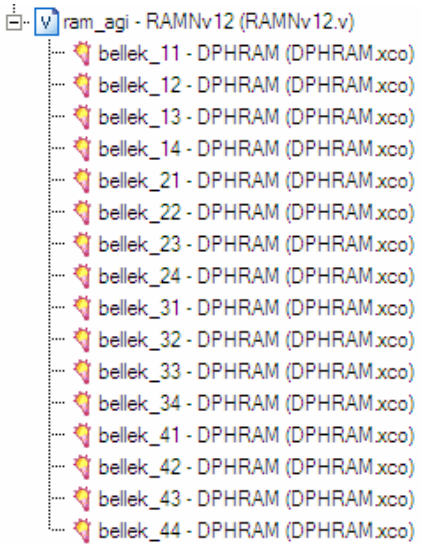
	HYSA Öykünleyici Çekirdek
Lojik Dilim Sayısı (tamamına oranı)	10731
FF Sayısı (tamamına oranı)	12296
LUT Sayısı (tamamına oranı)	17825
Block RAM Sayısı (tamamına oranı)	80
MULT18x18 Sayısı (tamamına oranı)	25
Ortalama Bir İterasyon Süresi	210.080 saat çevrimi

Çekirdek, toplam 16,384 norönu öykünleyebilmektedir. Öykünlenecek ağ boyutu 128x128'den daha ufak ise çekirdeğin ilgili parametreleri ayarlanarak bu gerçekleştirilebilir. Her bir noröna ait x[k], x[k+1], y[k], y[k+1] ve u değişkenleri bellekte tutulur. Her bir değişken 16 bit genişliğindedir ve ağın toplam bellek gereksinimi 1,310,720 bittir (160KB). Bunu saklamak için FPGA içerisinde bulunan, toplam kapasitesi 272 KB olan BlokRAM'lerden kullanılmıştır. Bellek elemanları, ikinci tasarımda olduğu gibi Bellek Dizisi modülü içerisinde bir araya getirilmiş ve tasarıma eklenmiştir.

Bu gerçeklemenin en önemli parçalarından biri Bellek Dizisi modülüdür. Bu modül CNPN modülündeki gibi birbirine paralel dizilmiş 16 adet RAM bloğundan ve bu RAM bloklarını kontrol eden bir kısımdan meydana gelmektedir. Bu RAM blokları FPGA tümdevresinde bulunan BlockRAM'lerden meydana getirilmiştir. Kullanılan FPGA'da her bir 2 KB kapasiteli 136 adet BlockRAM bulunmaktadır. RAM ağını meydana getirmek için ilk olarak mevcut BlockRAM'lerden oluşturulan dual-port hücresel RAM (DPHRAM) isimli, bir RAM komponenti tanımlanmıştır. Şekil 4.30'da DPHRAM komponenti gösterilmektedir. Şekil 4.31'de ise Bellek Dizisinin hiyerarşik yapısı verilmiştir. DPHRAM Kapasitesi 16 bit uzunluklu 5120 kelimedir. A portundan okuma ve yazma yapılabilir. B portundan ise sadece okuma yapılabilir. Adres veriyolu genişliği 13 bittir.

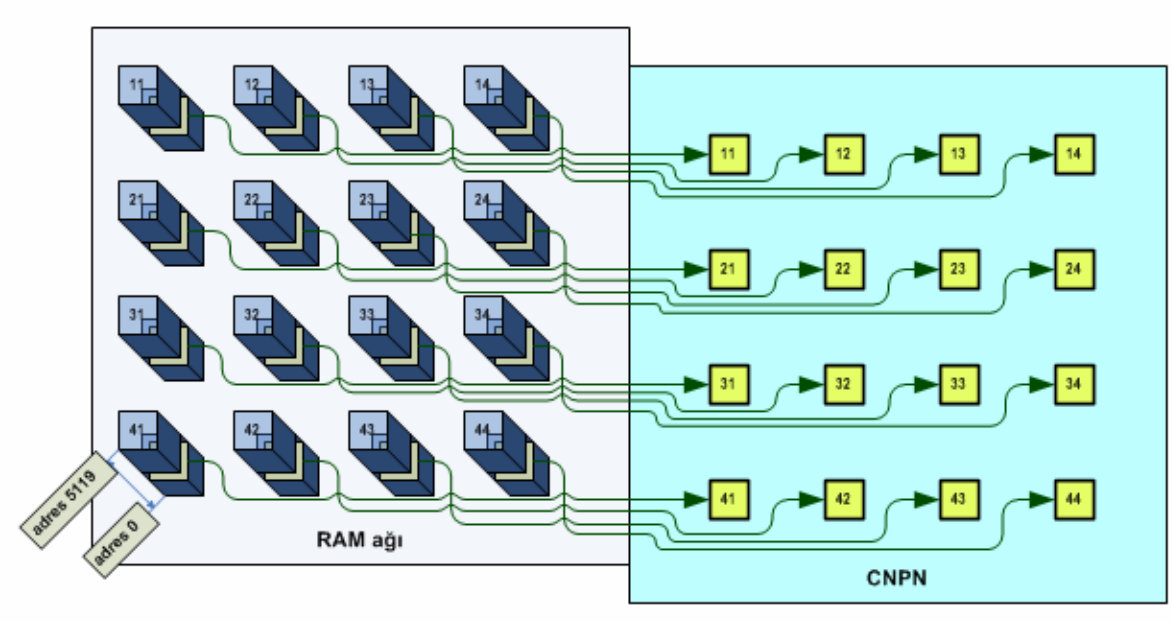


Şekil 4.30 : DPHRAM'in bellek haritası



Şekil 4.31 : Bellek Dizisi modülünün hiyerarşik yapısı

Bellek Dizisindeki her bir DPHRAM, CNPN modülündeki her bir NPE ile eşleşmektedir. Şekil 4.32'de bu yapı görselleştirilmiştir.

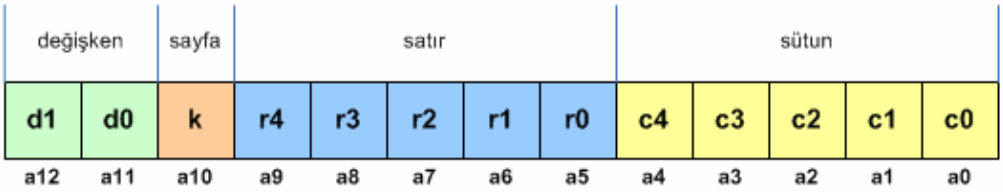


Şekil 4.32 : RAM ağı ile CNPN'nin eşleştirilmesi

RAM ağı modülünün çalışması paralel modülü olan kontrol devresinin kontrolündedir. Kontrol devresi RAM ağından HYSA'nın 4x4'lük bir bölümüne ait x, y, u değişkenlerinden bir seti okumasını ve çıkış portlarına yazmasını, Giriş portlarındaki verileri HYSA'nın 4x4'lük bir bölümüne ait x, y veya u değişkeni olarak kaydetmesini isteyebilir.

Bahsedilen veri akışının gerçekleştiği port DPHRAM komponentlerinin A portlarıdır. Kontrol devresi tüm A portları için ortak bir adres yolunu kullanarak aynı anda 16 DPHRAM'den 16 adet kelime okunmasını ya da yazılması sağlar. A portlarının ortak adres yolu 13 bit genişliğindedir. Gerçeklenen ağ 128 x 128 boyutundadır ve her seferinde 4x4'lük bir kısmı öykünlenmektedir. Aynı zamanda tüm ağ 4x4'lük parçalara ayrılarak DPHRAM dizisine yerleştirilmiştir. Şekil 4.32'de gösterildiği gibi, DPHRAM bir 5120 katlı bir bina, RAM ağı da bu binalardan oluşmuş bir site olarak düşünülmelidir. Ortak A portu adresi tüm sitede aynı katı ifade eder. Bu şekilde RAM ağında, belirli bir adres değeri için 16 değere aynı anda erişilir.

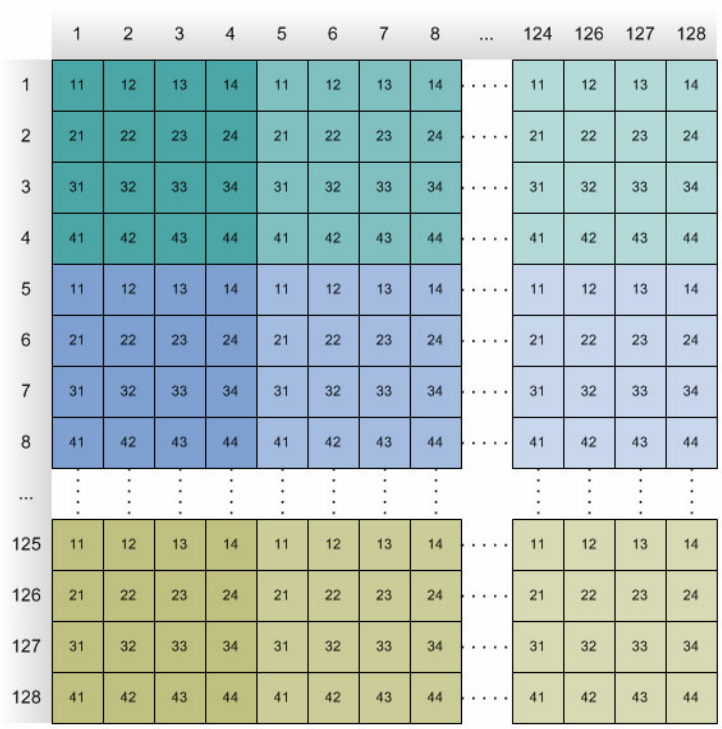
Şekil 4.33'de gösterilen A portu adresinin d1 ve d0 bitleri değişkeni, k biti de sayfayı belirtir. r5-r0 bitleri 4x4'lük parçanın 128x128'lik ağ içerisinde bulunduğu satırı, c4-c0 bitleri de de sütunu işaret eder. Şekil 4.30'daki bellek haritası bu adreslemeyi de göstermektedir.



Şekil 4.33 : Adres yolu bit isimleri

Kontrol modülü herhangi bir 4x4'lük ağ parçasının değişken değerlerinin okunması komutunu Bellek Dizisi modülüne sayfa, satır ve sütün adreslerini vererek gönderir. Bellek Dizisi modülü bu satır ve sütun adreslerine sayfa ve değişken adreslerini de ekler ve DPHRAM'lerden x (4x4'ün komşusu olan nöron değerleri ile birlikte), y ve u değişken setlerini okur. Bu toplamda 32 kelime x verisi, 16 adet y verisi ve 16 adet u verisi demektir. Bu veri 1024 bit genişlikli veriyolu üzerinden CNPN'ye aktarılır. Bellek Dizisi bir iterasyon boyunca 1. sayfadan değerleri okur, 2. sayfaya yazar. Takip eden iterasyonda 2. sayfadan değerleri okur ve hesaplanan değişken değerlerini 1. sayfaya yazar. Bu döngü süreklilik arzeder. u değişkeni için iterasyon içerisinde yeni değer hesaplanmadığından, u değişkenine yalnız bir sayfalık bellek alanı ayrılmıştır. Şekil 4.34'de tüm ağın 4x4'lük parçalara nasıl ayrıldığı gösterilmektedir. Renkli kareler simüle edilen nöronu ifade eder. Üst ve soldaki numaralar nöronun sırasıyla yataydaki ve düşeydeki konumlarıdır. Nöronun

üzerindeki numara ise, hangi DPHRAM komponentinde değerlerinin saklandığını gösterir.



Şekil 4.34 : Tüm ağın 4x4 parçalara ayrılması

Kontrol devresi CNPN'nin 4x4'lük ağ parçası için iterasyonu gerçekleştirmesinin esnasında hesaplanan değerlerin Bellek Dizisine yazılmasını sağlar. İterasyon içerisinde önce x[k+1] değerleri hesaplanır. Bu değerlerin hesaplanmasının hemen ardından Kontrol modülü RAM ağına değişken, sayfa, satır ve sütün değerlerini bildirerek, yaz komutunu gönderir. RAM ağı yaz komutunu aldığında, ilgili adrese CNPN'den gelen 256 bitlik (16değişken x 16bit) veriyolundaki değerleri yazar. Kontrol devresi CNPN'nin y[k+1] değerlerini hesaplamasının ardından, Bellek Dizisine gönderdiği adres değerinin değişken kısmını değiştirir ve tekrar yaz komutu gönderir. Bu komut ile, RAM ağı yeni adrese yine CNPN'den gelen 256 bitlik veriyi yazar.

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.

DPHRAM'lerin B portu ise sistemin bir diğer önemli özelliği için kullanılır. B portlarına da tek bir ortak B portu adresi ile ulaşılır ve adres kelimesi Şekil 4.33'dekinin aynısıdır. Yalnız bu portun kontrolü ve çıkış veriyolları VGA sürücü devreye bağlıdır. Bu port üzerinden, ağın iterasyonları sürerken, gerçek zamanlı olarak değişkenlerin o anki değerleri okunur ve sistem monitöründe ağ görüntüsü oluşturulur. B portu adresinin sayfa biti, yani k biti, aslında kontrol devresinin kontrolündedir. Kontrol devresi VGA sürücünün, CNPN'nin yazma yaptığı sayfaya değil, okuma yaptığı sayfaya erişmesini sağlamakla görevlidir. Bu ağ değişken değerlerinin güvenliği açısından önemlidir. B portu adresinin değişken bitleri ise FPGA kartı üzerindeki iki adet sürgü anahtar ile girilir. Bu sayede kullanıcı kart üzerindeki anahtaları açıp kapayarak, sistem monitöründe görmek istediği ağ değişkenin adresinin oluşmasını sağlar. Çizelge 4.13'de Bellek Dizisi'nin kapladığı alan ve erişim süreleri gösterilmiştir.

Çizelge 4.13 : RAM Ağı'nın başarımı.

	Bellek Dizisi
Lojik Dilim Sayısı (tamamına oranı)	1451
FF Sayısı (tamamına oranı)	1652
LUT Sayısı (tamamına oranı)	1862
Block RAM Sayısı (tamamına oranı)	80
4x4 Ağ Parçasının İterasyonu için Gereken Değerlerin Okunma Süresi	10 saat çevrimi
4x4 Ağ Parçasının İterasyonu Sonunda Değerlerin Yazılma Süresi	8 saat çevrimi
4x4 Ağ Parçasının Denetim için Erişim Süresi (yazma ve okuma eşit)	4 saat çevrimi

Bellek Dizisi, 16 bit kelime uzunluklu 5120 kelime kapasiteli, 16 adet çift portlu DPHRAM komponenti içerir. Toplam kapasitesi 160 kilobayttır. Toplam 1280 bit çıkış veriyolu, 256 bit giriş veriyoluna sahiptir. İterasyonu gerçekleştirmek ve denetleyici bilgisayarla arasında veri aktarmak için A portu, sistem monitöründe gerçek zamanlı görüntü oluşturmak için B portu kullanılır.

Çekirdeğin bir diğer alt modülü olan Parametre kütüğü, tüm ağın kullandığı parametreleri, diğer değişkenleri ve devrenin çalışması ile ilgili bazı bilgileri saklayan modüldür. Bu modülü RAM ağından ayrı tutan, sakladığı verilerin ağdaki tüm nöronlar için ortak olmasıdır. Yapısal olarak farkı ise, verileri RAM üzerinde değil, flipfloplardan oluşturulan tutuculardа saklamasıdır. Sakladığı her bir parametre için ayrı veriyolu bulunmakta ve bu veriyolu ile parametreyi kullanacak tüm

modüllerin ilgili girişlerine paralel bağlanılmaktadır. Parametre tutucularının geniş bir veriyolu ve birçok lojik elemanın girişini sürmeleri sebebi ile diğer tutuculara oranla daha yüksek gecikme süreleri vardır. Ama bu gecikmeler devrenin 25 MHz hızında çalışmasını olumsuz etkileyecek kadar büyük değildir. Çizelge 4.14'de saklanan parametreler açıklanmıştır.

Çizelge 4.14 : Parametre Kütüğünde saklanan parametreler.

Parametre Adı	Uzunluğu	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 fonksiyonun hesabında kullanılan x_{lmt} değeri
m	16 bit	g fonksiyonun 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_degeri	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 koşuğunu belirtir.

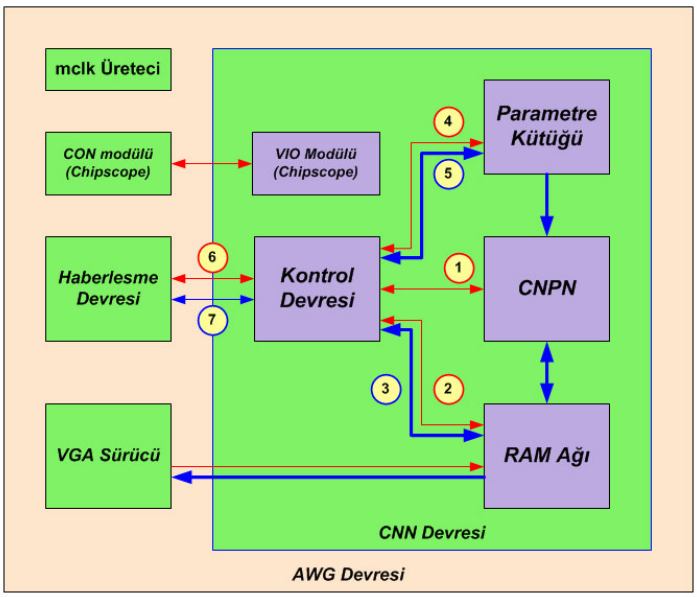
Parametre kütüğü parametre değerlerini çıkışlarından sürekli servis etmektedir. Paralelindeki kontrol devresi, parametre kütüğündeki parametre değerlerini 16 bit genişlikli başka giriş ve çıkış portlarından okuyabilir ya da değiştirebilir. Bu işlev denetleyici bilgisayar ile parametrelere erişmeyi mümkün kılar ki bu, çalışmanın geldiği önemli bir noktadır. Bahsedilen 16 bitlik veriyolundan iterasyon parametresi 2 seferde okunur veya yazılır. Ayrıca kontrol devresi tüm ağın her iterasyonu tamamlandığında parametre kütüğüne bir komut göndererek, koşulacak parametresinin bir azalmasını, iterasyon parametresinin bir artmasını sağlar. Ağın çalışacak kısmının sol, sağ, üst ve alt marjinleri, kontrol devresi tarafından her iterasyona başlarken okunur ve güncel değeri kullanılır. Koşulacak parametresinin değeri olabilen maksimum değerinde, yani 65535'de yapıldığında, parametre kütüğü, kontrol devresinin komutu ile iterasyon değerini arttırır ama koşulacak değerini azaltmaz. Bu sayede bu parametre değeri maksimuma getirildiğine, sistem iterasyonlarını durdurmaz, sürekli çalışır. Parametre kütüğünün başarımı Çizelge 4.15'de verilmiştir.

Çizelge 4.15 : Parametre Kütüğü'nün başarımı ve kaynak kullanımı.

	Parametre Kütüğü
Lojik Dilim Sayısı (tamamına oranı)	240
FF Sayısı (tamamına oranı)	293
LUT Sayısı (tamamına oranı)	380
Parametre Yazma Süresi	2 saat çevrimi
Parametre Okuma süresi	2 saat çevrimi

Kontrol devresi sadece Çekirdeği değil, tüm tasarımı kontrol etmekle yükümlüdür. CNPN'nin iterasyonu gerçekleştirmesi, RAM ağından değerlerin okunması ve yazılması, parametre kütüğündeki değerlerin okunması ve yazılması, ayrıca haberleşme devresinin kullanılması ve VGA sürücünün çalıştırılması kontrol devresinin denetimindedir.

Kontrol Devresi, Çekirdek içinde, paralelinde bulunan CNPN, RAM ağı ve parametre kütüğü modüllerini ve hiyerarşide üstünde olan haberleşme devresini kontrol etmektedir. Tüm sistemin çalışmasını sağlamaktadır. Diğer modüller açıklanırken kontrolörün çalışması da bir miktar anlatılmıştır. Detaylandırmak için öncelikle Şekil 4.35'deki birden altıya kadar numaralandırılmış yollar ve bunların Çizelge 4.16'da verilen açıklamaları incelenmelidir.



Şekil 4.35 : Kontrol devresinin bağlantıları.

Şekil 4.35'deki kırmızı yollar kontrol işaretlerinin taşındığı kontrol yollarını, mavi yollar verinin taşındığı veri yollarını simgelemektedir. Ok yönleri işaretin akış yönünü belirtir. Her hiyerarşik katmandaki bloklar farklı renktedir.

Çizelge 4.16 : Kontrol devresi bağlantılarının açıklamalı çizelgesi.

Yol Grubu	Yol Adı	Genişlik (bit)	Yönü	Taşınan İşaretin Açıklaması
1	x_hesapla	1	çıkış	Tüm NPE'lerin x(k+1) değerinin hesaplamasını başlatan işaret
	x_hesaplaniyor	1	giriş	NPE'lerin x(k+1) hesaplamakla meşgul olduklarını belirten işaret
	y_hesapla	1	çıkış	Tüm NPE'lerin y(k+1) değerinin hesaplamasını başlatan işaret
	y_hesaplaniyor	1	giriş	NPE'lerin y(k+1) hesaplamakla meşgul olduklarını belirten işaret
2	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 sütün belirten bitleri
	oku	1	çıkış	RAM ağının okumasını başlatan işaret
	okunuyor	1	giriş	RAM ağının okuma işlemi yürüttüğünü bildiren işaret
	yaz	1	çıkış	RAM ağının yazmasını başlatan işaret
	yaziliyor	1	giriş	RAM ağının yazma işlemi yürüttüğünü bildiren işaret
	Aen	1	çıkış	A portalsının izin işareti
	b_yaz	1	çıkış	RAM ağında bir parçaya yapılan işlemin yazma olduğunu belirten
	b_isle	1	çıkış	RAM ağında bir parçanın işlenmesini başlatan işaret
	b_isleniyor	1	giriş	RAM ağında bir parçanın işlenmekte olduğunu bildiren işaret
	Ben	1	çıkış	B portalarını izin işareti
	vga_degisken	2	çıkış	Sistem monitöründe gerçek zamanlı izlenecek değişkeni belirten
3	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 arttırıp, koşulacak değerini bir azaltan işaret
4	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
5	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
6	gidecek_hazir	1	çıkış	Haberleşme hattından bir baytlık veri gönderilmesini başlatan işar.
	gonderiliyor	1	giriş	Bir baytlık verinin gönderiliyor olduğunu belirten işaret
	gelen_var	1	giriş	Haberleşme hattından bir baytlık veri geldiğini bildiren işaret
	alimda_hata	1	giriş	Hattan veri alımı esnasında hata algılandığını belirten işaret
	gelen_alindi	1	çıkış	Haberleşme hattından gelen bir baytlık verinin alındığını bildiren iş.
7	giden_bayt	8	çıkış	Haberleşme hattından gönderilecek bir baytlık veri
	gelen_bayt	8	giriş	Haberleşme hattından gelen bir baytlık veri

Kontrol devresinin 2 temel modu vardır. Bunlardan ilki denetleyici bilgisayar ile haberleşme halinde olduğu 'bağlantı' modu, diğeri iterasyonları gerçekleştirdiği 'operasyon' modudur. Devreye ilk güç verildiği anda, bağlantı modunda çalışmaya başlar. Bağlantı modunda iken denetleyici, sistemin parametrelerini okuyabilir, değiştirebilir; bellekteki değişken değerlerini okuyup değiştirebilir. Ya da sistemi operasyon moduna geçirebilir.

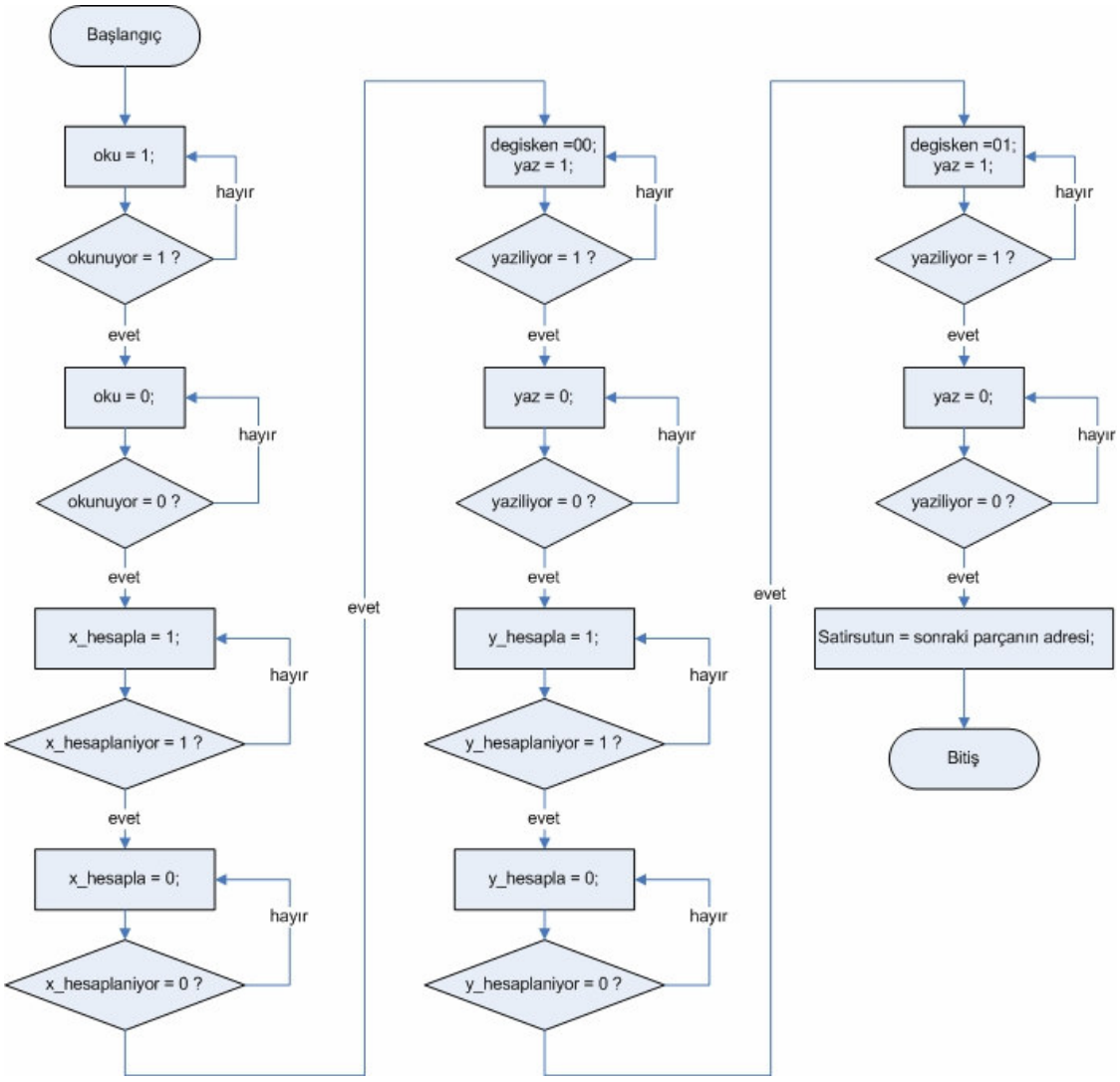
Operasyon modunda Çekirdek, 'kosulacak' parametresinin değeri adedinde iterasyonu gerçekleştirmek için çalışır. Koşulacak değeri sıfırlandığında hesaplamaları durdurur ve operasyon modunda beklemeye başlar. AWG operasyon modunda çalışıyorken veya bekliyorken, denetleyici RS-232 portundan bağlantı isteği yollayabilir. Kontrol devresi bu isteği alır. Bu istek, sürüyorsa o anki iterasyonun tüm ağ için tamamlanmasıyla, duruyorsa derhal kontrol devresi tarafından yerine getirilir ve sistem bağlantı moduna geçer. Denetleyici istediği işlemleri gerçekleştirir ve sistemi tekrar operasyon moduna geçirebilir. Kontrol

devresinin FGPA üzerinde kapladığı alan Çizelge 4.17'de, operasyon modunun basit akış grafı Şekil 4.36'da verilmiştir.

Çizelge 4.17 : Kontrol Devresinin kaynak kullanımı.

	Kontrol Devresi
Lojik Dilim Sayısı (tamamına oranı)	424
FF Sayısı (tamamına oranı)	328
LUT Sayısı (tamamına oranı)	837

Bu basit akış grafında bitiş ile yeniden başlama arasında tüm ağın işlenip işlenmediği, koşulacak değerinin sıfırlanıp sıfırlanmadığı, denetleyiciden bağlantı isteği gelip gelmediği kontrol edilmekte. Tüm ağ için bir iterasyon tamamlanmış ise k sayfa değeri değiştirilmekte, bu sayede bir önceki iterasyonda yazılan değerler okunacak hale getirilmektedir.



Şekil 4.36 : Operasyon modu için basit akış grafı.

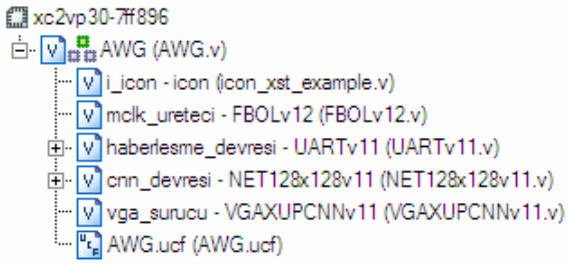
4.3.3 Çekirdeğin çevre birimleri

128x128 hücreli programlanabilir RO-HYSA devresi tasarımında Xilinx'in geliştirdiği Chipscope aracı da kullanılmıştır. Chipscope ile devre tasarımına dahili lojik izleyici modüller eklenebilmekte ve devre çalışırken belirlenen veriyolları ve tutucuların değerleri bu modüller sayesinde bilgisayarda izlenebilmektedir. Bunun için veriyolu ve tutucuların değerleri bilgisayara programlama kablosu üzerinden aktarılmaktadır. Üçündü tasarımın alt modülü olan i_con isimli yapı temel Chipscope modülüdür. Diğer Chipscope modüllerinin kontolü i_con'un görevidir.

Virtual IO (sanal giriş çıkış, i_vio) modülü Chipscope aracının sağladığı bir modüldür. Tasarım esnasında modüle bağlanmış, toplam genişliği 256 bit olan dahili veriyolları ve tutucuların değerlerini, tasarım yapılan bilgisayardan takip edebilmek için eklenmiştir. Bu modülün HYSA devresinden aldığı değerler, FPGA kartını programlamaya da yarayan USB kablosu üzerinden bilgisayara aktarılmaktadır. Bu hattaki veri akışı gerçek zamanlı değildir. Vio'nun kontrol ettiği tutucu ve yollarının değeri en fazla saniyede dört defa güncellenebilmektedir. Yine de tasarım için eşsiz bir araçtır. Devrenin çalışmasının ardından tasarımdan çıkarılmamış. İstendiğinde iç değerlerin yeniden izlenebilmesi için bırakılmıştır. Yine belirtilmelidir ki i_vio modülü Çekirdeğin işlevini gerçekleştirmesi için gerek duyulan bir modül değildir.

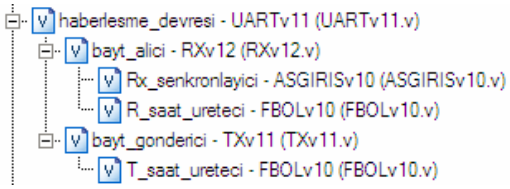
Üçüncü tasarımın önceki tasarımı aşan özelliklerinden en önemlisi interaktif çalışmasıdır. Bu tasarım sahip olduğu alt modüller ile, RS-232 portu üzerinden harici bir donanımla, örneğin bir bilgisayar ile haberleşebilir. 128 x 128 boyutlu HYSA üzerinde bulunan tüm hücrelere ait x, y ve u değişkenleri, tüm ağırlıklar ve tüm parametreler bu haberleşme kanalı üzerinden tasarlanan RO-HYSA'ya yazılabilir ya da RO-HYSA'dan okunabilir. Bu özellik devre kullanıcısına farklı parametreler ile farklı uzaylar oluşturarak çalışma ve çalışmasının sonuçların saklama yeteneği kazandırmıştır.

Haberleşme kanalı üzerinden x, y ve u değişkenlerinin yazılması ve okunması için geçmesi gereken zaman devrenin gerçek zamanlı çalışması prensibi ile çatışır. Bu yüzden, eski tasarımda olduğu gibi bu tasarımda da AWG'ye bir VGA görüntü oluşturan alt modül eklenmiş, kart üzerindeki anahtarlar ile x, y ve u değişkenlerinden istenen, harici bir VGA monitörden izlenebilir yapılmıştır. Şekil 4.37'de üçüncü tasarımın hiyerarşik yapısı verilmiştir.



Şekil 4.37 : 128x128 hücreli programlabilir RO-HYSA'nın hiyerarşik yapısı.

Şekil 4.37'deki 'xc2vp30-7ff869' isimli yapı kullanılan FPGA tümdevresidir. Haberleşme Devresi RS-232 protokolünü yürüten bir UART'tır. Bayt almak ve bayt böndermekle yükümlü iki alt modülü yazılmıştır. Her iki modülünde tamponları birer baytlıktır. Yani göndericiden bir baytlık veriyi göndermesi istenebilir. Ya da alıcı bir baytlık veri aldığında mutlaka tampondaki alınan bayt okunmalı ve tampon boşaltılmalıdır. Haberleşme devresi, HYSA devresinin kontrol modülü tarafından kontrol edilir. Şekil 4.38'de haberleşme devresinin hiyerarşik yapısı verilmiştir.



Şekil 4.38 : Haberleşme devresinin hiyerarşik yapısı

Haberleşme 115200 bit/saniye hızında asenkron olarak gerçekleştirilir. Asenkron haberleşme gerçekleştirdiği için RS-232 bağlantısının gelen veri (Rx) hattı senkronize edilerek okunmalıdır. Aksi taktirde doğru okunmama gerçekleşmektedir. Bunun için FPGA tasarımında uygulanması gereken bir takım yöntemler vardır. Varsayılan ayarlarda her giriş çıkışa yerleştirilen giriş çıkış tampon flipflopu Rx girişinde kaldırılmıştır ve bu giriş bir asenkron giriş olarak tanımlanmıştır. Haberleşme devresinin kaynak tüketimi Çizelge 4.18'de verilmiştir.

Çizelge 4.18 : Haberleşme devresinin kaynak kullanımı.

	haberlesme_devresi
Lojik Dilim Sayısı (tamamına oranı)	72
FF Sayısı (tamamına oranı)	81
LUT Sayısı (tamamına oranı)	129

Üçüncü tasarımın çalışma hızı, 128 x 128 boyutlu HYSA'nın gerçek zamanlı izlenmesini sağlayacak kadar yüksektir. FPGA kartı üzerindeki iki adet sürgü anahtar ile karta takılan VGA monitörde hangi değişkenin görüntüleneceği seçilir. Tüm ağa

ait x, y ya da u değişkenlerini monitör üzerine çizmek vga_surucu modülünün görevidir. İlk tasarımda 5x5'lik ağdaki her bir nöron 32*32 piksellik kareler ile temsil edilmiş ve ağ ekranın sol üst köşesine çizilmişken bu tasarımda 128x128'lik ağdaki her bir nöron, ikinci tasarımdaki gibi 2x2 piksellik kareler ile temsil edilmiş ve ağ ekranın tam ortasına çizilmiştir. Ayrıca önceden ekranda mavi renk negatif tüm değerleri, kırmızı renk ise pozitif tüm değerleri ifade ederken; bu tasarımda mavi ve kırmızı renk değerin işaretini, parlaklık ise genliğini ifade etmektedir. VGA sürücünün kaynak tüketimi Çizelge 4.19'da gösterilmektedir.

Çizelge 4.19 : VGA sürücünün kaynak tüketimi.

	vga_surucu
Lojik Dilim Sayısı (tamamına oranı)	127
FF Sayısı (tamamına oranı)	50
LUT Sayısı (tamamına oranı)	243

Yapılan tasarım kullanılan FPGA kaynaklarının hemen hemen tamamını harcamaktadır. Kapı, flipflop ve veriyolu sayısının çok olması tanımlanan devrenin gerçeklenmesini güçleştirmektedir. Kontrolsüz ilk sentezlemeler sonucunda yapılan analizlerde, devrenin bazı veriyollarının veri iletim sürelerinin bir saat çevriminden daha uzun olduğu ortaya çıkmıştır. Bu devrenin hatalı ya da güvensiz çalışmasına sebep olmuştur. Her ne kadar zamanlama kriterleri belirlenerek sentezleme işlemi tekrarlansa da veriyolu gecikme değeri ana saat işaretinin bir periyodunun (10 ns) altına indirilememiştir. Bu yüzden FPGA tümdevresine kart üzerindeki osilatörden verilen 100 MHz'lik ana saat işaretinin frekansı, mclk_ureteci isimli modülde 4'e bölünmüş ve 128x128 RO-HYSA devresinin diğer tüm alt modüllerinin saat girişleri bu 25 MHz'lik mclk işareti ile sürülmüştür. Saat işaretinin periyodunun 40 ns yapılması ile tüm veriyolları gecikme süreleri önemsiz hale gelmiş ve devrenin güvenli çalışması garanti edilmiştir. Ayrıca ana saat işareti düşürülerek sentezleyici de FPGA üzerine yerleştirici yazılım araçlarının çalışma süreleri azaltılmıştır. Sentezleme ve yerleştirme işlemleri, zor zamanlama kriterleri ile bir saati aşkın sürerken, 40 ns için verilen zamanlama kriteleri için yarım saatin altında sürmüştür. Bu sürelerin uzun olması en çok FPGA dilimlerinin neredeyse tamamının kullanılmak zorunda olmasıyla ilişkilidir.

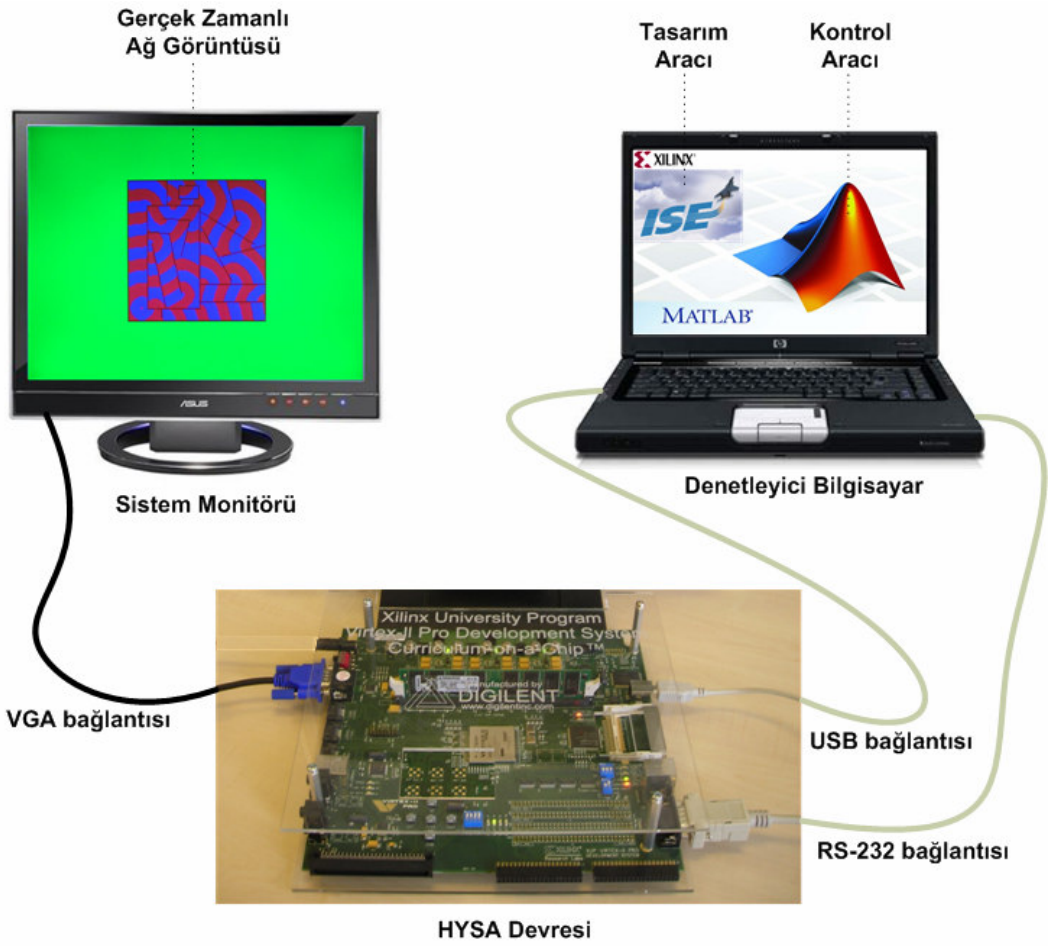
Yine Şekil 4.37'de alt blok olarak görülen AWG.ucf, tasarım için kullanıcının getirdiği kısıtlama ve tayinleri içerir. Örneğin üçüncü tasarımın giriş çıkışlarının, FPGA'nın gerçekte hangi pinlerine atanacağı bu blok içerisinde yazılıdır. Bununla birlikte devreyi sentezleyen yazılıma yönelik devrenin zaman kriterleri de bu blokta belirtirmiştir. Diğer alt modülle ayrı başlıklar altında açıklanmaktadır. Üçüncü tasarımın başarımı ve kaynak tüketimi Çizelge 4.20'de sunulmuştur.

Çizelge 4.20 : 128x128 programlanabilir ROHYSA'nın başarımı ve kaynak kullanımı.

	AWG Devresi
Lojik Dilim Sayısı (tamamına oranı)	10944
FF Sayısı (tamamına oranı)	12444
LUT Sayısı (tamamına oranı)	18004
Block RAM Sayısı (tamamına oranı)	80
MULT18x18 Sayısı (tamamına oranı)	25
Ortalama Bir İterasyon Süresi	210.080 saat çevrimi

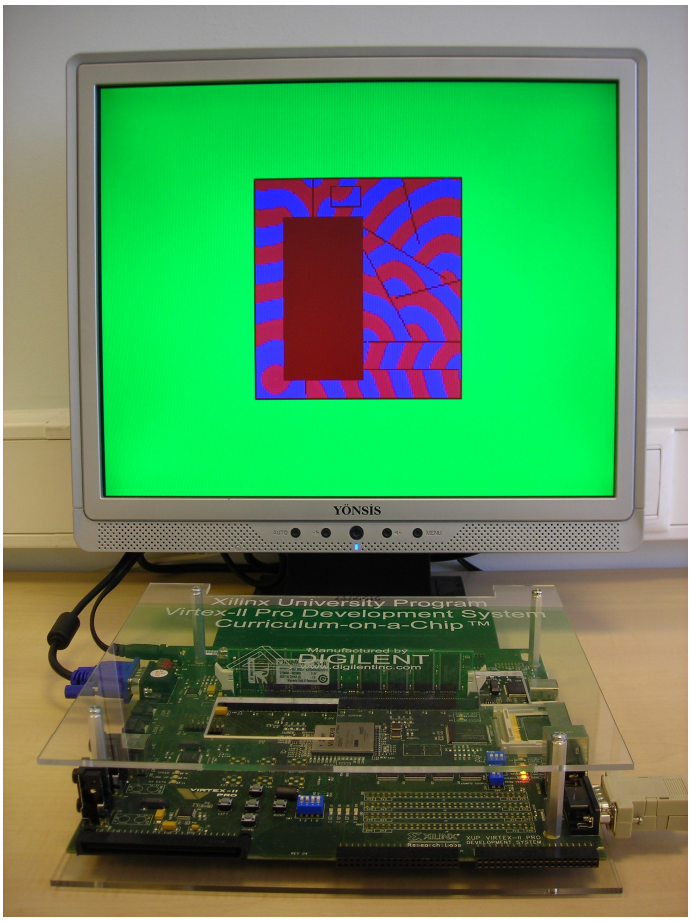
4.3.4 Gerçeklemenin bilgisayar yardımı ile kullanılması

Gelinen son noktada, Xilinx XC2VP3000 FPGA'sı üzerinde, başka harici bir bellek elemanına ihtiyaç duyulmadan, 128x128 boyutlu aktif dalga yayan ve tamamen kontrol edilebilen bir HYSA gerçeklenmiştir. Kullanılan FPGA'yı barındıran kartın resmi Şekil 4.39'da HYSA devresi olarak gösterilmektedir. Bu kartın üzerindeki VGA konnektörü, dijital-analog dönüştürücüsü ile tasarımda gerçeklenen VGA sürücü modülü ve FPGA tümdevresinin içindeki çift portlu bellek elemanları sayesinde HYSA'nın gerçek zamanlı görüntüsü, sistem monitöründen izlenebilmektedir. Ağa ilişkin üç değişken olan x, y ve u'dan monitörden izlenmek istenen değişkenin seçimi, kart üzerindeki sürgülü anahtarlar ile yapılmakta ve değişikliğe sistem anında cevap vermektedir. Gerçeklenen devrenin FPGA üzerinde konfigüre edilmesi için tasarımda olduğu gibi Xilinx'in ISE aracı denetleyici bilgisayarda çalıştırılmaktadır. Konfigürasyon USB kablosu üzerinden gerçekleşmektedir. Aynı zamanda denetleyici bilgisayar üzerinde Chipscope aracı çalıştırılarak, yine USB kablosu aracılığı ile, HYSA devresinin bazı iç tutucu ve veriyolu değerleri izlenebilmektedir. Lakin bu veri akışı gerçek zamanlı değildir. Zaten Chipscope sistemin çalışması için gerekli değildir. Yalnızca geliştirme aşamasında ihtiyaç duyulmuştur.



Şekil 4.39 : Sistemin yapısı

Sistemin denetleyici bilgisayar ile denetimi, RS-232 hattı üzerinden gerçekleşmektedir. Denetleyici yazılım olarak MATLAB kullanılmış, parametre okuma-yazma, değişken okuma-yazma, tüm ağ verisini yükleme ya da ağı kaydetme işlemleri için Matlab fonksiyonları yazılmıştır. Yazılan bu fonksiyonları kullanarak sisteme istenen başlangıç değerleri yüklenmekte, istendiği kadar iterasyon yaptırılmakta, iterasyonların tamamlanması ile sistemin o anki değerleri denetleyici bilgisayara alınabilmekte, ve Şekil 4.6'da gösterilen 16-bit kayan noktalı sayı formatındaki veriler analiz edilebilmektedir. Ayrıca bu alınan ağ değerleri bilgisayarda saklanabilmekte, bir başka zaman sisteme geri yüklenip, ağ öykünlemesi kaldığı yerden devam ettirilebilmektedir. Sistemin bu özelliği sayesinde farklı parametre değerlerinin sistem dinamiğine etkisi daha etkin incelenbilmektedir, sistemin kaydedildiği noktaya kadar tekrar çalıştırılması ile kaybedilecek zaman kazanılmış olmaktadır. Otodalga yaymakta olan sistemin, FPGA kartı ve sistem monitöründen oluşan son haline ait bir resmi Şekil 4.40'da verilmiştir.



Şekil 4.40 : Sisteme ait bir fotoğraf

Devrenin kullanılması için gereken denetleyici bilgisayarda koşturacak `set_global`, `connect`, `run`, `yaz`, `oku`, `resim_oku`, `resim_yaz` ve `yuvarla` fonksiyonları yazılmıştır.

Üçüncü tasarımdaki HYSA'nın kullandığı parametreler, devrenin işleyici ile ilgili değişkenler, haberleşme kanalı gibi global değişkenlerin çalışmaya başlanırken hazırlanması gerekmektedir. `set_global` fonksiyonu, 128x128 hücreli programlanabilir RO-HYSA devresi ile kullanıcı bilgisayarı arasıdaki haberleşme kanalını uygun bir COM objesi oluşturarak hazırlar ve tüm gerekli değişkenleri global olarak atar. Ayrıca sayı dönüşümlerinde kullanılacak kuantalayıcıları da hazırlar. Bu fonksiyon ilk çalıştırılması gereken fonksityondur.

`connect` fonksiyonu kullanıcı bilgisayarının devreye bağlanmasını sağlar. Global değişkenlerin hazırlanmasının ardından bu fonksiyon çalıştırılarak devre, bilgisayar ile haberleşme moduna alınır.

`yaz` fonksiyonu devrenin parametre kütüğünde sakladığı parametrelere ve bellek dizisinde sakladığı durum değişkeni ve giriş işaretine değerler yazmayı sağlar. HYSA'nın çalıştırılmasından önce uygun başlangıç durum değerlerinin ve parametrelerin yüklenmesi için bu fosiyon kullanılır.

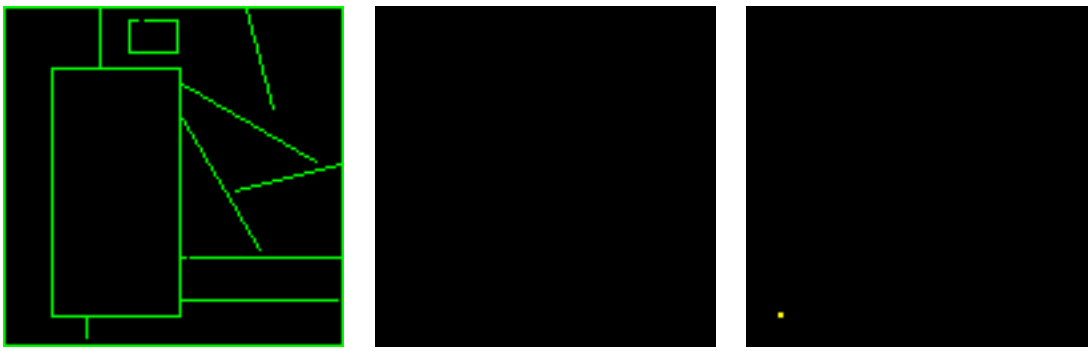
`yaz` fonksiyonunun ağın durum değişkenlerine yazacağı değerler bitmap dosyaları şeklinde oluşturulur ve saklanır. `resim_oku` fonksiyonu bitmap dosyalarından okuduğu değeri yaz fonksiyonun x, y ve u'ya yükleme esnasında kullanabileceği bir değişkene yazar. 24 bit renk derinlikli bitmap dosyada R ve B kanalları üzerinde saklanan 16 bitlik veri q2 kuantalayıcısı kullanılarak yarı duyarlı kayan noktalı sayıya dönüştürülür. G kanalı üzerinde ya 255 ya da 0 değeri saklanır ki bu da ilgili hücrenin pasif ya da aktif olacağını belirler. Pasif hücrenin R ve B kanallarındaki değerin önemi yoktur. Bu bitmaplerde her bir piksel HYSA'da bir hücreye karşılık gelir. x, y ve u için birer adet bitmap dosyası hazırlanır.

`oku` fonksiyonu, `yaz` fonksiyonun tersi işi görür ve HYSA'dan parametre veya durum değişkeni okumak için kullanılır.

`resim_yaz` fonksiyonu ile `oku` fonksiyonunun okuduğu x, y ve u matrisleri yine bitmap dosyaları yukarıda anlatılan prensiple yazılır.

HYSA'nın çalışmaya başlamasının ardından `run` fonksiyonu çağırılır. Bu foksiyon, devreyi operasyon moduna geçirerek öykünleme işlemini başlatır. `connect` fonksiyonu çağırılıncaya kadar veya koşulacak iterasyon parametresindeki iterasyon sayısı sıfırlandığında devre RO-HYSA'yı durdurur.

`yuvarla` fonksiyonu ise MATLAB'de yapılacak ağ benzetimlerinin 128x128 hücreli programlanabilir RO-HYSA devresinde yapılacak öykünlemeler ile eş sonuçlar üretmesi için eklenmiştir. Sistemin çalışması için olmazsa olmaz değildir. `yuvarla` fonksiyonu, denormalize sayıları sıfırlar ve çift duyarlı sayıları kuantalayıp yarı duyarlı hale getirir. Böylece benzetimdeki aritmetik toplama ve çarpma işlemleri gerçeklenen NPE'lerindeki toplayıcı ve çarpıcıların yaptığı şekilde olur. Aşağıda anlatılan fonksiyonların kullanılması ile yapılan bir öykünleme örneği verilmiştir.

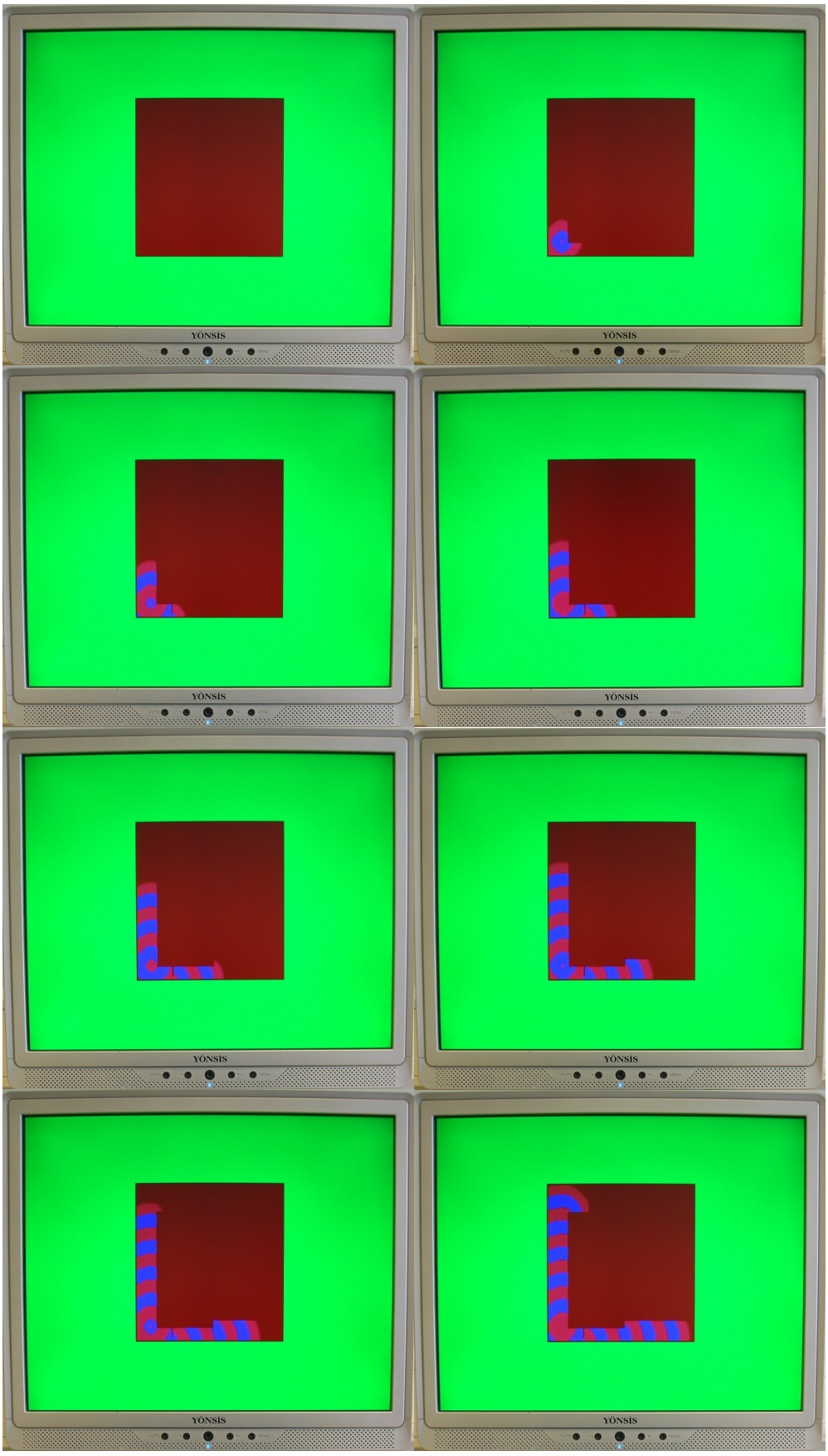


X[0] Y[0] U

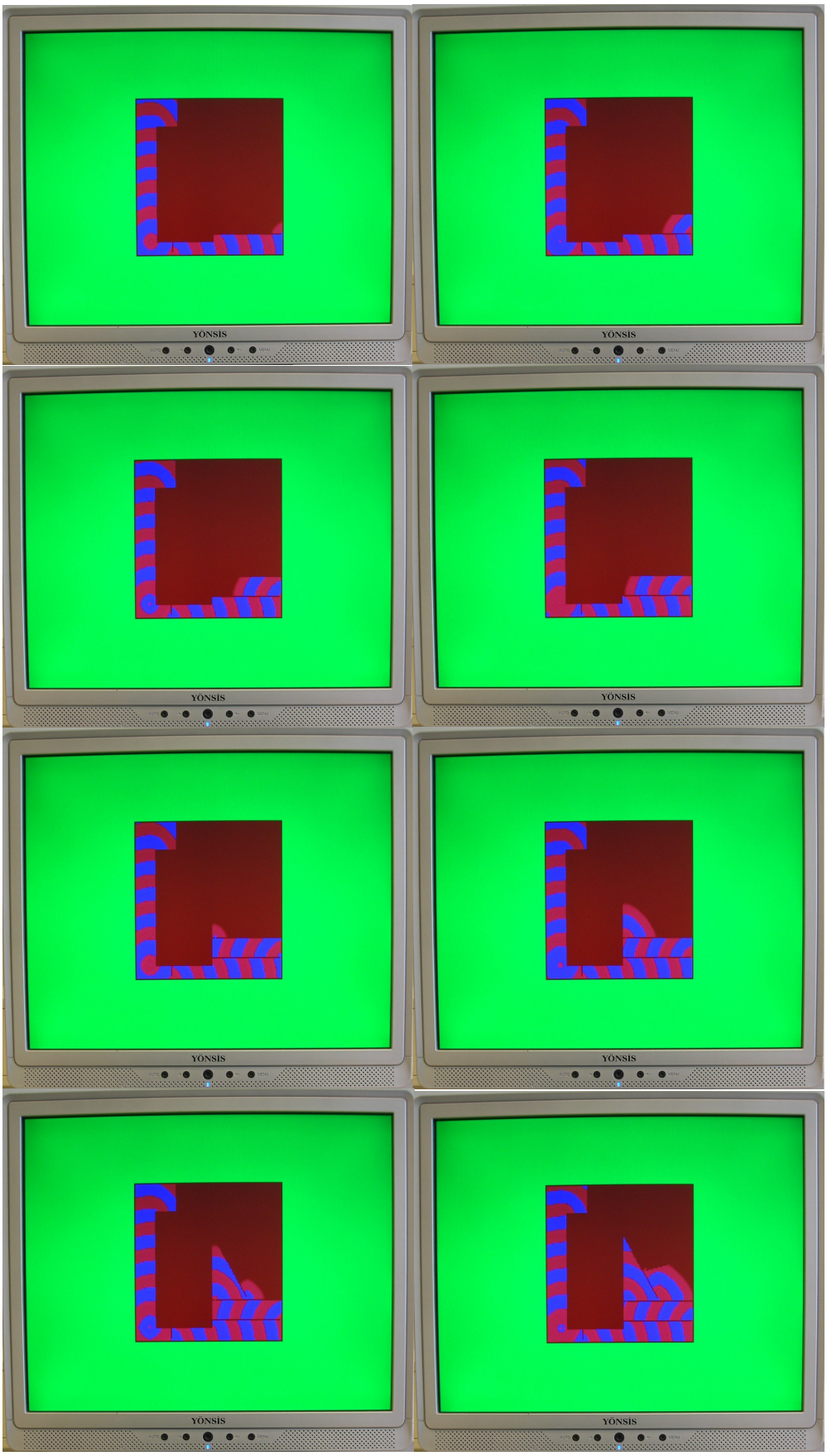
Şekil 4.41 : Sisteme yüklenen başlangıç değerleri.

HYSA devresi FPGA üzerine kurulduktan sonra, ağa başlangıç değerleri olarak Şekil 4.41'de gösterilen X[0], Y[0] ve U matrisleri yüklenmiştir. Siyah renk sayısal değer olarak 0'ı, yeşil pasif hücreleri, sarı da 10'u ifade etmektedir.

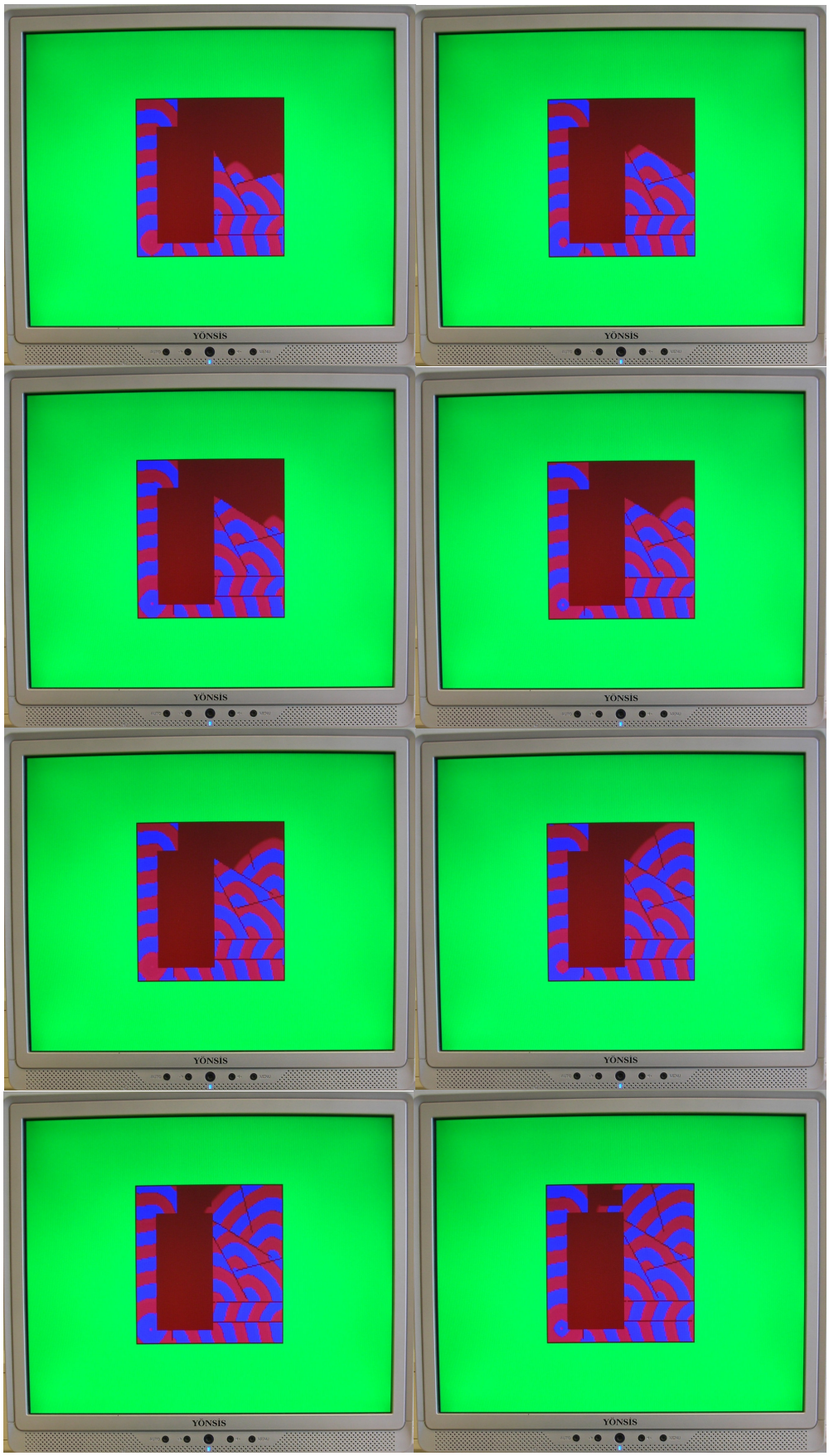
Başlangıç değerlerinden sonra parametreler, $\alpha = 3$; $\beta = -4$; $\varepsilon = 0.15$; $\sigma = -0.1$; $m = -8$; $\lambda = 1.1$; $T = 0.15$; $x_{sabit} = 0$; $a_{dogu} = 0.09$; $a_{kuzey} = 0.09$; $a_{batı} = 0.09$; $a_{guney} = 0.09$ yüklenmiştir. Bu başlangıç koşulları altında HYSA'nın her 100 iterasyon sonundaki x değişkenine ait ağ görüntüsü, yani X matrisi, dört sayfa boyunca süren Şekil 4.42'de verilmiştir.



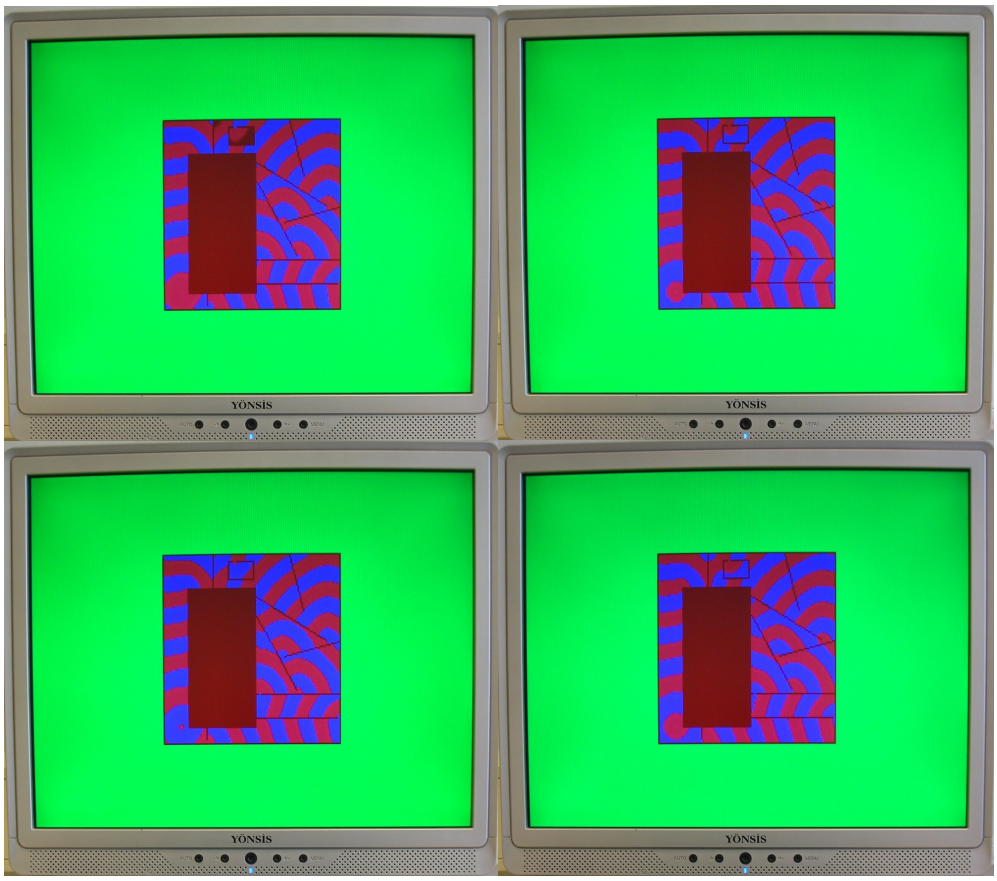
Şekil 4.42 : Otodalga yayan ağın, her 100 iterasyonda monitörden çekilen resimleri.



Şekil 4.42 : Otodalga yayan ağın, her 100 iterasyonda monitörden çekilen resimleri (devam).



Şekil 4.42 : Otodalga yayan ağın, her 100 iterasyonda monitörden çekilen resimleri (devam).



Şekil 4.42 : Otodalga yayan ağın, her 100 iterasyonda monitörden çekilen resimleri (devam).

5. GERÇEKLENEN SAYISAL RO-HYSA İLE GÖZLENEN YENİ UZAY-ZAMAN DALGALARI

128x128 hücreli programlanabilir RO-HYSA devresinin gerçeklenmesinin ardından, devrenin programlanabilme ve denetlenebilme yeteneği kullanılarak bazı parametre aralıkları taranmış bu taramalar ile çeşitli dalga formları elde edilmiştir. Bu taramalar esnasında türlü otodalga, spiral dalga [25], yürüyen dalga ve dama desenli dalgalar gözlenmiştir [15-18]. Bu bölümde belirtilen başlangıç koşulları ve parametreler ile elde edilen dalga desenlerinden örnekler sunulmuştur. Altbölüm 4.3.4 'te anlatılan fonksiyonlar kullanılarak, Şekil 5.1'de gösterilen U, X[0] ve Y[0] matrisleri RO-HYSA'ya başlangıç koşulları olarak yüklenmiştir. Şekillerde siyah renk 0 başlangıç koşuluna, yeşil renk ise pasif hücrelere karşılık gelmektedir.

U X[0] Y[0]

Şekil 5.1 : Başlangıç koşulları.

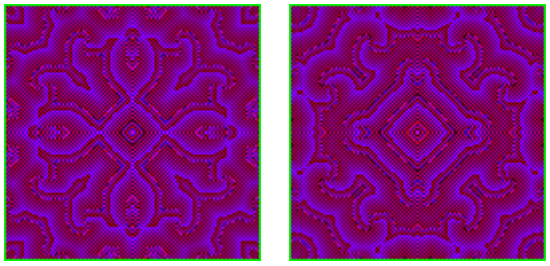
U ve Y[0] matrislerinin tüm elemanları 0 değerindedir. X[0]'da ise sınır hücreler (yeşil) pasif olarak işaretlenmiş ve bu çerçeve içinde kalan aktif hücreler 0 sayısal değeri yüklenmiştir. HYSA parametrelerine atana değerler de Çizelge 5.1'de verilmiştir.

Bu başlangıç koşullarında Şekil 5.2'deki desenler elde edilmiştir. Bu desenler şekilde resmin altında belirtilen m parametresi ile ve belirtilen iterasyon sayısı kadar sonra üretilen desenlerdir.

Çizelge 5.1 : HYSA parametre değerleri.

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

Her bir desenin elde edilmesinden sonra HYSA'ya ilgili U, X[0] ve Y[0] başlangıç koşuları ve parametreler tekrar yüklenmiş ve öykünleme baştan başlatılmıştır.



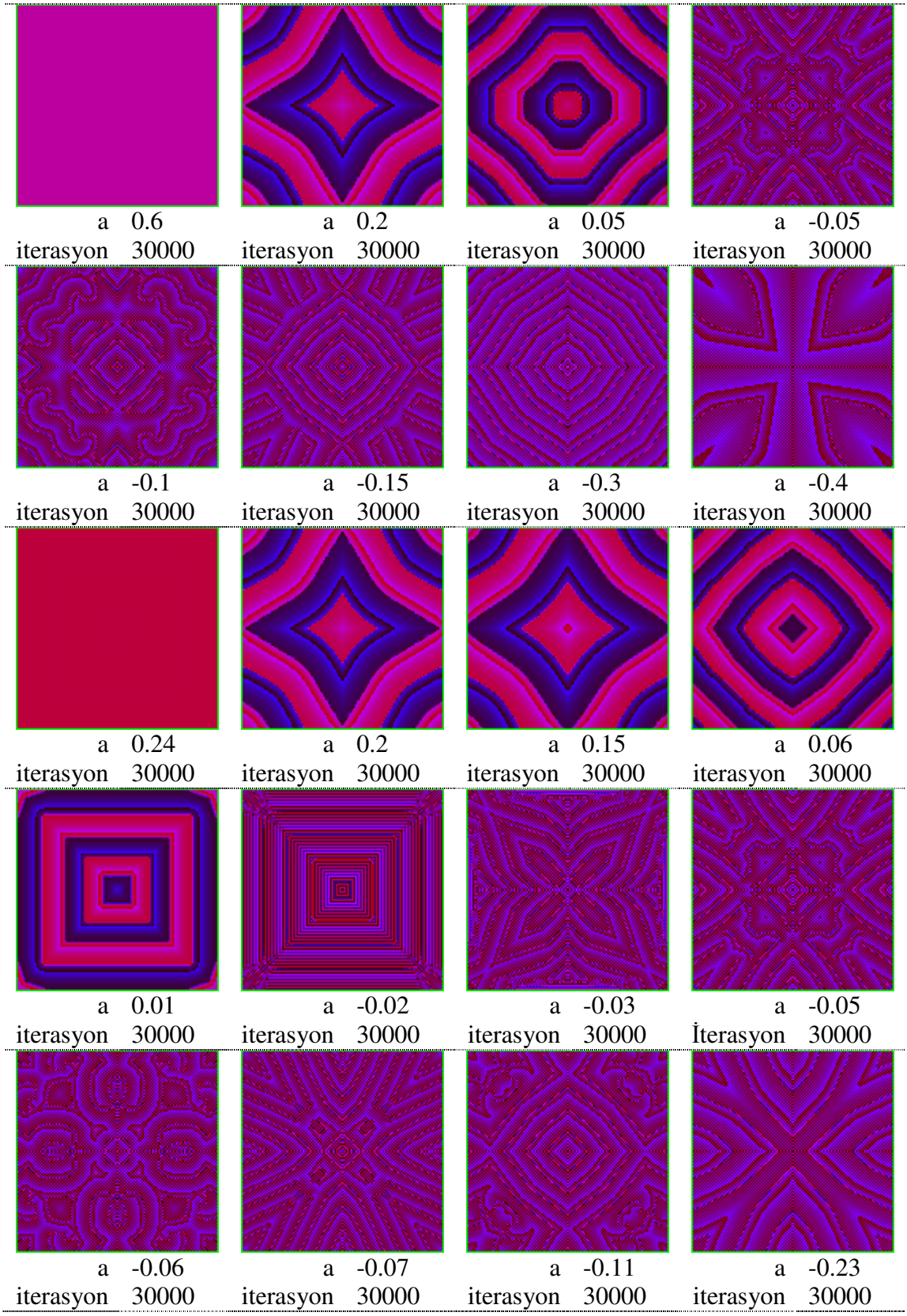
m -2 iterasyon 20000 | m -8 iterasyon 25000

Şekil 5.2 : Elde edilen dalgalar ve desenler.

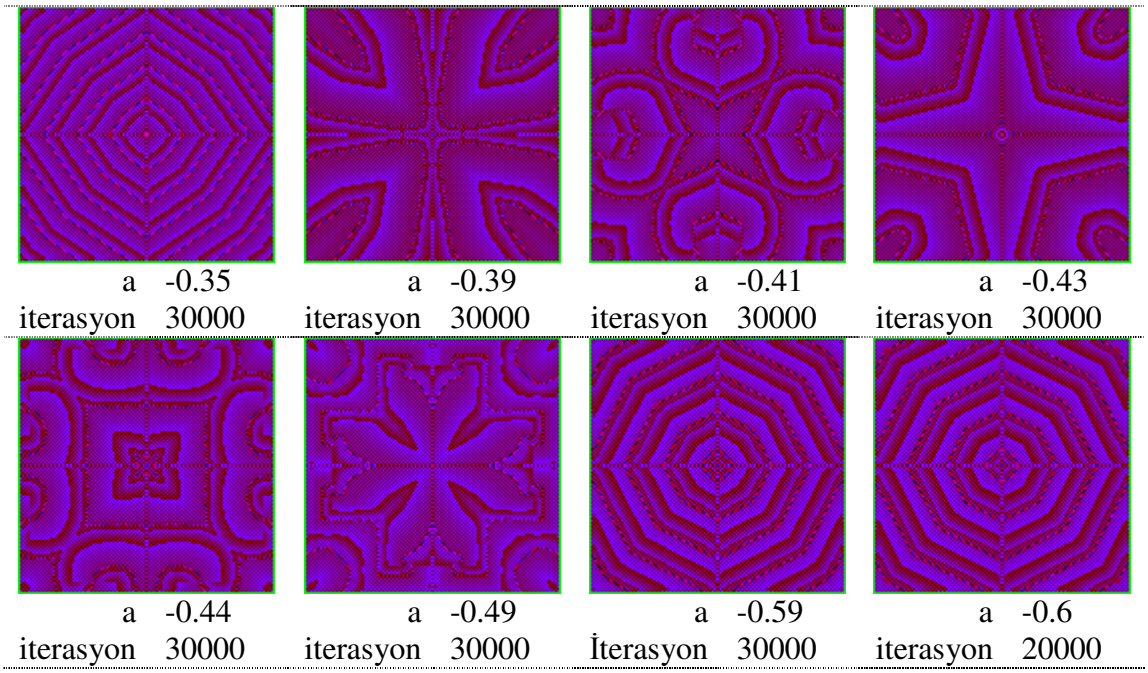
Şekil 5.1'deki U, X[0] ve Y[0] başlangıç koşulları ve Çizelge 5.2'deki parametre değerleri ile yapılan öykünmeler sonucunda Şekil 5.3 ve Şekil 5.4'deki sonuçlar elde edilmiştir.

Çizelge 5.2 : HYSA parametre değerleri.

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

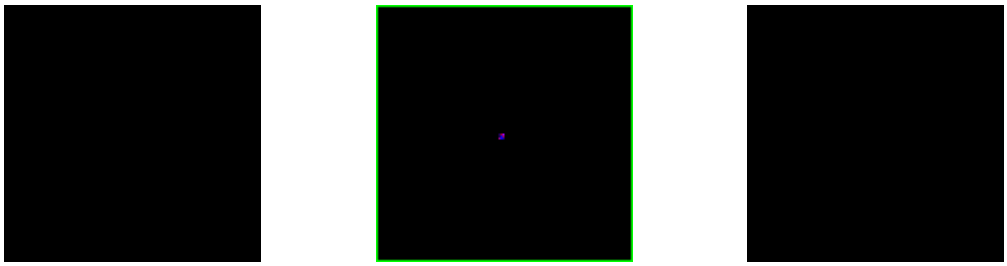


Şekil 5.3 : Elde edilen dalgalar ve desenler.



Şekil 5.4 : Elde edilen dalgalar ve desenler

Şekil 5.2, Şekil 5.3 ve Şekil 5.4'de HYSA'nın dört köşesindeki hücre dalga kaynağı olarak davranmaktadır. Bu şekilde üretiken desenlerin çift simetri eksenine sahip olduğu görülmektedir. Devam eden örnekler için Şekil 5.5'deki başlangıç koşuları kullanılmıştır. Bu başlangıç koşullarının Şekil 5.1'dekinden farkı X[0] matrisindedir. X başlagıç matrisinin yaklaşık orta bölgesindeki dokuz hücreye çok küçük başlangı değerleri atanmıştır.



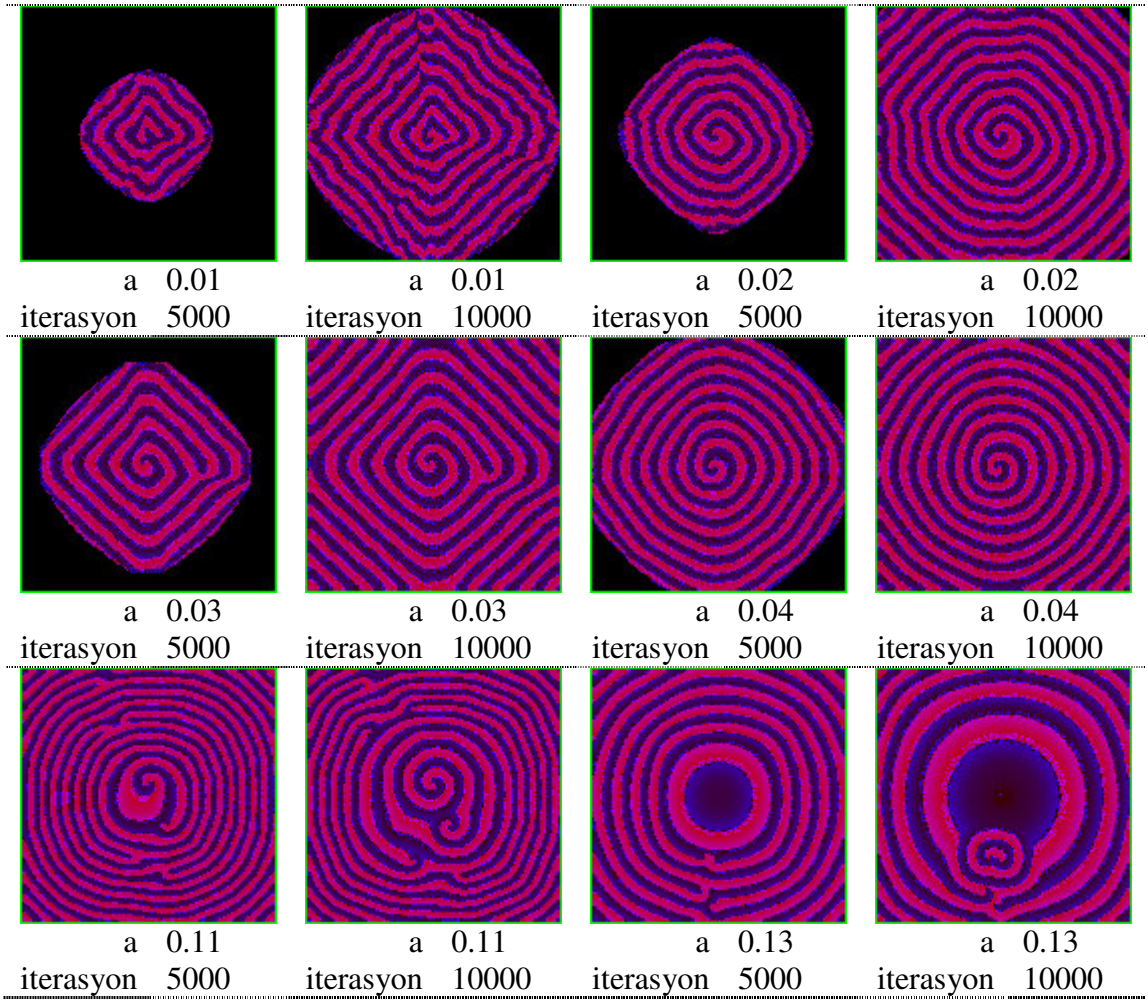
U X[0] Y[0]

Şekil 5.5 : Başlangıç koşulları.

Şekil 5.5, Şekil 5.6 ve Şekil 5.7'deki örneklerde Çizelge 5.3'deki parametreler uygulanmıştır x_{sabit} değerinin 0 yapılmasına dikkat edilmelidir. Böylece ortadan bşlayan dalgalar pasif hücrelere temas edinceya kadar pasif hücrelerde kaynak etkisi gözlenmemektedir.

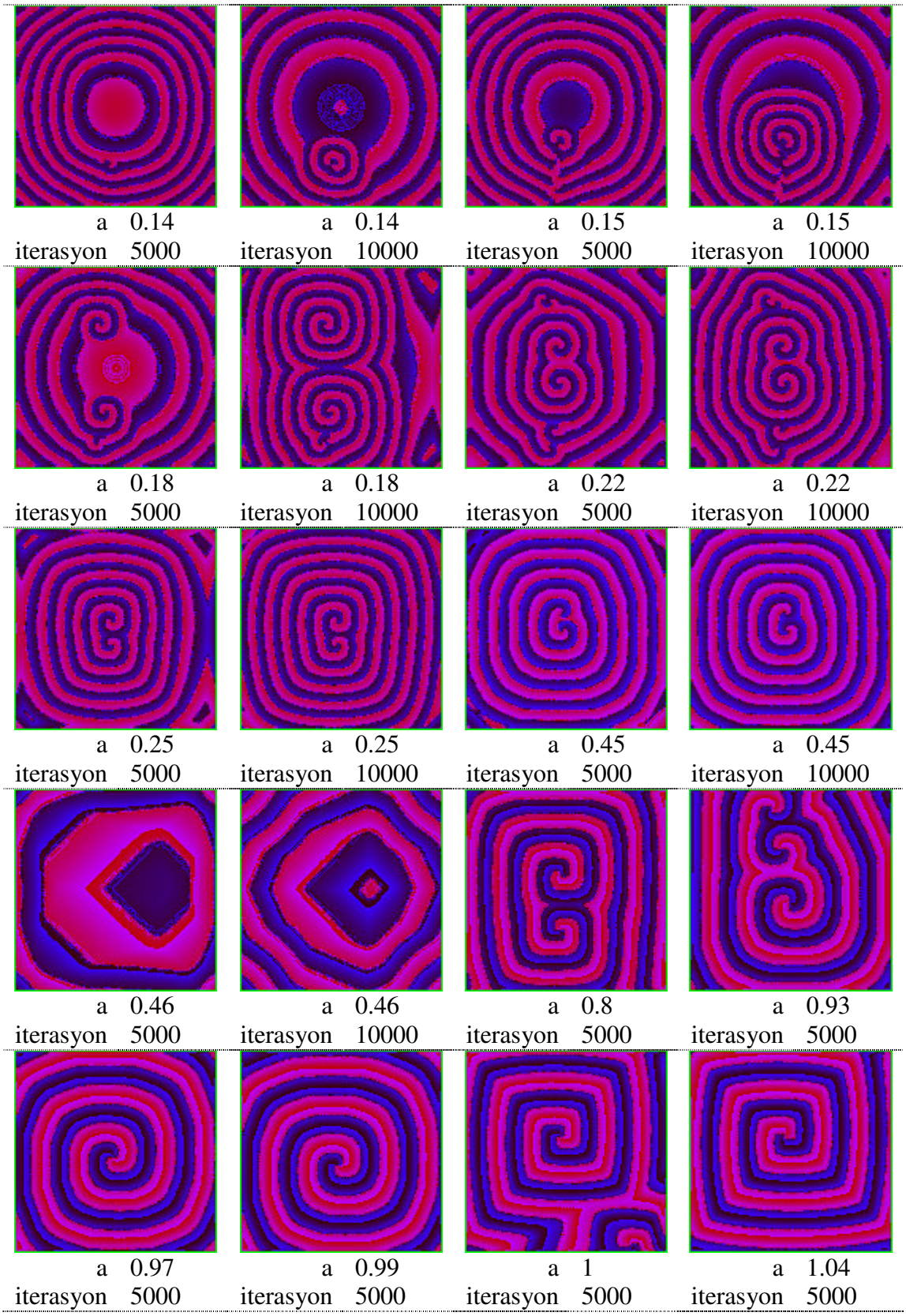
Çizelge 5.3 : HYSA parametre değerleri.

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



Şekil 5.6 : Elde edilen dalgalar ve desenler.

Bu örneklerde, başlangıç koşullarında orta bölgedeki farklılığın spiral dalgaları başlatabileceği saptanmıştır. Bu özel başlangıç koşullarının tespiti de yine tarama sürecleri, analizlerle ve çeşitli denemelerle yapılmıştır.

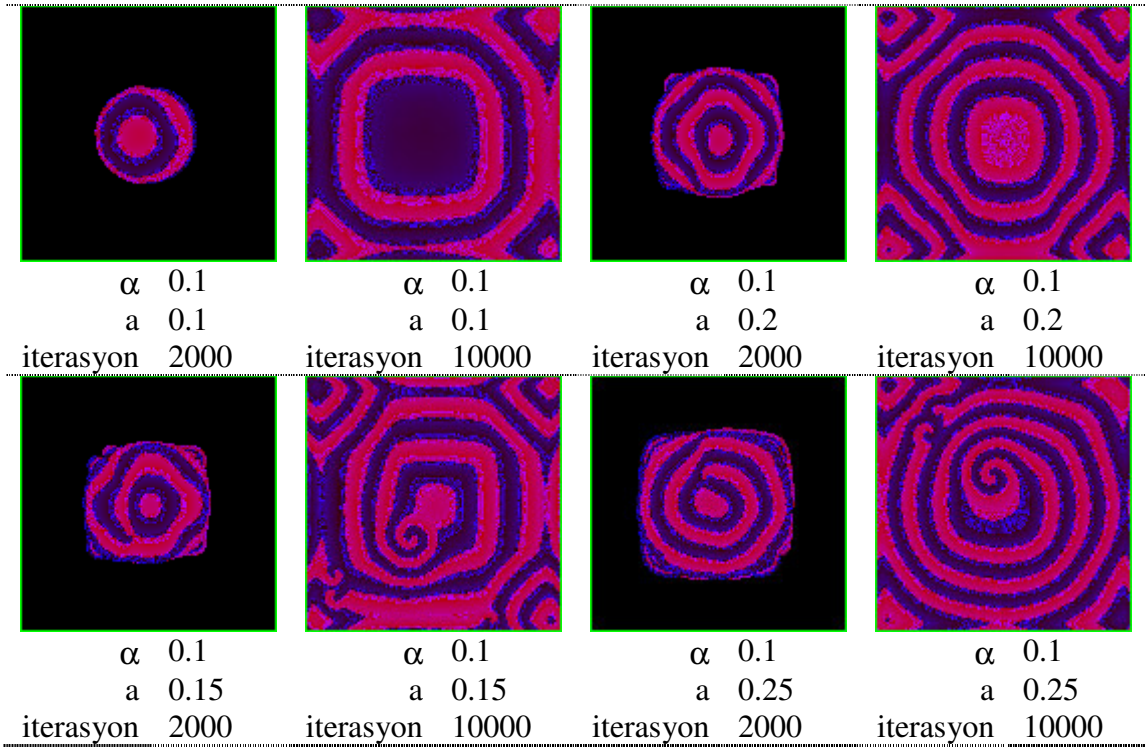


Şekil 5.7 : Elde edilen dalgalar ve desenler.

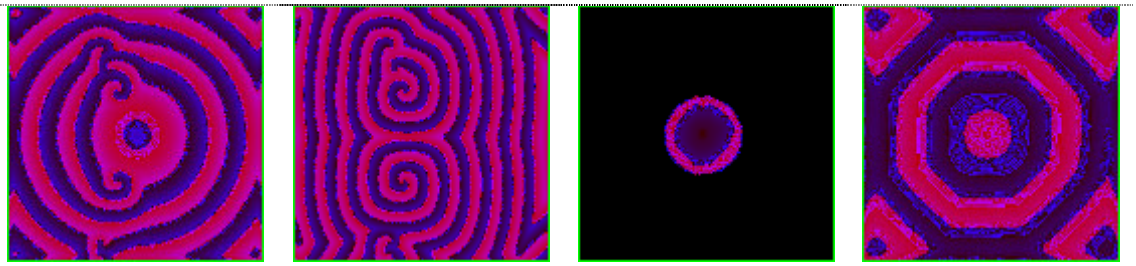
Şekil 5.5'deki başlangıç koşulları Çizelge 5.4'deki parametre değerleri ile denendiğinde elde edilen sonuçlar Şekil 5.8 – Şekil 5.12'de verilmektedir.

Çizelge 5.4 : HYSA parametre değerleri.

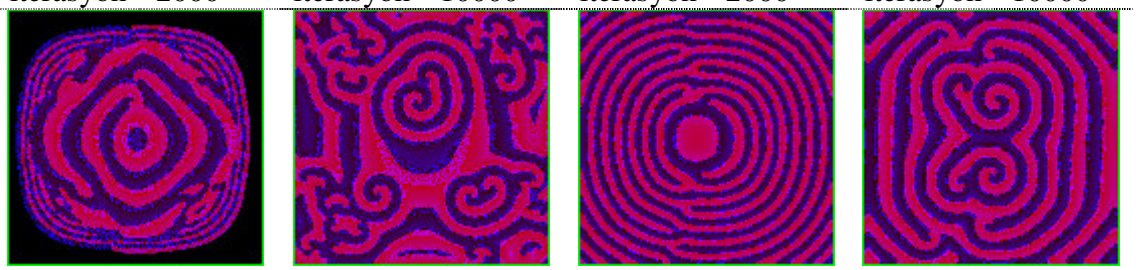
Parametre	Değeri
α	değişken
β	-4
ε	0.1
σ	-0.1
a	değişken
m	-20
λ	1.05
T	0.08
x_{sabit}	0



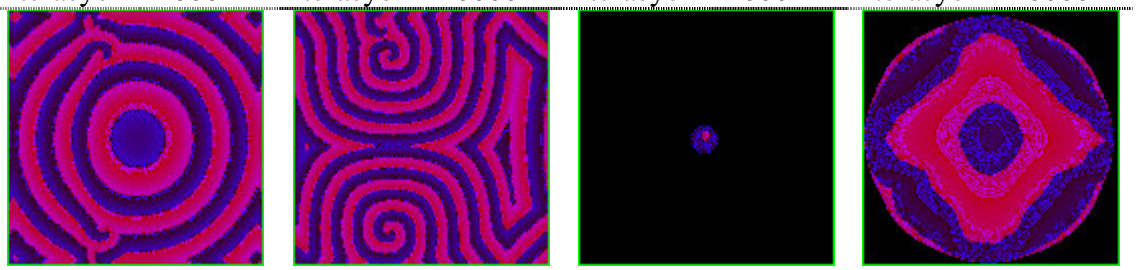
Şekil 5.8 : Elde edilen dalgalar ve desenler.



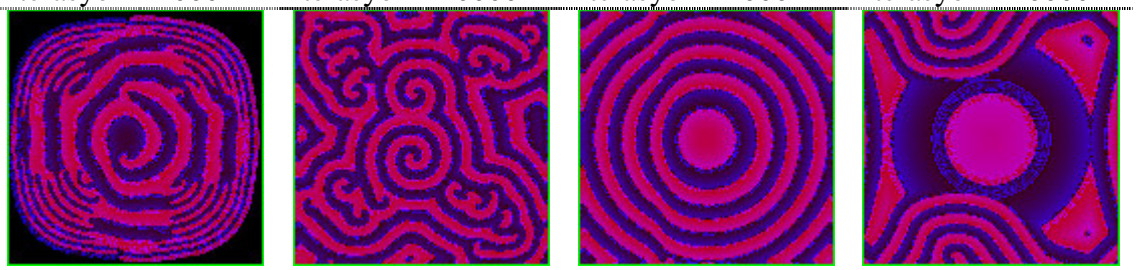
α 0.3 a 0.1 iterasyon 2000	α 0.3 a 0.1 iterasyon 10000	α 0.1 a 0.05 iterasyon 2000	α 0.1 a 0.05 iterasyon 10000



α 0.15 a 0.05 iterasyon 2000	α 0.15 a 0.05 iterasyon 10000	α 0.2 a 0.05 iterasyon 2000	α 0.2 a 0.05 iterasyon 10000

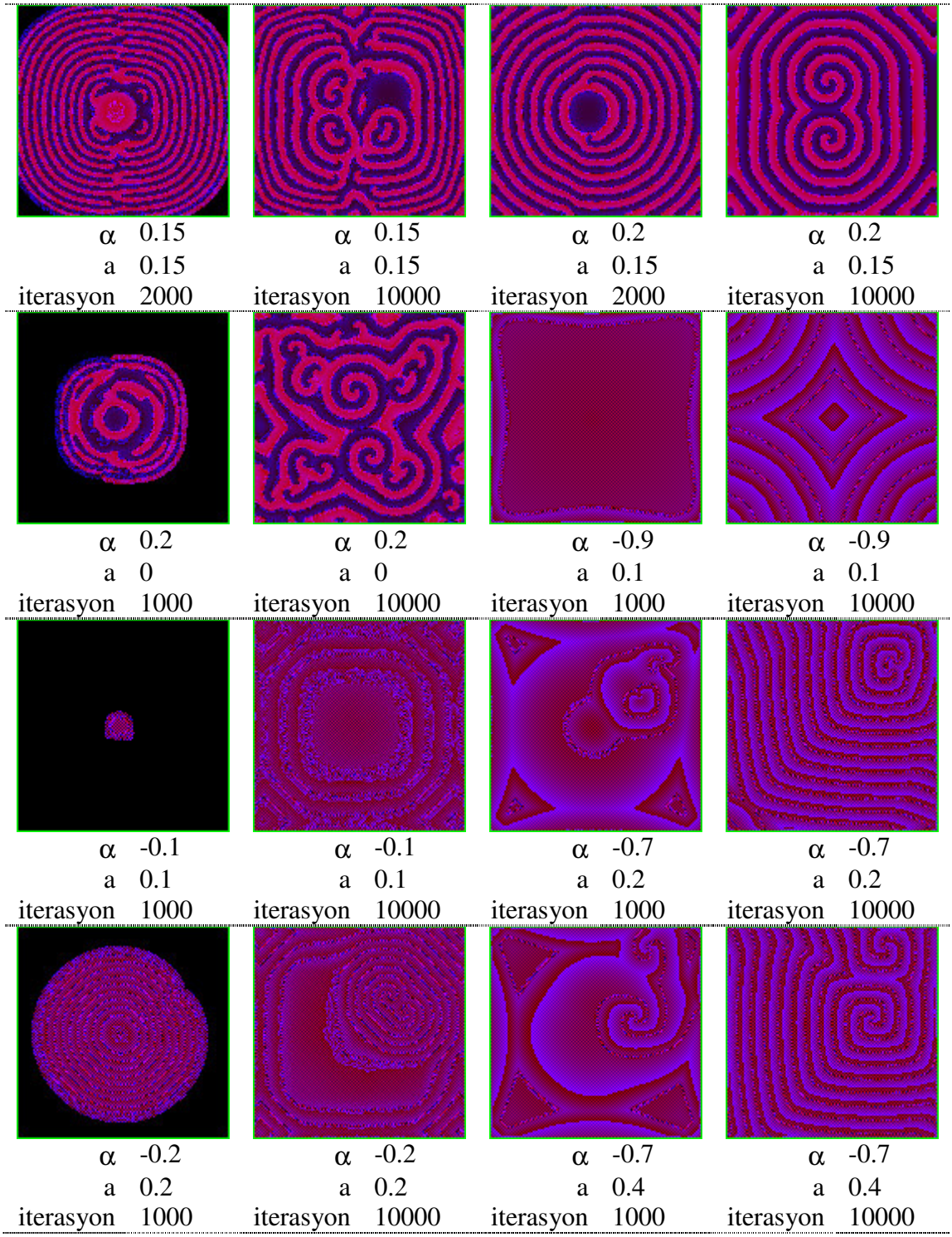


α 0.3 a 0.05 İterasyon 2000	α 0.3 a 0.05 iterasyon 10000	α 0.05 a 0.1 iterasyon 2000	α 0.05 a 0.1 iterasyon 10000

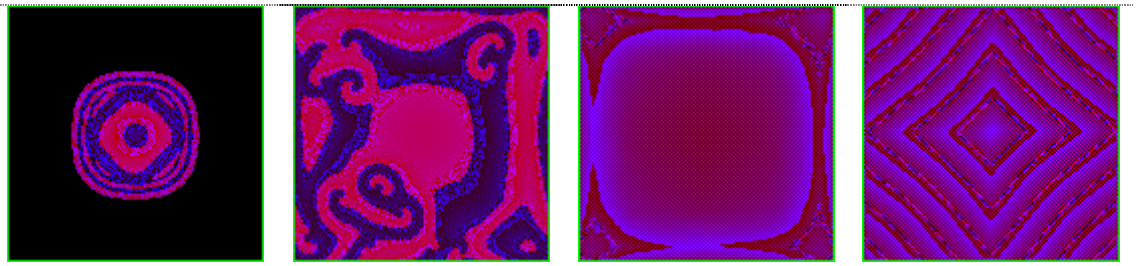


α 0.15 a 0.1 iterasyon 2000	α 0.15 a 0.1 iterasyon 10000	α 0.25 a 0.1 iterasyon 2000	α 0.25 a 0.1 iterasyon 10000

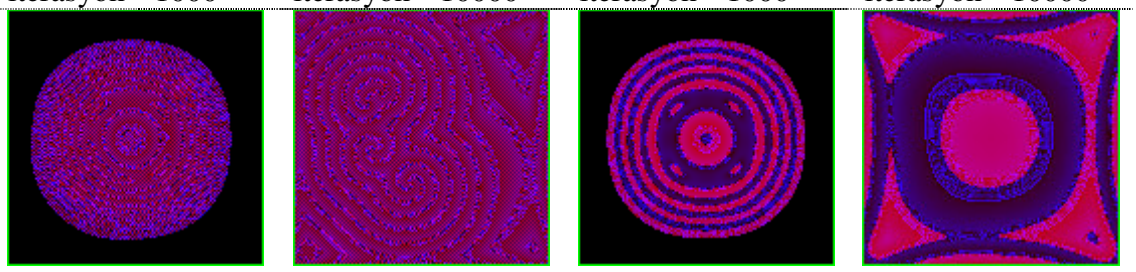
Şekil 5.9 : Elde edilen dalgalar ve desenler.



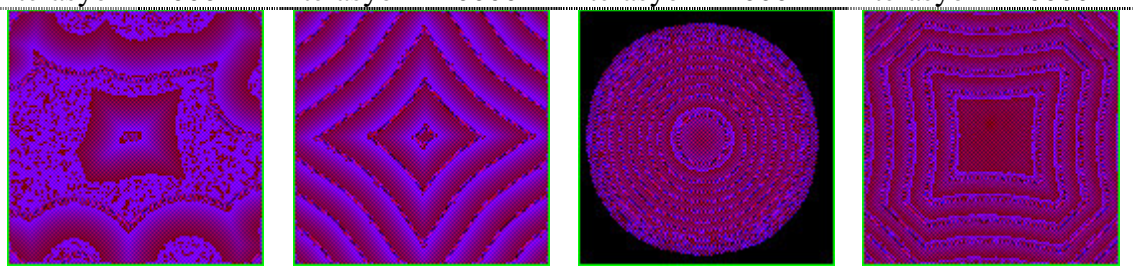
Şekil 5.10 : Elde edilen dalgalar ve desenler.



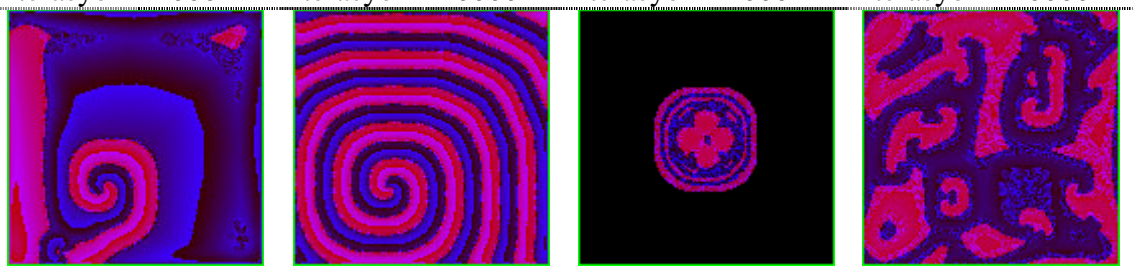
α 0.2 a -0.2 iterasyon 1000	α 0.2 a -0.2 iterasyon 10000	α -0.8 a -0.5 iterasyon 1000	α -0.8 a -0.5 iterasyon 10000



α -0.3 a -0.5 iterasyon 1000	α -0.3 a -0.5 iterasyon 10000	α 0.3 a -0.5 iterasyon 1000	α 0.3 a -0.5 iterasyon 10000

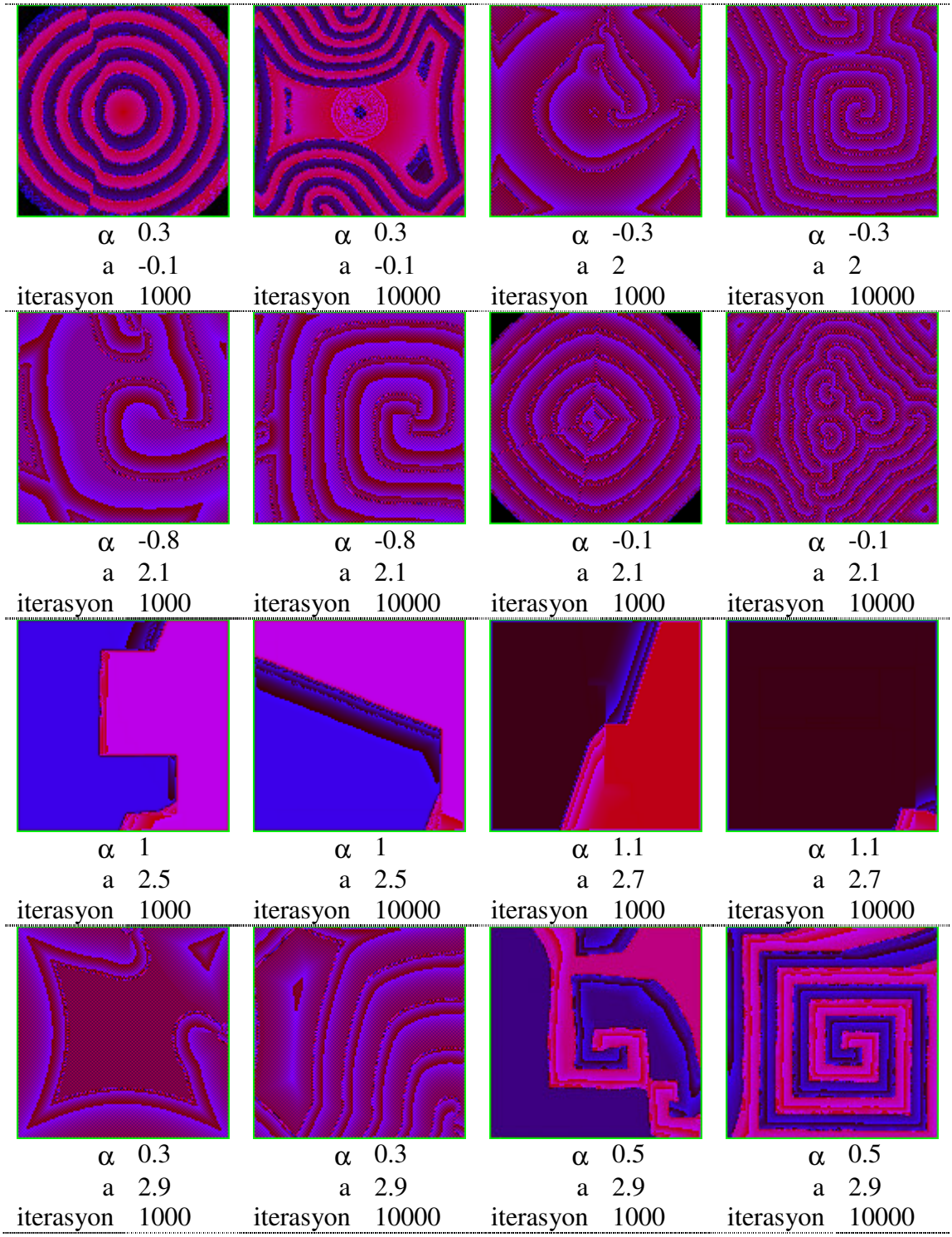


α -0.6 a -0.4 iterasyon 1000	α -0.6 a -0.4 iterasyon 10000	α -0.3 a -0.4 iterasyon 1000	α -0.3 a -0.4 iterasyon 10000



α 1 a -0.4 iterasyon 1000	α 1 a -0.4 iterasyon 10000	α 0.2 a -0.3 iterasyon 1000	α 0.2 a -0.3 iterasyon 10000

Şekil 5.11 : Elde edilen dalgalar ve desenler.



Şekil 5.12 : Elde edilen dalgalar ve desenler.

6. GERÇEKLENEN RO-HYSA'NIN YOL BULMA UYGULAMALARI

Bölüm 5. 'te yapılan gözlemler sonucunda oluşan fikirler, yol bulma uygulamaları için yürüyen dalgaların öncelikle kullanılmasını sevketmiştir. Bu bölümde sırasıyla, yürüyen dalgaların kullanıldığı iki ayrı yol bulma algoritması anlatılmış ve bunlardan ikincisinin robot uygulaması sunulmuştur.

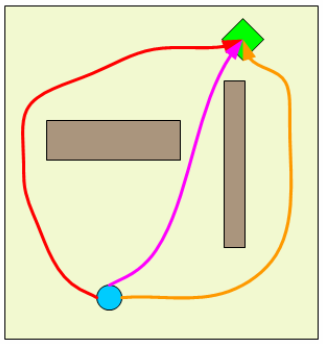
6.1 Dalga Çeperi Difüzyonuna Dayalı Algoritma

Gerçeklenen HSYA ve onu barındıran sistemin aktif dalga uygulamaları geliştirmek, dalga dinamiklerini incelemek gibi bir çok görevde kullanılabilecek stabil ve hızlı bir sistem olduğu ortaya konmuştur. Aktif dalgaların sayısal simülasyonu için gereken işlemlerin FPGA üzerinde hızla gerçekleştirilmesinin getirdiği en büyük avantaj dalgaların istenen dinamiğini sağlayan parametrelerin tarama yöntemi ile saptanabilmesi olmuştur. Bu taramalar sonucunda bilgisayar simülasyonunda tespiti oldukça zor olan spiral dalgaların tespiti de gerçeklenmiştir. Ayrıca FPGA üzerindeki ağa denetleyici bilgisayardaki MATLAB ile erişilebilmesi sayesinde spiral dalgaların oluşmasına sebep olan başlangıç koşulları kaydedilebilmiştir. Bu kayıt aynı spiral dalgaların istendiğinde yeniden üretilebilmesini sağlamıştır.

Bu önemli keşifle birlikte, projenin amacına yönelik de tespitler yapılmıştır. Yine denetleyici bilgisayarın istendiğinde HSYA'nın resmini alabilmesi, saklayabilmesi ve işleyebilmesi sayesinde dalga çeperlerinin ilerleyişi temel alınarak yol bulma problemi için temel bir algoritmanın geliştirilmesi sağlanmıştır. Bu uygulanan algoritma ile HSYA üzerindeki herhangi bir noktadan belirlenen bir noktaya olan tüm yolların paralel olarak üretildiği gözlenmiştir.

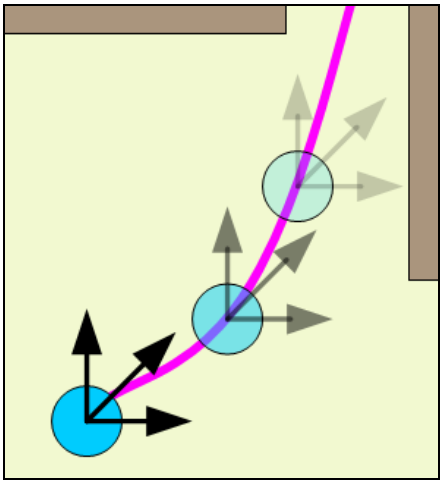
Hedefe götüren yolun saptanması mobil robotlar için hayati problemlerden biridir. Şekil 6.1 bu problemi tasvir etmektedir. Mavi renkli obje robot, yeşil renkli bölge de robotun ulaşması gereken hedeftir. Bu durumda şekilde kırmızı, pembe ve turuncu oklar ile gösterilen üç yoldan hangisinin seyir süresi daha kısadır sorusu hemen ilk akla gelen problemdir. Bu üç alternatif yol verilmediği taktirde mavi objeden yeşil

bölgeye uygun kısa bir yol tanımlanması istendiğinde buna cevap verebilecek türlü yöntemler ve algoritmalar literatürde yerini almıştır.



Şekil 6.1 : Yol problemi

Mavi renkli robot için yeşile giden, engeller ile kesişmeyen, kısa bir rotanın tayini başlangıç için gereklidir. Bu rota saptandığı anda bu kez robotun rota üzerinde hareket edebilmesi için zaman içerisinde hangi manevraları yapması gerektiğinin hesaplanması da ayrı bir problemdir. Şekil 6.2 de bu yönelim problemini açıklamaktadır. Şekilde tasvir edildiği gibi robotun manevra kabiliyeti dahilinde bulunan manevraların hangisini yaparak rotayı izleyeceği de bir sorundur.

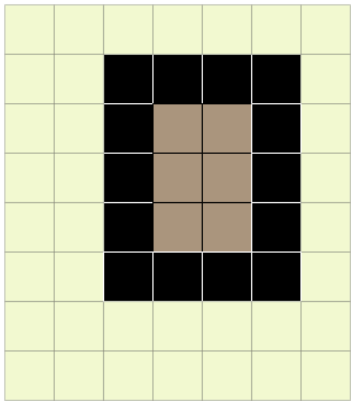


Şekil 6.2 : Yönelim problemi

HYSA ağının bu iki soruna sunduğu çözüm, bulunan yöntemler sayesinde birlikte gerçekleşmektedir. Çözülmeye çalışılan problem gerçek dünya ile ilişkilidir. Bu yüzden problemin gerçek dünya değişkenlerinin kullanılan sistemdeki değişkenlere oturtulması gerekmektedir. 128 x 128 boyutlu ağ robotun üzerinde hareket edeceği, hedefi ve başlangıç noktasını, engelleri barındıran düzlemsel uzaya karşı düşmektedir. Ağın boyutu bu düzlemsel uzayın modellenmesinde kullanılan bir

çözünürlük parametresidir. Örneğin gerçek uzay 256 x 256 cm boyutlarında ise HSYA üzerindeki her bir nöron gerçek uzayda 2 x 2 cm'lik bir bölgeye karşı düşer. Sistemde çözünürlük sabittir, fakat gerçek uzayın boyutları istenildiği şekilde seçilebilir.

Denetlenebilir HSYA'da daha önce bir x_{sabit} değeri tanımlanmıştır. Pasif hücreler NPE ile simüle edildiğinde değerleri değişmez, x_{sabit} olarak kalır. Pasif hücreler, dalgaların yayılmasını sağlayan dinamik davranışı göstermezler. Ve tüm bunlar sayesinde ağda sınır koşulunun oluşturulmasını sağlarlar. Dolayısı ile robotun gezeceği uzayda bulunan engel objeler ya da duvarlar ağdaki ilgili nöronların pasif olarak işaretlenmesi ile oluşturulmuş olur. Şekil 6.3'de gösterilen yapı HSYA üzerinde bir engel modellini vermektedir. Açık sarı renkli norönlar uygun başlangıç x değerleri atanmış olanları temsil ederken siyah renkli nöronlar pasif hücrleri temsil etmektedir. Kahverengi nöronların ise x başlangıç değerlerinin ne olduğu önemli değildir. Çünkü aktif dalga açık sarı nöronlar üzerinde oluşturulup yayılmakta, siyah nöronlar sebebi ile dalga etkisi kahverengi nöronlara ulaşmamaktadır.



Şekil 6.3 : HSYA'da bir engel

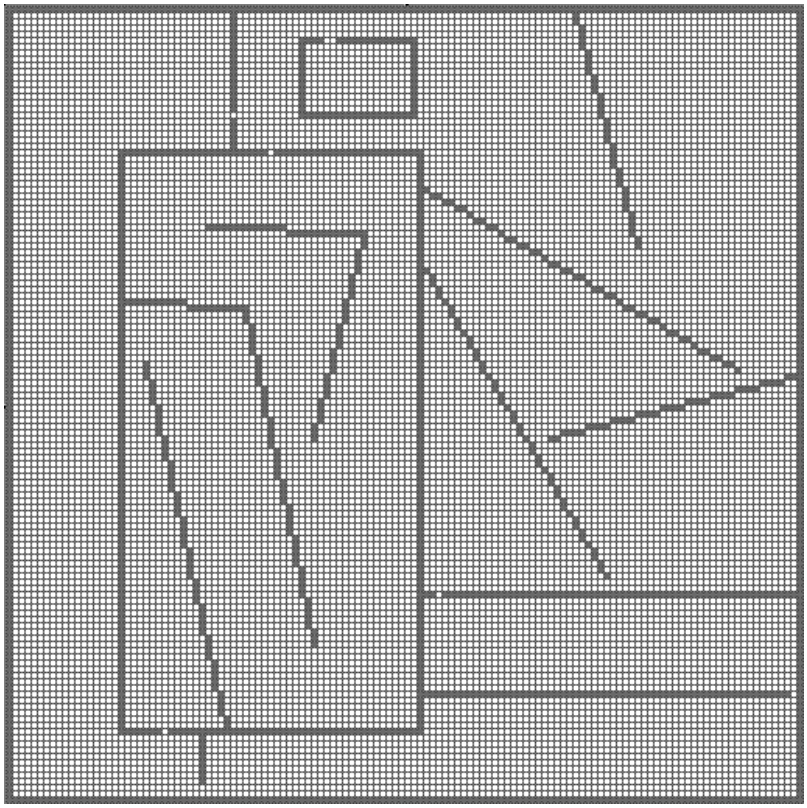
Ayrıca ağın çalışabilmesi için belirtilen sınır oluşturma yöntemi ile tüm ağın sınırları da belirlenmelidir. Yani 128 x 128 nöronluk tüm ağın ilk son sütun ve satırlarına da pasif etiketi atanmalıdır. Bunun yapılmasını gerekli kılan FPGA içine gömülen donanımın özelliğidir.

Yol probleminin çözümü dalgaların tek bir kaynağı olması, bu kaynak nöronun da matematiksel modelde u değişkeni ile oluşturulması önerilmiştir. Düzgün bir dalga formu için sisteme Çizelge 6.1'deki parametreler yüklenmiştir.

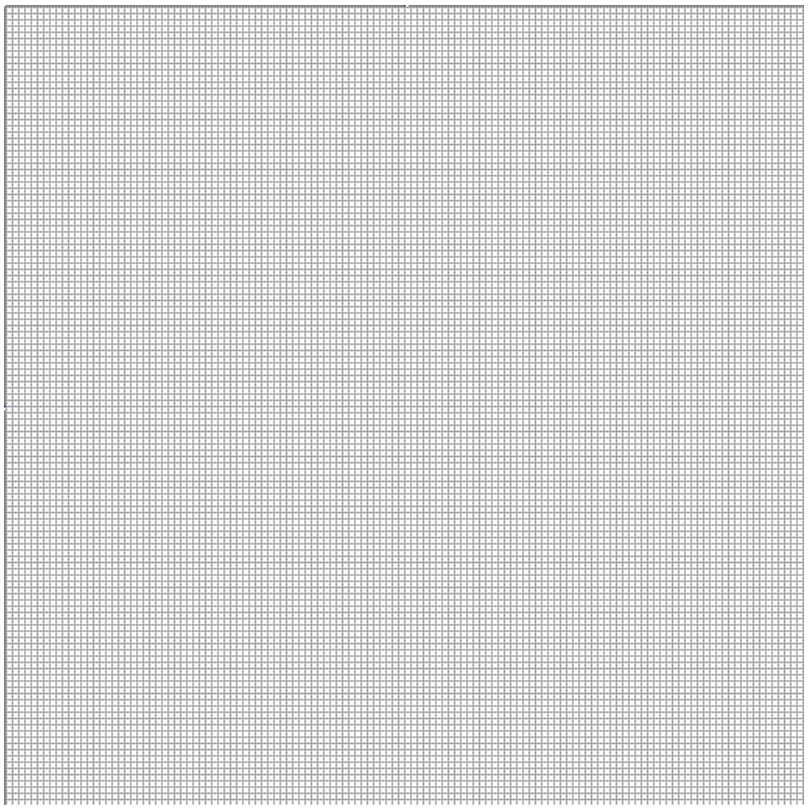
Çizelge 6.1 : Problemin tespiti için belirlenen uygun parametreler.

Parametre Adı	Değeri
a(tüm yönler)	1.1
λ	1.05
m	-20
α	2.80
β	-4
ε	0.1
σ	-0.1
T	0.08

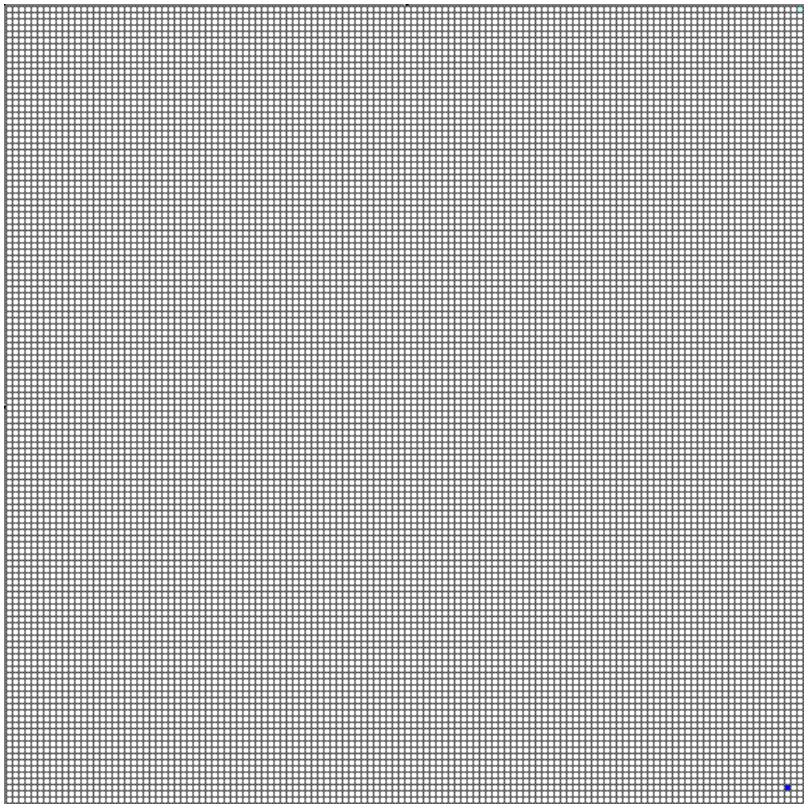
Arenanın yerine geçecek model bir uzay tasarlanmış ve bu uzay x değişkenleri olarak FPGA'daki ağa yüklenmiştir. Şekil 6.4'de x değişken haritası verilmiştir. Beyaz norönlara 0.00 başlangıç değeri atanmış, koyu renkli nöronlar pasif hücre olarak etiketlenmiştir.Şekil 6.5'de yüklenen y değişkenlerini göstermektedir. Şekilden anlaşıldığı gibi y değeri olarak bütün ağa 0.00 değeri yüklenmiştir.Şekil 6.6'da ise biri hariç tüm nöron değerleri 0.00 olan bir u değişken matrisi gösterilmiştir. Ağa yüklenen u değişkenleri bunlardır. Burada 0'dan farklı olan değer 2'dir. Şekilde mavi renk ile temsil edilmiştir ve matrisin sağ alt köşesine yakın bir yerdedir.



Şekil 6.4 : X[0] başlangıç matrisi



Şekil 6.5 : Y[0] başlangıç matrisi



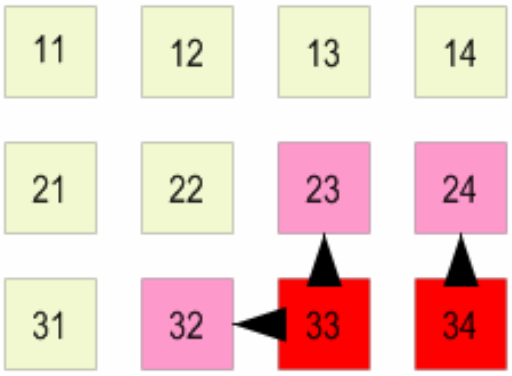
Şekil 6.6 : U başlangıç matrisi

FPGA üzerindeki HSYA'ya denetleyici bilgisayar tarafından bu değerlerin yüklenmesi halinde ilk iterasyondan itibaren u değeri 0'dan büyük olan nörondan başlayan bir yürüyen dalga yayılmaya başlamaktadır.

Algoritmanın gerçeklenmesine ait işlem yükü, dalgaların oluşturulması ve yayılmasından ayrı olarak denetleyici bilgisayar üzerinde gerçekleştirilmektedir. Bunun için FPGA ve bilgisayar arasında yoğun bir şekilde ağ değişken verisinin aktarılması gerekmektedir.

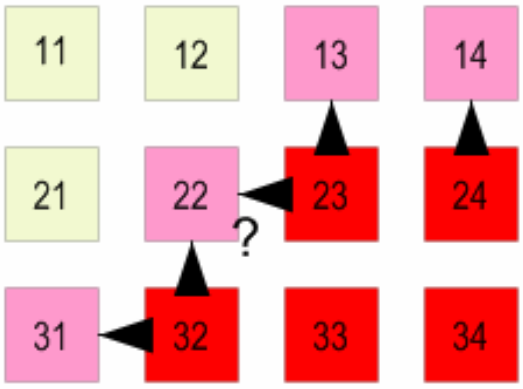
Algoritma temel olarak ağ üzerinde yayılan dalga çeperinin gradyentinin vektörel olarak hafızalanmasına dayanmaktadır. Başlangıçta durağanlıkta olan HSYA'nın her bir norönu üzerine bir dalgaya ait çeper geldiği anda bu dalganın geliş yönü hesaplanabilir. Bu dalga çeperinin o nöronun bulunduğu yerdeki normal vektörüdür. Bu normal vektörü her hücre için yalnız bir kere hesaplanmalıdır. Çünkü ağ başlangıçta durağanlıkta, dalga yayılmaya başladıktan sonra da sürekli değişim içindedir. Bizim için problemin çözümünde işe yarayacak dalga çeperi, herhangi bir nöronun üzerinden geçen ilk dalga çeperidir. İlk dalga çeperinin yakalanması herhangi bir nöron için ilk değer değişiminin algılanması anlamına gelmektedir.

Bir nörona dalga çeperi ulaştığının saptanmasının ardından dalga yönü de o nöronun dört komşusunun hangisinden dalganın ulaştığının tespiti ile belirlenir. Dört komşuluk ile birbirine bağlı olan nöronların üzerine de dalga ancak bu dört yönden birinden ulaşmış olmaktadır. Şekil 6.7'de kırmızı ile gösterilen sağ alt nöronlardan yayılan dalganın pembe ile gösterilen nöronlara hangi vektörler ile ulaştığı gösterilmiştir. Pembe nöronlar henüz daha o anda başlangıçtan farklı değer almıştır. Yani dalga çeperi o anda o norönlar üzerindedir.



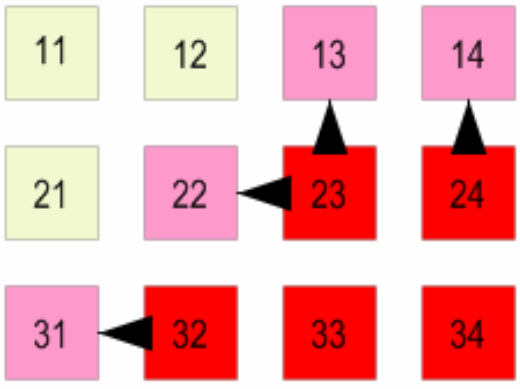
Şekil 6.7 : Dalganın yayılışı

Dalga, Şekil 6.8'deki gibi yayılmasını sürdürdüğünde aynı yöntem ile vektörler saptanırken 22 etiketli nöronda bir problem ile karşılaşılmıştır. Bu nörona dalga hangi komşusundan ulaşmaktadır. Bunun tespiti için hemen hangi komşusunun x genliğinin daha büyük olduğuna bakılabilir. Bu daha doğru bir yöntem olmakla birlikte, işlem süresini uzatmaktadır.



Şekil 6.8 : Dört komşuluk problemi.

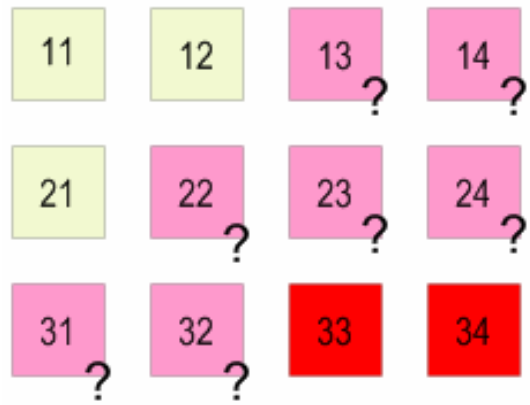
Bu durumda komşulara öncelik atayarak da bir çözüme varılması denenmiştir. Önerilen algoritmada öncelik sırası: sağ komşu > üst komşu > sol komşu > alt komşu şeklindedir. Bu öncelik atama yöntemi algoritma denetleyici bilgisayar üzerinde koşarken gereksiz gibi görünmesine rağmen, FPGA'ya alındığında daha az işlem yükü demek olduğundan, gayet anlamlı hale gelecektir. Dolayısıyla Şekil 6.8'deki problemin çözümü Şekil 6.9'daki gibi olmaktadır.



Şekil 6.9 : Öncelikli komşuluk ile bulunan çözüm.

Böylece u değişkeni ile belirlenen başlangıç nöronundan dalganın nasıl yayıldığı, dalga çeperinin her ilerleyişinde ilerleme vektörünün saptanması ve bunun 128x128 boyutlu bir vektör matrisinde ilgili nörona denk düşen yere kaydedilmesi ile olur.

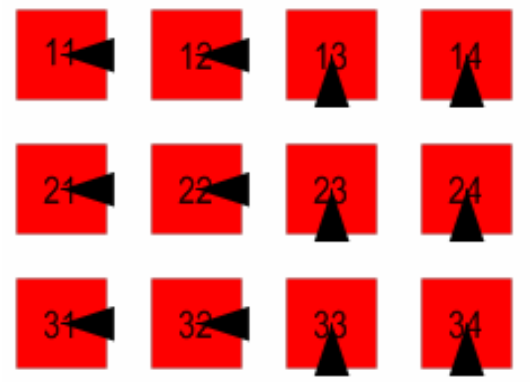
Bu algoritmada önemli olan bir nokta dalga çeperinin her bir nöron ilerleyişinde fark edilip vektörün üretilmesi gerektiğidir. Aksi taktirde Şekil 6.10'da gösterilen problem ile yüzleşilir.



Şekil 6.10 : Hızlı dalga ilerleyişi sorunu.

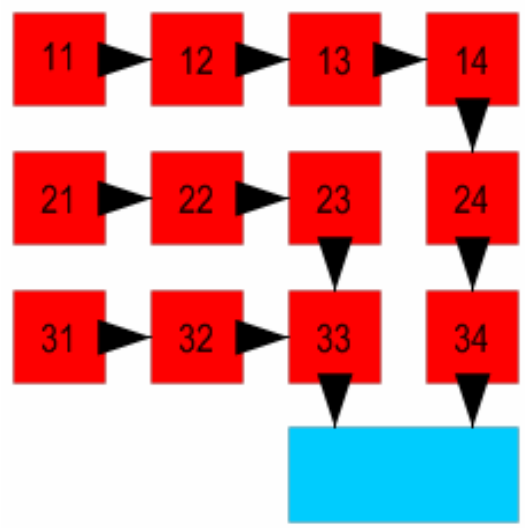
Dalganın ilerleyiş hızının en fazla her seferinde bir hücre kadar olacağını garanti edecek sıklıkla HSYA görüntüsü örneklenmelidir. Çizelge 6.1'de verilen parametreler dalganın bir iterasyonda en fazla bir hücre kadar yayılmasını sağlar. Denetleyici bilgisayarın HSYA görüntüsünü FPGA'dan her iterasyonda alması ile algoritmayı güvenli koşturacak datayı elde etmiş olur.

Denetleyici bilgisayar her iterasyonda gelen ağ görüntüsünü, yani X[k] matrisini, bir değişkende tutar. Aynı görüntünün bir önceki değeri, yani X[k-1] de bir başka değişkende tutulmaktadır. Algoritma koştuktan sonra yani vektörleri barındıran değişkene yeni vektörler işlendikten sonra şimdiki görüntü eski görüntünün üzerine yazılır ve FPGA'ya bir iterasyon daha koşturması komutu gönderilir. Bu süreç vektörleri tutan matrisin pasif hücreler hariç tamamıyla dolması ile son bulur. Şekil 6.11 örnek olarak gösterilen ağda son durumu vermektedir.



Şekil 6.11 : Tüm vektörler.

Bu hesaplanan vektörleri tersine döndürerek, herhangi bir nörondan bu döndürülmüş vektörleri takip ederek dalganın başladığı bölgeye ulaşırız. Şekil 6.12 bu örneği göstermektedir.

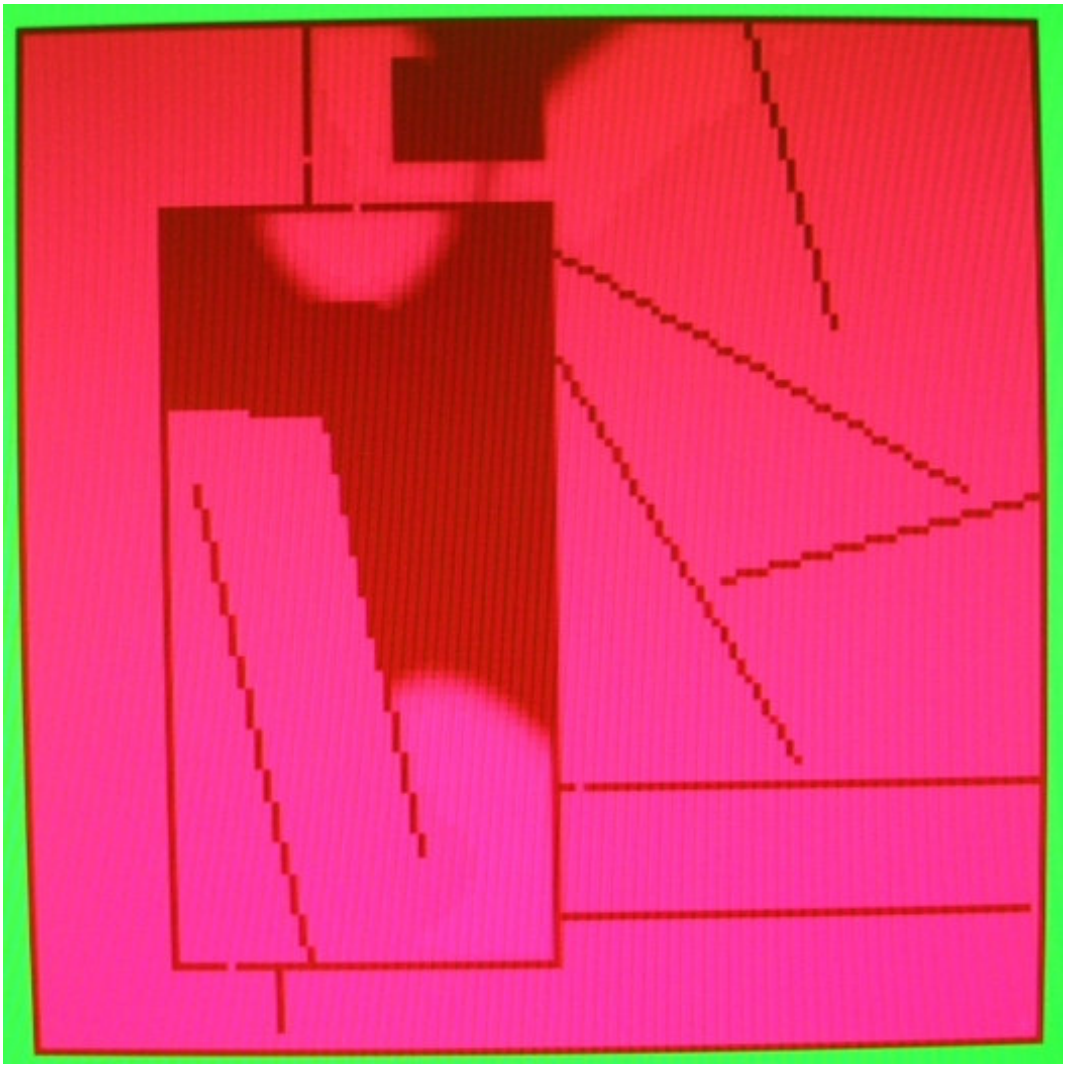


Şekil 6.12 : Başlangıca geri dönme.

Şekil 6.12'ye bakıldığında 11 nöronundan mavi bölgeye ulaşmak için okları takip etmek yeterlidir. Diğer tüm nöronlar gibi 23 ya da 31 nöronu için de bu geçerlidir. Bu algoritma ile elde edilen sonuç, ağ üzerindeki tüm diğer noktalardan başlangıç noktasına olan yolun her bir adımını veren vektörler dizisidir.

İşte robot için gerekli olan da tam olarak budur. Robotun gitmesi hedeflenen noktadan yayılan dalgaların ürettiği vektörler ters çevrilir. Robotun bulunduğu noktadan hedeflenen noktaya gitmesi için yapması gereken ilk olarak bulunduğu noktadaki vektörü izleyerek komşu nörona geçmesidir. Bundan sonraki adımlar ise vardığı noröndaki vektörü izleyerek bir diğer komşuya geçmesidir. Takip işlemi sonunda robot mutlaka dalgaların yayıldığı kaynağa varacaktır.

Şekil 6.13'de, Şekil 6.6'daki mavi ile belirtilen noktadan yayılmaya başlayan dalgaların ilerleyiş sürecinden bir kesit gösterilmiştir. Bu tür dalgalara, daha önce de anıldığı gibi, yürüyen dalga denmektedirç. Dalgaları taşıyan nöronlar başlangıç değerinden başka bir sonlu değere ulaşmakta ve o değer yaklaşıklığında kalmaktadır. Bu durum, sistemin en temel parçası olan relaksasyon osilatörünün (kare dalga osilatörünün) özel bir davranışıdır. Bu yüzden Şekil 6.13'deki ağ görüntüsünde sadece pozitif değerleri temsil eden kırmızı dalgalar görülmekte, negatif değerler görülmediği için mavi renkli dalga görünmemektedir.

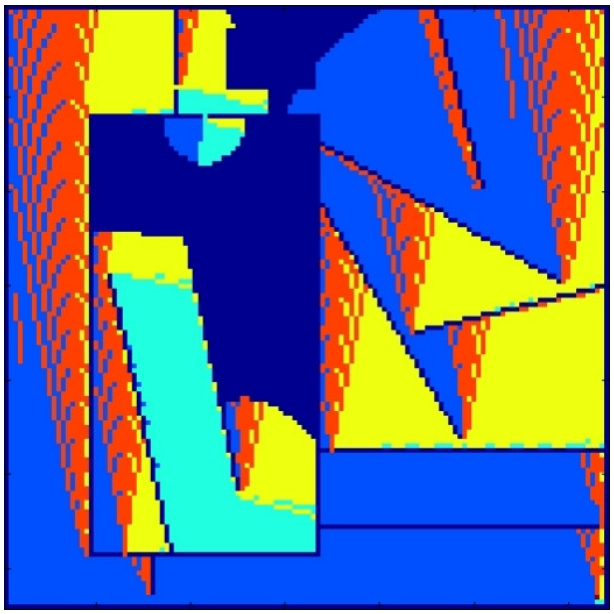


Şekil 6.13 : Dalgaların yayılması sürecinden bir görüntü.

Şekil 6.14'de ise ağın Şekil 6.13'deki haline gelinceye kadar üretilen vektörler gösterilmektedir. Haritada kullanılan 6 rengin anlamı Çizelge 6.2'de verilmiştir.

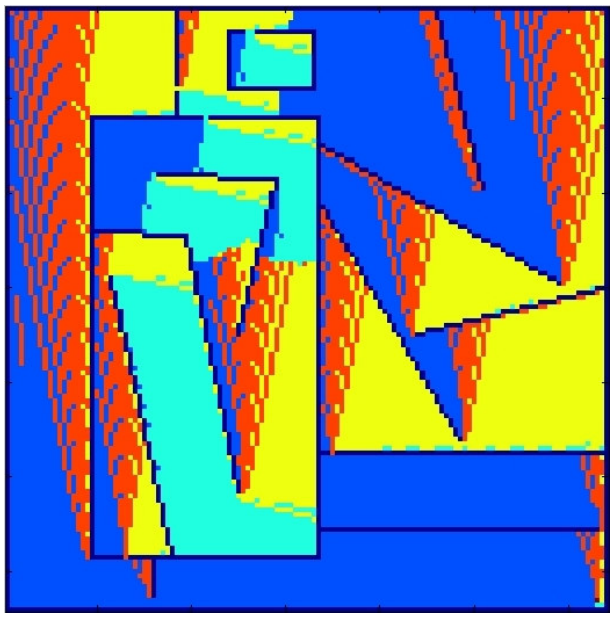
Çizelge 6.2 : Vektör haritasının kılavuzu.

Renk	Anlamı
Lacivert	Pasif nöron ya da henüz dalganın ulaşmadığı aktif nörol
Kırmızı	Dalga kaynağı olan nöron
Açık Mavi	Kuzey yönlü vektör
Koyu Mavi	Batı yönlü vektör
Turuncu	Güney yönlü vektör
Sarı	Doğu yönlü vektör



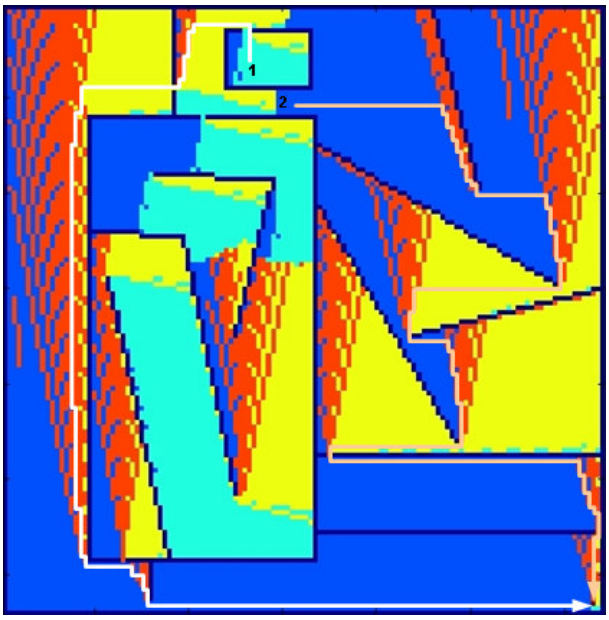
Şekil 6.14 : Vektör haritası, tamamlanmamış.

Şekil 6.15'de ise dalgaların tamamen yayılması beklenmiş ve vektör haritasının tamamı üretilmiştir. Bu andan sonra iterasyonlar sürdürülse dahi vektör haritasında değişiklik olmaz.



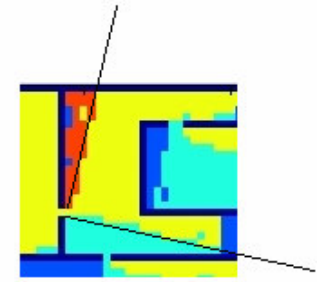
Şekil 6.15 : Vektör haritasının tamamı.

Şekil 6.16'da tüm vektör haritası üzerinde iki adet örnek yol gösterilmiştir. Hedef nöron sağ alt köşededir. Beyaz yol 1 numaralı nörondan hedefe giden en kısa yolu, pembe yol ise 2 numaralı nörondan hedefe giden en kısa yolu göstermektedir.

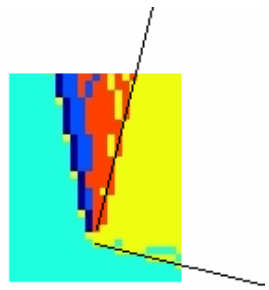


Şekil 6.16 : Örnek yollar: 1 ve 2'den başlayan.

Şekil 6.17 ve Şekil 6.18'de dalga dinamiğinin vektörler üzerindeki etkisine dikkat çekilmiştir. Geçitlerden geçen ya da köşelerden kıvrılan dalganın dağılımı benzerlik göstermektedir. Şekillerdeki siyah doğrular sarı renkli doğu yönlü vektörlerin turuncu renkli güney ve açık mavi renkli kuzey yönlü vektörlerden ayrıldığı sırtı göstermektedir. Faklı parametreler ile oluşturulan diğer aktif dalgaların geçit geçme ve köşe kıvrılma karakteristikleri farklıdır.



Şekil 6.17 : Geçitten sonra dalga yayılımı.



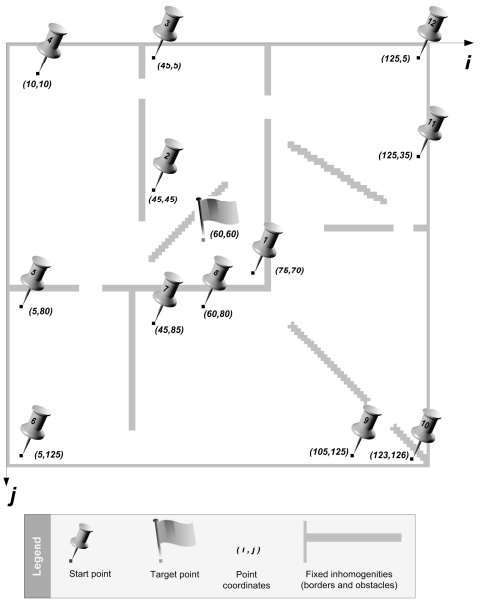
Şekil 6.18 : Köşeden sonra dalga yayılımı.

Algoritma denetleyici bilgisayar üzerinde koşturulduğu için tamamlanması uzun zaman almaktadır. Fakat aynı algoritma FPGA'ya gömüldüğünde hem çok kısa sürecek hem de yarı paralel gerçekleşecektir. Çünkü algoritmanın işlemleri aktif dalganın yayılması için yapılan işlemler ile iç içe yapılabilecektir. Bu algoritmanın gerçek zamanlı çalışma performansına erişmesi için FPGA'ya gömülmesi gerekmektedir.

Dalga Çeperi Difüzyonuna Dayalı Algoritma, taxicab geometrisinde en kısa yolu vermektedir. Taxicab geometrisinde iki nokta arasındak uzaklık, noktaların koordinatlarını mutlak farkının toplamı olarak tanımlıdır. Öklid uzaklığını yaklaşımı ile en kısa yol sonucunu verememektedir.

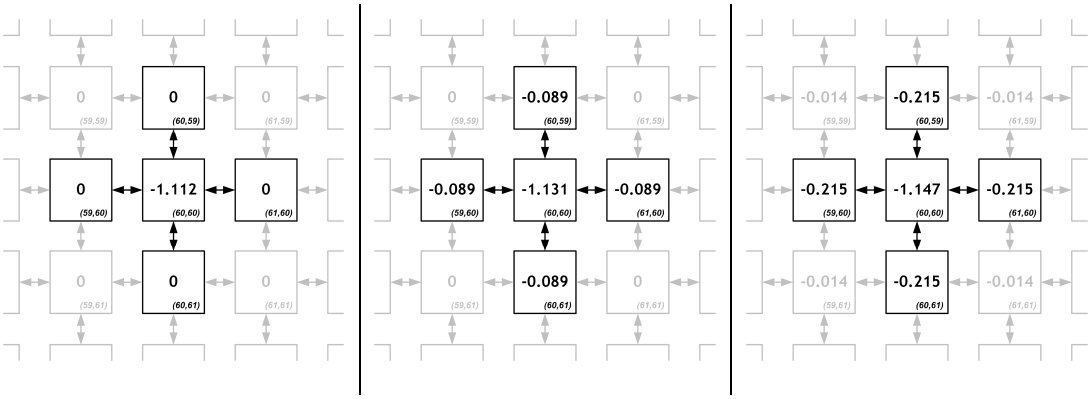
Dalga çeperinin hareketinin izleyen denetleyici bilgisayar olduğunda algoritmada küçük değişiklikler basitçe yapılabilmektedir. Bölümün başından buraya kadar dalga yayılmasında çeperin, bir hücreye transferi esnasında, kaynak hücrenin tespiti yapılırken bir öncelik kullanılmıştır. Algortimanın bu kısmı değiştirilerek daha iyi bir sonuç araştırılmıştır.

Bu örnek için Şekil 6.19'daki harita üzerinde çalışılmıştır. Bu harita X[0] matrisi olarak HYSA'ya yüklenmiştir. Ağın (60,60) hücresi dalga kaynağı olarak seçilmiş ve başlangıç değeri -1.122 olarak ayarlanmıştır. Haritadaki gri bölgelere gelen hücreler pasif hücre olarak etiketlenmiştir. Yürüyen dalga yayan parametreler verilerek iterasyonlara başlanmıştır.



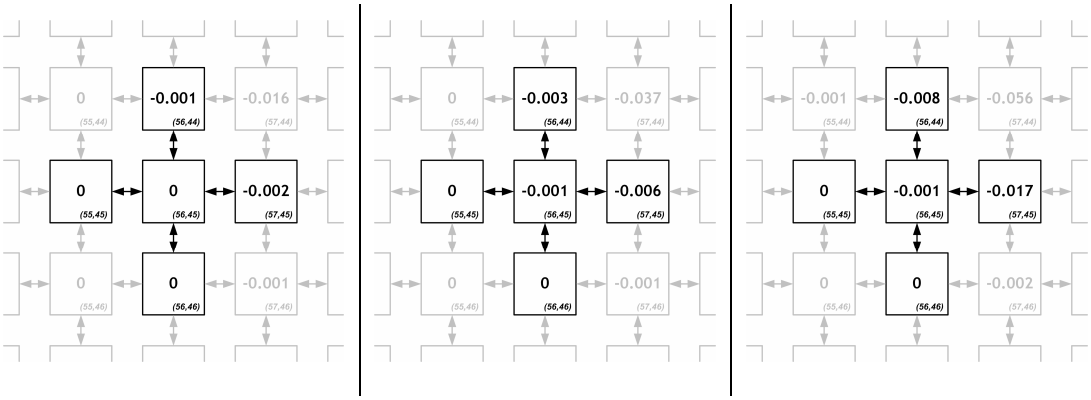
Şekil 6.19 : Yeni arena.

Başlangıç hücresinin ve komşularının x durumlarının 1., 2. ve 3. iterasyonlarda aldığı değerler Şekil 6.20'de gösterilmektedir.



Şekil 6.20 : Başlangıç hücresi ve komşularının ilk üç iterasyon sonunda x durumları.

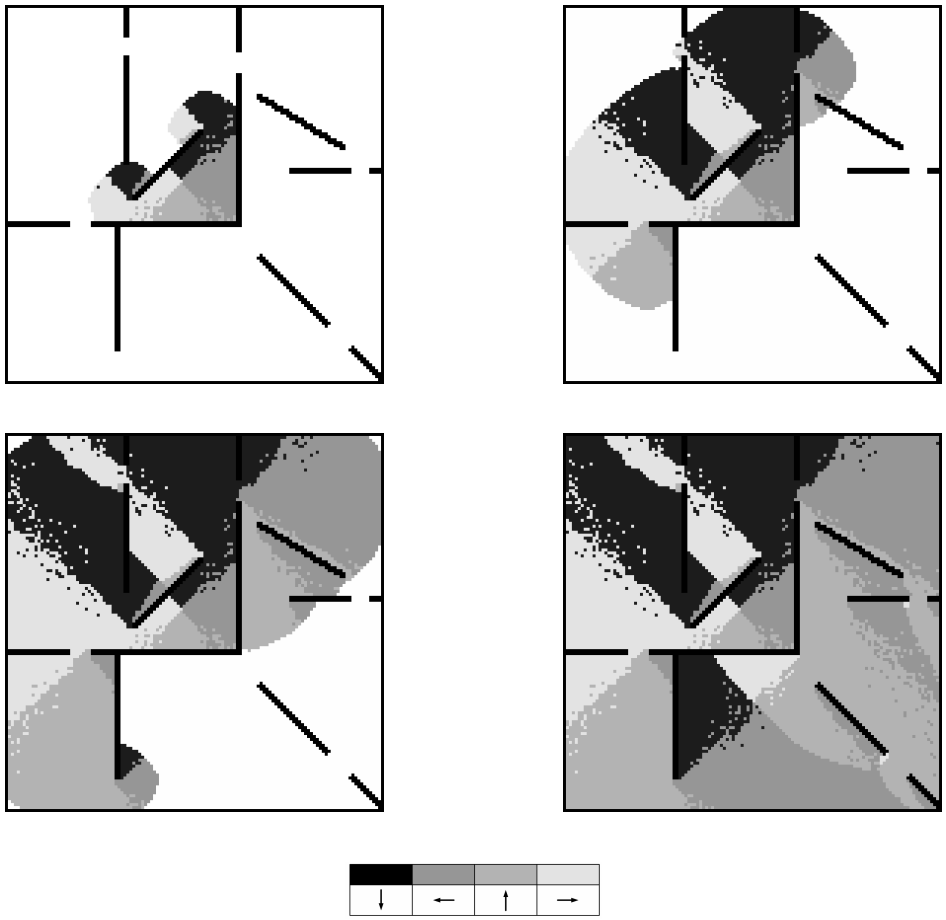
Dalga ilerleyişi sürdürülmüş ve çeper örnek hücre (56,45)'e vardığında x durumlarına bakılmıştır. Şekil 6.21'de bu hücre ve komşuları için x durumlarının değerleri verlmiştir.



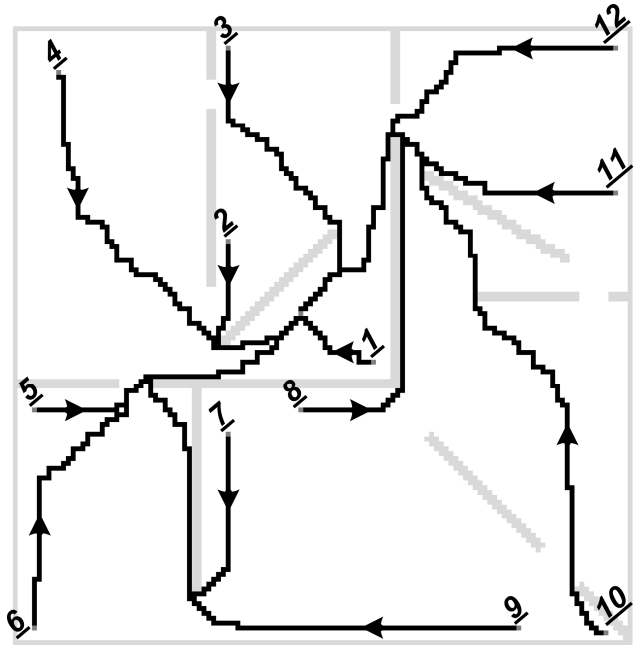
Şekil 6.21 : (56,45) hücresi ve komşularının 103.-105. iterasyonlarda x durumları.

Algoritmadaki küçük değişikliğe göre, örnek hücreye dalgayı ileten hücre, komşularından mutlak x genliği en büyük olan olarak seçilecektir. Yani Şekil 6.21'deki durum için (56,45) hücresine yürüyen dalga doğu komşusu olan (57,45)'den ulaşmıştır.

Tüm hücreler için dalga yayılım vektörleri bu kurala göre belirlendiğinde Şekil 6.22'deki sonuçlara ulaşılır. Bu şekil dalganın ilerleyişi ile saptanan yayılım vektörlerini göstermektedir. Şekil 6.23 ise Şekil 6.19'da seçilen 12 başlangıç noktasından hedef noktaya bu vektörleri takip ederk ulaştıran yolları göstermektedir. Bu yollar Şekil 6.16'daki yollar karşılaştırılarak algoritmada yapılan küçük değişikliğin etkisi anlaşılabilir.



Şekil 6.22 : Dalganın yayılması ile hesaplanan vektörler ve altta lejant.



Şekil 6.23 : Belirlenen 12 noktadan hedef noktaya algoritma ile bulunan yollar.

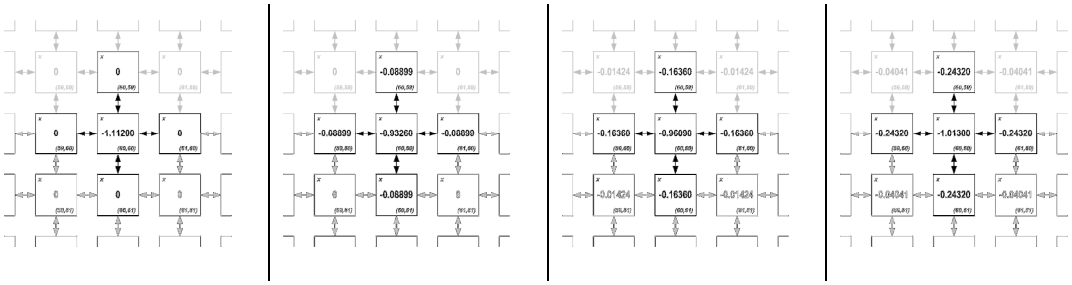
6.2 Dalga Birikimi ve Gradyentine Dayalı Algoritma

Bu algoritma, Dalga Çeperi Difüzyonuna Dayalı Algoritma'nın iki belirgin olumsuzluğunu gidermek amacıyla çalışılmış ve bulunmuştur. Bunlardan ilki önceki algoritmada dalga çeperini izleyen denetleyici bilgisayarın her iterasyonda ağ görüntüsünü (X matrisini) okuma ihtiyacı, diğeri de Şekil 6.16ve Şekil 6.23'de elde edilen yolların arenadaki engellerden güvenli bir uzaklıktan geçmemesi, ayrıca çok keskin dönüşler yapmasıdır. Her iterasyonda ağ görüntüsünün alınması haberleşme kanalına aşırı yük bindirilmesine ve algoritmanın hesaplanma süresinin uzamasına sebep olmaktadır. Yolların güvenli uzaklıktan geçmemesi algoritmayı kullanacak robotun engellere çarpma riskini doğurmakta, keskin dönüşler de robotun yo üzerinde ilerleme hızını düşürmektedir.

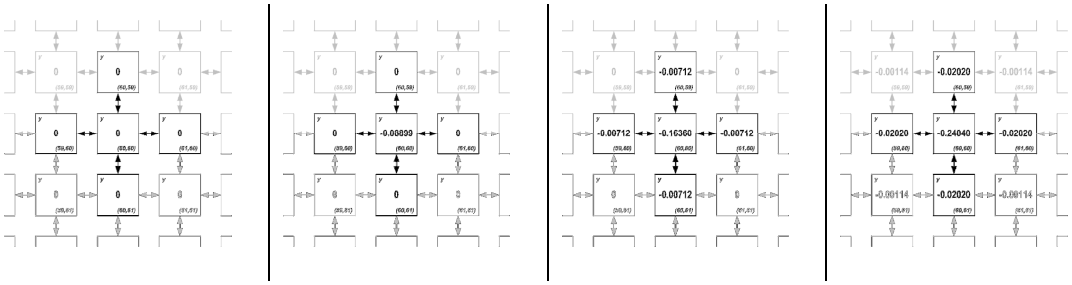
Dalga Birikimi ve Gradyentine Dayalı Algoritma'da, model durum değişkenlerinden x kullanılarak ağda bir yürüyen dalga yayılmış, y kullanılarak da bu yürüyen dalganın integrali alınmıştır. Dalga tüm ağı kapladığında, dalgayı üzerinde biriktiren Y matrisi topografik bir haritayı andırır. Öyleki bu haritada hedef nokta en derinde, dalganın en son ulaştığı, yani en geç ulaştığı hücre de aktif hücreler arasında en yüksekte bulunur. Pasif hücreler, yani engeller ise tıpkı duvarlar gibi, en yüksekteki aktif hücreden de yukarıda olurlar. Bu topografik datanın gradyetleri, yine sadece HYSA üzerinde, yani ne X ne de Y matrisinin denetleyici bilgisayara alınmasına gerek duyulmadan hesaplanır. HYSA algoritmik işlemlerin tamamını kendisi yapar ve işlemleri bitirdiğinde denetleyici bilgisayar bahsedilen topografik datanın i ve j yönlü gradyentlerini HYSA üzerinden okuyabilir ve bu gradyent datası ile yolların tespitini sağlayabilir.

Kısaca anlatılan bu algoritmanın ilk basamağında yine HYSA'ya denetleyici bilgisayar tarafından başlangıç koşulları yüklenir. U ve Y[0] matrisleri tamamen 0'lardan oluşur. X[0] başlangıç koşulu olarak bir önceki algoritmanın son kımındaki değerler kullanılır. Yani Şekil 6.19'daki harita, kaynak hücre $x_{60,60}[0] = -1{,}112$ olacak şekilde x durum dğişkenleri olarak yüklenir. Parametreler $\alpha = 0$, $\beta = 0$, $\varepsilon = 1$, $\sigma = 0$, $a_{i,j+1} = a_{i-1,j} = a_{i,j-1} = a_{i+1,j} = 1$; $m = -20$, $\lambda = 1$, $x_{sabit} = 0{,}00005$ olarak ayarlanır. HYSA, 750 iterasyon koşturulur.

Bu yüklenen başlangıç koşulları ve parametreler ile (60,60) hücresinin ve komşularının x ve y durum değişkenlerinin ilk dört iterasyon sonundaki değerleri Şekil 6.24'de ve Şekil 6.25'de sırasıyla gösterilmiştir. X durum matrisi üzerinde yürüyen dalga yayılmakta ve maksimum genliği -1 yaklaşıklığında olmaktadır. her iterasyonda Y durum matrisi üzerine X durum matrisi eklenmektedir.



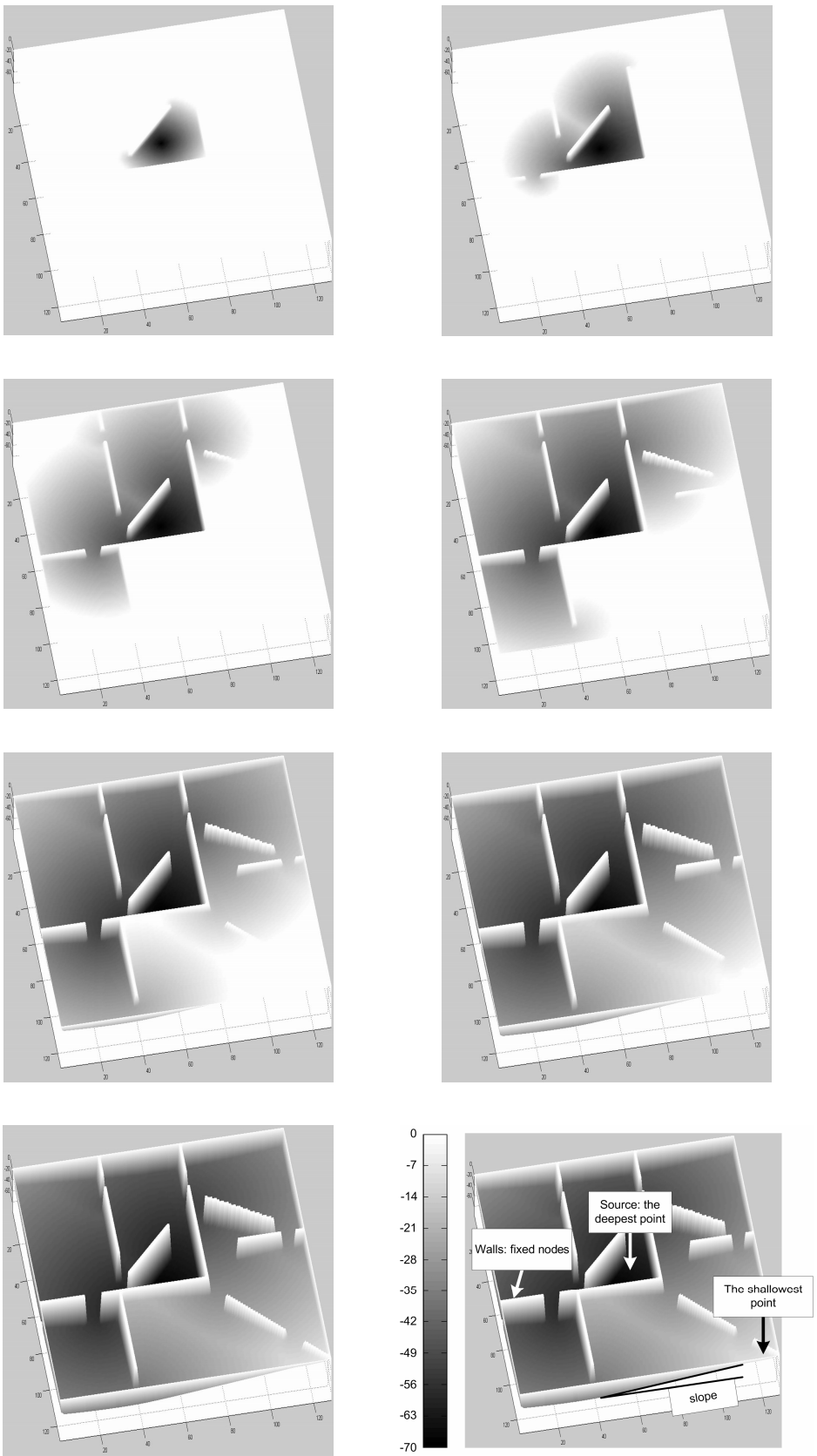
Şekil 6.24 : Başlangıç hücresi x durum değişkenin ilk dört iterasyon değerleri.



Şekil 6.25 : Başlangıç hücresi y durum değişkenin ilk dört iterasyon değerleri.

Y durumları, iterasyonlar sürdükçe sonsuza ıraksayacaktır. Ama algoritma için gereken iterasyon sayısı sonludur (750). Dikkat edilmesi istenen nokta Şekil 6.25'de başlangıç hücresinin y durum değerinin diğer tüm hücrelerden giderek daha az hale geliyor olduğudur. İşte bu topografik verinin oluşumudur. 750 iterasyon sona erdiğinde yürüyen dalga ağdaki aktif hücrelerin tamamına ulaşmaktadır. İterasyon sayısı seçilen haritaya göre denenerek bulunmuştur. Şekil 6.26'da HYSA'nın Y durum değişkenlerinin yavaş yavaş oluşturduğu topografik grafik gösterilmektedir.

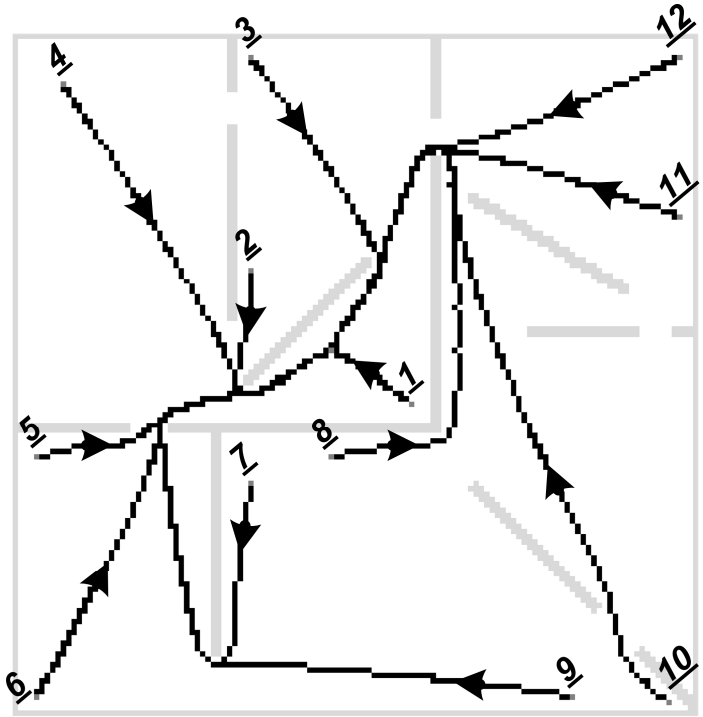
Dalga yayılımı tamamlandığında denetleyici bilgisayar tarafından parametreler güncellenir ve $\alpha = -1$, $\beta = 1$, $\varepsilon = 0$, $\sigma = 0$, $a_{i,j+1} = 0$, $a_{i-1,j} = 0$, $a_{i,j-1} = 0$, $a_{i+1,j} = 0$, $m = 0$ yapılır. Bir iterasyon koşturulur. Sonra, $\beta = 0$, $a_{i-1,j} = -1$, $a_{i+1,j} = 1$ parametrelerinde değişiklik yapılır ve bir iterasyon daha koşturulur. Bu işlemler ile HYSA, Y matrisini X matrisine kopyalar sonra X'deki datanın düşey (i ekseninde) gradyentini hesaplar. Takibinde parametrelerde $\beta = 1$, $\varepsilon = 1$, $\sigma = -1$, $a_{i,j+1} = 0$, $a_{i-1,j} = 0$, $a_{i,j-1} = 0$, $a_{i+1,j} = 0$ değişikliği yapılır. Bir iterasyon koşturulur. Ardından, $\beta = 0$, $\varepsilon = 0$, $\sigma = 0$, $a_{i,j-1} = -1$, $a_{i,j+1} = 1$ değişikliği yapılır ve bir iterasyon daha koşturulur.



Şekil 6.26 : Y durum matrisi üzerindeki dalga birikiminin görselleştirilmesi.

Bu işlemlerle de HYSA, X ve Y datasının yerlerini değiştirir ve yine X'deki datanın yatay (j eksenindeki) gradyentini hesaplar. 754 iterasyon tamamlandığında X durum matrisi arenanın topografik haritasına ait yatay gradyentleri, Y durum matrisi de düşey gradyentleri üzerinde tutuyor olur. Bu andan itibaren uygulamada kullanılacak robot hangi hücre üzerinde ise, o hücrenin x ve y durum değişkenlerinde tutulan vektörlerin, vektörel toplamı ile elde edilen vektör doğrultusunda ve şiddetinde ilerler. Sorgulanacak tüm noktalar için hesaplanacak vektörler, robotu en kısa yoldan hedef noktaya doğru yönlendirecektir.

Bu algoritma ile üretilen bilgi öncekine göre daha çok ve daha kıymetlidir. Öncekinde robot için ancak dik açılar ile manevra yapmak gerekmekteyken bu algoritmada ara açı değerlerine yönlenilebilmektedir. Öncekinde vektör genliği hesaplanamazken bu algoritmada robotun hız kontrolünü de yapmayı sağlayacak vektör genliği bilgisi üretilmektedir. Yine önceki algoritmanın tamamlanma süresi saatleri bulabilmekte iken bu algoritma için toplam süre saniyeler ile ölçülmektedir.



Şekil 6.27 : Dalga Birikimi ve Gradyentine Dayalı Algoritma ile bulunan yollar.

Şekil 6.27'de bu algoritma ile bulunan yollar gösterilmiştir. Bu yollar Şekil 6.23'deki yollar ile karşılaştırılarak algoritmaların sonuçları arasındaki fark belirlenmelidir. Dalga Birikimi ve Gradyentine Dayalı Algoritma ile Bölüm 4.3'de anlatılan 128x128

hücreli RO-HYSA devresi gerçek zamanlı robot uygulamasında denenebilir hale gelmiştir.

6.3 Yol Bulma Uygulamasının Bir Mobil Robot ile Testi

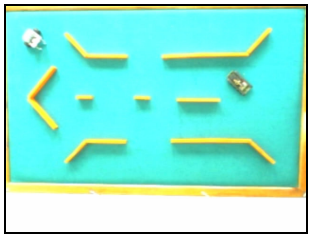
Bölüm 6.1 ve Bölüm 6.2'deki anlatılan algoritmaların mobil robota uygulanabilmesi için bir test platformu hazırlanmıştır. Test platformunun resimleri Şekil 6.28'de verilmiştir. Homojen aydınlatma için ışık kaynakları platform üzerinde bulunmaktadır. Platformun üzerine, platformu kuşbakışı gören bir kamera yerleştirilmiştir. Yeşil renkli platform düzlemi mattır. Mat yüzey, aydınlatma ışığının kameraya yansımamasını sağlar. Engeller için turuncu renk, hareket düzleminin rengi ile kontrast olması için seçilmiştir. Platformun kontrol masasında Bölüm 4.3'de tasarımı anlatılan 128x128 hücreli programlanabilir RO-HYSA gerçeklemesini barındıran FPGA kartı, denetleyici bilgisayar, HYSA monitörü ve bilgisayar monitörü bulunmaktadır.

Şekil 6.28 : Kurulan test platformu ve kullanılan mobil robotun resimleri.

Platform üzerinde yerleştirilen kameradan alınan görüntü ile hareket düzleminin sınırları, engellerin, robotun ve hedefin yerleri bulunmalıdır. Görüntü denetleyici

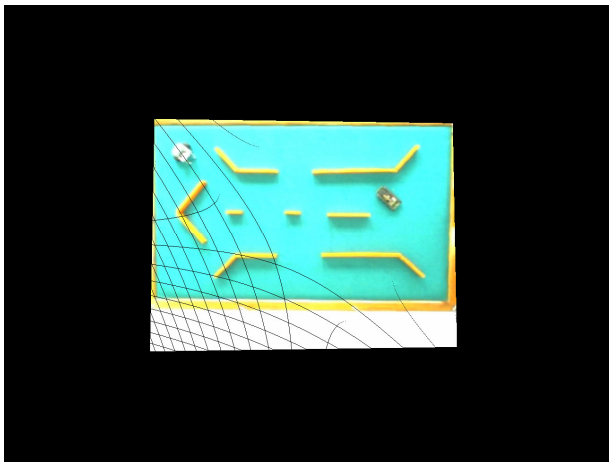
bilgisayar üzerinde işlenmekte, işlenen data HYSA devresine gönderilmekte ve HYSA'nın algoritma sonucunda ürettiği data tekrar denetleyici bilgisayara yollanmaktadır. Mobil robotun hareket bilgisi yine denetleyici bilgisayar tarafından kablosuz olarak robota aktarılmaktadır.

Şekil 6.29'da tepe kameradan alınan bir görüntü verilmiştir. Alınan bu kuşbakışı görüntünün çözünürlüğü 640x480'dir.



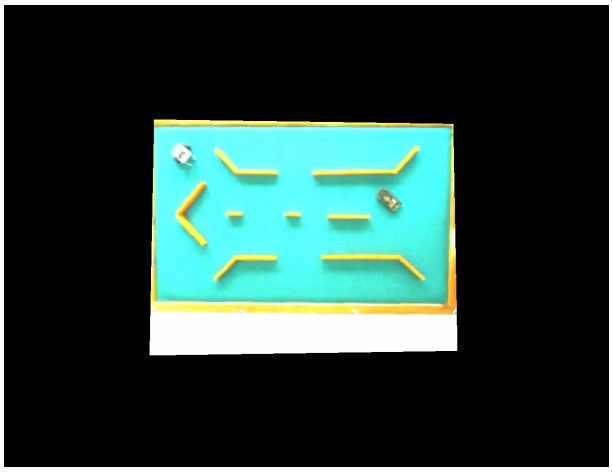
Şekil 6.29 : Tepe kameradan alınan bir görüntü.

Tepe görüntüsüne projektif dönüşüm uygulanır. Bu dönüşüm ile kamera normali ile test platformu normali paralel hale, ayrıca görüntüdeki platform sınırları görüntü kenarlarına paralel hale getirilir. Şekil 6.30'daki resim projektif dönüşüm uygulanmış tepe görüntüsünü vermektedir.



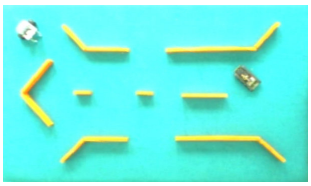
Şekil 6.30 : Projektif dönüşüm uygulanmış resim.

Projectif dönüşüm esnasında görüntü üzerinde bazı noktaların değeri hesaplanamaz. Şekil 6.30'da görüntü üzerindeki siyah yaylar değeri hesaplanamayan piksellerden oluşmaktadır. Bu değerler, interpolasyon ile kestirilir ve tepe görüntüsü Şekil 6.31'deki hali alır.



Şekil 6.31 : İnterpolasonla düzeltilmiş tepe görüntüsü.

Dönüşüm ve interpolasyon işlemleri sonucunda oluşan resimlerin çözünürlükleri orijinal görüntüden büyüktür. Uygulama için gereken paltform görüntüsü de orijinal görüntünün bir parçasıdır. Bu aşamada görüntünün fazla yerleri kırpılır ve Şekil 6.32'deki resme ulaşılır.



Şekil 6.32 : Fazla kenarları kırpılmış tepe görüntüsü.

Kırpılan görüntüdeki beyaz renkli mobil robot, siyah renki görünen hedef aracın konumları hesaplanır ve bunlar görüntüden çıkarılır. Geriye görüntüde platform düzlemi ve engeller kalır. Engeller saptandıktan sonra morfolojik işlemlerden genişletme (dilate) uygulanır. Engellerin genişletilmesinin sebebi, algoritmanın bulağı yolun duvardan güvenli bir mesafe öteden geçmesini sağlamaktır. Genişletme işleminden sonra engeller yeşil renge, platfom siyah renge boyanır. Görüntünün uzun kenarı 128 piksel olacak şekile görüntü küçültülür. Elde edilen görüntünün kenarlarındaki pikseller de yeşile boyanır. Kısa kenar görüntüye yeşil bir bölge eklenerek uzun knar boyuna yani 128 piksele uzatılır. Elde edilen görüntü Şekil 6.33'de verilmiştir.

Şekil 6.33 : RO-HYSA'nın başlangıç koşulu olarak kullanabileceği tepe görüntüsü.

Tepe görüntünün son hali 128x128 hücreli programlanabilir RO-HYSA devresine başlangıç koşulu olarak yüklenmeden önce salga kaynağı olacak hücreye denk düşen piksele -1.122 değerine karşılık gelecek renk atanır ve elde edilen resim RO-HYSA'ya yüklenir.

Parametrelerinde algoritmalarda anlatıldığı değerlere ayarlanmasının ardından yapılacak öykünleme ile RO-HYSA hedefe tüm platformdan ulaşmayı yarayacak gradyent verisini üretir.

Teper görüntüsünden RO-HYSA'ya yüklenebilir verinin hesaplanması için yapılan işlemlerin süreleri Çizelge 6.3'de sunulmuştur.

Çizelge 6.3 : Görüntü işleme adımları ve süreleri

İşlemler	Süreler (s)
resim alım süresi	5 -14 arası
resim döndürme süresi	1.27
siyah çizgilerin yok edilme süresi	4.24
çalışılacak alan belirleme süresi	0.01
sabit engel bulma süresi	2.41
hedef bulma süresi	0.01
robot bulma süresi	0.01
toplam işlem süresi	8 – 22 arası

7. SONUÇLAR

Çalışmanın sonucunda, RO-HYSA üzerinde yayılan aktif dalgalar kullanılarak en kısa yol problemine sunulan çözümün başarılı olduğu görülmüştür. Sistemin parçası olan denetleyici bilgisayarın işlevleri de FPGA içerisine gereken devrelerin tasarlanması ile gömüldüğü ve platformu kuşbakışı gören kameranın robot üzerine alınması halinde, daha hızlı ve daha kompakt bir çözüm sunulabileceği görülmüştür.

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

Yarı duyarlı kayan nokta aritmetiği yerine sabit noktalı aritmetik de analiz edilerek sayı formatı değiştirlebilir. Böylelikle RO-HYSA devresinin kullandığı bellek miktarı düşürülebilir ya da daha büyük boyutlu bir ağ, aynı miktarda bellek ile öykünlenebilir. Aynı zamanda kayan nokta aritmetiğinin uzun toplama işlemi süresinden kurtularak daha hızlı bir tasarıma ulaşılabilir.

Gelişmekte olan yeniden konfigüre edilebilir sayısal cihazlar teknolojisi takip edilerek tasarımların daha hızlı cihazlar üzerinde gerçeklenmesi de mümkündür.

Gerçeklenen RO-HYSA, en kısa yolun bulunması probleminin yanı sıra aktif dalga temelli başka uygulamalarda da rahatlıkla kullanılabilecektir. Bu tezde gelinen son nokta, üç boyutlu hücresel yapay sinir ağlarının sayısal gerçeklenmesi, mobil robot için yerel yörünge planlanması, dinamik ortamlar için yörünge planlanması, çoklu robot çoklu hedef çözümleri konularında çalışmalara ışık tutmaktadır.

KAYNAKLAR

[1] **L. Chua and L. Yang,** 1988. "Cellular neural networks: theory," *Circuits and Systems, IEEE Transactions on,* vol. 35, no. 10, pp. 1257–1272.

[2] **L. Chua and L. Yang,** 1988. "Cellular neural networks: applications," *Circuits and Systems, IEEE Transactions on,* vol. 35, no. 10, pp. 1273–1290.

[3] **M. Yalcin,** 2008. "A simple programmable autowave generator network for wave computing applications," *IEEE Transactions on Circuits and Systems II-Express Briefs,* vol. 55, no. 11, pp. 1173–1177.

[4] **A. Adamatzky, P. Arena, A. Basile, R. Carmona-Galan, B. Costello, L. Fortuna, M. Frasca, and A. Rodriguez-Vazquez,** 2004. "Reaction-diffusion navigation robot control: from chemical to VLSI analogic processors," *Circuits and Systems I: Regular Papers, IEEE Transactions on,* vol. 51, pp. 926–938.

[5] **P. Arena, A. Basile, L. Fortuna, and M. Frasca,** 2004. "CNN wave based computation for robot navigation planning," *in Proc. Of the 2004 IEEE ISCAS,* Toronto, Canada.

[6] **P. Arena, L. Fortuna, M. Frasca, G. Vagliasindi, and A. Basile,** 2005. "CNN wave based computation for robot navigation on ACE16K," *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on,* vol. 6, pp. 5818–5821.

[7] **P. Arena, S. De Fiore, L. Fortuna, and L. Patane,** 2008. "Perception-action map learning in controlled multiscroll systems applied to robot navigation," *CHAOS,* vol. 18, p. 043119.

[8] **K. Ito, M. Hiratsuka, T. Aoki, and T. Higuchi,** 2006. "A shortest path search algorithm using an excitable digital reaction-diffusion system," *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences,* vol. E89A, pp. 735–743.

[9] **K. Ito, M. Hiratsuka, T. Aoki, T. Higuchi,** 2006. "A Shortest Path Search Algorithm Using an Exicitable Digital Reaction-Diffusion System," *IEICE Trans. Fundamentals,* vol. E89-A, no. 3.

[10] **I. Gavrilut, V. Tiponut, A. Gacsadi,** 2006. "Path Planning of Mobile Robots by Using Cellular Neural Networks", *10th International Workshop on Cellular Neural Networks and their Applications, CNNA 2008,* Istanbul, Turkey.

[11] **M. Hiratsuka, K. Ito, T. Aoki, and T. Higuchi,** 2008. "Shorthest Path Search Using a Reaction-Diffusion Processor," *Int. Journal of Unconventional Computing,* vol. 4, pp. 113-123.

[12] **T. Roska, L. Chua,** 1993. "The CNN Universal Machine: An Analogic Array Computer," *IEEE Trans. Circuits Syst. II,* vol. 40, pp. 163-173.

[13] **Z. Nagy and P. Szolgay,** 2003. "Configurable multilayer CNN-UM emulator on FPGA," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications,* vol. 50, no. 6, pp. 774–778.

[14] **K. Kayaer, V.Tavsanoglu,** 2008. "A new approach to emulate CNN on FPGAs for real time video processing," *11th Int.l Workshop on Cellular Neural Networks and their Applications, CNNA 2008,* pp. 23–28.

[15] **R. Yeniceri and M. Yalcin,** 2008. "An implementation of 2D locally coupled relaxation oscillators on an FPGA for real-time autowave generation," *11th International Workshop on Cellular Neural Networks and their Applications, CNNA 2008,* pp. 29–33.

[16] **R. Yeniceri, M. Yalcin,** 2008. "A Programmable Hardware for Exploring Spatiotemporal Waves in Real-time", *11th International Workshop on Cellular Neural Networks and their Applications, CNNA 2008,* p. 7.

[17] **R. Yeniceri, M. Yalcin,** 2009. "An Emulated Digital Wave Computer Core Implementation", *accepted to European Conference on Circuit Theory and Design 2009 (ECCTD'09)*, Antalya, Turkey.

[18] **R. Yeniceri and M. Yalcin,** 2009. "Path planning on cellular nonlinear network using active wave computing technique*," Proceedings of SPIE Europe Microtechnologies for the New Millenium Symposium,* vol. 7365.

[19] **L. Chua and T. Roska,** 1993. "The CNN paradigm," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on,* vol. 40, pp. 147–156.

[20] **T. Roska and A. Rodriguez-Vazquez,** 2002. "Toward visual microprocessors," *Proceedings of the IEEE ,* vol. 90, pp. 1244–1257.

[21] **T. Roska,** 2007. "Circuits, computers, and beyond boolean logic," *Int. J. Circuit Theory and Applications,* vol. 35(5-6), pp. 485–496.

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

[23] **M Yalcin, J. Suykens, J. Vanderwalle,** 2004. "Experimental Observations of Autowaves on the ACE16k CNN Chip", *Proc. Of the 8th IEEE Int. Workshop on CNNA,* Budapest, Hungary.

[24] IEEE754, 1985. "IEEE 754: Standard for binary floating-point arithmetic," IEEE, http://grouper.ieee.org/groups/754/, Tech. Report.

[25] **R. Yeniceri, M. Yalcin,** 2009. "A Testbed of 2D Locally Coupled Relaxation Oscillators for Spiral Waves Generation", *accepted to 4th International Scientific Conference on Physics and Control (PHYSCON 2009)*, Catania, Italy.

ÖZGEÇMİŞ

Ad Soyad: Ramazan YENİÇERİ

Doğum Yeri ve Tarihi: Denizli, 1985

Adres: İTÜ Elektrik Elektronik Fakültesi,

Oda 1109, 34469, Maslak, İstanbul, Türkiye

Lisans Üniversite: İstanbul Teknik Üniversitesi, Elektronik Müh.

Yayın Listesi:

▪ **Yeniçeri R.**, Yalçın M. E., "A Testbed of 2D Locally Coupled Relaxation Oscillators for Spiral Waves Generation", *accepted to 4th International Scientific Conference on Physics and Control (PHYSCON 2009)*, Catania, Italy, 1-4 September 2009.

▪ **Yeniçeri R.**, Yalçın M. E., "An Emulated Digital Wave Computer Core Implementation", *accepted to European Conference on Circuit Theory and Design 2009 (ECCTD'09)*, Antalya, Turkey, 23-27August 2009.

▪ **Yeniçeri R.**, Yalçın M. E., "Path Planning on Cellular Nonlinear Network Using Active Wave Computing Technique", *Proceedings of SPIE Europe Microtechnologies for the New Millenium Symposium,*, Dresden, Germany, 3-5 May 2009.

▪ **Yeniçeri R.**, Usta A., Yalçın M. E., "ITUcam, FPGA Tabanlı Görüntü Yakalama ve İşleme Kartı Gerçeklemesi", *1. Gömülü Sistemler ve Uygulamaları Sempozyumu (GÖMSİS 2008)*, İstanbul, 3-5 Kasım 2008. (Türkçe)

▪ **Yeniçeri R.**, Yalçın M. E., "A Programmable Hardware for Exploring Spatiotemporal Waves in Real-time", *11th International Wokshop on Cellular Neural Networks and their Applications (CNNA 2008)*, Santiago de Compostela, Spain, 14-16 July 2008.

▪ **Yeniçeri R.**, Yalçın M. E., "An Implementation of 2D Locally Coupled Relaxation Oscillators on an FPGA for Real-time Autowave Generation", *11th International Wokshop on Cellular Neural Networks and their Applications (CNNA 2008)*, Santiago de Compostela, Spain, 14-16 July 2008.

▪ Koyuncu E., Ceylan O., **Yeniçeri R.**, "Kamera Denetimli Yapay Sinir Ağları ile Hareketli Cisim Yörüngesi İzleyen Zeki Taret Savunma Sistemi Tasarımı", *Kara Harp Okulu, Savunma Bilimleri Enstitüsü, 1. Savunma Bilimleri Araştırmaları Yarışması Eser Özetleri Kitabı*, Ankara, 13 Mart 2006. (Türkçe)

▪ Koyuncu E., Ceylan O., **Yeniçeri R.**, "Bilgisayarla Görü Tabanlı, Cisim Yörünge Doğrultusu İzleyen Robot Kol Tasarımı", *Otomatik Kontrol Türk Milli Komitesi, Otomatik Kontrol Ulusal Toplantısı (TOK'05)*, İstanbul, 2-3 Haziran 2005. (Türkçe)