BİLECİK ÜNİVERSİTESİ

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

LORENZ KAOTİK SİSTEMİNİN SABİT NOKTALI SAYI GÖSTERİMİ İLE FPGA ÜZERİNDE GERÇEKLENMESİ VE ANALİZİ

Emre Güngör

Yüksek Lisans Tezi

Tez Danışmanı

Yrd. Doç. Dr. Enver Çavuş

BİLECİK, 2011

BİLECİK ÜNİVERSİTESİ **BİLECİK ÜNİVERSİTESİ** **FEN BİLİMLERİ ENSTİTÜSÜ** **BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**	**YÜKSEK LİSANS** **JÜRİ ONAY FORMU**

Bilecik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 26/12/2011 tarih ve 22 sayılı kararıyla oluşturulan jüri tarafından 04/01/2012 tarihinde tez savunma sınavı yapılan Emre Güngör'ün "LORENZ KAOTİK SİSTEMİNİN SABİT NOKTALI SAYI GÖSTERİMİ İLE FPGA ÜZERİNDE GERÇEKLENMESİ VE ANALİZİ" başlıklı tez çalışması Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak oy birliği/oy çokluğu ile kabul edilmiştir.

JÜRİ

ÜYE
(TEZ DANIŞMANI) : Yrd. Doç. Dr. Enver ÇAVUŞ

ÜYE: Yrd. Doç. Dr. Cihan KARAKUZU

ÜYE: Yrd. Doç. Dr. İhsan PEHLİVAN

ONAY

Bilecik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ……/……../…… Tarih ve ……../…….. sayılı kararı.

İMZA/MÜHÜR

ÖZET

Tez çalışmasında Lorenz kaotik sistemi, kayan ve sabit noktalı sayı gösterimleri ile yazılımsal olarak oluşturulmuştur. Sabit noktalı sayı gösterimiyle de Verilog dilinde donanımsal olarak tasarlanan sistem, FPGA üzerinde gerçeklenmiştir. Bu çalışmayı literatürdeki diğer çalışmalardan ayıran en önemli nokta ise Lorenz kaotik sisteminin gerçeklenmesinde Verilog kodlarının herhangi bir otonom yazılım yardımı olmadan oluşturulmasıdır. Tasarımın bütün parçaları, çalışma kapsamında ayrıntılarıyla yazılmış olup farklı sayıda bit uzunluklarıyla sabit noktalı sayı gösterimi kullanılarak sistem gerçeklenmiştir. Ayrıca kaotik sistemlerin belirlenen kısıtlar ve koşullar altında farklı parametreler kullanılarak, bilgisayar ortamında kurgulanan benzetimi ile FPGA üzerinde gerçeklenen tasarımın aynı sonuçlar vermesi ve bu sistemlerin donanımsal olarak sentezlenmesi sağlanmıştır. Bu sayede Lorenz kaotik sisteminin FPGA üzerindeki kaynak kullanım sonuçları, Xilinx Virtex 2 modelleri üzerinde elde edilmiştir.

Anahtar Kelimeler

FPGA, Verilog, Matlab, Lorenz, Kaotik Sistemler

ABSTRACT

In this thesis, the software model of Lorenz chaotic system is implemented using floating and fixed point number representations. Furthermore, a parametric, fixed point design of the system is realized on FPGA. Compared to other studies in literature, the most significant difference of this work is the realization of Lorenz chaotic system without any automatic code generation tools. All of the system design components are realized using fixed point number representation with varying bit lengths. Also, software simulation and hardware design outputs are matched under different conditions with various bit lengths. Consequently, speed and resource usage of Lorenz chaotic system is obtained using Xilinx Virtex II FPGA.

Keywords

FPGA, Verilog, Matlab, Lorenz, Chaotic Systems

TEŞEKKÜR

Tez çalışmamda büyük emeği geçen danışmanım, Yrd.Doç.Dr. Enver ÇAVUŞ hocama, tez konumun belirlenmesine yardımcı olan ve desteğini hiçbir zaman esirgemeyen Yrd. Doç. Dr. İhsan PEHLİVAN hocama, destek ve anlayışlarından dolayı iş arkadaşlarıma ve hayatımın her safhasında emek vererek, eğitimim konusunda ellerinden gelen her şeyi yapan ve beni bu güne getiren aileme teşekkürü borç bilirim.

İÇİNDEKİLER

ÇİZELGELER DİZİNİ

Sayfa No

ŞEKİLLER DİZİNİ

Sayfa No

SİMGELER VE KISALTMALAR DİZİNİ

FPGA: Alan Programlanabilir Kapı Dizisi (Field Programmable Gate Array).

FSM : Sonlu Durum Makinesi (Finite State Machine).

HDL : Donanım Tanımlama Dili (Hardware Definition Language).

LUT: Başvuru Çizelgesi (Lookup Table).

MATLAB: Adını Matris Laboratuarı kelimesinin kısaltılmasından alan yazılım geliştirme aracı.

M-Q-N : M bitlik tamsayı ve N bitlik Kesirli sayı gösterimi (Örnek:16Q16, 8Q12).

RTL : Yazmaç Transfer Seviyesi (Register Transfer Level).

SENTEZ : HDL kodundan lojik kapı yapısına dönüştürme işlemi.

SIMULINK: Dinamik sistem modelleme ve benzetiminin yapılabildiği Matlab yazılımında eklenebilen bir yazılımdır.

VERILOG : Bir HDL dili.

1. GİRİŞ

Kaos, karmaşıklık ve düzensizlik anlamlarına gelmektedir. Kaos teorisi literatürde kaos kuramı veya kargaşa kuramı olarak da geçmektedir. Matematiksel bir tümevarım ya da fiziksel teori olmamakla birlikte kaos teorisi, gerçekliklerin bir bütün olarak eğilimlerini açıklamaya yaramaktadır. Genelde rastsal olarak görünen kaotik olaylar aslında rastsal olaylar değillerdir. Başlangıç parametrelerinin hassasiyeti ve sistemin zamanla değişiminden dolayı dışarıdan rastsal bir olay ve sistemler olarak görülür. Ancak bu sistemlerin kendine has bir düzeni vardır. Dinamik sistemler içerisinde incelenen kaos, bilinen en karmaşık kararlı hal davranışıdır.

Kaotik sistemler içerisinde yer alan buhar, gaz ya da dumanın havadaki hareketi düzensiz, bağımsız ve rastlantısal olarak düşünülmektedir. Fakat görülen bu dinamiğin arkasında, ortama etki eden birçok parametre bulunduğundan bu olayların bir sistem içinde ele alınması gerekir. Ortamda gerçekleşen etkenlerin ve parametrelerin sayısının çokluğu sistemin incelenmesini ve anlaşılırlığını zorlaştırır. Parametrelerin çokluğu ve değişkenliği, parametrelerin aynı zamanda çıktı olmasından ileri gelmektedir. Bu nedenle, dünyanın bir ucundaki bir kelebeğin kanat çırpması, dünyanın diğer ucundaki bir fırtınanın parametresi olabilmektedir. Bu sebeple kaotik sistemler, başlangıç koşullarına karşı hassas ve sistem sonuçlarının uzun vadede tahmin edilemediği karmaşık sistemlerdir.

Bu yüzyılın ortalarında kaotik sistem teorisi, lineer teori üzerinde çalışan bilim adamları tarafından belirlenmiştir. İlk kaos teorisi Henri Poincare tarafından 1980'lerde önerilmiştir. Henri Poincare sonsuza doğru artmayan, belirli bir noktaya ulaşmayan ve periyodik olmayan yörüngelerin olduğunu bulmuştur (Poincaré, 1890). Daha sonra, hava durumu tahminlerinde, gazların ve dumanın davranışlarında kaotik özellikler gözlenmiştir. G.D. Birkhoff (Birkhoff) , A.N. Kolmogorov (Kolmogorov, 1941) , J.E. Littlewood (Littlewood, 1945) gibi birçok araştırmacı fiziksel olaylara dayandırılan sistemlerdeki dinamik davranışı incelemişlerdir ve ilk olarak 1961'de Edward Lorenz kaos terosini hava durumu tahminlerinde kullanmaya başlamıştır(Lorenz,1963). Kaotik özelliklerin birçok farklı sistemde yer alan bir durum olmasının keşfedilmesiyle birlikte,

kaos teorisinin uygulama alanları, evrende gerçekleşen olaylardan günlük yaşamda karşılaşılan durumlara kadar geniş bir yelpazeye yayılmıştır.

Uygulama alanlarından bazıları: jeoloji, matematik, programlama, mikrobiyoloji, bilgisayar bilimleri, ekonomi, finans, meteoroloji, fizik, felsefe, politika ve robotik sayılabilir. Laboratuar koşullarında incelenen elektriksel devreler, lazerler, akışkan dinamikleri, manyetik ve mekanik araçlar dışında hava şartlarının gözlemine dayanan sistemler de kaotik davranış gösterirler. Ayrıca Bilgisayar Bilimleri disiplini içinde yer alan iletişim alanında, rastgele sayı üretiminde ve şifreleme algoritmalarının yazımında kullanılmaktadır (Yu ve Bai, 2010). Kâğıt para ya da kitaplar üzerinde taklit olmadığının anlaşılmasını sağlayan filigran şekil ve yapılarının oluşturulmasında da kaotik sistemler kullanılmaktadır (Li vd., 2009).

Kaotik sistemlerin geniş bir uygulama alanı olmasından dolayı, bu sistemlerin donanımsal olarak gerçeklenmesi önemlidir. Örneğin hava tahmininde donanımsal olarak yapılan analizler ile daha doğru ve hızlı sonuçlar elde edilebilmektedir. Benzer olarak şifreleme sistemlerinde kullanılan kaotik sistemler, devresel olarak gerçeklenerek, hızlı bir biçimde şifreleme ve çözümleme ihtiyaçlarını karşılamaktadır. Literatürde kaotik sistemlerin dijital olarak FPGA üzerinde tasarımı ilk olarak 2002 yılında Lorenz kaotik sistemi için yapılmıştır (Aseeri vd., 2002). Bu çalışmada Matlab Simulink yazılımı ve Xilinx'in System Generator yazılımları kullanılarak otomatik sistem kodları oluşturulmuş ve FPGA üzerinde gerçekleme yapılmıştır. Literatürde benzer biçimde bir çok farklı kaotik sistem de ele alınmıştır. Benzer çalışmalardan Yu Simin ve Lu Jinhu'nun kaotik Chua devresi ve donanımsal gerçeklemesi (Yu ve Lu, 2007), Wang ve arkadaşlarının Chen kaotik sisteminin tasarımı ve gerçeklemesi birer örnek olarak verilebilir (Wang vd., 2011). Ayrıca gerçek zamanda kaotik denklemlerin uygulanabilirliği iletişim alanında imkân sağlamaktadır. Sinyallerin kaotik sistemler yardımıyla şifrelenmesi ve maskelenmesiyle iletişim güvenliği sağlanabilmektedir (Ding & Pan, 2010) (Azzaz, 2009).

Yukarda özetlenen kaotik sistemler üzerindeki donanımsal çalışmalar, genel olarak kayan noktalı sayı sistemi kullanılarak gerçeklenmiştir. Fakat donanımsal olarak daha etkin gerçeklenme özelliğine sahip sabit noktalı sayı gösterimi kullanılarak kaotik sistemlerin tasarımına gerektiği kadar değinilmemiştir. Bu tez çalışmasında, kaotik

sistemlerin donanımsal olarak basit, hızlı ve etkin bir biçimde tasarlanması ve gerçeklenmesi amaçlanmıştır. Donanımsal olarak tasarım ve gerçekleme sırasında ise sistem karmaşıklığının ölçümlenmesi ve analizlerinin yapılması tez açısından önemlidir. Bu çalışmanın kapsamı, Lorenz kaotik sisteminin kayan ve sabit noktalı sayı gösterimleri ile Matlab dilinde tasarlanması ve sabit noktalı sayı gösteriminin Verilog dilinde yazılarak FGPA üzerinde gerçeklenmesidir. Bu çalışmayı diğer literatürdeki çalışmalardan ayıran en önemli nokta ise Lorenz kaotik sisteminin gerçeklenmesinde Verilog kodlarının herhangi bir otonom yazılım yardımı olmadan oluşturulmasıdır. Tasarımın bütün parçaları çalışma kapsamında ayrıntılarıyla yazılmış olup farklı sayıda bit uzunluklarıyla sabit noktalı sayı gösterimi kullanılarak sistem gerçeklenmiştir. Ayrıca kaotik sistemlerin belirlenen aynı kısıtlar ve koşullar altında farklı parametreler kullanılarak, bilgisayar ortamında kurgulanan benzetimi ile FPGA üzerinde gerçeklenen tasarımın aynı sonuçlar vermesi ve bu sistemlerin donanımsal olarak sentezlenmesi sağlanmıştır. Bu sayede amaçlanan Lorenz kaotik sisteminin FPGA üzerinde gerçeklenmesi sağlanarak, donanımsal seviyede kaotik sistemlerin kaynak kullanım sonuçları elde edilmiştir.

Tezde kullanılan ve incelenen kaotik sistemler ile çözümleri Bölüm 2'de ele alınmıştır. Ayrıca çalışmada kullanılan Lorenz sistemi ve literatürdeki uygulamaları yine aynı bölümde bahsedilmektedir. Sistemin yazılımsal olarak tasarım ve modellenme aşamaları ise Bölüm 3 içerisinde anlatılarak, donanımsal tasarım metodları ile birlikte Matlab ve Verilog dilleri üzerinde nasıl gerçekleştirildikleri Bölüm 4 içerisinde ele alınmıştır. Ayrıca sistem tasarımında önem arz eden modüller ve sistemin farklı bit uzunluklarında tanımlanması, yine aynı şekilde Bölüm 4'de yer almaktadır. Sistemin benzetimi ve karşılaştırılması, gerçeklenmesi ve gerçeklenen donanımsal özelliklerin literatür ile karşılaştırılması da Bölüm 5 içerisinde belirtilmiştir. Çalışmada yapılan analizler ışığında elde edilen sonuçlar ve sistemin geliştirilebilir özellikleri sonuç bölümü olan Bölüm 6'da ifade edilmektedir.

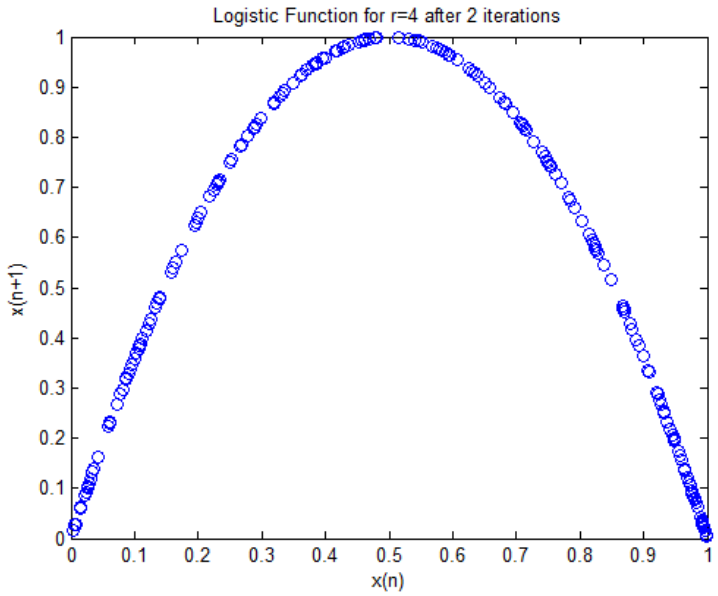
2. KAOTİK SİSTEMLER VE ÇÖZÜM YÖNTEMLERİ

Literatürde birçok kaotik sistem mevcuttur. Kaotik sistemleri incelerken ve gerçeklerken ele alınan bazı kıstaslar; sistemlerin karmaşıklığı, yaygın kullanımı ve literatüre olan katkısıdır. Bu kıstaslar altında tez çalışmasının kapsamına göre en uygun olarak düşünülen kaotik sistemler ele alınmıştır. Bölüm 2.1'de incelenen Logistic Map ve Bölüm 2.2'de incelenen Henon Map, kaotik sistemleri anlamak ve belirli koşullar altında davranışlarını inceleme açısından en kolay olan sistemlerdir. Bu çalışmada kullanılan Lorenz kaotik sistemi ise Bölüm 2.3'de incelenmiştir.

2.1. Logistic Map Kaotik Sistemi

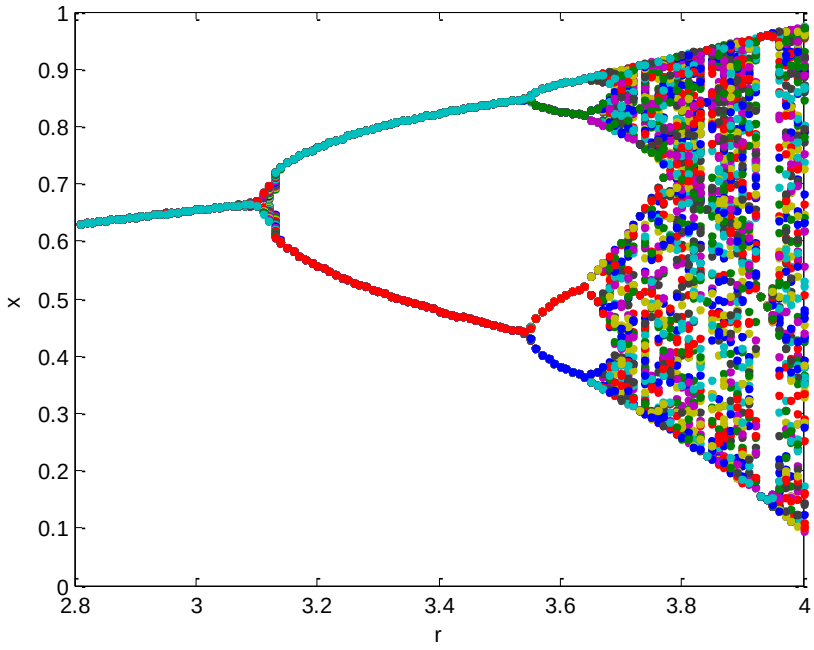
Logistic map, ikinci dereceden bir polinomdur. Pierre François Verhulst tarafından bulunan denklem, 1976 yılında biyolojist Robert May tarafından ayrık zaman demografik modelin lojistic denkleme benzerliğini göstermesiyle popüler hale gelmiştir. Logistic Map, toplum içerisindeki nüfus değişiminin matematiksel bir modeli olarak ortaya çıkmıştır (Verhulst, 1845).

$$x_{n+1} = r x_n (1 - x_n) \tag{2.1}$$



Şekil 2.1 - İki boyutlu Logistic Map x_n değerinin x_{n+1} değerine göre değişimi.

Eşitlik 2.1'de verilen denklemin çözümünün Şekil 2.1'de görüldüğü Logistic Map, her bölgede kaotik özellik göstermemektedir (Weisstein, 1999-2011). Örneğin, Eşitlik 2.1 deki denklem bazı bölgelerde periyodik sonuçlar verirken *r* parametresinin değerlerine bağlı olarak kaotik özellik vermeye başlamaktadır. Bu sebeple Logistic Map, genellikle basit lineer olmayan dinamik denklemlerin kaotik özelliği gösterebildiğine güzel bir örnek oluşturmaktadır. Şekil 2.2'te *r* parametresinin sisteme etkisi görülmektedir. Bu parametre sistemde nüfusun çoğalması ve açlık seviyesinin oranınını ifade etmekle birlikte bu oranın artması, sistemi sonuç olarak sabit ve belirli çıktılar vermek yerine tahmin edilemeyen kaotik bir sisteme dönüşmektedir. Şekil 2.2'de *r* parametresi 2,8-3 arasında alındığında x değeri 0,6-0,7 arasında sabit bir değer almaktadır. Eğer *r* parametresi 3,8 değeri alırsa elde edilen x değerleri 1-0,1 arasında kaotik bir biçimde değişmektedir.



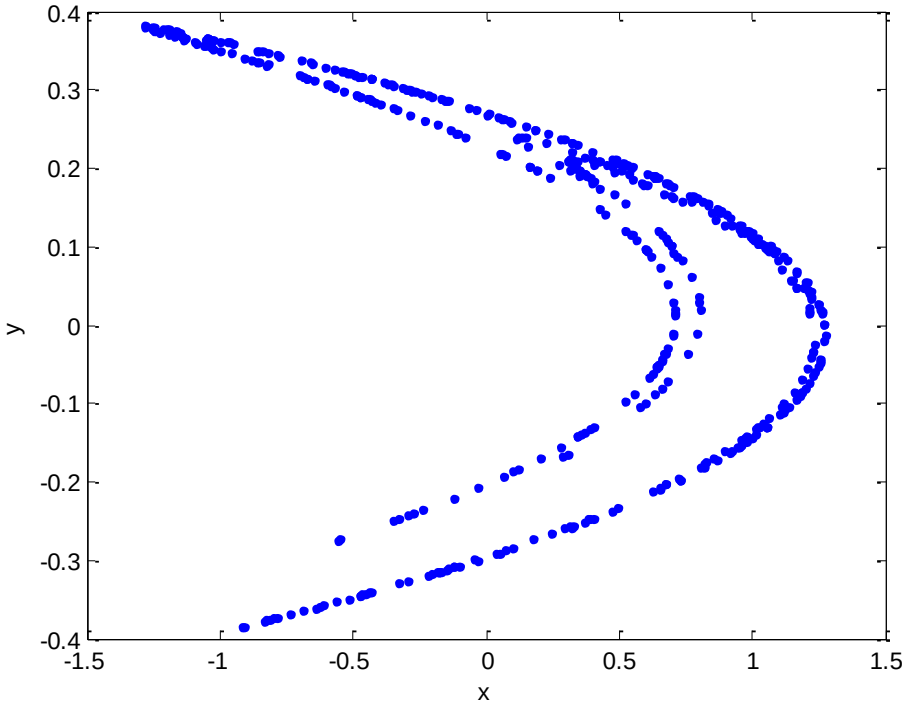
Şekil 2.2 - Farklı r değerlerinin Logistic Map üzerindeki etkisi.

2.2. Henon Map Kaotik Sistemi

Henon Map, ayrık zamanlı bir dinamik sistem olarak tanımlanır. Kaotik davranış gösteren dinamik sistemler içerisinde en çok çalışılmış sistemlerden biri olarak gösterilebilir. Henon Map, Michel Hénon tarafından ortaya konmuş, bir sonraki bölümde açıklanan Lorenz modelinin basitleştirilmiş halidir. İşlevsel olarak bir düzlemdeki x ve y noktasını başka bir noktaya aktarır (Hénon, 1976).

$$x_{n+1} = y_n + 1 - ax_n^2,$$
$$y_{n+1} = bx_n. \tag{2.2}$$

Eşitlik 2.2'de verilen Henon Map denklemlerinde önemli olan iki parametre bulunmaktadır. Bunlar a ve b parametreleridir. Standart kabul edilen a=1,4 ve b=0,3 değerleri için sistem kaotik özellik göstermektedir. Standart a ve b değerlerine göre Henon çekicisinin grafiği Şekil 2.3'de görülmektedir.



Şekil 2.3 - a = 1.4 ve b = 0.3 için Henon Map çözümü.

2.3. Lorenz Kaotik Sistemi

Lorenz çekicisi, Edward N. Lorenz tarafından 1963 yılında geliştirilen, Lorenz osilatörünün uzun vadeli davranışının lineer olmayan dinamik sistem karşılığıdır. Lorenz osilatörü, kaotik akış gösteren üç boyutlu dinamik bir sistemdir. Literatürde Lorenz çekicisi, genel anlamda kaotik sistem uygulamalarının bir temsili olarak ele alındığından tez çalışması için önemlidir. Lorenz denklemini literatüre kazandıran Edward Lorenz, kaotik akışın lineer olmayan diferansiyel denklemler ile modellenmesini sağlamıştır. Edward Lorenz ayrıca kaos tanımı ve koşullarını kendi yazdığı "Essence of Chaos" adlı eserinde açıklamıştır (Lorenz, 1963) (Lorenz, 1995).

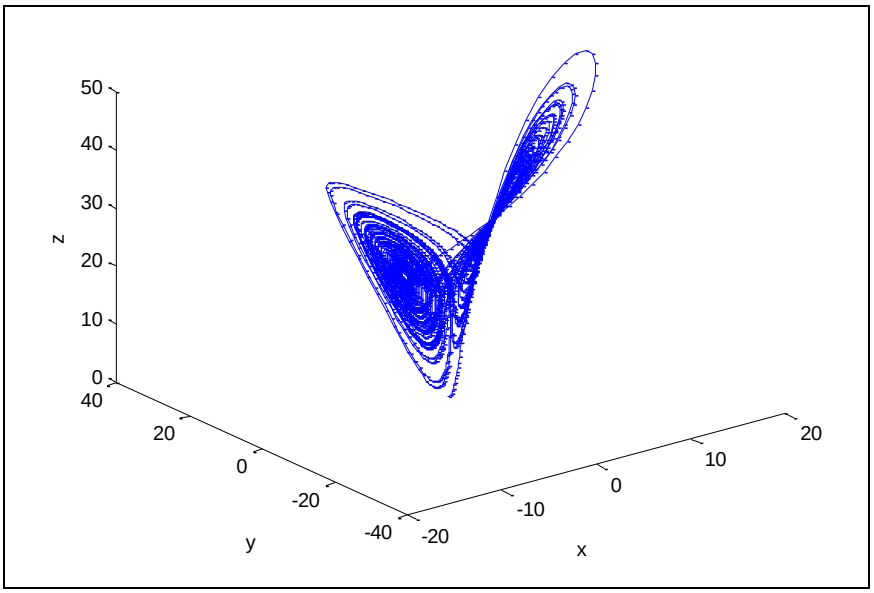
Bu tez çalışmasında, literatürde yaygın olarak bilinen Lorenz kullanılmaktadır. Ayrıca literatürde yapılan çalışmalarda Lorenz sistemi üzerinde performans ve uygulama bakımından ölçütlerin belirlenmiş olması tezde kullanılan kaotik sistem seçiminde tercih sebebi olmuştur. Eşitlik 2.3'de Lorenz sisteminin matematiksel denklemi verilmiştir:

$$\frac{dx}{dt} = \sigma(y - x)$$
$$\frac{dy}{dx} = x(\rho - z) - y \qquad (2.3)$$
$$\frac{dz}{dt} = xy - \beta z$$

Lorenz sistemi, Eşitlik 2.3'te görüldüğü şekilde üç faklı denklemle tanımlanır. σ sayısı, Prandtl sayısını ifade etmektedir (White,2006). ρ sayısı ise Rayleigh sayısını gösterir (Turcotte ve Schubert, 2002). Genelde sayısal olarak, $\sigma = 10$, $\beta = \frac{8}{3}$, ρ ise değişken değer almaktadır. Çalışmada kullanılan bu parametreler Şekil 2.4'de görüldüğü gibi $\sigma = 10$, $\beta = 2{,}5$, $\rho = 28$ olarak alınmıştır.

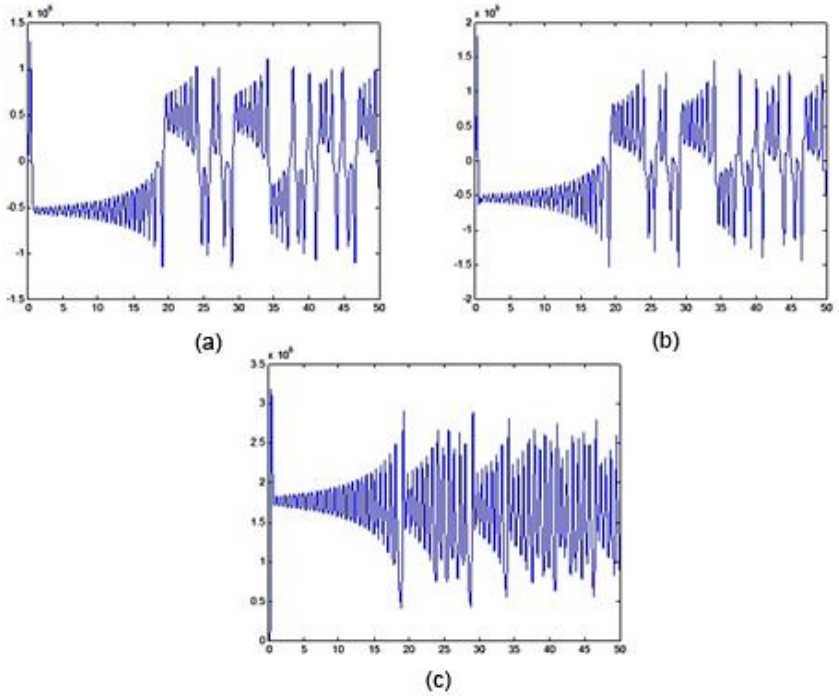
```
nbit=6; %integer part
mbit=6; %fraction part

%Change of Parameters for Fixed Purposes
b = 2.5; % set the parameters Floating:8/3;
sig = 10;
r = 28;
x0 = [0; 1; 0]; % initial conditions

t= 0:0.01:50;
hvalue=0.01;
```

Şekil 2.4 - Matlab Lorenz parametre tanımlamaları.



Şekil 2.5- Lorenz çekicisinin üç boyutlu görünümü.

Eşitlik 2.3'ün çözümü sonucunda elde edilen Şekil 2.5'te grafiği görülen Lorenz sisteminin çıkışlarını her bir koordinat sistemi için ayrı ayrı inceleyecek olursak Şekil 2.6'daki gibi düzlemlerin zamana göre değişimlerini elde ederiz. Buradaki *x*, *y*, *z* değişkenleri zamana göre kaotik bir özellik göstermektedir.



Şekil 2.6 - Lorenz sistemi x,y ve z durum değişkenlerinin zamana göre değişimi.

2.3.1. Lorenz'in 4. dereceden Runge-Kutta ile çözümü

Nümerik analizde Runge-Kutta metotları, tipik diferansiyel denklemlerin belirli ya da belirsiz aşamalı metotların çözümünün aldığı yaklaşık değerlerin bulunmasında önemli bir yere sahiptir. Difransiyel denklemlerin çözüm yöntemleri karşılaştırıldığında, Euler metodu en popüler fakat aynı zamanda hata oranı en yüksek olan bir çözüm yöntemidir. İkinci ve üçüncü dereceden olan Runge-Kutta çözüm metodları ise Euler metoduna göre daha doğru sonuçlar vermektedir. Buna karşın 4.dereceden Runge-Kutta çözümü daha karmaşık olmakla birlikte hata oranı göz önüne alındığında çalışma için uygun bir çözüm yöntemi olarak karşımıza çıkmaktadır. RK4 olarak da geçen Runge Kutta 4. dereceden çözümü, başlangıç koşulları Eşitlik 2.4'te görüldüğü şekilde ele alındığında *y*'nin değişim hızı, *y* ve *t* (zaman) girişlerini kullanan bir fonksiyondur. Başlangıç zamanı olarak ele alınan t_0 zamanında *y* değeri y_0 olarak tanımlanmıştır.

$$y' = f(t, y), \quad y(t_0) = y_0 \tag{2.4}$$

Başlangıç koşulları verilen problemin çözümü ise Eşitlik 2.5, Eşitlik 2.6 ve Eşitlik 2.7 kullanılarak elde edilmektedir.

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{2.5}$$

$$t_{n+1} = t_n + h \tag{2.6}$$

$$\left.\begin{aligned} k_1 &= hf(t_n, y_n) \\ k_2 &= hf\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1\right) \\ k_3 &= hf\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2\right) \\ k_4 &= hf(t_n + h, y_n + k_3) \end{aligned}\right] \tag{2.7}$$

Eşitlik 2.5'teki y_{n+1} değeri; Eşitlik 2.6'daki t_{n+1} zamanı ve Eşitlik 2.7'de elde edilen *k* değerleri kullanılarak hesaplanır. Ayrıntıya inildiğinde y_{n+1} değeri bulunurken, y_n ile Eşitlik 2.9'daki delta değeri toplanır. Delta değeri, Eşitlik 2.7'deki *k* değerleri toplamlarının ortalama ağırlığı hesaplanarak bulunur. Bu *k* değerlerini hesaplarken

Eşitlik 2.8'deki eğim parametresi ile çözümlenmek istenilen fonksiyon çarpılır. Bu şekilde sistemin RK4 ile çözümlenmesi sağlanır(WH. Teukolsky vd., 2007).

$$h\,(slope) = h\,f(t, y) = \Delta t\,(dy / dt) = \Delta y. \tag{2.8}$$

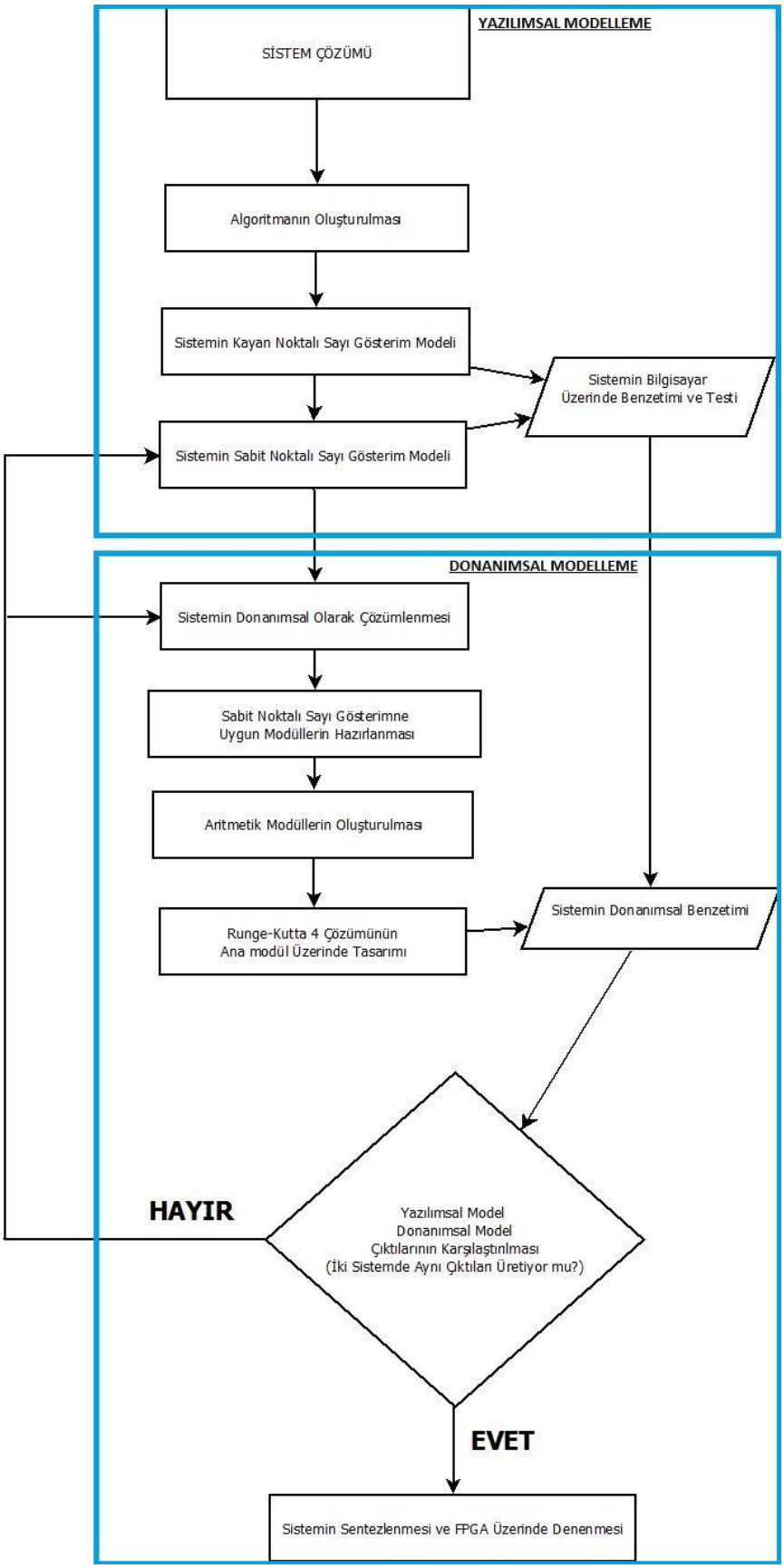
$$\text{delta} = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{2.9}$$

RK4 metodu dördüncü dereceden bir metot olduğundan her adımdaki ortalama hatanın derecesi büyük-O-gösterimine (big-O-notation) göre h^5 derecesindedir. Ayrıca toplam hatanın derecesi ise h^4 türünden hesaplanır (Butcher, 2003).

Lorenz gibi başlangıç koşulları ve en ufak değişimin çok büyük farklar ortaya koyduğu bir sistemde Euler entegralinin kullanılması çok büyük hata oranlarına yol açacaktır. Daha karmaşık veya daha basit olarak birçok farklı tümlev alma yöntemleri bulunmasına karşın Lorenz sistemi için en uygunu RK4 olarak öne çıkmaktadır. Bunun sebebi RK4'ün O(5) yani diferansiyel denklem çözümünün Taylor serisinde 5.dereceden bir hata vermesidir (Fiedler, 2006).

3. LORENZ SİSTEMİNİN YAZILIMSAL MODELLEMESİ

Lorenz Sisteminin donanımsal olarak FPGA üzerinde gerçeklenebilmesi için sistemin ilk olarak algoritma seviyesinde tasarlanması gerekmektedir. Lorenz kaotik sisteminin yazılımsal olarak tasarlanması, donanımsal aşamada sistem ifadesi ve kontrolü için gereklidir. Yazılımsal modelleme için sistemin çeşitli aşamalardan geçmesi gerekmektedir. Bu aşamalar, sistemin algoritmasının tanımlanması, kayan noktalı sayı gösteriminde sistem çözümü, ve son olarak sabit noktalı sayı gösteriminde sistemin modellenmesidir. Yazılımsal model oluşturulduktan sonra, farklı sayı gösterim formatlarında elde edilen sistem çözümleri karşılaştırılabilir. Bu sayede sistemin donanımsal olarak nasıl ifade edilmesi gerektiği anlaşılmaktadır. Yazılımsal modellemede ele alınması gereken aşamalar Şekil 3.1'de görüldüğü gibi birbirleriyle bağlantı içerisindedir. Bu bölümde Şekil 3.1'de görülen sistemin algoritma seviyesinde tasarımı ve sayı gösterim sistemlerinde ifadesi anlatılacaktır. Sayı gösterim ifadelerinden kayan ve sabit noktalı sayı gösterimleri bölüm 3.1 ve 3.2'de sırasıyla incelenmektedir.



Şekil 3.1- Sistemin yazılımsal, donanımsal modelleme ve gerçekleme aşamaları.

3.1. Kayan Noktalı Sayı Gösterim Sistemi

Kayan noktalı (floating point) sayı gösterim sistemi, ondalıklı sayıların donanımsal olarak ifade edilebilmesi için geliştirilmiş bir sayı temsil standardıdır. Kayan noktalı sayı gösteriminin biçimi ve kuralları IEEE (Institute of Electrical and Electronics Engineers) tarafından IEEE-754 formatı olarak belirlenmiştir. Bu sistem içerisinde sayı dönüşüm, ifade, aritmetik hesaplamalar ve yuvarlama işlemleri format dahilinde belirlenmiştir.

Dijital sistemlerde bilindiği üzere sayı tanımlamaları ikili sayı sisteminde saklanması gerekir. Fakat ondalıklı sayıların ikili sayı sisteminde donanımsal olarak tanımlanması ve kullanılması için çözülmesi gereken problemler vardır. İkili sayı sisteminde sayıların iki ve katlarına bölündüğünde oluşan kesirli sayılar gösterim bakımından sonlu bir ifadeye dönüşmektedir. Diğer durumlar için ikili sayı gösterimi sonsuz olacaktır. Bu nedenle kesirli sayıların gösterimi, sayıların karşılaştırılması bakımından önemlidir. Çizelge 3.1'de görüldüğü gibi 10 ile 0,2 sayısının çarpımı kesirli sayı aritmetiğinde 2'ye eşit olmamaktadır. Çözüm yaklaşık kesir değerinde temsil edilebilen kısım kadar değer alacaktır.

Çizelge 3.1 - İkili sayı sisteminde kesirli sayı gösterimi (Wikipedia, 2011).

Kesir	Ondalık	İkili	Yaklaşık Kesir Değeri
1/1	1 ya da 0.999…	1 ya da 0.111….	½ + ¼ + 1/8 + …
½	0.5 ya da 0.499..	0.1 ya da 0.0111…	¼ + 1/8 + 1/16 + …
1/3	0.333…	0.010101…	¼+ 1/16 + 1/64 + …
¼	0.25 ya da 0.2499…	0.01 ya da 0.00111…	1/8 + 1/16 + 1/32 + …
1/5	0.2 ya da 0.199….	0.00110011…	1/8+1/16+1/128...
1/6	0.1666…	0.0010101…	1/8+1/16+1/128...
1/7	0.142857142857…	0.001001…	1/8+1/16+1/128...
1/8	0.125 ya da 0.124999…	0.001 ya da 0.000111…	1/16+1/32+1/64...

Kesirli sayıların gösterimi ikinin katları hariç belirli bir sonlu yapıda olmadığından kesirli sayı ifadeleri ikili sayı sisteminde belirtilirken veri kaybı yaşanmaktadır. Ayrıca 0,0111 şeklinde kesirli bir sayıyı hafıza birimlerinde tutmak için belirli bir formatın tanımlanması ve işlemlerin aynı format üzerinde yapılması

gerekliliği ortaya çıkar. Bu gereklilikler ışığında sabit noktalı sayı sisteminden daha fazla ondalıklı veriyi tutabilen IEEE 754 formatı, kesirli sayı tanımlamalarında kabul gören bir standart olarak karşımıza çıkmaktadır.

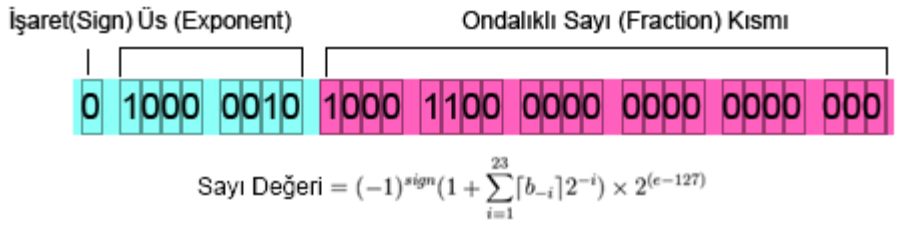
IEEE 754 formatında bazı noktalar performans ve kullanım açısından tez kapsamında yapılan çalışma için önemlidir. Bu durum IEEE 754 formatının sayı tanımlamalarında sabit bir bit alanı kullanma zorunluluğundan kaynaklanmaktadır. 3.2'te görüldüğü gibi kullanılan sayı sisteminin hassasiyetine uygun verinin en az 16-bitlik bir sayı sisteminde gösterim zorunluluğu bulunmaktadır. Genel olarak kaotik sistemlerin FPGA üzerinde gerçeklenmesi üzerine incelenen çalışmalarda, kaotik sistemlerin gerçeklenmesinde 32-bitlik (tek hassasiyetli) sayı formatı kullanılmıştır (IEEE, 2009).

Çizelge 3.2 - IEEE 754-2008 temel formatlar ve özellikleri (IEEE, 2009).

Adı	Genel Adı	Tabanı	Basamak (Digits)	Min. Üs	Maks. Üs	Ondalık Hassasiyet	Ondalık Maks. Üs
Binary16	Half Precision	2	10+1	-14	+15	3.31	4.51
Binary32	Single Precision	2	23+1	-126	+127	7.22	38.23
Binary64	Double Precision	2	52+1	-1022	+1023	15.95	307.95
Binary128	Quadruple Precision	2	112+1	-16382	+16383	34.02	4931.77
Decimal32		10	7	-95	+96	7	96
Decimal64		10	16	-383	+384	16	384
Decimal128		10	34	-6143	+6144	34	6144

Sayı gösterim formatları incelenecek olursa, seçilen format üzerinde yapılan işlemlerin sonunda elde edilen sonuçların karşılaştırılması için ondalıklı sayı sistemine çevrilerek değerlerin incelenmesi gerekmektedir. Bu durumda IEEE-754 standardında tanımlanan kayan noktalı ve tezde kullanılan sabit noktalı sayı gösterim sistemlerinin dönüşümleri Şekil 3.2 ve Şekil 3.3'de görülebilmektedir.

IEEE-754 formatında 32-bitlik tek hassasiyetli (Single Precision) bir sayının gösterimi Şekil 3.2'de görüldüğü gibi ilk biti işaret biti, sonraki 8 bitlik bir üs değeri ile sayının ondalıklı kısmının tutulduğu 23 bitlik bir alandan meydana gelmektedir. Şekil 3.2'de 12,375 sayısının IEEE-754 formatında gösterimi ifade edilmektedir.



Şekil 3.2 - IEEE-754 formatında tek hassasiyetli 32-bit kayan sayı sistemi ifadesi ve değerinin hesaplanması.

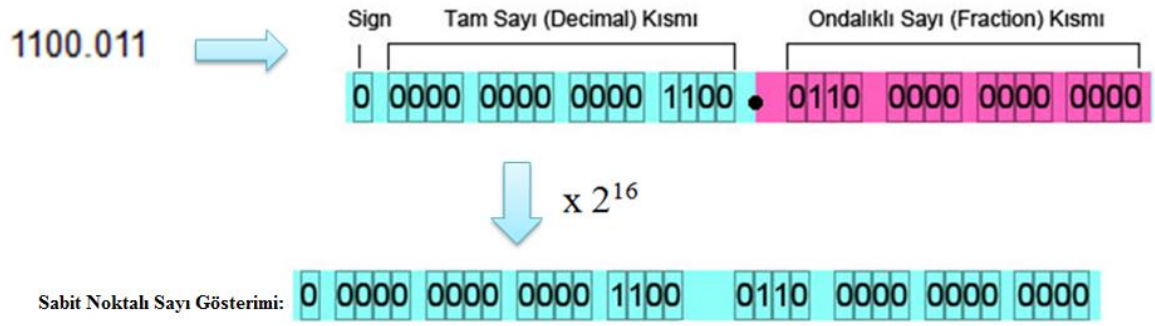
Elde edilen IEEE-754 formatında belirtilen sayı değeri ise ondalıklı sayı kısmının 2'lik sayı sistemi uyarınca toplanarak elde edilen sonuç 1 ile toplandığında üs değerine göre sayı ile çarpılarak ondalık sistemdeki karşılığı elde edilir. En son işlemde ise sayının negatif ya da pozitif olduğu işaret (sign) biti ile belirlenir.

Kayan noktalı sayı gösteriminde toplam bit sayısı ve sayı temsil hassasiyeti sabittir. IEEE-754 formatının avantajı 32-bitlik sabit noktalı sayı gösterimine oranla daha fazla ondalıklı sayıyı temsil edebilmesidir

3.2. Sabit Noktalı Sayı Gösterim Sistemi

Sabit noktalı sayı gösterim sistemi, sabit sayıda basamak kullanılarak ondalıklı sayıların gösterilmesidir. Fakat bu gösterimde sayının ifade edildiği kesirli sayı bölümü genişletilerek tam sayıya dönüştürülmektedir. Ondalıklı sayıların tam sayıya dönüşümünde sayıların genişletilmesi, sayının temsil formatı büyüklüğüne göre değişir.

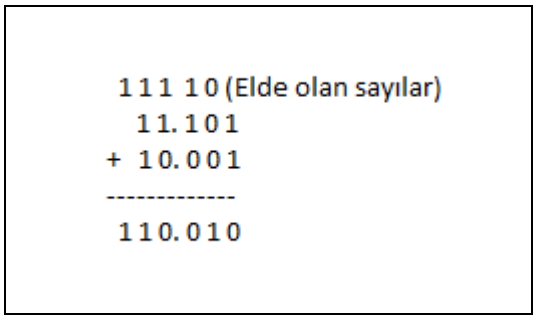
Sabit noktalı sayı gösteriminde, IEEE-754 formatında ifade ettiğimiz 12,375 sayısını 16 bit tamsayı ve 16 bit kesirli sayı olarak ele alıp 16Q16 ile ifade edecek olursak sayımız ikili bit sisteminde Şekil 3.3'de görüldüğü gibi 1100,011 şeklinde ifade edilmektedir. Donanımsal olarak sayıların tanımlanmasında 16Q16 sisteminde ilk 16 bit ondalıklı kısmı ifade ettiğinden tam sayı kısmının hesaplanması için elde edilen sayının 16-bit sağa kaydırılması gerekmektedir. Aynı şekilde 16Q16 gösteriminde belirtileceği zaman ise sayı 16 bit ile sola doğru kaydırılır yani 2^{16} ile sayı genişletilir.



Şekil 3.3 - Sabit sayı gösteriminde 16Q16 formatında sayının temsili.

Şekil 3.3'de sabit noktalı sayı gösterimi ile ifade edilen sayıda işaret (sign) biti 16Q16 ifadesinin içerisinde katılmamıştır. 2'ye tümleyen sistemde işaret bitini hesaba kattığımızda tamsayı ifade kısmı 15-bit uzunluğunda ifade edilmelidir.

Sabit noktalı sayı sisteminde kullanılan aritmetik işlemler, format dahilinde belirlenen sayı temsil uzunluklarına göre ele alınmaktadır. Kesirli sayılar, belirtilen formatta yapılan işlemlerden sonra yuvarlama için kontrol edilir. Sonrasında ise sayının doygunluk kontrolü yapılır. Doygunluğa ulaşan sayı, temsiline göre en yüksek ya da en düşük değeri alabilir. Eğer sayı sisteminde tamsayı kısmı 2 bit, ondalıklı sayı kısmı ise 3 bit ile gösterilirse, Şekil 3.4'da belirtilen toplama işleminin sonucunda elde edilecek sayı doygunluğa ulaştığından sonuç 5-bitlik gösterimde en büyük ifade olan 11.111 değerini alacaktır.



Şekil 3.4 - İkili sayı sisteminde kesirli sayıların toplanması.

Sabit noktalı sayı gösterimi, kayan noktalı sayı gösterimine göre sayı temsil oranında daha az ondalıklı sayı temsili sağlamasına karşın farklı bit uzunluklarında sayı gösterimi yapılabilmesi, aritmetik işlemlerin donanımsal olarak kolay bir biçimde oluşturulması ve kesirli sayı ifade dönüşümünün kolaylığı bakımından avantaj sağlamaktadır.

3.2.1. Sabit noktalı sayı sisteminde yuvarlama işlemi

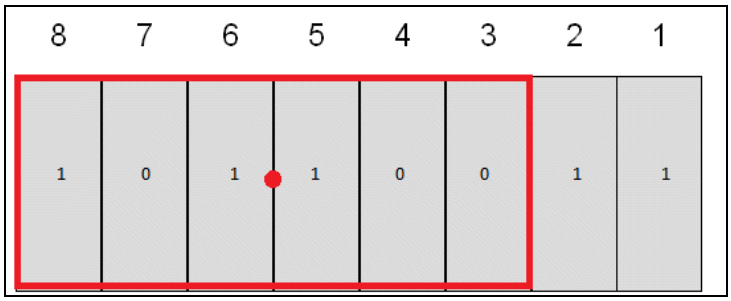
Dijital sistemlerde ifade edilmek istenen sayılar, donanımsal sınırlamalar nedeniyle istenilen hassasiyette saklanamamaktadır. Bu sebeple sayıların sınırlar dahilinde en az hata oranı ile gösterilmesi gerekmektedir. Yuvarlama işlemi gerçek sayıların ifadesinin donanımsal ortamda en az hata oranı ile saklanması için gerçekleştirilen ve veri kaybını azaltan bir yöntem olarak karşımıza çıkmaktadır.

Tez çalışmasında gerçekleştirilen matematiksel işlemlerin sonucunda, sayıların sabit noktalı sayı gösterimine uygun bir şekilde yuvarlanması önemlidir. Matlab yazılımı üzerinde hesaplanan değerler ile Verilog üzerinde elde edilen değerler arasındaki eşitliğin belirlenmesinde, sayıların yuvarlanması ile veri kaybına sebep olan değişimler tez çalışması içerisinde aynı şekilde ele alınmıştır. Bu sayede çıkış değerleri üzerinde eşleşme sağlanmış ve Lorenz sistemi donanımsal olarak tasarlanmıştır.

Tasarlanan sistemler içerisindeki matematiksel işlemler sonucunda veri kaybı iki şekilde ortaya çıkmaktadır. Birincisi elde edilen sayının gösterim yapılacak dijital formattan daha büyük ya da negatif sayılar için daha küçük olması durumunda sistemdeki sayı formatında doygunluk oluşacağı için maksimum ve minimum sayılar ele alınarak işlem devam ettirilmektedir ve veri kaybı yaşanmaktadır. İkinci veri kaybının yaşandığı durum ise elde edilen sayıdaki en önemsiz bitlerin sayı formatına uyması için atılması gerektiğinde ortaya çıkmaktadır. Bu durumda yapılan bütün nümerik işlemlerde yuvarlama işlemi uygulanması gerekmektedir.

İkili sayı sisteminde yuvarlama işlemi kolay bir şekilde yapılabilmektedir. Yuvarlama işlemi, veri kaybının gerçekleştiği yani atılacak sayının en yüksek anlamlı bit değerine göre yapılmaktadır. Şekil 3.5'de görülen tamsayı kısmının 3-bit ve ondalıklı sayı kısmının 3-bit ile oluştuğu 3Q3 şeklinde belirtilen bir sayı gösterim sisteminde yuvarlama işlemi, en düşük anlamlı bitler olan birinci ve ikinci bitlerin atılmasıyla yapılmaktadır. En düşük anlamlı birinci ve ikinci bitler atıldığı sırada ikinci bitin değeri üçüncü bite eklenerek yuvarlama işlemi yapılmış olur. İkili sayı sisteminde elde edilen sonuç "101.101" ve bunun onluk sistemdeki karşılığı 5,625 olacaktır. İkiye tümleyen sayı olarak ele alınırsa sayı, "101.101" olacak ve onluk sistemde belirlenen karşılığı "-2,375" olacaktır. Sistemde yapılan yuvarlama işlemi ondalıklı sayıları atmak

değil belirtilen sayı aralığının dışında kalan düşük anlamlı bitler üzerinden en az hata oranına ulaşmaktır.



Şekil 3.5 - İkili sayı sisteminde 3-3lük bit sayı gösterimi.

Şekil 3.5 üzerinde görülen sayıda oluşan hata oranı, ikiye tümleyen sisteminde gerçek değer "-2,40625" iken elde ettiğimiz yuvarlama sonucunda "-2,378"dir. Burada oluşan hata "2,40625-2,378= 0,02825" olmaktadır. Eğer sayımız yuvarlama işlemi yapılmadan "101.100" şeklinde alınmış olsaydı elde edeceğimiz değer "2,5" olacaktı. Bu durumda hata oranımız "2,5-2,40625=0,09375" olacaktı. Yuvarlama yaptığımız durumda belirttiğimiz sayı, verilen sayıya "0,09375-0,02825=0,0655" kadar daha yakın bir değerdir.

Lorenz gibi başlangıç koşullarına bağımlı bir sistem, hassas ölçümler gerektirdiğinden sistemde oluşan en ufak sapmalar istenilen sonuçtan çok fazla uzaklaşılmasına neden olur. Bu sebeple yuvarlama işleminin önemi, istenilen sayılardan minimum hata oranı ile sapma sağlanmasıdır. Belirlenen sayı gösterim formatında dijital olarak kaç bitlik bir tanımlama yapılmak isteniyorsa buna uygun bir şekilde sistemdeki veri kaybı göz önünde bulundurulmalıdır.

3.3. Matlab Üzerinde Lorenz Sistem Algoritmasının Modellenmesi

Lorenz sisteminin yazılımsal olarak çözümlenebilmesi için sistemin başlangıç koşulları, sistem parametreleri ve sistemin toplam çözüm aralığı belirlenmelidir. Yazılımsal modelleme, belirlenen bu koşullar dahilinde bilgisayar üzerinde sistemin tasarlanmasıdır.

Lorenz kaotik sistem çözümünün gerçekte aldığı değerler ile yazılımsal ve donanımsal sistem sonuçları arasında farklar bulunmaktadır. Bu fark sayıların donanımsal olarak ifade edilebilmesi için belirli sınırlamaların bulunmasından

kaynaklanır. Sayının kaç bit ile ifade edileceği, sistem hassasiyetinin ne olacağı gibi durumlar ve donanım maliyetleri bu noktada sınırlamaların birer kaynağıdır. Bu nedenle kullanılan sayılar ve işlem sonuçları, donanımda sayı gösterim sistemleri kullanılarak yaklaşık olarak tutulmaktadır.

Matlab yazılımında sisteminin RK4 çözümünün algoritması Şekil 3.6'de görüldüğü gibi tanımlanmıştır (Hall, 2002). Şekil 3.6'de görülen *feval* komutu ile Lorenz sistemi içinde bulunduran *frhs* fonksiyonu ilk değerler kullanılarak hesaplanmaktadır. Eğer farklı bir sistem RK4 ile çözümlenmek isteniyorsa Matlab içerisinde *frhs* fonksiyonu değiştirilmelidir.

Yazılımsal olarak tanımlanan Şekil 3.6'deki Lorenz sistemi çözümü, kayan noktalı sayı gösteriminde ifade edilerek oluşturulmaktadır. Sistem fonksiyonları Şekil 3.6'de görüldüğü gibi birçok fonksiyon tez çalışmasında tek Matlab yazılım dosyası içerisinde birleştirilmiştir.

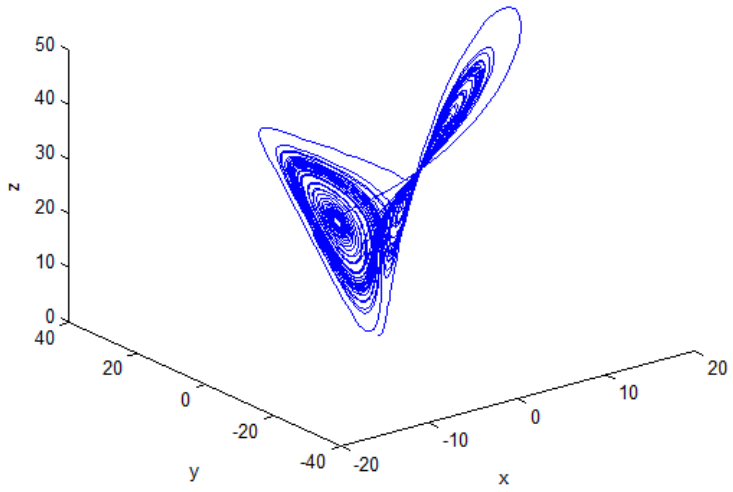
```
% oderk.m
% C. Hall
% [t, x] = oderk(frhs,tspan,x0)
% implements runge-kutta 4th order algorithm
% frhs = 'filename' where filename.m is m-file
% tspan = 1xN matrix of t values
% x0 = nx1 matrix of initial conditions
% returns t=tspan, x=nxN matrix
%                  x(:,i) = x(t(i))
function [tspan,x] = oderk(frhs,tspan,x0)
    N=length(tspan);
    n=length(x0);
    x0=reshape(x0,n,1);
    x=[x0 zeros(n,N-1)];
    w=x0;
    for i=1:N-1
        h=tspan(i+1)-tspan(i);
        t=tspan(i);
        K1=h*feval(frhs,t,w);
        K2=h*feval(frhs,t+h/2,w+K1/2);
        K3=h*feval(frhs.t+h/2.w+K2/2):
        K4=h*feval(frhs,t+h,w+K3);
        w=w+(K1+2*K2+2*K3+K4)/6;
        x(:,i+1)=w;
    end
    x=x';
```

Şekil 3.6 - Matlab'da RK4 gerçeklenmesi (Hall, 2002).

Matlab'da kayan noktalı sayı sistemi kullanılarak, yani herhangi bir kesim ya da veri kaybı oluşturulmadan, Şekil 3.6'deki RK4 çözümü kullanılarak Şekil 3.7'da görülen Lorenz sisteminin çözümlenmesi sağlanmıştır. Tanımlanan x(1); x eksenindeki değer kümesini, x(2); y eksenindeki değer kümesini, x(3); z eksenindeki değer kümesini temsil etmektedir. Lorenz sisteminde kayan noktalı(fixed point) sayı gösteriminde yapılan çözümün sonucu Şekil 3.8'de görüldüğü gibidir.

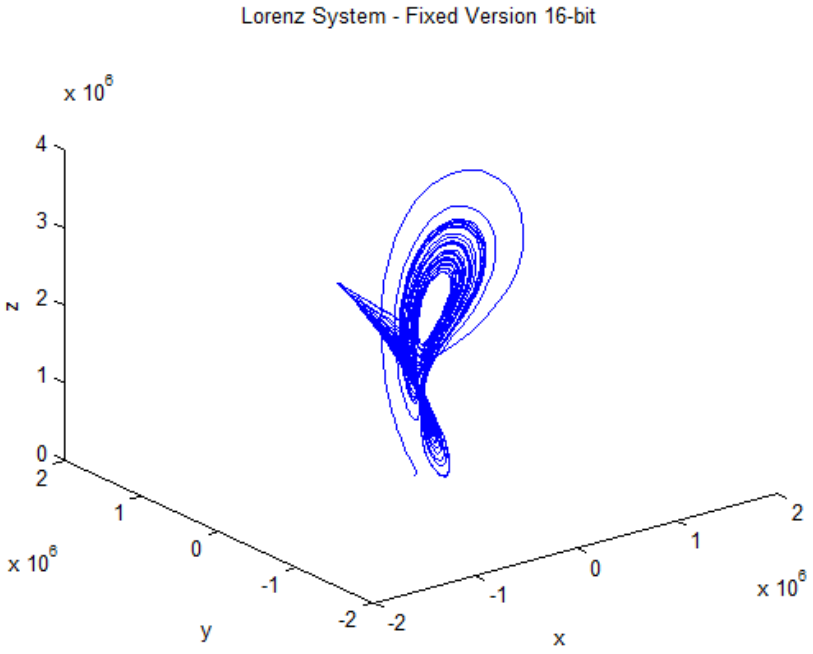
```
clear
clc
b=2.5;
sig = 10;
r = 28;
x0 = [0; 1; 0];
t = 0:0.01:50;
tspan=t;

N=length(tspan);
n=length(x0);
x0=reshape(x0,n,1);
x=[x0 zeros(n,N-1)];
w=x0;
for i=1:N-1
h=tspan(i+1)-tspan(i);
t=tspan(i);
wx=w;
K1=h*  [sig*(wx(2)-wx(1));-wx(1)*wx(3)+r*wx(1)-wx(2);wx(1)*wx(2)-b*wx(3)];
wx=w+K1/2;
K2=h*  [sig*(wx(2)-wx(1));-wx(1)*wx(3)+r*wx(1)-wx(2);wx(1)*wx(2)-b*wx(3)];
wx=w+K2/2;
K3=h*  [sig*(wx(2)-wx(1));-wx(1)*wx(3)+r*wx(1)-wx(2);wx(1)*wx(2)-b*wx(3)];
K4=h*  [sig*(wx(2)-wx(1));-wx(1)*wx(3)+r*wx(1)-wx(2);wx(1)*wx(2)-b*wx(3)];
w=w+(K1+2*K2+2*K3+K4)/6;
x(:,i+1)=w;
end
x=x';
```

Şekil 3.7 - Lorenz sisteminin RK4 ile çözümü.



Şekil 3.8 - Lorenz kayan noktalı sayı sisteminde çözümü.

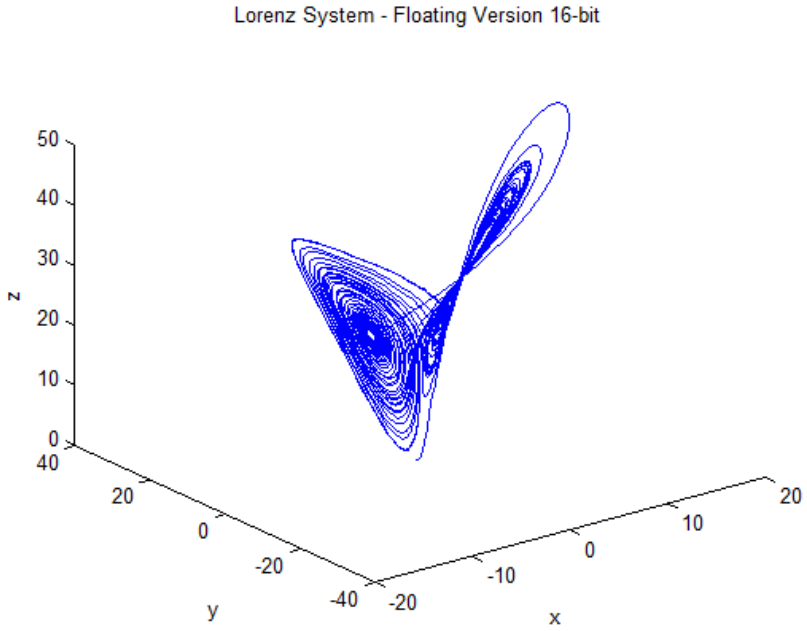
Sabit noktalı sayı ifadesinde, Lorenz sisteminin çözümünü elde edebilmek için tanımlanan sayı gösterimine uygun yazılımı tasarlamak gerekmektedir. Bu durumda dikkat edilmesi gereken koşul ise tamsayı ve ondalıklı bölümün kayan sayı sisteminden sabit sayı sistemine çeviren fonksiyonu belirlemektir. 16Q16'lık bir sistemde yani 16-bit tamsayı kısmı ve 16-bit ondalıklı kısmı olan bir sayı sisteminde bitlerin sola doğru 16-bit kaydırılması gerekmektedir. Bu sayede elde edilen sayı 2'nin 16ıncı katı olacaktır. Daha sonrasında elde edilen sayıda yuvarlama işlemi uygulanarak 2^{16} ile genişletilmiş sayının ondalıklı kalan kısımlarından kurtulmak gerekmektedir. Sonrasında 16-bit sağa kaydırma işlemi uygulandığında 16-bitlik kesirli sayı gösterimine ulaşılmış olur. 16-bitlik tam sayı kısmında sayıyı ifade edebilmek için taşma kontrolü (overflow check) yapılması gerekmektedir. Eğer sayı taşma durumunda ise sayı doyum (saturation) değeri olan 16Q16'lık sistemdeki en yüksek ya da en düşük sayısal değeri alacaktır. Ayrıca belirlenen 16Q16'lık her işlem sonucunda elde edilen sayının yine aynı formata dönüştürülmesi gerekmektedir.



Şekil 3.9 - Lorenz sistemi tam sayı modeli 16Q16 gösteriminin çözümü.

Lorenz sistemi, FPGA üzerinde Şekil 3.9'de görülen grafikte olduğu gibi 16Q16'lık sistemde bitlerin 16 bit sola doğru kaydırılmış durumunu göstermektedir. Matlab yazılımındaki tasarımın benzetimi sonucunda elde edilen değerler ile Şekil 3.9'de görülen değerler incelenerek karşılaştırma yapılır. Şekil 3.10'te ise elde edilen

tam sayı modelinin çözümü gösterilmektedir. Kayan sayı sistemi ile oluşan fark, sistemin hata oranını vermektedir.



Şekil 3.10 -Lorenz sistemi tam sayı modeli çözümü.

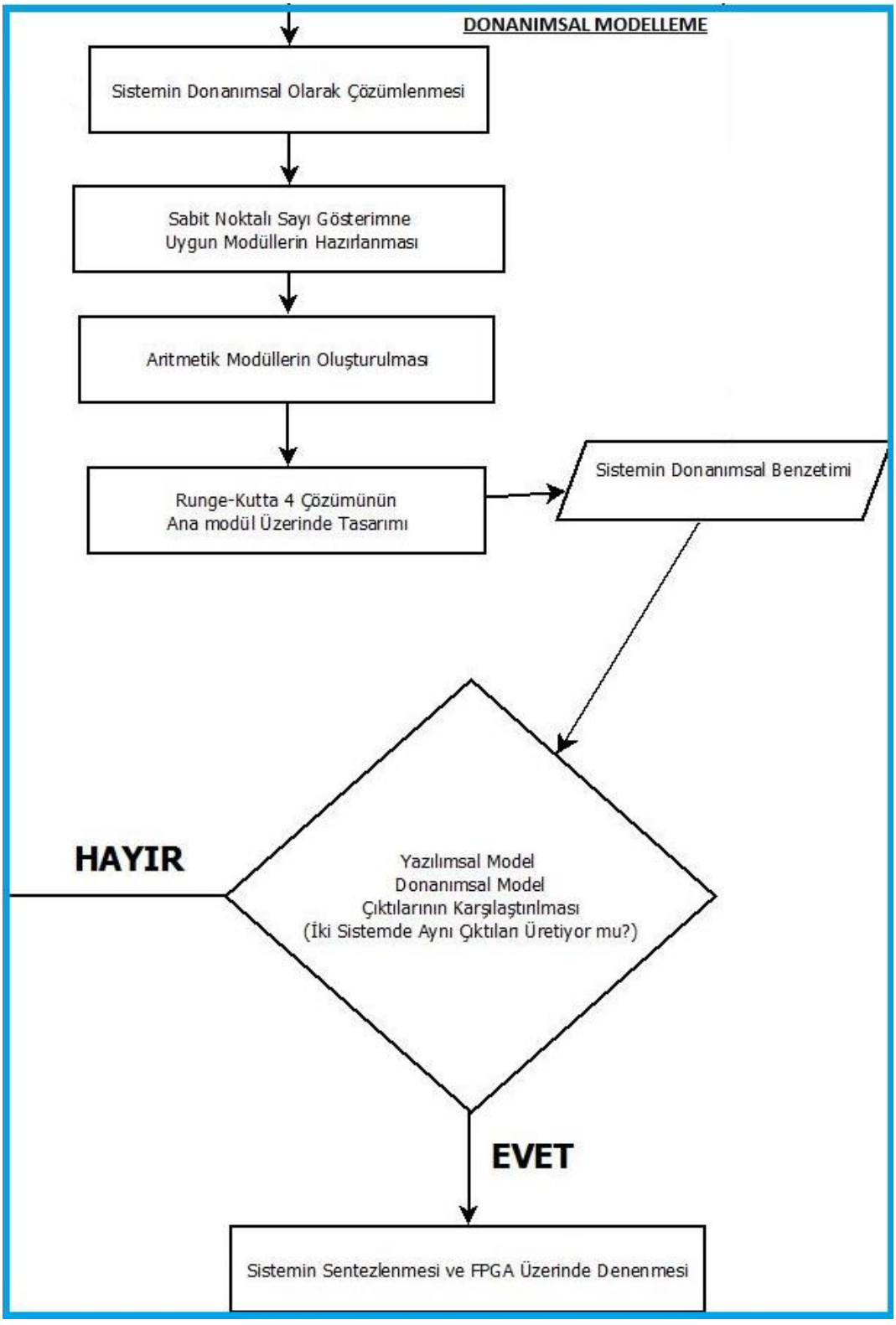
Matlab gibi yüksek seviyeli bir dil ile lojik bir sistemi karşılaştırabilmek için iki sistemde de aynı koşulların sağlanması gerektiğinden, açıklanan sabit sayı üzerinde Lorenz sisteminin modellenmesi Matlab yazılımı ile sağlanmıştır. Bu sayede Verilog dili kullanılarak yapılan tasarım ile bilgisayar ortamında oluşturulan model karşılaştırılabilmektedir.

4. LORENZ SİSTEMİNİN DONANIMSAL MODELLEMESİ

Lorenz kaotik sisteminin gerçeklenebilmesi için donanımsal olarak modellenmesi gerekmektedir. Donanımsal modelleme, sistemin belirlenen sınırlar dahilinde lojik devre olarak oluşturulabilir hale getirilmesidir. Donanımsal modellemede Verilog dili kullanılarak sistemin lojik yapısı tanımlanmaktadır.,

Bu bölümde donanımsal modelin oluşturulabilmesi için Şekil 4.1'deki sistemin donanımsal modellemesi, sabit sayı gösterimine göre modüllerin tanımlanması, aritmetik modüllerin oluşturulması ve sistemin benzetimi çerçevesinde;

- sayısal tasarım,
- kapı seviyede modelleme,
- yazmaç ve davranışsal seviyede modelleme,
- donanımsal sistem modülleri,
- parametrik sistem yapısı konuları ele alınacaktır.

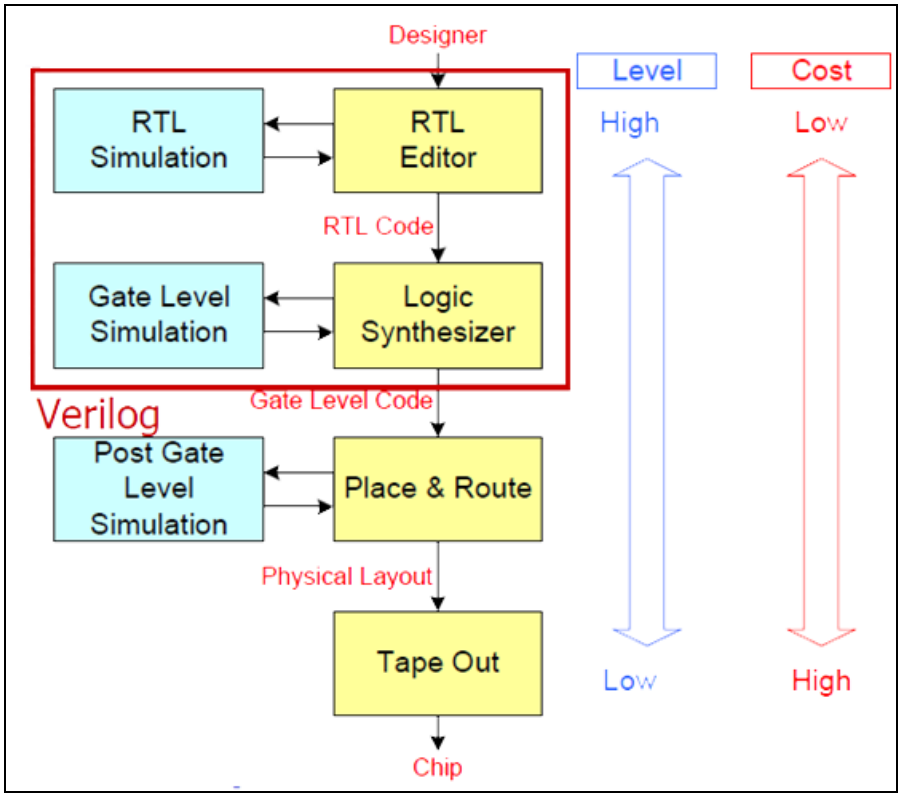


Şekil 4.1 - Lorenz sisteminin donanımsal modelleme ve sentez aşamaları.

4.1. Sayısal Tasarım

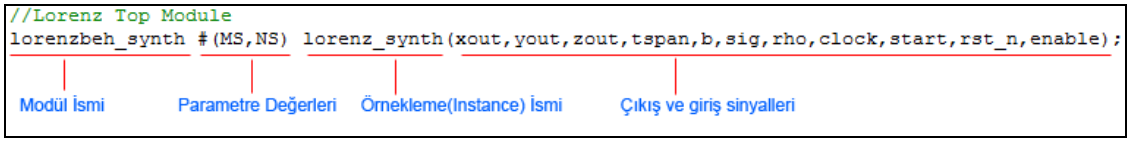
Sayısal tasarım, oluşturulmak istenen sistem işleyişinin lojik yapılara dönüştürülmesi ve lojik devresinin tasarlanması işlemidir. Lorenz kaotik sistemini lojik devre olarak gerçekleyebilmek için Verilog dilinde gerekli bütün modüller Bölüm 4.2.1'de tanımlanmıştır. Tasarım aşamasında Verilog dili kullanılırken mantıksal tasarım ilkeleri göz önünde bulundurulmuştur.

Sayısal tasarım yapılırken Şekil 4.2'deki şemada görüldüğü gibi tasarımcı oluşturmak istediği devreyi Verilog dili kullanarak tasarlarken kapı seviyesi, RTL (Yazıcı Transfer Seviyesi) ya da davranışsal(behavioural) seviyede kodlarını yazar. RTL ve Kapı seviyesindeki gerçeklenme, sentezleme yoluyla oluşturulur. RTL modeli kullanılarak yapılan benzetimler ise yine bu katmanda yer alır. FPGA yongasının programlanması için oluşturulacak gerekli donanımsal özelliklerin, girişlerin ve çıkışların FPGA üzerinde tanımlanması gerekmektedir. Ayrıca donanım üzerine sığdırma (fitter) işleminin yapılması gerekmektedir. Bütün bu tanımlamalar yapıldıktan sonra lojik devre FPGA üzerine aktarılmış olur.



Şekil 4.2 - FPGA tasarım yöntemi (Chao, 2005).

Verilog programlamada tasarım hiyerarşik bir şekilde yapılabilmektedir. Oluşturulan her modül bir diğer modül içerisinde çağrılarak kullanılabilir. Şekil 4.3'de benzetim için hazırlanan modül içerisinde ana modülün somutlaştırılarak kullanımı gösterilmektedir.



Şekil 4.3 - Verilog modülünün örnek somutlaştırılması.

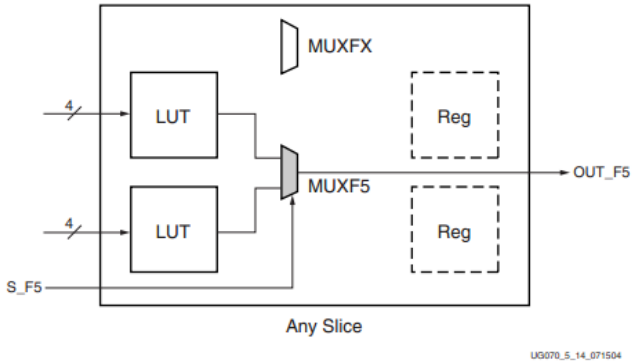
4.1.1. FPGA, Verilog, Sentezleme Nedir?

FPGA, alan programlanabilir kapı dizisi anlamına gelmekle birlikte üretim sonrasında lojik olarak donanımsal devrelerin oluşturulması için tasarlanmış bir üründür. Bu programlanabilir bölgeler kullanılarak istenilen donanımsal özellikler Verilog veya VHDL donanım tasarım dilleri aracılığıyla oluşturulmaktadır. FPGA, alanlar içerisinde programlanabilir anahtarlardan ve bağlantı bölümlerinden oluşmaktadır.

FPGA üzerinde devre oluşturabilmek için, donanım tanımlama dilleri kullanılmaktadır. Tezde önerilen yöntem, Verilog donanım tanımlama dili kullanarak Lorenz kaotik sisteminin FPGA üzerinde çalışabilir hale getirilmesidir. Yazılan kod Verilog 2001 standartlarına uymaktadır. Verilog dilinde hazırlanan mantıksal modüllerin programlanması ise Xilinx ISE ya da Quartus2 – 9,1 yazılımları kullanılarak yapılmış ve sistemin benzetimi gerçekleştirilmiştir. Benzetim sonrasında elde edilen sonuçlarla MATLAB ortamında oluşturulan benzetim sonuçları karşılaştırılmıştır.

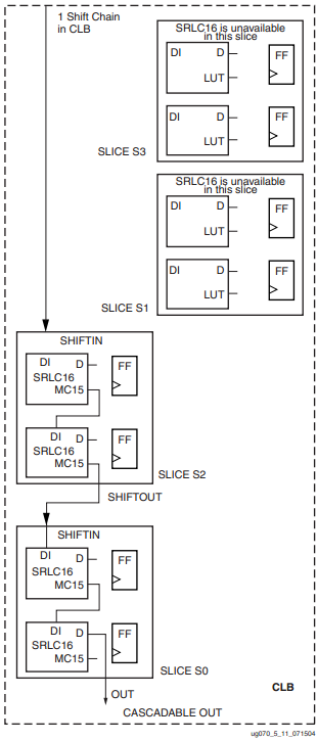
Lojik sentez, dijital devrelerde lojik bağlaşımların otomatik oluşturulma işlemine denir. Sentezleme aşamasında soyut devre davranışı RTL (Yazmaç Transfer Seviyesi) kullanılarak yapılan tasarımla mantıksal kapılara dönüştürme işlemi gerçekleştirilir. Sentezleme işlemi için tasarım, HDL programlama dilleri kullanılarak yapılır. Bu diller arasında en popüler olanları Verilog ve VHDL'dir (Jiang ve Devadas, 2008). Tezde kullanılan kaotik sistem, Verilog ile tasarlanarak sentezleme işlemi yapıldıktan sonra oluşturulan devre, FPGA üzerinde çalışır hale getirilmiştir.

FPGA'de lojik elemanlar birleşerek dilim(slice) olarak tanımlanan yapıları ortaya çıkarmaktadırlar. Literatürde Lorenz ve diğer kaotik sistemlerin, FPGA üzerindeki uygulamalarına performans açısından bakıldığında genellikle ölçütler FPGA üzerindeki dilim sayıları ile belirlenmektedir. Şekil 4.4'deki yapı basitleştirilmiş bir dilim yapısının örneğini göstermektedir (Xilinx, 2008).



Şekil 4.4 - Basitleştirilmiş Virtex 4 dilim yapısı (Xilinx, 2008).

Şekil 4.5'deki ardışık kaymalı yazmaç yapısında görüldüğü gibi daha karmaşık lojik devrelerin oluşturulmasında dilim yapısı kullanılır. Bu sayede dilim yapısı, lojik devreleri alan bazında karşılaştırma olanağı sağlamaktadır.



Şekil 4.5 - Ardışık kaydırmalı yazmaç (Xilinx, 2008).

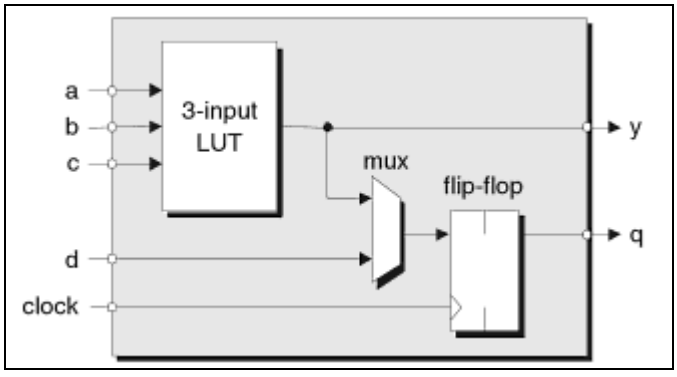
Donanımsal olarak dilim tanımlamaları her donanım mimarisinde değiştiğinden, uygulamalar üzerinde karşılaştırmalar kısıtlı yapılabilmektedir. Farklı FPGA donanım mimarilerindeki dilim yapıları ve eşdeğerleri Çizelge 4.1'de görülmektedir (CORE, 2011).

Çizelge 4.1 - FPGA mimarilerinde dilim eşdeğerleri (CORE, 2011).

Logic Bloğu	Virtex-4 Slice Eşdeğeri
Xilinx Virtex-4 slice (reference)	1
Xilinx Virtex-5 slice	2
Altera ALM	1,3
Actel VersaTile	0,25

Sentezleme işlemi sonucunda elde edilen donanımsal bilgiler FPGA üzerine yüklenerek çalıştırılabilir. Bu işlemden önce donanımsal olarak giriş, çıkış tanımlamaları yapılarak tasarımın ihtiyaç duyduğu FPGA mimarisi belirlenmelidir. Ayrıca FPGA üzerinde Verilog kodunun sentezlenmesinden sonra oluşan lojik devrenin kapladığı alan bilgileri dilim sayıları ile birlikte Xilinx ISE yazılımı tarafından gösterilmektedir.

Literatürde yapılan çalışmalarda kullanılan Virtex II FPGA modelinde lojik hücreler, Şekil 4.6'te görüldüğü gibi LUT (başvuru çizelgesi), kapan (flip-flop) devresi ve diğer lojik hücrelerle olan bağlantılardan oluşmuştur. Önemli donanımsal bir yapı olan LUT ise "*ve*", "*veya*", "*toplam*" gibi dört girişlik mantıksal yapıları oluşturabilmek için, birleşimsel lojik (combinational logic) yapısını kullanmaktadır. Lojik hücrelerden iki tanesi birleşerek Şekil 4.4'te görülen lojik dilim (slice) yapısını oluşturur (NI, 2009).

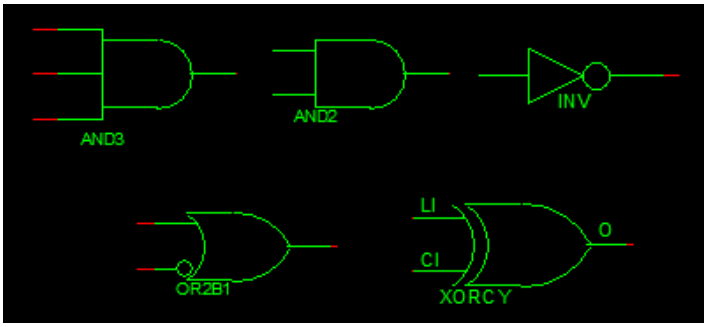


Şekil 4.6 - Basitleştirilmiş lojik blok yapısı (Maxfield, 2004).

4.1.2. Kapı seviyesinde programlama

Kapı seviyesinde programlama, temel seviyede tasarım elementlerini içeren Verilog programlama türüdür. Temel lojik kapıların kullanımını esas almaktadır. Kapı seviyesinde programlamada amaç, kapı yapılarını kullanılarak hazır modüllerin oluşturulması ve bu modüller ile istenilen devre dizaynının yapılmasıdır. Bu tanımlanan bloklar, Verilog içerisinde tanımlı olabilir veya kullanıcı tarafından da tanımlanabilmektedir.

Hali hazırda gelen temel kapılar, Şekil 4.7'deki gibi sırasıyla "*ve"* (and3,and2) kapısı, "*değil"* (inv) kapısı, "*veya"* (or) kapısı, "*xor"* kapısı gibi temel lojik devre elemanlarını kapsar. Kapı isminden sonra gelen numaralar girdi sayısını temsil etmektedir.



Şekil 4.7 - Temel kapı tanımlamaları.

Kapı seviyesinde programlama yapılırken, genellikle başlangıçta kullanılacak lojik mimarinin tasarımı yapılır. Sonrasında ise yazılması gereken Verilog kodu bu mimari kullanılarak oluşturulur.

4.1.3. Yazmaç ve davranışsal seviyede programlama

RTL ve davranışsal seviye programlama, kapı seviyesinde programlamadan daha üst seviye bir yapı kullanmaktadır. Dijital devre tasarımcıları, soyut seviye tanımlamalarını, tasarımların karmaşıklığına göre sınıflandırmışlardır. Tasarım karmaşıklığı arttıkça tasarımcı tasarımını daha yüksek seviyede incelemek ve oluşturmak zorundadır. Transistörlerin birleşimiyle oluşturulan kapı ve diğer bileşenler, transistör seviyesinde tasarım olarak adlandırılır. Birleşimsel ve ardışık devreler ise temel lojik kapılarını kullanmaktadırlar. Bu lojik kapıların kullanıldığı tasarım seviyesine lojik seviye ya da kapı seviyesinde tasarım denilmektedir. Eğer tasarım prosedürü yazmaçlar arasında veri transferini gerektiriyorsa ya da toplayıcılardan

yazmaçlara veri yollarını kullanarak iletişim sağlanması gerekiyorsa bu tasarıma yazmaç transfer seviyesi dizaynı ya da kısaca RTL dizaynı denilmektedir. RTL modellemede, yazmaç seviyesinde lojik yapıların tasarlandığı düşünülürse, davranışsal seviye modellemede de lojik yapıların davranışının tanımlandığı bir katman olarak düşünülebilir (Vahid, 2010) (Tzartzanis, 1998).

Davranışsal modellemede kodlar genellikle "*initial*" ve "*always*" blokları içerisinde yazılır. Başlangıç anlamına gelen "*initial*" bloğu yürütüm zamanının başında bir kere çalıştırılırken "*always*" bloğu ise içerisinde belirlenen koşullara göre komutlar sürekli olarak çalıştırılır. Şekil 4.8'deki gibi "*always*" bloğu içerisinde koşul belirtilmemişse verilen zaman sabiti süresinde girilen kod satırı tekrarlanır.

```
//Clock Signal Generation Cycle Time #1 - Saat Sinyal Üretimi
initial
begin
   clock=1'b0; //set clk value to 1
end

always
   #1 clock= ~clock; //Toggle clock signal every 1 time units- Saat değişimi
```

Şekil 4.8 - Verilogda initial ve always blokları

Bloklar içerisinde yazmaç ve ağ tipinde tanımlanan değişkenlerin ataması yapılabilmektedir. Şekil 4.9'da ise yordamsal olarak davranışsal modellemede belirtilen yazmaç ve ağ yapıları atamalarının yapılışı görülmektedir.

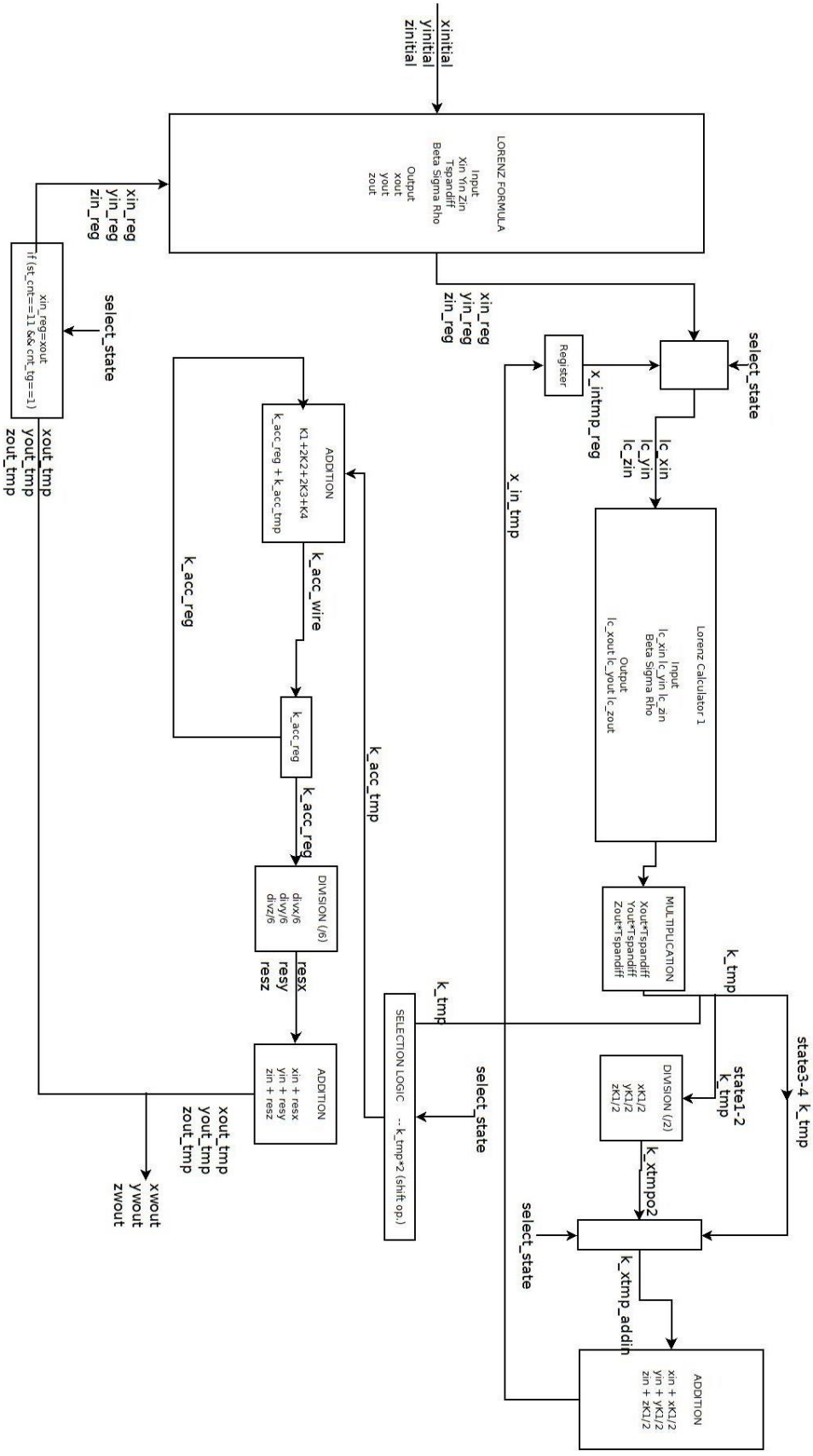
```
//Control input data- Giriş Verileri Kontrolü
initial
begin
   enable=1'b1;
   x=xin;y=yin;z=zin;

   //8Q12 -- 8-12 bit:20bit
   tspan=21'b0_0000_0000_0000_0010_1001;//0.01
   rho=  21'b0_0001_1100_0000_0000_0000;//r=28
   b=    21'b0_0000_0010_1000_0000_0000;//b=2.5
   sig=  21'b0_0000_1010_0000_0000_0000;//sig=10


   #200 $finish;//Terminate at time unit 200 - 200de çıkış
   end

always@(posedge clock)
begin
    x<=xout;y<=yout;z<=zout;
end
```

Şekil 4.9 - Verilogda yordamsal atama işlemleri

4.2. Verilog Üzerinde Lorenz Sisteminin Modellenmesi

Lorenz kaotik sisteminin donanımsal olarak modellenmesi, RK4 çözümü, Lorenz denklemleri ve bu hesaplamalar için gerekli olan aritmetik işlemlerin tanımlanmasıyla yapılmaktadır.

RK4 ile çözümlenecek olan Lorenz sistemin için gerekli olan Eşitlik 2.7'de verilen *k1*, *k2*, *k3* ve *k4* değerlerinin ana modülde hesaplanması gerekmektedir. Bu işlemleri yapabilmek için sonlu durum makinesi (FSM) tanımlanmıştır. Bu sayede oluşturulan modüller kullanılarak *k* değerleri hesaplanır. Hesaplanan *k* değerleri bağlantılar aracılığı ile yazmaçlara aktarılır. Yazmaçlarda tutulan bu değerler, ihtiyaç anında gerekli modüllere aktarılır. En son olarak hesaplanan bütün *k* değerleri akümülatör yardımıyla toplanarak bölme modülüne gönderilir. Şekil 4.10'da görülen blok diyagram ana modülde tanımlanan sistemin genel işleyişini göstermektedir.

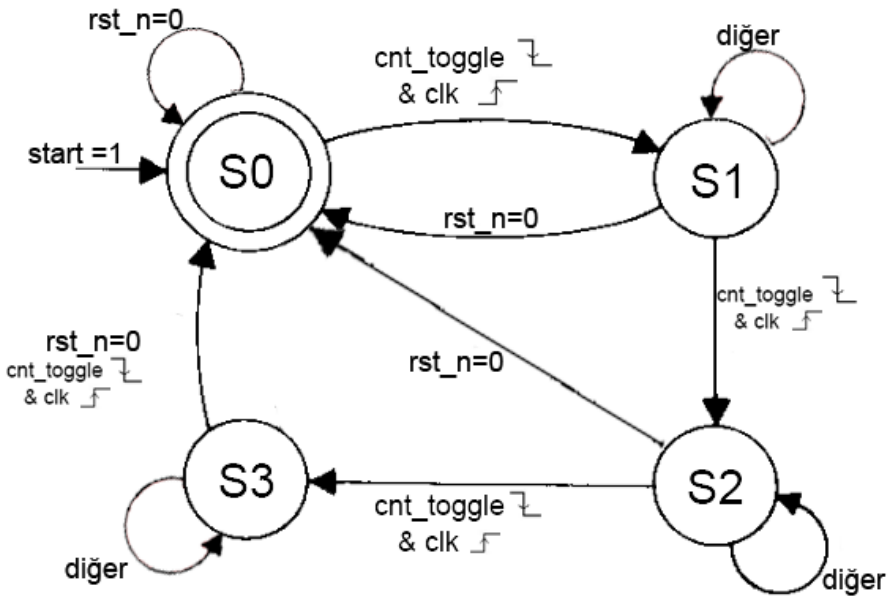


Şekil 4.10 - Verilog Lorenz sistemi blok diyagramı.

Şekil 4.10'da verilen blok diyagramında görüldüğü gibi sistem içerisindeki sonlu durum makinesi gerekli sinyalleri;

- Lorenz denklemlerini hesaplayan modül öncesinde,
- Toplama ve bölme işlemleri öncesinde,
- Akümülatöre giden *k_acc_tmp* sinyali öncesinde,
- Bir sonraki dürüm (iterasyon) koşulu sağlandığında *st_cnt* sinyaline göre atayarak sistemin çalışmasını sağlamaktadır.

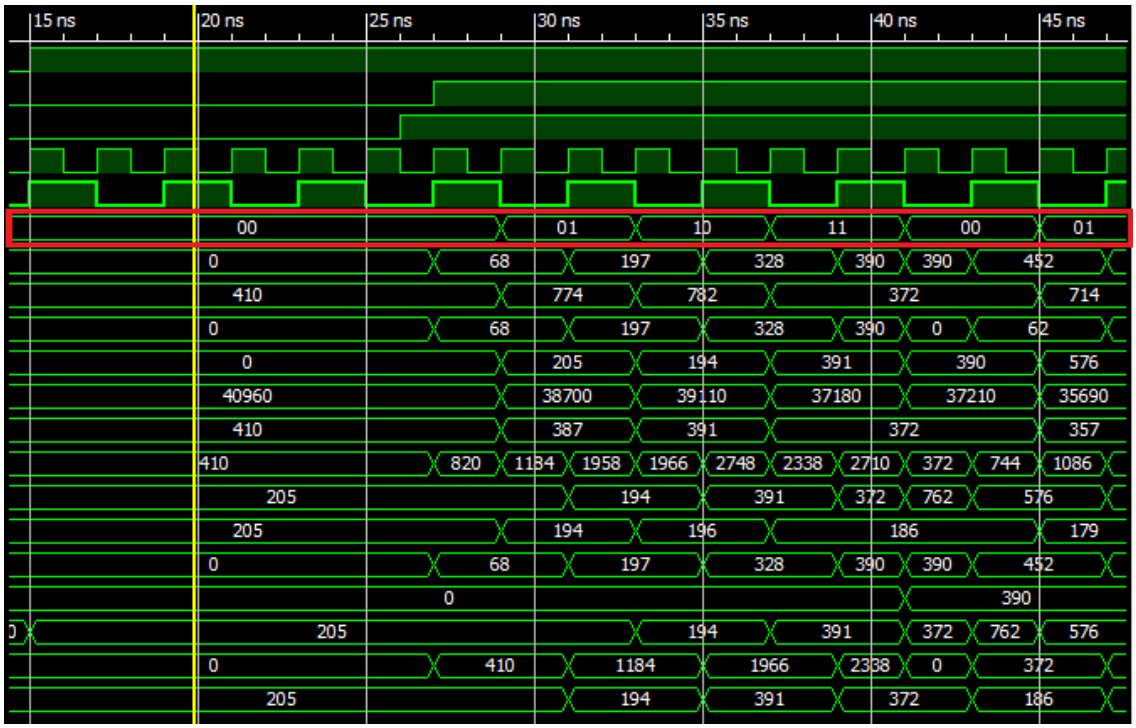
Şekil 4.11 sonlu durum makinesini ve durumlarını etkileyen sinyalleri göstermektedir.



Şekil 4.11 - Sonlu durum makinesi ve gelen sinyallere göre davranışı.

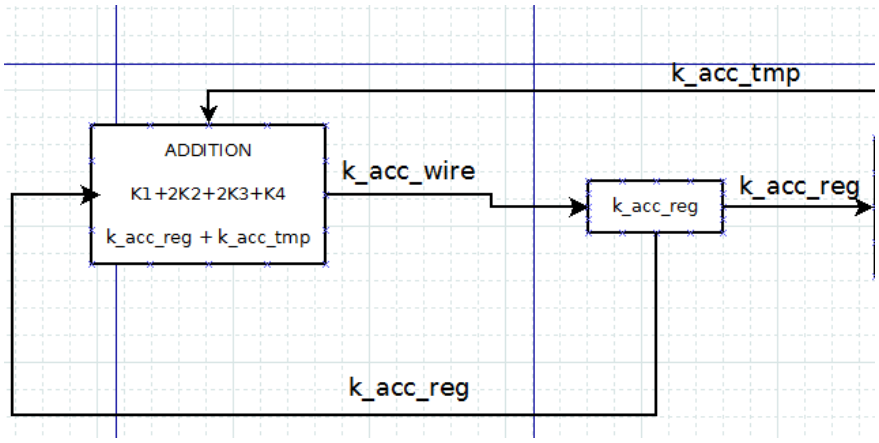
Tasarlanan yapıya göre belirlenen Şekil 4.10'daki sistem içerisinde; saat sinyali olan *clk*, başlangıç sinyali olan *start* ve sıfırlama sinyali olan *rst_n*'e göre sistemde durum sinyali olan *st_cnt* değeri değişmektedir. Bu değişim Şekil 4.12'de görülmektedir. Kırmızı ile belirtilen alan, durum sinyalinin aldığı değerleri vermektedir. En üstte görülen sinyal sıfırlama sinyali, kırmızı ile işaretlenmiş bölgenin üzerindeki iki sinyal ise saat ve durum sinyallerini kontrol eden *cnt_toggle*'ı göstermektedir. Saat

sinyali ile sıfırlama sinyali arasında kalan iki sinyalden biri sistemin başlangıç sinyali diğeri ise bunu kontrol eden sinyaldir.



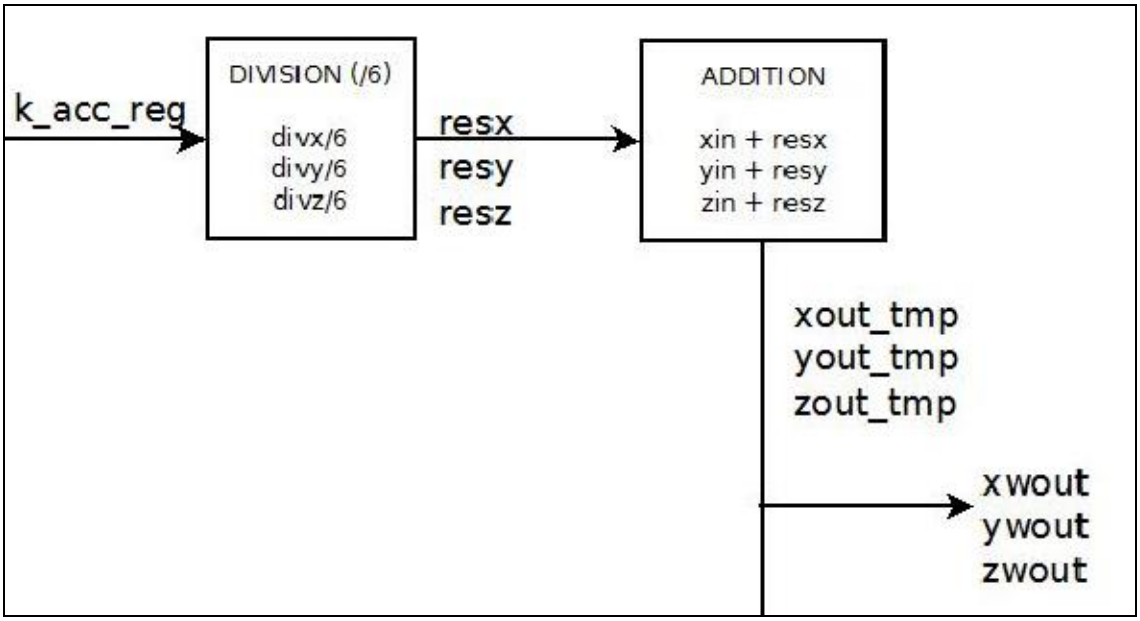
Şekil 4.12 - Sonlu durum sinyali ve kontrolü.

Şekil 4.13'deki yapı ise Eşitlik 2.9'da görülen RK4 çözümünde yer alan delta içerisinde *k* değerleri toplamını ifade eden akümülatör yapısıdır. Yazmaçta tutulan *k_acc_tmp* sinyali yardımıyla her farklı duruma geçiş esnasında Lorenz modülü hesaplanarak oluşturulan *k* değerleri akümülatörde saklanmaktadır. Dört durum sinyali değişimi sonucunda elde edilen toplam değerler, *k_acc_reg* yazmacında toplanarak bölüm modülüne aktarılmaktadır. Bu sayede Eşitlik 2.9'da hesaplanması gereken ortalama ağırlık Verilog dilinde tanımlanmıştır.



Şekil 4.13 - Tasarlanan sistemdeki akümülatör yapısı.

Lorenz sisteminde yapılması gereken bütün işlemler yapıldıktan sonra Şekil 4.14'de tasarımı görülen yapıda *st_cnt* sinyali 3 değerini aldığı zaman, akümülatördeki sinyal bölüm modülüne girerek Eşitlik 2.5'teki hesaplanan y_{n+1} değerinin değişim kısmını hesaplar. Sonrasında ise y_n değeri ile hesaplanan değer toplanarak y_{n+1} çıkış değerleri elde edilmiş olur. Bu değerler *xwout*, *ywout*, *zwout* sinyalleriyle dışarı verilirken yine aynı sinyal değerleri bir sonraki hesaplama için ana modülde sisteme giriş olarak verilmektedir.



Şekil 4.14 - Lorenz sisteminin Verilog üzerinde çıkış değerlerinin elde edilmesi.

4.2.1. Verilog modülleri

Verilog modülleri, sistem işleyişini gerçekleştirmek için oluşturulan parçalar olarak görülebilir. Lorenz sisteminin modellenmesi için yazılan modüller altı farklı kategoride yer almaktadır. Bunlar test modülü olan stimulus modülü, bütün sistemin tasarımını ve işleyişini barındıran ana modül, Lorenz denklemlerinin hesaplamalarını yapan lorenz modülü, çarpım işlemlerini gerçekleyen ve kontol eden çarpım modülü, bölme işlemini yapan bölme modülü ve girilen sinyale göre toplama ya da çıkarma işlemini ikili tümleyen sisteme göre gerçekleştiren toplama-çıkarma modüllerinden oluşmaktadır.

4.2.1.1. Lorenz Stimulus modülü

Stimulus modülü, Verliog dili ile yazılmış olan Lorenz sisteminin benzetimi için oluşturulmuş bir modüldür. Bu modül, belirlenen parametre ve giriş değerlerine göre sistemin davranışını ISIM veya ModelSIM yazılımlarında test edebilmek için hazırlanmıştır. Ayrıca stimulus modülü yardımı ile Lorenz sisteminden elde edilen çıktılar dosyaya yazdırılmıştır. Bu dosyadaki veriler, Matlab yazılımında oluşturulan Lorenz sisteminin benzetim sonuçları ile karşılaştırılmıştır. Bu sayede sistemlerin doğruluğu ve işleyişi kontrol edilmiştir.

```
//Monitoring & Recording Waveform
//İzleme ve Dalga Biçimi Kaydı
initial  begin
   $dumpfile ("LorenzrhsWaveForm1.vcd");
   $dumpvars;
end
```

```
//File Save - Dosya Kayıt
integer file, filesaveok;

initial
begin
   file = $fopen("VerliogLorenz_1.txt", "w");
   #4800 $fclose(file);
end
```

```
//Recording Values to File-Değerlerin Dosyaya Kaydı
$fwrite(file,"%d \t %d \t %d\n",x,y,z);
```

Şekil 4.15 - Stimulus modülünde sinyal ve dosya kayıt işlemi.

Şekil 4.15'de görüldüğü gibi stimulus içinde tanımlanan kodlar kullanılarak sinyal verilerinin saklandığı "LorenzrhsWaveForm1.vcd" ve Lorenz sisteminin verdiği çıktıların kaydedildiği "VerilogLorenz_1.txt" dosyası oluşturulmaktadır.

```
initial
begin
   clock=1'b0;
end

always
   #1 clock= ~clock;


initial
begin
   rst_n= 1'b1;
   start=1'b0;
   #5 rst_n= ~rst_n;
   #10 rst_n= ~rst_n;
   #11 start= 1'b1;
end
```

Şekil 4.16 – Saat (clock) ve sıfırlama (rst_n) sinyallerinin tanımlanması.

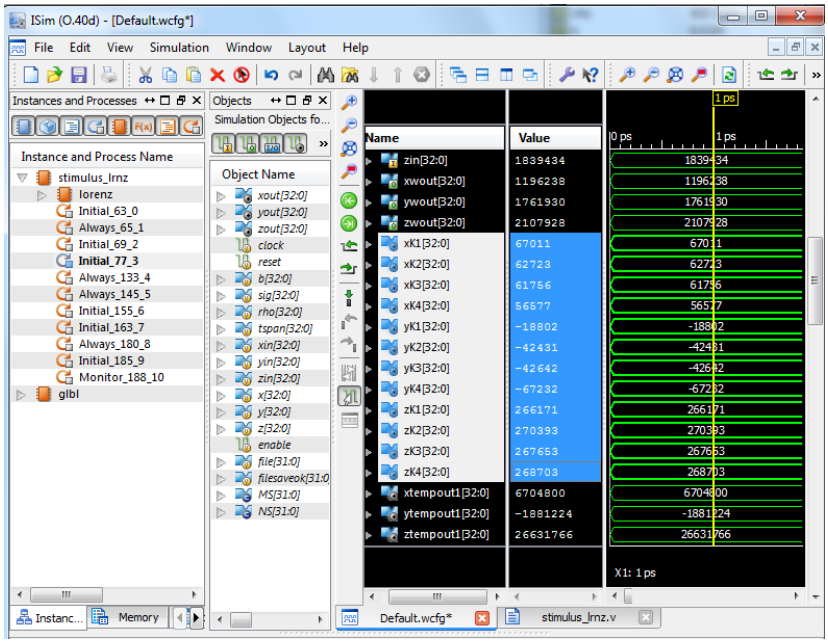
Şekil 4.16'da stimulus içerisinde, saat (clock) süresi ile sistemin sıfırlama sinyali olan *rst_n* tanımlanmıştır. Başlangıçta 0 değerini alan sıfırlama sinyali 5 zaman birimi sonrasında 1 değerini almaktadır. 10 birim zaman sonra yani başlangıç anından toplam 15 birim zaman geçtiğinde sinyal tekrar 0 olur. 26. birim zamanda ise sıfırlama sinyali 1 değerini alarak benzetim sonuna kadar bu şekilde devam etmektedir. Birim zaman sabitinin büyüklüğü ise modül başında yer alan zaman aralığı "*`timescale 1ns/1ps*" kodu ile belirlenir.

Sistem parametrelerinin giriş değerleri olan ve Eşitlik 2.3'te belirtilen rho, *sigma*, *b* ve *tspan* gibi değerler stimulus tarafından atanmaktadır. Şekil 4.17'de görülen stimulus modülünde, belirli zaman aralıklarında istenilen değerler hem dosyaya hem de benzetimde ekrana yazdırılmaktadır.

```
always@(posedge clock)
begin
if(start)
begin
   if(clockfirst==0 && clk_cntr==2)
      clockfirst=1;
   else if(startfirst==0)
      clk_cntr=8;
   else
      clk_cntr=clk_cntr+1;
   startfirst=1;
   if(clk_cntr==8)
   begin
      clk_cntr=0;
      $fwrite(file,"%d \t %d \t %d\n",x,y,z);
      //Monitor
      $display("\t\ttime,\ttspan values,\tclock");
      $monitor($time,
      "clock = %d,\tx=%d -\ty=%d-\tz=%d\t_____\txin=%d
      clock,x,y,z,xin,yin,zin,rho,sig,b,tspan);
   end
end
end
```

Şekil 4.17 - Parametrelerin konsola ve dosyaya yazımı ve koşulları.

Şekil 4.18'de stimulus modülü kullanılarak oluşturulan Lorenz kaotik sisteminin çıktıları, ISIM yazılımında görülmektedir. Buna göre sistem sinyalleri ile donanımsal tepkiler takip edilebilmektedir.

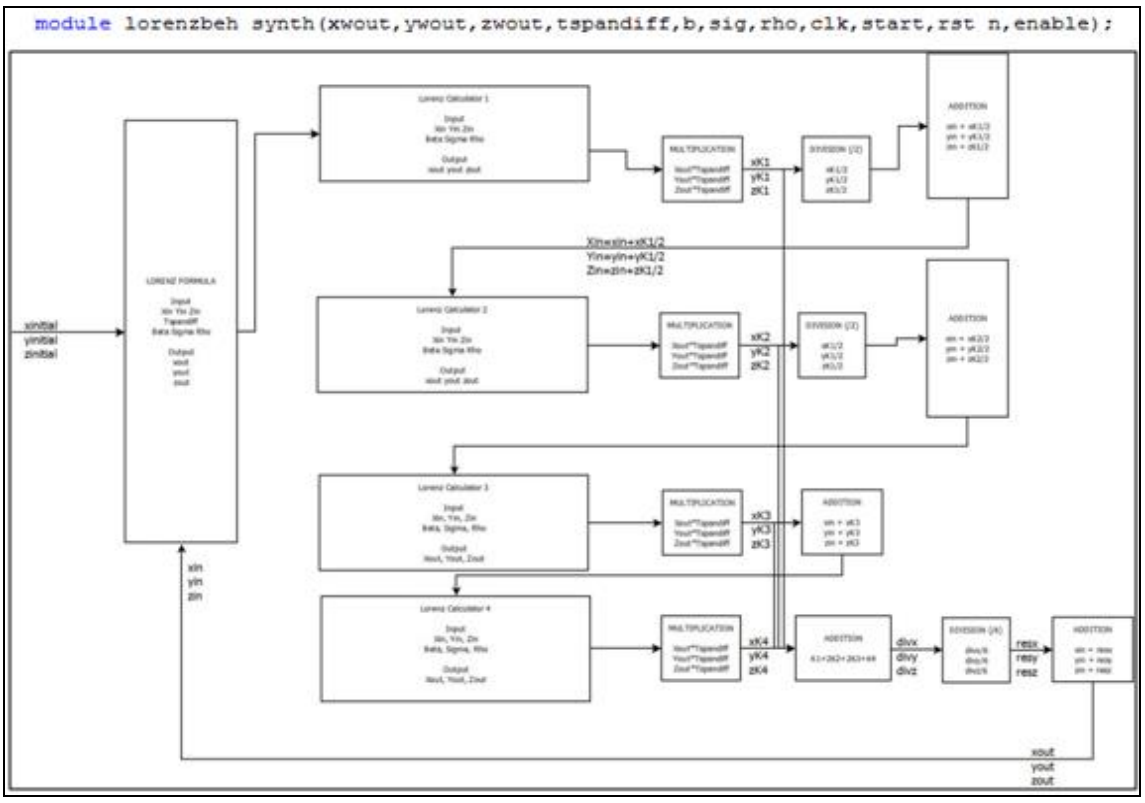


Şekil 4.18 - Xilinx ISE içerisinde yer alan ISIM benzetim yazılımı.

4.2.1.2. Ana modül

Lorenz modülü bütün işlemlerin yapıldığı ana modül olarak tanımlanabilir. Bu modül içerisinde birçok modül bulunmaktadır. RK4 için gereken hesaplama modülleri, Lorenz denklemlerinin bulunduğu *lorenz_calculator* modülü yine bu modül içerisinde yer almaktadır. Şekil 4.10'te belirtilen blok diyagramda görülen sistem bu ana modülün grafiksel bir gösterimidir.

Ana modülün parametrelerini inceleyecek olursak, modül *xwout, ywout, zwout* isimli üç farklı boyut için üç farklı çıktı tanımlanmıştır. *tspandiff* parametresi RK4 çözümü için gerekli olan *h* parametresini ifade etmektedir. *b, sig, rho* parametreleri Lorenz denkleminde var olan beta, sigma ve rho girdilerini belirtmekle birlikte *b*=2,5, *sig*=10, *r*=28 olarak alınmıştır. *h* parametresine ise 0,01 değeri atanmıştır. Ana modül olan *lorenzbeh_synth* modülündeki diğer parametrelerden; saat sinyalini *clk*, sistemin başlama sinyalini *start*, negatif kenar reset sinyalini *rst_n* belirtmektedir. *enable* sinyali sistemin çalışması için gerekli olan sinyali sağlamak üzere tanımlanmıştır.



Şekil 4.19 - Lorenz ana modülü tanımlaması ve işlem diyagramı.

Ana modülde yapılan işlemleri Şekil 4.10 üzerinde incelemek gerekirse, her modülün 4 kere çağırılması gerekmektedir. Sistem tasarımının ilk safhalarında Şekil 4.19'da görüldüğü gibi her bir modülden dörder adet oluşturularak RK4 çözümüne ulaşılması düşünülmüştür. Fakat oluşturulan alanın azaltılması istendiği için sistem Şekil 4.10'daki duruma uygun olarak modellenerek sonlu durum makinesi kullanılmıştır.

4.2.1.3. Lorenz hesaplama modülü

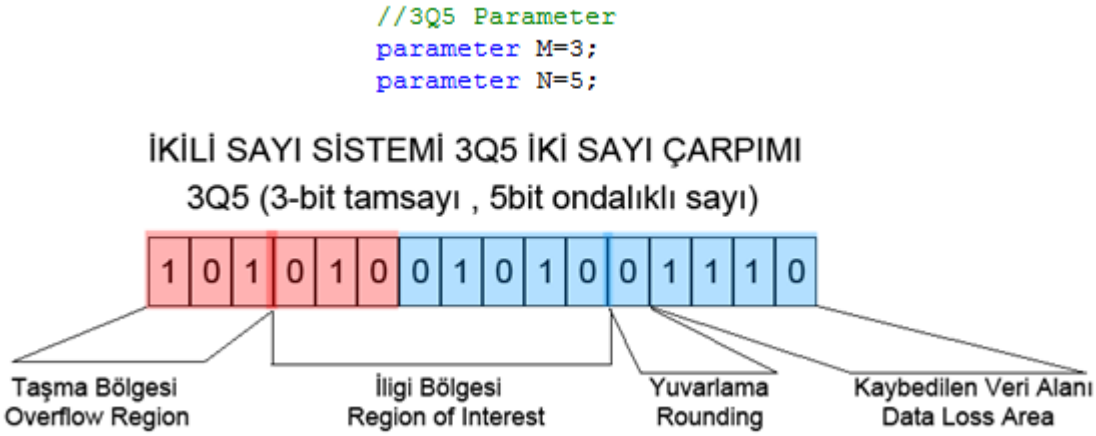
Lorenz hesaplama modülü, bütün Lorenz denklemlerinin hesaplandığı Verilog modülüdür. Bu modülde Bölüm 2.3'te belirtilen Lorenz kaotik sisteminin denklemleri tanımlanmıştır. Ana modülde belirlenen "M" ve "N" parametreleri, sistemin kaç bitlik sayı gösterimiyle çalışacağını belirtmektedir. Lorenz modülü içerisinde verilen girdiler ise bu parametrik değerlere göre hesaplanmaktadır. Ayrıca bu modül içerisinde farklı kaotik denklemler benzer şekilde modül haline getirilerek çözüm sağlanabilir.

Eşitlik 2.3'te yer alan denklemlerin çözümlemesi için modülün gelen saat sinyaline göre bir kez çalıştırılması yeterlidir. Girdi olarak verilen zaman aralığında hesaplanan her değer, bir sonraki Lorenz modülünde kullanılmak ve akümülatöre gönderilmek için yazmaçlara aktarılır.

4.2.1.4. Çarpma modülü

Çarpma modülü, sistemin veri kaybı ve hata oranı açısından önemli bir modülüdür. RTL seviyesinde modellemede çarpma işlemi için lojik olarak ek bir işlem yapılmasına gerek yoktur. Çarpma işlemi için belirtilen parametre kadar yer ayrılması, işlemin yapılması için yeterlidir. Fakat çarpım sonucunda elde edilen sayılar dikkatli incelenmelidir. Çarpma modülüne gönderilen iki aynı formatta sayı çarpıldığında sonuç olarak Şekil 4.20'de görüldüğü gibi parametrik formatın iki katı büyüklüğünde bir veri elde edilir. Eğer Şekil 4.20'de görülen 3Q5 formatında olan sayı sisteminde veri işlemek için 3-bit tamsayı kısmı ve 5-bit ondalıklı kısım kullanılacak ise parametre değişkenleri olan M ve N sırasıyla 3 ve 5 değerlerini alır. Elde edilen sonuç ise M ve N parametreleri toplamının iki katı olacaktır. Elde edilen sayının ilk 3-bitlik kısmı, sonucun sayı sistemi içerisinde belirtilip belirtilemeyeceğini belirler. Eğer taşma bölgesinde sayı var ise yani sıfırdan farklı bir değerde ise sayı doyuma ulaşır ve 3Q5'lik sistemde alınabilecek en büyük veya en küçük değeri alır. Eğer bu 3-bitlik kısımdaki değer sıfır ise, elde edilen

sayı 3Q5 formatında tanımlanabilir olduğundan ilgi bölgesi dışındaki yuvarlama işleminin yapılacağı *N*.bite bakılır. *N*.bit değerine göre yuvarlama işlemi uygulanır. En az anlamlı bit bölgesi olan 0 ile *N-1* bitlik bölge ise dikkate alınmaz ve atılır. Çarpma işlemi sonucunda elde edilen sayı, ilgi bölgesine yuvarlama işlemi uygulanarak bulunur.



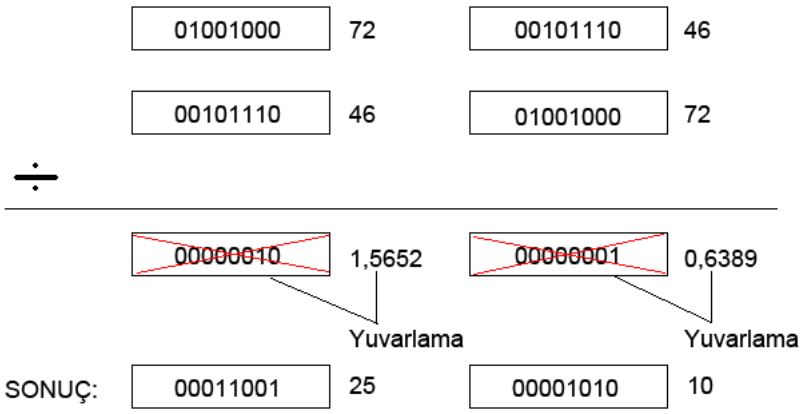
Şekil 4.20 - Çarpım modülünde elde edilen çarpım sonucu.

4.2.1.5. Bölme modülü

Bölme modülü, oluşturulan sistemdeki bütün bölme işlemlerinin ele alındığı ve istenilen formatta bölüm sonuçlarının hesaplandığı modüldür. Bölme modülünün farklı versiyonlarına optimizasyon bakımından ihtiyaç duyulmuştur. Ayrıca sentezlemede sayıların iki ve katlarına bölünebilmesi, sentezleme aşamasında sorun olmazken değişken sayılarla bölme işleminin sentezlenmesi için farklı bir modüle ihtiyaç duyulmuştur. Sabit sayılarda RTL seviyesi modellemede bölme modülüne gerek kalmadan bölme işareti ile bölme işlemi yapılır. Fakat 2,3,5 böleni gibi sabit sayılar dışında bölme işlemi için aynı yöntemle sistem sentezlenememektedir. Bu sebeple oluşturulan birleşimsel bölücü modülünde; sayı genişletilmesi, çıkarma işlemleri ve bit kaydırma işlemleri kullanılarak döngüler içerisinde bölme işlemi yapılabilmektedir.

Bölme modülünde bölüm işlemi sonucu elde edilen sayı, belirlenen sayı sistemine göre daha küçük bir sayı formatına dönüştüğü için ondalıklı kısmı temsil eden *N* ile genişletilmelidir. *N*-bit sola kaydırma işlemi yapıldığında istenilen sonuca ulaşılmaktadır. Daha iyi anlaşılması için Şekil 4.21'deki örnek incelenebilir. 4Q4'lük bir sayı sisteminde ilk temsil edilen 72 sayısı kayan noktalı sayı sisteminde 4,5'luk bir değeri ifade etmektedir. 46 sayısı ise kayan noktalı sayı sisteminde 2,8750 sayısını

göstermektedir. Kayan noktalı sayı sisteminde sonuç 1,5652 olmakta iken bu sayı tezde önerilen yöntem kullanıldığında 25,0435 olmaktadır. Virgülden sonraki sayı kısmı bu sistemde gösterilemediğinden dolayı sonuç 25 olur. Sistemin hatası ise 1,5652 – 1,5625 =0,0027 olmaktadır. Şekil 4.21'de elde edilen sonuç, herhangi bir ön işlem yapılmadan hesaplanmış olsaydı istenilen sonuç bulunamazdı. Bu sebeple bölünen sayı *N*-bit sola kaydırılıp yani $72x2^4$ şeklinde genişletilip 46 sayısına bölündüğünde istenilen sonuca ulaşılmaktadır. Bunun sebebi ise sayı sisteminde belirtilen bütün sayıların 2^N kadar çarpılarak sola kaydırma işleminin yapılmasıdır. Böleceğimiz sayıyı 2^N ile genişletmek, yapılan işlemin sonucunun istenilen *M* ve *N* parametrelerinde yani *MQN* sayı formatında kalmasını sağlar. Sonuçlar, onluk sayı sisteminde yanlış gibi görülmesine rağmen bu sayılar $1/2^N$ ile çarpılarak yuvarlandığında eğer gösterim sisteminde taşma yok ise sistemin doğru çalıştığı test edilebilir. Kayan noktalı sayı gösterim sistemi sonuçlarında yapılan kontrol ve işlemler ile sabit noktalı sayı gösterim sonuçları elde edilir. Bu sonuçlar ile Verilog'dan alınan sonuçlar karşılaştırılır.



Şekil 4.21 - Bölme modülünde işlem mantığı.

Bölme işleminde dikkat edilmesi gereken bir diğer nokta ise sayının *N* bit sola kaydırılmasına karşın bölünen sayının bölen sayı değerinden küçük olması durumudur. Bu durumda elde edilen sayı 0 ile 1 arasında olacağından sayıyı ikili sayı sisteminde donanımsal olarak göstermek imkânsızdır. Çözüm olarak sistemde yuvarlama işlemi sonrası elde edilen sayı, pozitif ise 0 ya da 1 değerini, negatif ise -1 değerini sonuç olarak çıkışa vermelidir. Tez içerisinde oluşturulan bölme modülü, bütün bu koşulları sağlayan bir yapıda tasarlanmıştır. Fakat kontrol edilmesi gereken durumların çokluğu bölme modülünün karmaşıklığını artırmaktadır.

4.2.1.6. Toplama ve çıkarma modülü

Toplama ve çıkarma modülü, tek modül içerisinde toplama ve çıkarma işlemlerinin ele alındığı bir modüldür. 2'li tümleyen sistemde toplama ve çıkarma işlemleri için farklı modüller oluşturmasına gerek duyulmamaktadır. Sınır değerlerini aşma, doyuma ulaşma ve yuvarlama kontrolleri, belirlenen sayı gösteriminde toplama ve çıkarma işlemlerine uygun olarak tanımlanmıştır. Toplama ve çıkarma işlemlerinden hangisinin yapılacağı, bir bitlik giriş sinyali ile belirlenmektedir.

4.2.2. Sistemin parametrik olarak ele alınması

Sistemdeki parametre ifadesi, sabit noktalı sayı gösteriminde sayıların kaç bitlik donanımsal alan ile tanımlanacağını gösterir. Verilog dilinde oluşturulacak donanımsal yapı parametrik olarak tanımlanabilir. Bu özellik Şekil 4.22'de görüldüğü gibi "*parameter*" komutu kullanılarak uygulanır.

```
//Parameter for stimulus (MS.NS)
parameter MS=8;
parameter NS=12;
```

Şekil 4.22 - Sayı gösterimi için belirlenen parametreler.

Parametreler yardımıyla sistemdeki yazmaçların ve ağ tanımlamalarının kaç bitlik olacağı Şekil 4.23'te görüldüğü gibi tanımlanır.

```
reg signed [(MS+NS):0]b,sig;
reg signed [(MS+NS):0]rho;
reg signed [(MS+NS):0]tspan;
reg signed [(MS+NS):0]xin,yin,zin; ///Input x values

reg signed[(MS+NS):0] x,y,z;
wire signed[(MS+NS):0]xout,yout,zout;
```

Şekil 4.23 - Parametrik yazmaç ve ağ tanımlamaları.

Ayrıca tanımlanan modül içerisinde kullanılacak farklı alt modüller bulunuyorsa, bunların Verilog tanımlamaları Şekil 4.24'te görüldüğü gibi olmalıdır. Parametre kullanılacak ise modül isminden sonra "#" işareti eklenerek parantez içerisinde parametrelerin değerlerini belirtmek gerekmektedir. Bu sayede belirtilen modül içindeki bütün parametrik tanımlamalar üst modülde belirlendiği şekilde oluşturulacaktır.

```
//Lorenz Top Module
lorenzbeh_synth #(MS,NS) lorenz_synth(xout,yout,zout,tspan,b,sig,rho,clock,start,rst_n,enable);
```

Şekil 4.24 - Parametrik Lorenz yapısının tanımlanması.

5. SİSTEM BENZETİMİ, SENTEZLENMESİ VE ANALİZİ

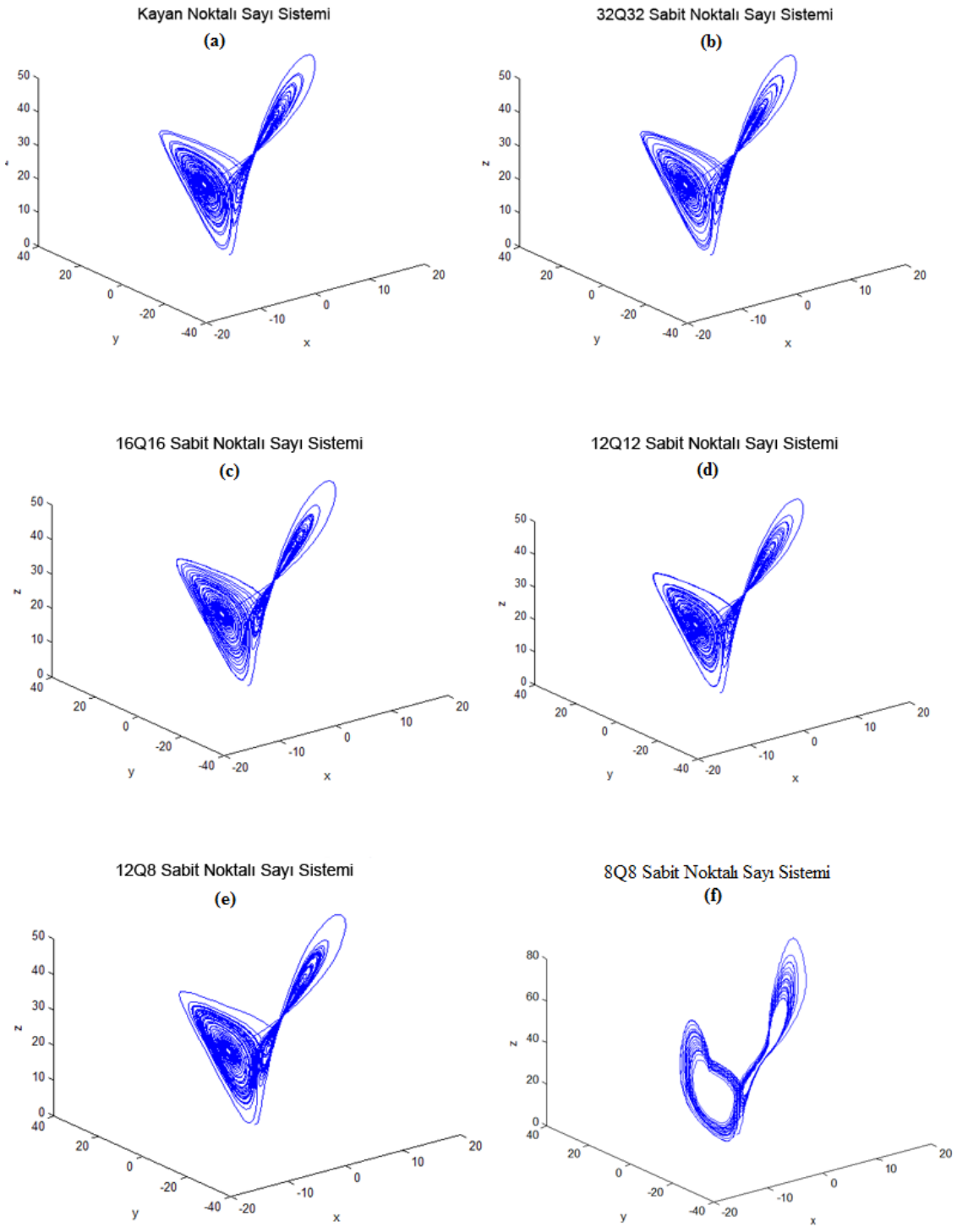
Benzetim, belirlenen koşullar altında sistem tasarımının ve geçerliliğinin test edilmesini sağlar. Sistemin doğru sonuçlar üretip üretmediğinin analizindeki ilk aşama, Verilog ve Matlab dilleri ile oluşturulmuş tasarımların benzetim sonuçlarının karşılaştırılmasıdır. Bu aşamada Xilinx ISE yazılımındaki ISIM ve ModelSIM araçları kullanılarak donanımsal benzetim sonuçları elde edilmektedir. Sistem kontrolü için Matlab üzerinden elde edilen yazılımsal benzetim sonuçları ile donanımsal benzetim sonuçları karşılaştırılır.

FPGA üzerinde sentezleme işlemi, tasarım ve benzetim işlemleri yapıldıktan sonra sistemin donanımsal olarak oluşturulmasıdır. Bu esnada oluşturulan modüller, donanımsal olarak lojik kapılara dönüştürülür. Xilinx ISE yazılımı kullanılarak sentez işlemi yapmak için projenin ayarlar bölümünden oluşturulması planlanan hedef donanım seçilir. Sentez sonrası elde edilen sonuçlar, FPGA üzerinde alan ve hız bakımından değerlendirilir. Alan karşılaştırması için dilim ve LUT yapısı kullanılırken hız değerlendirmesi için minimum saat frekansı kullanılır.

5.1. Lorenz Sisteminin Benzetimi ve Matlab ile Karşılaştırılması

Çalışma kapsamında Verilog'da tanımlanan Lorenz sistemi ile Matlab'da benzetimi yapılan sistemin aynı değerleri üretmesi önemlidir. Kaotik uygulamalarda donanımsal çözüm ile yazılımsal çözüm eşitliği sağlanmazsa bu sistemler hatalı çalışacak ve senkron bozulacaktır. Bu nedenle bölme modülü ile diğer modüllerde ele alınan yuvalama işleminin kullanılması gerekmektedir. Fakat bu karar FPGA üzerinde donanımsal olarak daha fazla alan gereksinimini ortaya çıkarır.

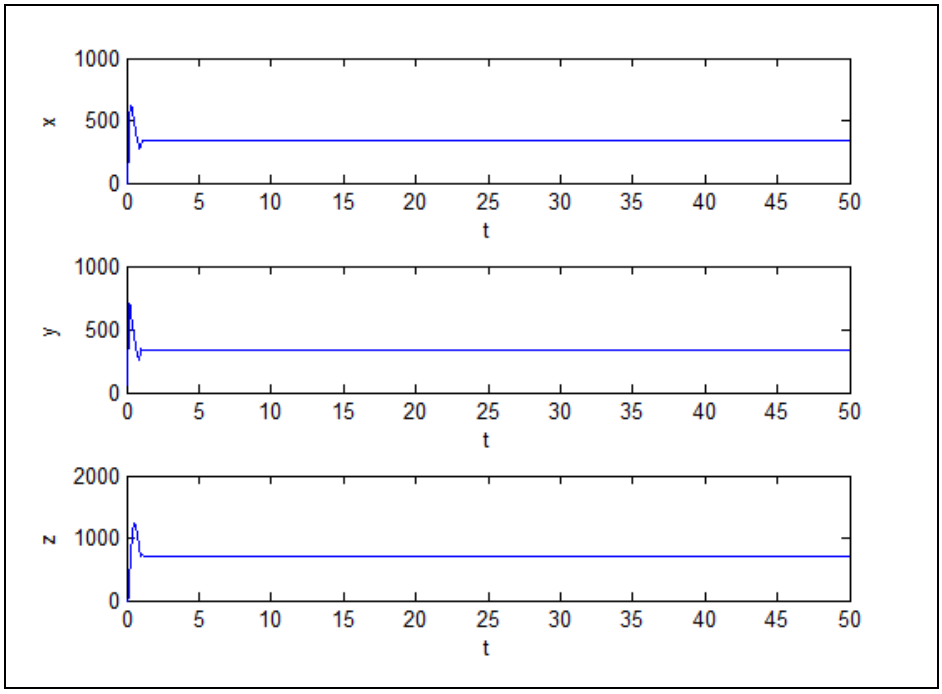
Azzaz ve arkadaşlarının önerdiği şekilde FPGA üzerinde oluşturulacak bir sistem, donanımsal olarak az yer kaplar (Azzaz vd., 2009). Buna rağmen, sistem oluşturulurken yapılan varsayımlardan dolayı elde edilen donanımsal veriler ile MATLAB yazılımındaki benzetim değerleri farklılık gösterecektir. Bu tarz bir yaklaşım, eğer yazılımsal sistem ile donanımsal sistem sonuçlarının eşitliği önemli değilse kullanılabilir.



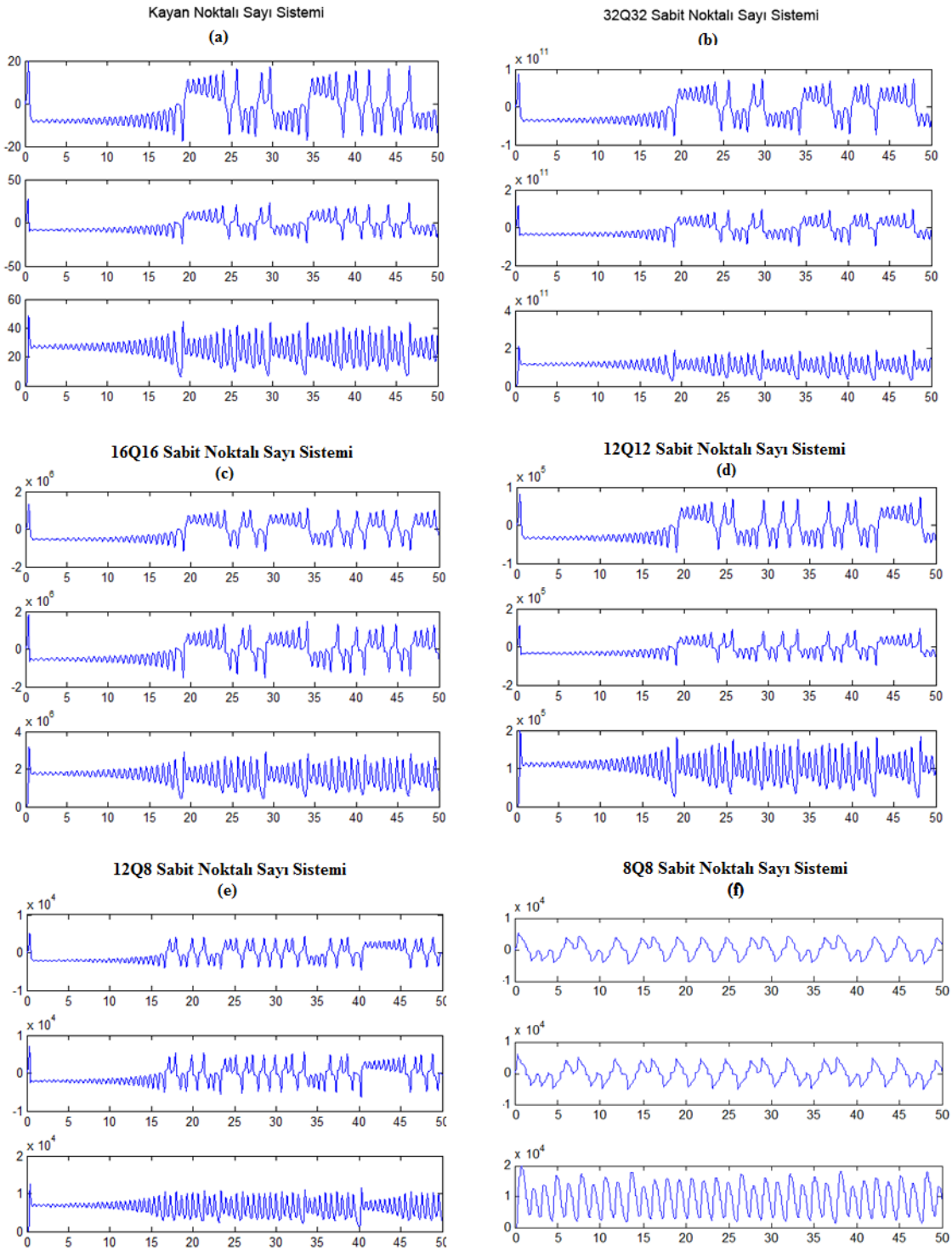
Şekil 5.1 - Kayan ve sabit noktalı sayı sistemlerinde yazılımsal benzetim çözümleri.

Şekil 5.1'de MATLAB yazılımı ile elde edilen benzetim sonuçları yer almaktadır. Bu şekillerde, kayan noktalı sayı gösteriminin ve farklı bit uzunluklarında sabit noktalı sayı gösterimlerinin Lorenz kaotik sistem sonuçlarına etkisi görülmektedir. Grafiklerdeki değişimin sebebi, sistemler arasındaki sayı gösterim kapasitesinin farklı

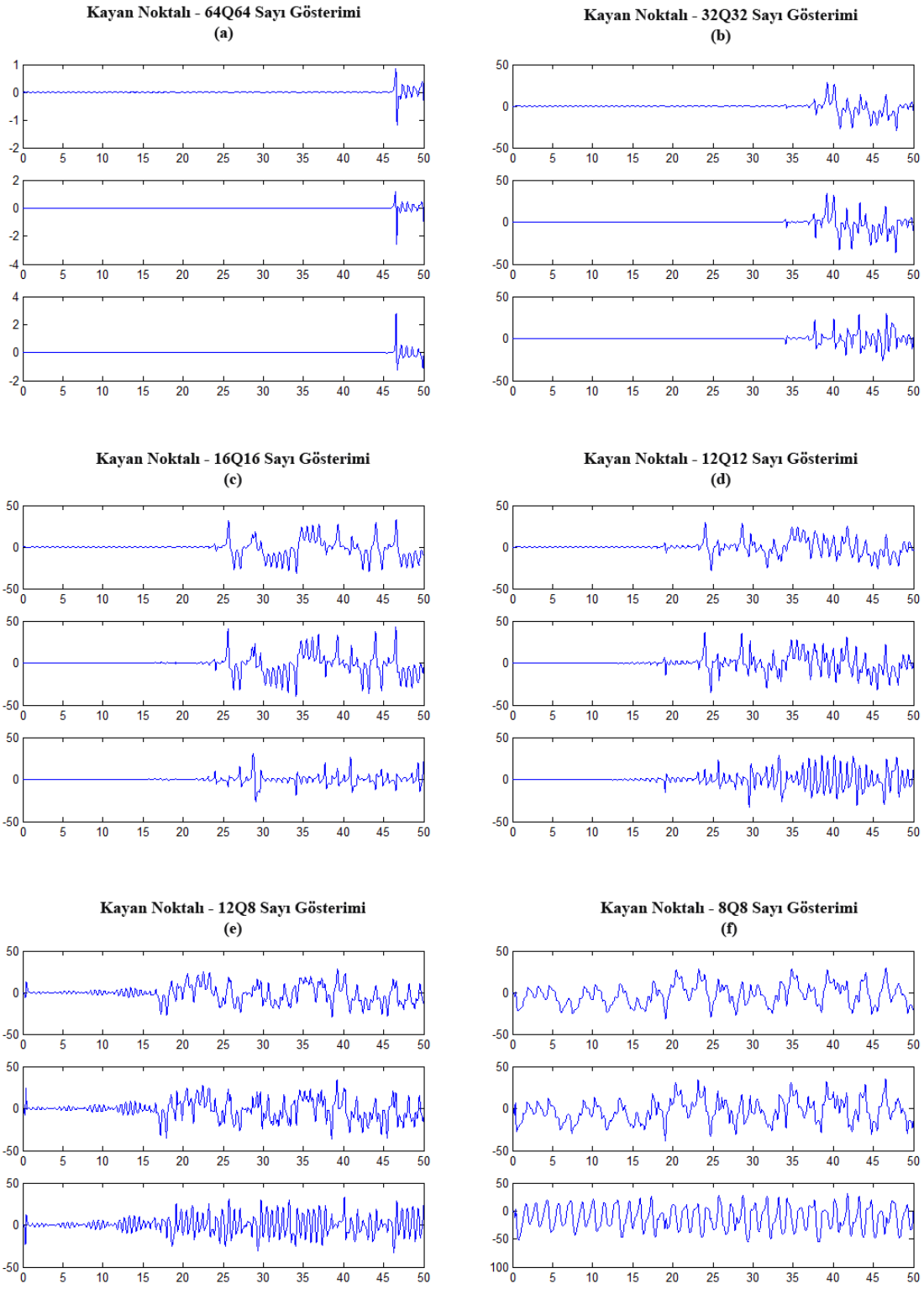
olmasından kaynaklanmaktadır. Düşük bit gösterimlerinde belirtilen sayılar, sürekli olarak negatif ve pozitif alanlarda doyuma ulaştıklarından sistem kaotik özelliğini yitirebilmektedir. Şekil 5.2 ve Şekil 5.3'te görüldüğü üzere sayı ifadesi uzunluğunun azalması, elde edilen sayı değerlerinin değişiminin de azalmasına neden olur. Düşük sayı formatlarında elde edilen değerler, sabit olarak belirli bir aralıkta değer alırlar. Dolayısıyla sistem istenilen sonucu verememektedir. Bu nedenle sistemin parametrik olması çok düşük bit seviyelerinde aynı performansı vereceği anlamına gelmez. Şekil 5.1 ile Şekil 5.3 karşılaştırıldığında, sayısal gösterimin sistemi nasıl etkilediği anlaşılmaktadır. Kayan noktalı sayı sistemi ile sabit noktalı sayı sistemi arasında yüksek bit gösterimlerinde dahi IEEE 754 formatının hassasiyetinden kaynaklı fark bulunmaktadır. Sayı gösteriminde kullanılan bit sayısı azaldıkça fark artarken, yüksek seviye bit gösterimlerinde bu fark azdır. Şekil 5.2'de ise düşük bit ile sayı temsilinden kaynaklı olarak, sistem başlangıç anından kısa süre sonra tek noktada takılı kalmaktadır.



Şekil 5.2 - Verilog 6Q6 sabit noktalı sayı sistemi benzetimi zamana karşı x,y,z değerleri.



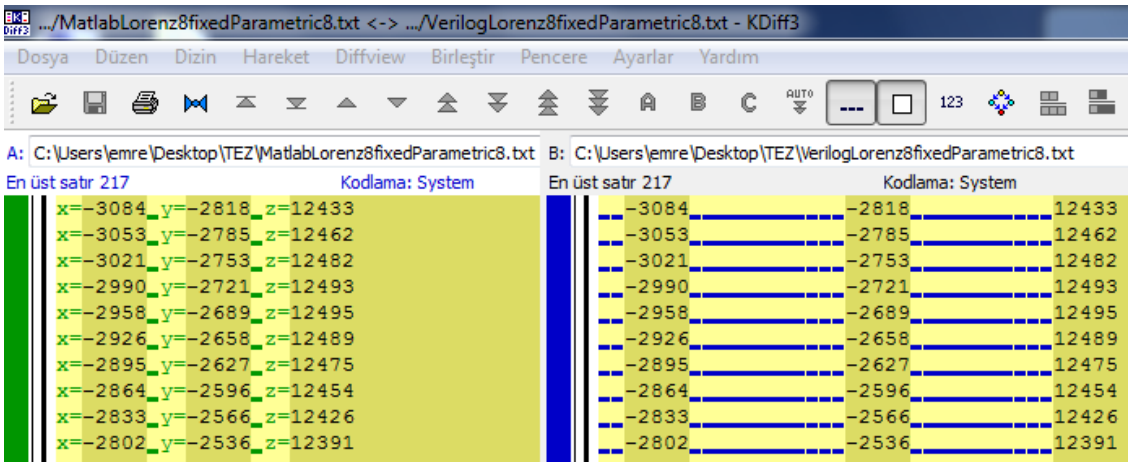
Şekil 5.3 -Kayan ve sabit noktalı sayı sistemlerinde benzetimlerin zamana karşı x,y,z değerleri.



Şekil 5.4 - Kayan noktalı ile sabit noktalı sayı gösterimlerinde Lorenz sisteminde oluşan fark grafikleri.

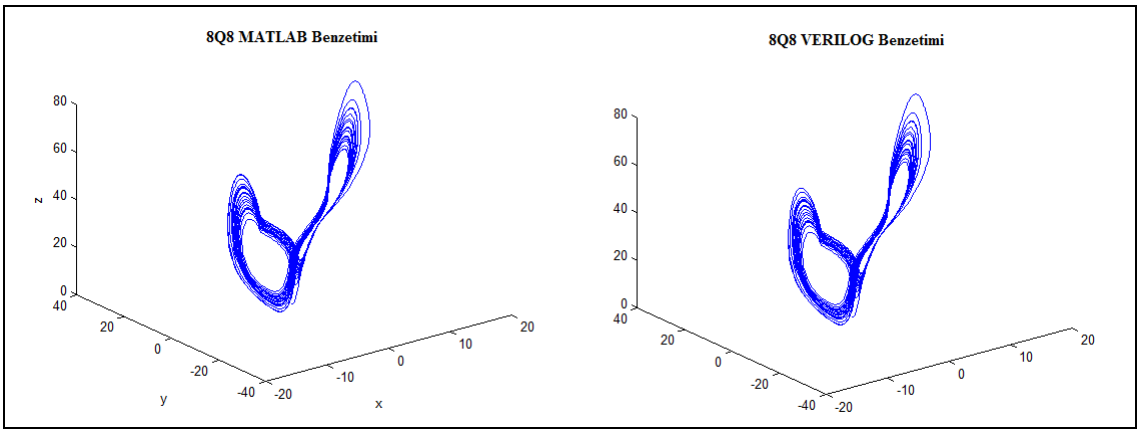
Lorenz kaotik sisteminde kayan noktalı ve sabit noktalı sayı gösterim sistemleri arasındaki değişimlerden kaynaklanan fark grafikleri Şekil 5.4'de görülmektedir. Lorenz

kaotik sistemi için bütün çıkışlar aynı zamanda bir sonraki işlem için giriş olduğundan sistemdeki küçük bir hata katlanarak büyür. Bu sebeple sistemde oluşan hata oranı sabit olmaz. Şekil 5.3'te kayan noktalı sayı gösteriminde oluşturulan sistem çözümü ile farklı bit uzunluklarındaki sabit noktalı sayı gösterimi çözümleri karşılaştırılarak Şekil 5.4'teki fark grafikleri elde edilmiştir. Bu grafiklerde, başlangıç koşullarından benzetim süresi sonuna kadar 50 birim zamanlık sürede 0.01 sürelik adımlar dahilinde oluşan sistem çözüm değişimleri ele alınmıştır. Kısaca Şekil 5.4'de görülen fark grafikleri, iki farklı sayı gösterim sisteminde oluşan hataların birbirlerine göre nasıl değişim gösterdiğini tanımlamaktadır.



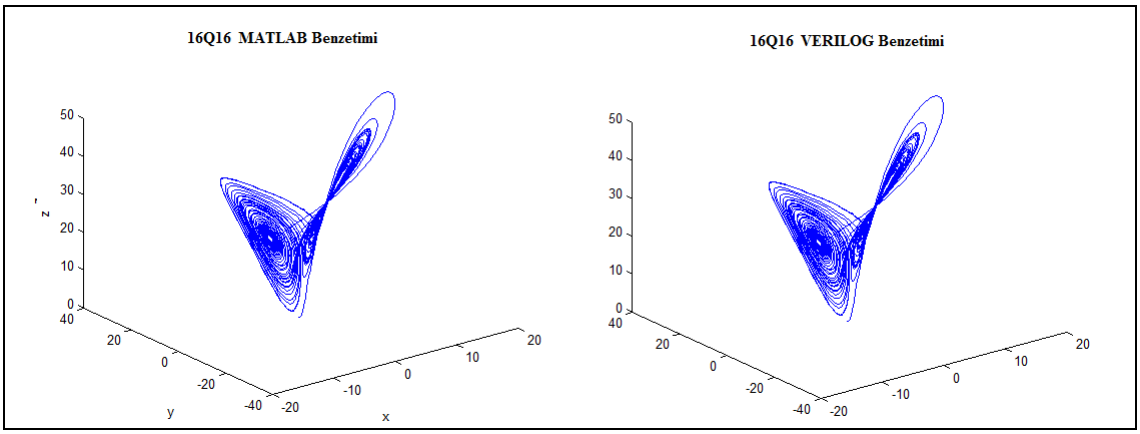
Şekil 5.5 - Kdiff yazılımı ile 8Q8 Matlab ile Verilog benzetim sonuçlarının karşılaştırılması.

Verilog ve Matlab dillerinde modellenen Lorenz sistemlerinden alınan çıkış değerleri dosyaya yazdırılmıştır. Dosyaya yazdırılan bu değerler Kdiff isimli yazılım aracılığıyla Şekil 5.5'te karşılaştırılmıştır. Lorenz sistemi, kaotik sistemlerin uygulanması bakımından Logistic Map ve Henon Map'ten daha önemlidir. Bunun nedeni ise karmaşıklığı ve aldığı değerler bakımından önemli bir örnek olmasıdır. En ufak bir hata, Şekil 5.5'te görülen sistem çıkış değerlerinde büyük sapmalara neden olabilmektedir. Bu yüzden elde edilen değerlerin çeşitliliği, benzetim ve sistem tasarımının doğruluğu açısından önem arz etmektedir.



Şekil 5.6 - 8Q8 bitlik Matlab ile Verilog benzetim sonuçlarının grafiksel karşılaştırılması.

Şekil 5.5'de, yazılımsal ve donanımsal benzetim sonuçlarının grafikleri karşılaştırılmaktadır. Grafikte görüldüğü üzere, 8Q8 seviyesinde Matlab ve Verilog dillerinde oluşturulan modellerin çözümleri aynı çıktıları vermektedir. Sabit noktalı sayı sistemi, kayan noktalı sayı sistemine göre temsil farkı olmasına rağmen değişik parametrelerde kullanılabilmektedir. Şekil 5.7'da 16 bit tamsayı ile 16 bit ondalıklı sayı gösteriminde Matlab ve Verilog sistemlerinin karşılaştırılması yapılmıştır. Şekil 5.7'da görüldüğü gibi Verilog HDL dilinde tasarlanan sistem, istenilen sonucu vermektedir.



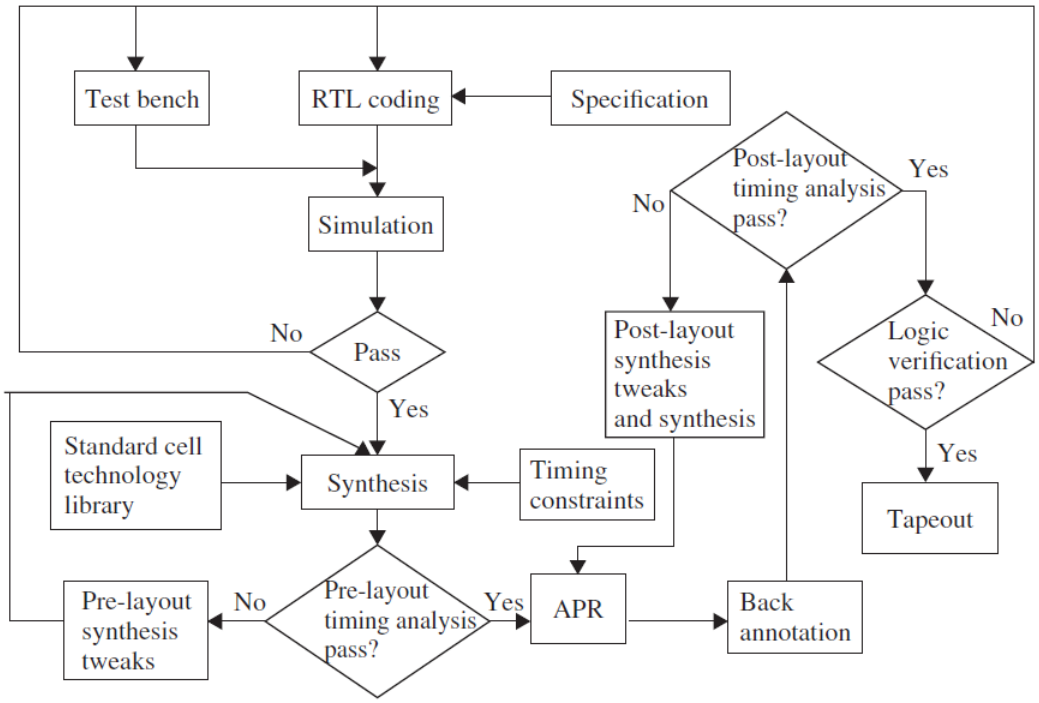
Şekil 5.7 - 16Q16 bitlik Matlab ile Verilog benzetim sonuçlarının grafiksel karşılaştırılması.

5.2. Sentezlenebilirlik Koşulları ve Sentezleme

Benzetimin düzgün çalışması sistemin lojik seviyede oluşturulabileceğini göstermemektedir. Lojik seviyede devrenin oluşturulabilmesi için sistemin sentezlenebilir olması gerekmektedir. Verilog dilinde tasarlanan sistemde, sözdizimi hatası(syntax error) bulunmasa da sistem sentezi işleminde hatalar bulunabilmektedir. Çalışmada Lorenz sistemi sentezlenirken karşılaşılan hatalar, sonlu durum makinesinde

bütün koşulların ele alınmamasından veya bölme işlemindeki sorunlardan kaynaklanmıştır.

RTL seviyesindeki tasarımdan sonra sistem, sentezleme aşamasına gelmektedir. Sonrasında sistemin Şekil 5.8'de görülen zamanlama ve devre kontrolleri yapılmaktadır (Lee, 2003). Bu safhada Xilinx ISE tarafından oluşturulamayan bir modül ya da komut varsa yazılım hata verir. Ayrıca sistem performansını artırabilecek öneriler yine bu aşamada Xilinx ISE tarafından belirtilmektedir.



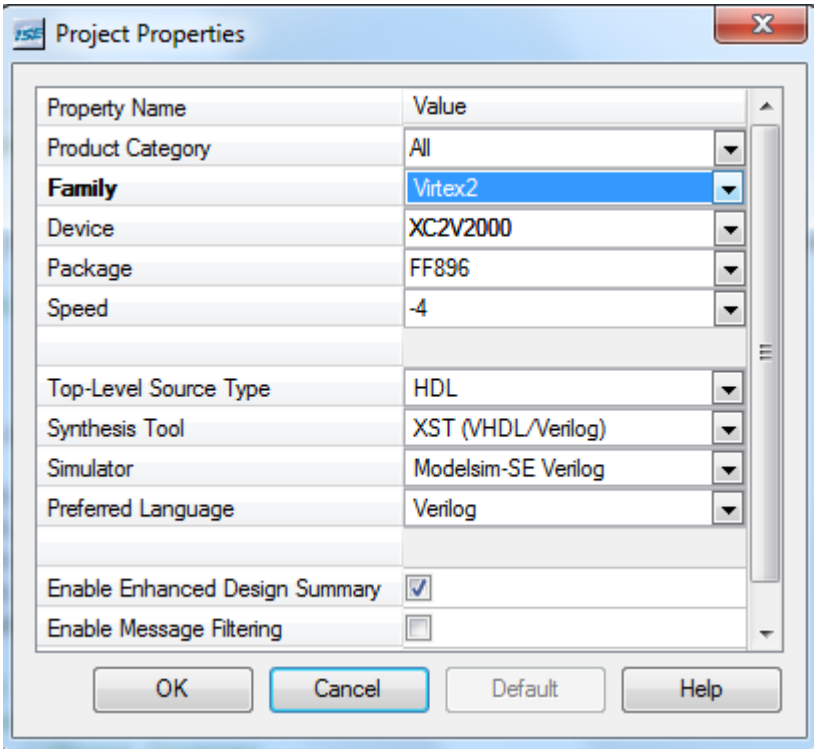
Şekil 5.8 - Lojik sistem tasarımı için akış şeması (Lee, 2003).

Yazılan bölme modülünde, kontroller dışında bölüm işlemi için bir alt modül daha yazılmasına rağmen sabit sayılar için bir değişiklik yapılmamıştır. Sentezleme işlemi sırasında RTL seviyesinde sabit olan 2 ve 6 sayıları için bölme işlemi hatasız gerçekleştirilirken farklı sayılarda sentezleme işlemi gerçekleştirilememektedir. Bu sebeple değişken sayılarla bölme işlemi yapılması gerektiğinde, sentezlenebilir olarak tanımlanan birleşimsel bölme modülü kullanılmaktadır.

5.3. Sentez Sonuçları

Tezde tasarlanan Lorenz sistemi, farklı sayılardaki bit gösterim formatlarında sentezlenmiştir. Sentezleme işlemi için literatürdeki benzer uygulamalarda kullanılan

donanımlar esas alınmıştır. Bu sebeple Şekil 5.9'deki Xilinx'in Virtex II serisinden olan 2v2000ff896 modeli ile 2v1000fg456 modeli sentez işlemi için seçilen donanımlardandır.



Şekil 5.9 - Xilinx ISE proje ayarları donanım ve özelliklerinin belirlenmesi.

8Q8 formatında Lorenz kaotik sistemi Verilog modelinin Xilinx Virtex-II üzerindeki sentez sonuçları Çizelge 5.1'te görüldüğü gibidir.

Çizelge 5.1 - 8Q8 Xilinx xq2vp40-5fg676 FPGA sentez sonuçları.

Kaynak Kullanım Özeti (yaklaşık değerler)			
Lojik Kullanımı	**Kullanılan**	**Mevcut**	**Kullanım Oranı**
Dilim Sayısı	1685	19392	8%
Flip-Flop Dilim Sayısı	211	38784	0%
4 girişli LUT Sayısı	3129	38784	8%
Bağlı IOBs Sayısı	122	416	29%
MULT18x18 Sayısı	8	192	4%
GCLK Sayısı	1	16	6%

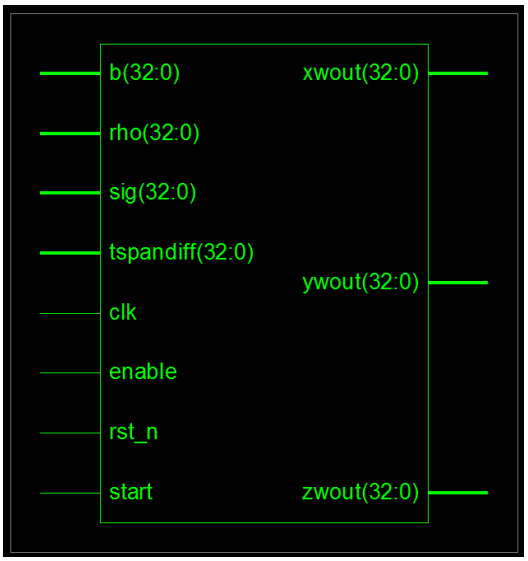
Çizelge 5.1 ve Çizelge 5.2'deki tablolardan anlaşılacağı üzere, bit gösterim sayısı ile kullanılan toplam dilim yapısı arasında doğru orantı bulunmaktadır. 16Q16 seviyesinde gerçeklenen sistem istenilenden daha fazla dilim yapısına ihtiyaç duymasına rağmen yazılımsal model ile aynı sonuçları vermesi açısından önemlidir.

Bununla birlikte gerçeklenen sistem, optimize edilmiş sistemlere göre daha az hata oranı bulundurmaktadır.

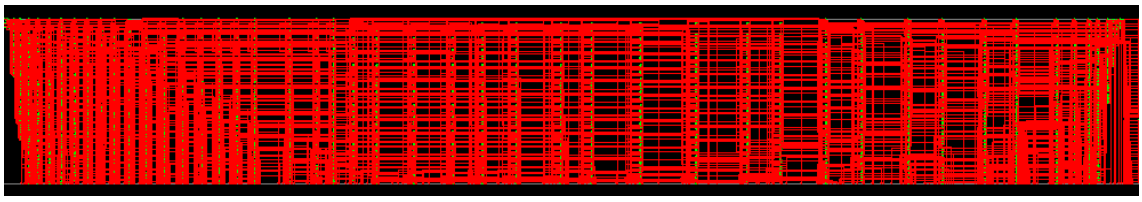
Çizelge 5.2 - Farklı bit gösterimlerinde Xilinx xq2vp40-5fg676 FPGA sentez sonuçları.

Kaynak Kullanım Özeti (yaklaşık değerler)						
Lojik Kullanımı	**8Q8**	**8Q12**	**12Q12**	**16Q16**	**Mevcut**	**Kullanım Oranı**
Dilim Sayısı	1685	2544	3356	5560	19392	8% - 28%
Flip Flop Dilim Sayısı	211	262	310	400	38784	0%
4 girişli LUT Sayısı	3129	4779	6367	10166	38784	8% - 26%
Bağlı IOBs Sayısı	122	150	178	234	416	29% - 56%
MULT18x18 Sayısı	8	32	32	32	192	4% - 16%
GCLK Sayısı	1	1	1	1	16	6%

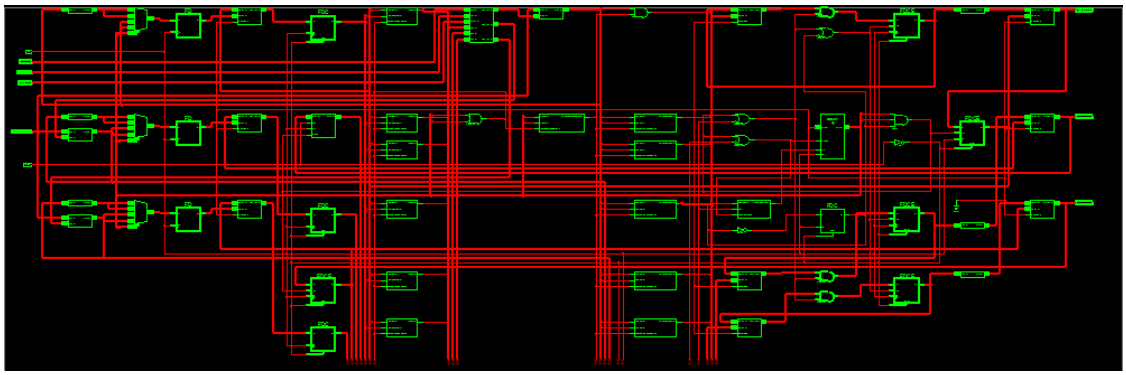
Sentez işlemi sonucunda elde edilen veriler ile sistemin lojik yapısı Şekil 5.11'da görüldüğü gibi Xilinx ISE yazılımı tarafından oluşturulmaktadır. Başlangıçta Şekil 5.10'daki ana modül giriş ve çıkışlarını gösteren bir devre bloğu gözükmektedir. Sonrasında modülün içine girildiğinde Şekil 5.11'daki lojik yapıların ve bağlantıların gösterildiği şema ortaya çıkmaktadır.



Şekil 5.10 - 16Q16 Ana modül devre bloğu.



Şekil 5.11 - Ana modül lojik şeması.



Şekil 5.12 - Ana modül RTL şeması.

RTL seviyesinde sistemin oluşumu, sentez bölümü altındaki RTL şeması bölümünden elde edilmektedir. Lorenz kaotik sisteminin RTL şeması Şekil 5.12'de görülmektedir.

Sentez sonuçları, Lorenz kaotik sisteminin donanımsal olarak tasarlanarak dijital devrenin oluşturulabildiğini gösterir. Bu sayede başlangıçta tanımlanan sistem özelliklerini gerçekleyen devre, çalışabilir hale getirilir.

5.4. Donanımsal Hız, Kaplanan Alan ve Optimizasyon

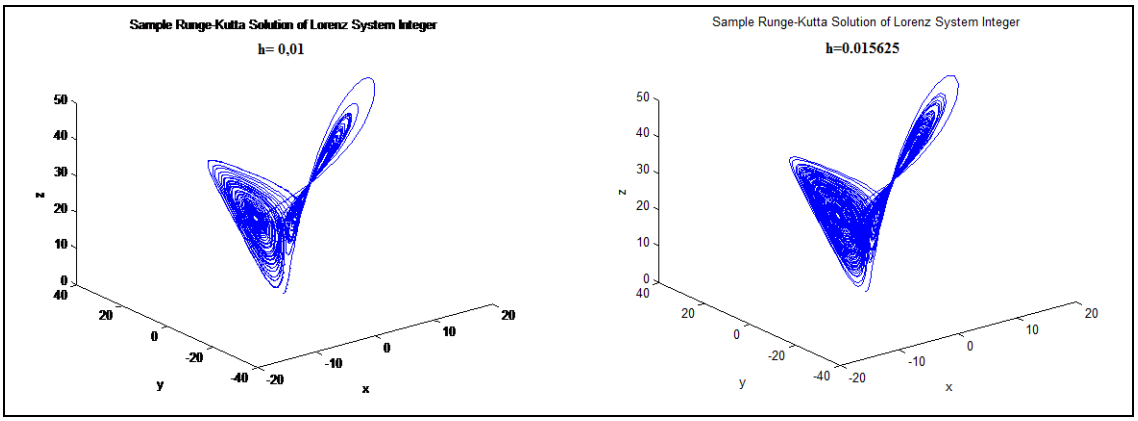
Donanımsal hız, sistemin saat frekansının büyüklüğüne bağlı olarak tanımlanırken kaplanan alan, FPGA üzerinde gerçeklenen sistemin ihtiyaç duyduğu dilim sayısını ifade eder. Donanımsal olarak hız ve kaplanan alan arasında bir ilişki bulunmaktadır. Sistemin hızını artırmak için yapılan değişimler donanımsal olarak kullanılan alanı artırmaktadır. Kaplanan alanın azaltılması ise sistem hızının yavaşlamasına neden olur.

Sistem optimizasyonu, alan-hız ilişkisini değiştirmeden daha az alan ve daha hızlı donanımsal yapı için sistemde yapılan değişiklikleri ifade eder. Hızın artırılması ve donanımsal alanın azaltılması, sistem hatasının daha fazla olmasına sebep olur. Sistemin kayan noktalı sayı gösterimine daha yakın olması ise kullanım alanını artırmaktadır. Ayrıca sabit noktalı sayı gösterim formatlarında parametrik olarak farklı bitlerde işlem yapılacaksa sistem bu parametrik yapıya göre tasarlanacağından iyileştirme yani optimizasyon seçenekleri kısıtlanır.

Çarpma modülü işlem yapabilmek için bölme modülüne göre daha fazla bit alanına ihtiyaç duyar. Sentezlenebilir bölme modülü ise daha fazla kontrol içerdiğinden

karmaşık bir yapıya sahiptir. Akümülatörden elde edilen sayının, altıya bölünmesi ile 1/6'yla çarpılması aynı manaya geleceğinden işlemler çarpma modülü kullanarak yapılabilir. Fakat bu durum, her bit seviyesi için geçerli olmamaktadır. Bunun nedeni 1/6 sayısının 0,1667 şeklindeki gibi ifade edilebilmesidir. Örneğin 0,5704 gibi bir sayı kayan noktalı sayı sisteminde gösterilerek bölüm işlemi yapılacak olunursa 0,5704/6=0,0951 sonucu elde edilir. Fakat bu sayının ondalıklı kısmı 8-bit ile ifade edilirse elde edilen sonuç 0,0958 olacaktır. Yani 0,0007'lik bir hata oluşacak ve elde edilen bölüm değerleri farklılık gösterecektir. Bu nedenle belirli bir bit seviyesinin altında 1/6 ile çarpım işlemi hatalı sonuçlar üreteceğinden, sistemden alınan değerler ile Matlab yazılımı ile elde edilen değerler birbirini tutmayacaktır. Fakat aynı sayı 12-bitlik ya da 16-bitlik bir ondalıklı sayı gösterimi ile gösterildiği durumda istenilen 0,0951 sayısına ulaşılabilmektedir. Sonuç olarak bölme modülünün çıkarılması toplam sistemde avantaj sağlasa da istenilen sayı sisteminde işlem yapma özelliği yani parametrik özellik yitirilmektedir. Bölme işleminin kaldırılması sadece belirli parametrelerde iyi bir performans verir. Bu sebeple Lorenz kaotik sisteminin farklı bit seviyelerinde oluşturulması yerine sabit bit seviyesinde oluşturulması bölme modülünün kaldırılmasına olanak verir. Böylece sistemde iyileştirme sağlanabilir.

Sistemde iyileştirme bakımından RK4 denklemlerinde dikkat edilmesi gereken bir diğer husus, kullanılan *h* parametresidir. 0,001 şeklinde aldığımız değer, belirttiğimiz ondalıklı sayının en az 8-bit ile ifade edilme zorunluluğunu ortaya çıkarmaktadır. Fakat *h* parametresi n-bitlik bir gösterimde $1/2^n$ şeklinde ifade edildiği zaman bu zorunluluk ortadan kalkacaktır. Bu sebeple *h* değerinin değişim aralığı dikkatli belirlenmelidir. Şekil 5.13'de *h* parametresi değişiminin Lorenz çekicisi üzerindeki etkisi görülebilmektedir. Bu nedenle *h* aralığı belirli bir değerden düşük olmamalıdır.



Şekil 5.13 - Lorenz denkleminde h parametre değişiminin çözüme etkisi.

Sistemdeki sayılarda uygulanan yuvarlama işlemi donanımsal olarak kullanılan alanı artırmaktadır. Yuvarlama işlemi ile hata oranının azaltılması sağlanmıştır. Yuvarlama işlemi uygulanmazsa FPGA üzerindeki gerekli donanımsal alan büyük ölçüde azalacaktır. Ortaya çıkan hata oranı, bit gösteriminin belirlendiği parametrelere göre değişim gösterse de tez çalışmasında önemli olan sistemdeki kaotik özelliğin korunmasıdır. Lorenz sisteminde oluşan hata ve bu hataların analizi ile ilgili durumlar Lian ve Liu tarafından incelenmiştir (Lian ve Liu, 2000).

Yuvarlama işleminin önemsiz olduğu durumlarda çarpım ve bölme işlemleri için bit kaydırma yönteminin kullanılması sistem sonucunu etkileyecektir. Fakat bu bit kaydırma yönteminde elde edilen sonuçlar toplam hatayı artıracaktır. Bu durum aynı zamanda yazılımsal benzetim sonuçları ile gerçeklenen donanımsal çıktıların birbirinden farklı olmasına neden olacaktır. Bu sebeple sistem kullanımı donanımla sınırlı kalacaktır.

5.5. Literatürdeki Çalışmalar ile Karşılaştırılması

Literatürdeki Lorenz sisteminin FPGA ile gerçeklenmesi üzerinde yapılan çalışmalarda genellikle Matlab/Simulink yazılımının System Generator aracı kullanılmıştır (Atoche vd., 2006). Ayrıca VHDL kullanılarak 16Q16'lık sistemde oluşturulan Lorenz kaotik sistemi de bulunmaktadır. Daha ayrıntılı inceleyecek olursak, System Generator aracı kullanılarak elde edilmiş sistem tasarımı ile Xilinx 2v2000ff896 cihazı kullanılarak gerçekleştirilen sentez işlemi sonuçları Çizelge 5.3'te görüldüğü gibidir (Atoche vd., 2006). Aynı cihaz üzerinde 16Q16'lık bir sistem kullanarak çalışmada gerçekleştirilen sentez sonuçları ise Çizelge 5.4'te görülmektedir. Tezde

oluşturulan 16Q16'lık sistem, Çizelge 5.3'te görülen sistemden daha iyi bir alan performansı vermesine karşın kullanılan yöntem farklılıklarından dolayı daha fazla hata oranına sahiptir.

Gerçeklenen Lorenz kaotik sistemi kullanılarak düşük bit seviyelerinde daha az alan gerektiren kaotik sistemler oluşturulabilmektedir. Ayrıca oluşturulan bu sistem Matlab yazılımı ile aynı sonuçları vermektedir.

Çizelge 5.3 - Xilinx 2v2000ff896-4 donanımında kayan noktalı sayı sistemi sentez sonuçları (Atoche vd., 2006).

Kaynak Kullanım Özeti (yaklaşık değerler)			
Lojik Kullanımı	**Kullanılan**	**Mevcut**	**Kullanım Oranı**
Dilim Sayısı	6055	10752	56%
Bağlı IOBs Sayısı	193	624	30%
GCLK sayısı	1	16	6%

Çizelge 5.4 - Xilinx 2v2000ff896-4 donanımında elde edilen sabit noktalı sayı sistemi 16Q16 sentez sonuçları.

Kaynak Kullanım Özeti (yaklaşık değerler)			
Lojik Kullanımı	**Kullanılan**	**Mevcut**	**Kullanım Oranı**
Dilim Sayısı	5217	10752	48%
Flip Flop Dilim Sayısı	406	21504	1%
4 girişli LUT Sayısı	10164	21504	47%
Bağlı IOBs Sayısı	234	624	37%
MULT18x18 Sayısı	32	56	57%
GCLK sayısı	1	16	6%

Otomatik kod oluşturma aracı kullanılmadan yapılan Azzaz ve arkadaşlarının oluşturduğu 16Q16'lık sistem, otomatik kod üretim yöntemi ile oluşturulan sistemden çok daha iyi bir sonuç vermektedir. Alan olarak bakıldığında Çizelge 5.5, Çizelge 5.3'te görülen sistemden ve tezde oluşturulan sentez sonuçlarından daha iyi bir alan performansı göstermektedir. Bu sistemler incelendiğinde tez çalışmasında elde edilen Çizelge 5.6 ile Çizelge 5.5 verileri arasındaki temel alansal fark, sistem gerçekleştirilirken ele alınan yaklaşımlardan kaynaklanmaktadır. Azzaz ve arkadaşlarının önerdikleri sistemde çarpma ve bölme işlemleri bit kaydırma yöntemi ile yapılmıştır (Azzaz, 2009). Bu sebeple, Azzaz ve arkadaşlarının önerdikleri sistem ile tezde önerilen sistem 16Q16 yapısında olmalarına rağmen Çizelge 5.5'te görüldüğü

üzere Azzaz ve arkadaşlarının tasarımları daha az donanımsal alan kullanmaktadır. İncelenen tasarımında yazılımsal ve donanımsal sonuçlar karşılaştırılmadığı için tezde kullanılan yöntem ve sistem farklılık göstermektedir. Tez kapsamında tasarlanan sistemde çarpma ve bölme modüllerinin yanı sıra yuvarlama, taşma kontrolü gibi durumlar ile parametrik sayı ifade özelliği kaynak tüketimini büyük ölçüde artırmaktadır.

Çizelge 5.5 - 16Q16 sistemde Xilinx 2v1000fg456-4 donanımında sentez sonuçları (Azzaz, 2009).

Kaynak Kullanım Özeti (yaklaşık değerler)		
Lojik Kullanımı	**Kullanılan**	**Mevcut**
Dilim Sayısı	1926	5120
Flip Flop Dilim Sayısı	791	10240
4 girişli LUT Sayısı	2718	10240
Bağlı IOBs Sayısı	11	324
MULT18x18 Sayısı	40	40
GCLK sayısı	1	16

Çizelge 5.6 - Xilinx 2v1000fg456-4 donanımında elde edilen 16Q16 sentez sonuçları.

Kaynak Kullanım Özeti (yaklaşık değerler)			
Lojik Kullanımı	**Kullanılan**	**Mevcut**	**Kullanım Oranı**
Dilim Sayısı	5207	5120	101%
Flip Flop Dilim Sayısı	400	10240	3%
4 girişli LUT Sayısı	10155	10240	99%
Bağlı IOBs Sayısı	234	324	72%
MULT18x18 Sayısı	32	40	80%
GCLK sayısı	1	16	6%

Sistemler üzerinde değerlerin birbirini tutması yani eşzamanlı olarak sistemlerin çalışması önemsenmiyorsa çarpma ve bölme modüllerindeki karmaşık yapılar giderilerek sistemde genel bir iyileştirme yapılabilir. Fakat sistemin toplam kaynak kullanımı, temel olarak çözüm yönteminin hassasiyeti ve kaotik sistemin yapısına bağlıdır.

6. SONUÇ

Sonuç olarak, Lorenz kaotik sisteminin FPGA üzerinde gerçeklenmesi farklı bit seviyelerinde sabit noktalı sayı gösterim sistemleriyle sağlanmıştır. Literatürde kullanımı görülen Matlab Simulink yazılımının yerine Verilogda bütün parametreler ve koşullar göz önüne alınarak sistem gerçeklenmiştir. Parametrik sistem oluşturulurken, Matlab'da yazılan sistem modeli ile Verilog'da yazılan model arasında değerlerin eşitliği sağlanmıştır. Bu sayede yazılımsal ve donanımsal olarak oluşturulan iki Lorenz sistemi de aynı değerleri vermektedir.

Gerçeklenen sistemde geliştirilebilir özellikler bulunmaktadır. Özellikle Bölüm 4.4'te bahsedilen h parametresinin değişimi ve sistemindeki işlemler üzerinde yuvarlamanın kaldırılması toplam FPGA üzerinde kullanılan alanı azaltacaktır. Fakat bu işlem, hata oranının daha yüksek olmasına yol açacaktır. Eğer sadece kaotik özelliğin korunması hata oranından daha önemliyse bu durumda yapılan sistem iyileştirilebilir. Çarpım ve bölme modülleri yerine bit kaydırma yöntemi kullanılması sistemde donanımsal alanı azaltmasına karşın hataların artmasına ve sistem değerlerinin yazılımsal ve donanımsal farklılık göstermesine yol açacaktır. Lorenz sisteminin çözümünde kullanılan RK-4 nümerik çözüm yöntemi yerine hata oranı daha fazla olan diğer sistemlerin kullanılması donanımsal alanda iyileştirme sağlar. Bu durumda sistem çözümünün yazılımsal ve donanımsal eşitliğinin sağlanması için MATLAB üzerinde de sistem modelinin benzer biçimde tanımlanması gerekir.

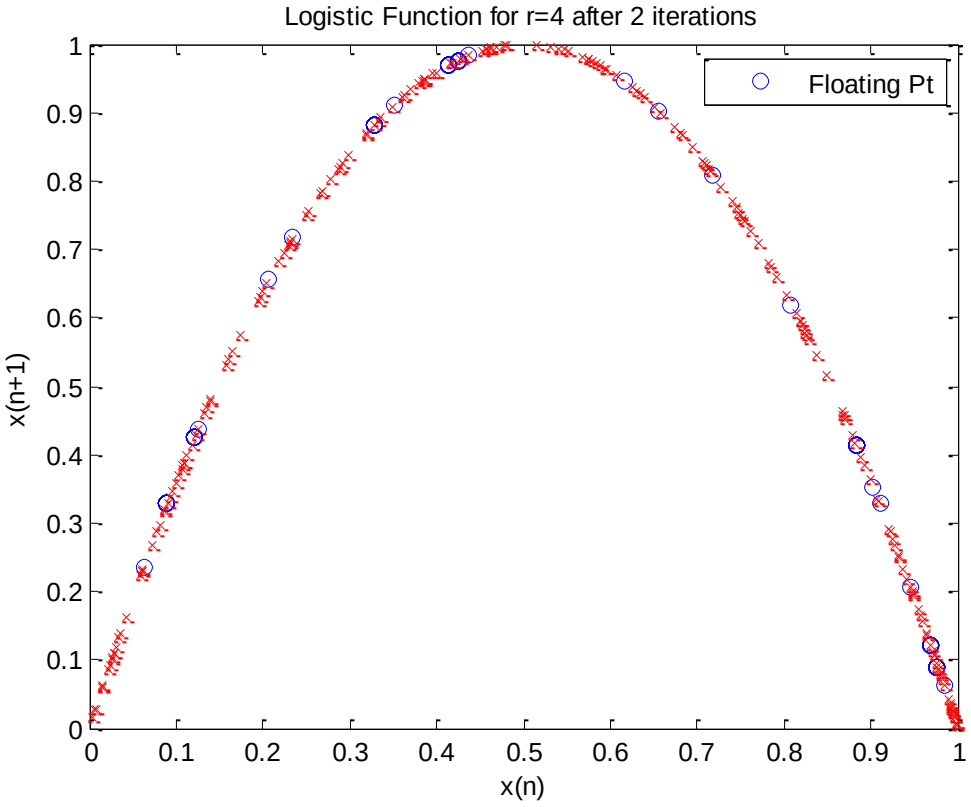
Tezde gerçeklenen Lorenz sistemi yerine farklı kaotik sistemler de FPGA üzerinde gerçeklenebilir. Oluşturulan Lorenz matematiksel hesaplama modülü, istenilen kaotik modele göre yeniden yazılması ile sistem 4.dereceden Runge Kutta çözümüne göre hesaplama yapacaktır. Farklı bir nümerik çözüm önerilirse ana modülde buna uygun olarak değiştirilmelidir.

Yapılan çalışma, donanımsal olarak hedeflenen gerçekleme işlemini yapmakla birlikte ileriki kaotik sistem uygulamaları için bir temel oluşturmaktadır. Bu sayede kaotik sistemlerde genel amaçlı kullanılabilecek bir Verilog modeli oluşturulmuştur.

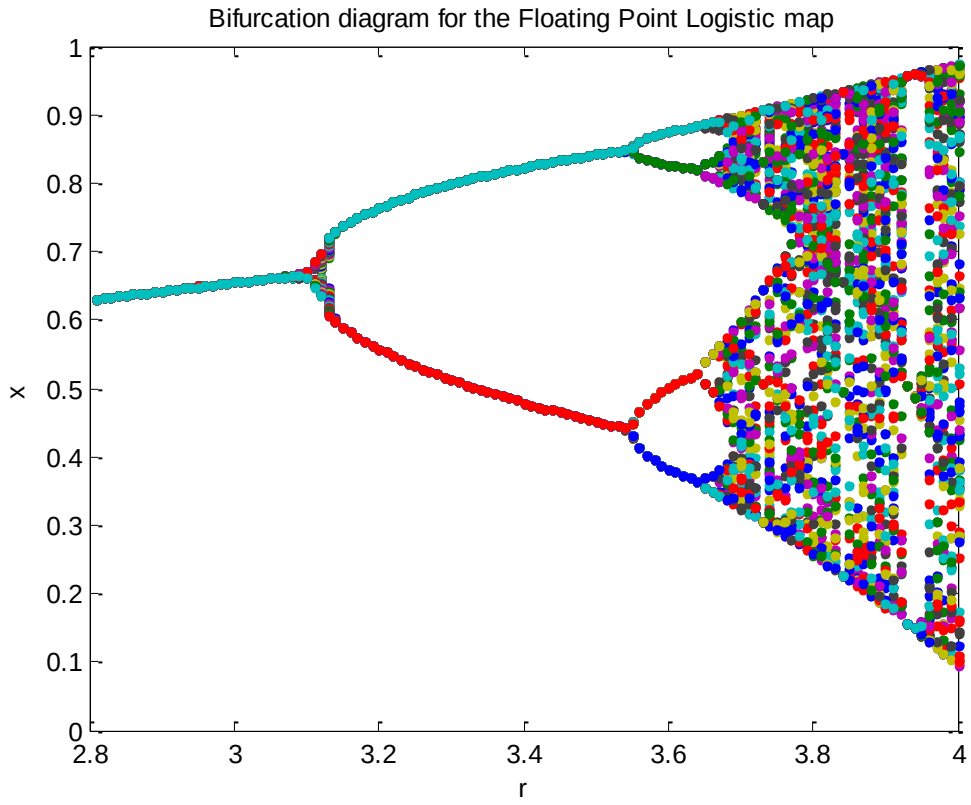
7. EKLER

7.1. Logistic Map Sistem Modellemesi

Lorenz Sistemi birçok durumu birleştiren karmaşık bir sistem olmasından dolayı göreceli olarak daha basit bir sistem olan ve kaotik özelliği belirli bölgelerde gösteren Logistic Map Matlab ve Verilog programlama dilleri kullanılarak tez kapsamında yazılmıştır. Şekil 7.1'de sabit noktalı ve kayan noktalı sayı gösterim sistemlerinde elde edilen x_n ve x_{n+1} değerlerinin birbirine göre değişimleri verilmiştir. Şekil 7.2'de ise sistemin çözüm çıktıları ve r parametre değişiminin kayan nokta sisteminde sistem çözümlerine olan etkisi görülmektedir.

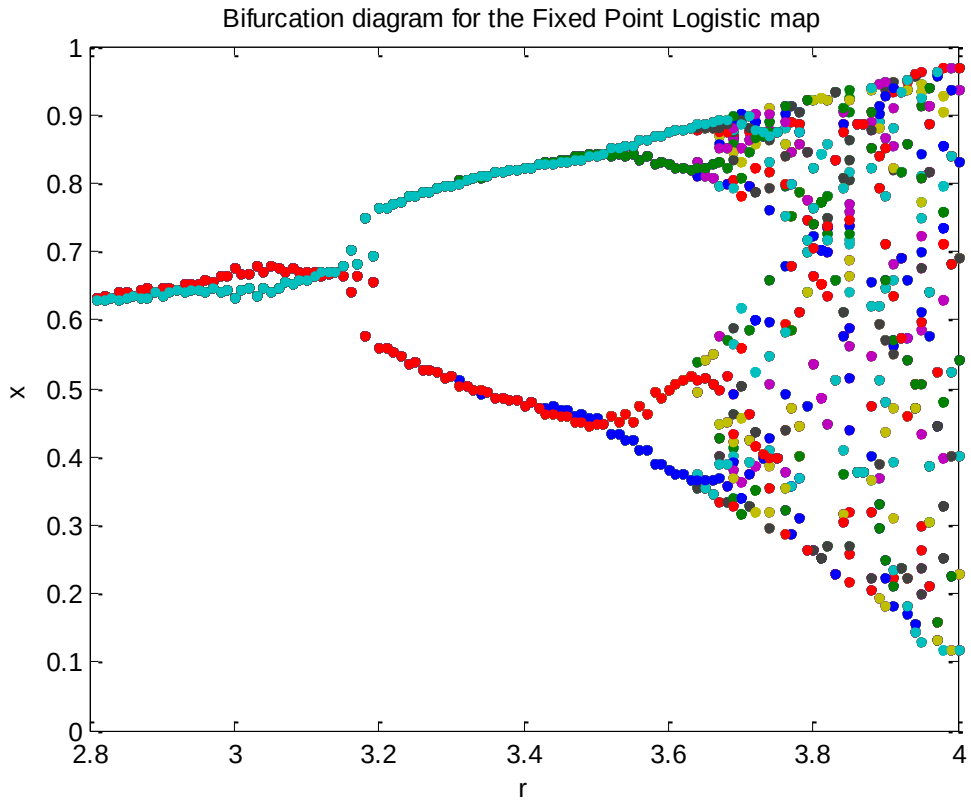


Şekil 7.1 - Logistic Map sisteminin x_n ile x_{n+1} değerlerinin değişim grafiği.



Şekil 7.2 - Kayan noktalı sayı sisteminde Logistic Map'deki x değerlerinin r parametresine göre çatallanması.

Şekil 7.3'te görülen çatallanma (bifurcation) diyagramı, Logistic Map denklemini farklı *r* parametrelerinde aldığı çıkış değerlerinin grafiğini göstermektedir. 3,2 değerinden düşük *r* değerlerinde sistem tek tür çıkış üretirken 3,7 ve üzeri *r* değerlerinde sistem kaotik hale gelmektedir. Bu sistemde eğer sayı gösterimini belirleyen bit sayısı çok düşürüldüğü takdirde sistem 3,7 ve üzeri *r* değerlerinde dahi sayı gösterimi için belirlenen bir sayısı düşük olduğundan kaotik özelliğini yitirebilmektedir.



Şekil 7.3 – Sabit noktalı sayı sistemi üzerinde Logistic Map'in x verilerinin r parametresine göre çatallanma diyagramı

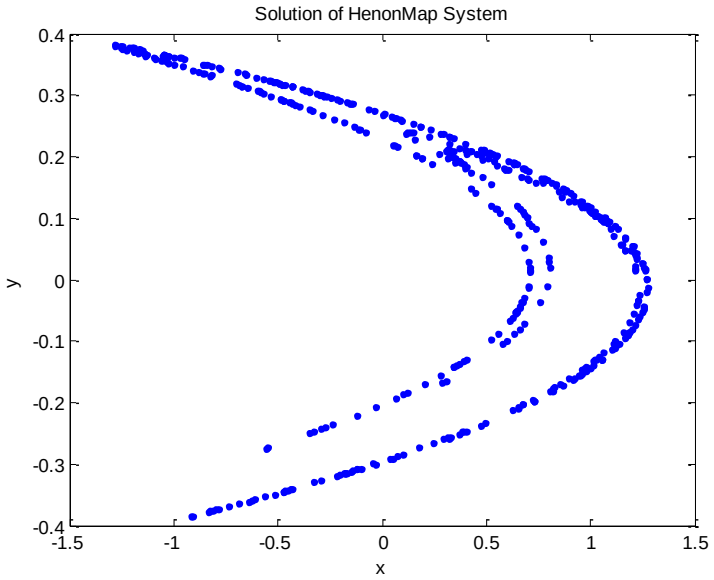
Şekil 7.4'te ise Logistic Map sisteminin parametrik modelinin Matlab kodu yer almaktadır. Eşitlik 2.1 uyarınca bütün değerler genişletilerek sabit noktalı sayı gösteriminde denklem her 0,01 birimlik *dt* zaman aralığında hesaplanarak Şekil 7.1'deki çözüme ulaşılmaktadır. Sabit noktalı sayı gösteriminde elde edilen değerlerin yayılımı Şekil 7.3'te görüldüğü gibi olmaktadır.

```
%r Parameter Value
r=4;
%rounded value
rm=round(r*2^m);
% x0=1.26;

%initial x value
x0=0.125;
%rounded value
x0m=round(x0*2^m);

%t: Iteration
t = 0:1:200; % [0 0.01 0.02 ... 50]

% %%Bifurcation diagram rv change interval
 rv=0.1:0.01:4;
 rvlength=length(rv);

N=length(t);
nx=length(x0);

xfx=[x0m zeros(nx,N-1)]; %x0 initial condition with filled in time simpen zero
x=[x0 zeros(nx,N-1)]; %x0 initial condition with filled in time simpen zeros

% %For Bifurcation diagram
 zrs=[zeros(rvlength,N)];
 zrsfx=[zeros(rvlength,N)];
 zrs(:,1)=x0;
 zrsfx(:,1)=x0m;
 xr=zrs;
 xrfx=zrsfx;

% rv=0:0.1:4;
%LogisticMap Function
for i=1:N-1
x(i+1)=r*x(i)*(1-x(i));
xfx(i+1)=round( (round( (rm*xfx(i))/2^m )  * ( (1*2^m) - xfx(i) )) /2^m );

%Round it and calculate ever value as *2^m
end
xfx=xfx';
x=x';

%Bifurcation diagram for the Logistic map Calculations
k=1;
for rvx=0:0.01:3.9
    for i=1:N-1
        xr(k,i+1)=rvx*xr(k,i)*(1-xr(k,i));
```

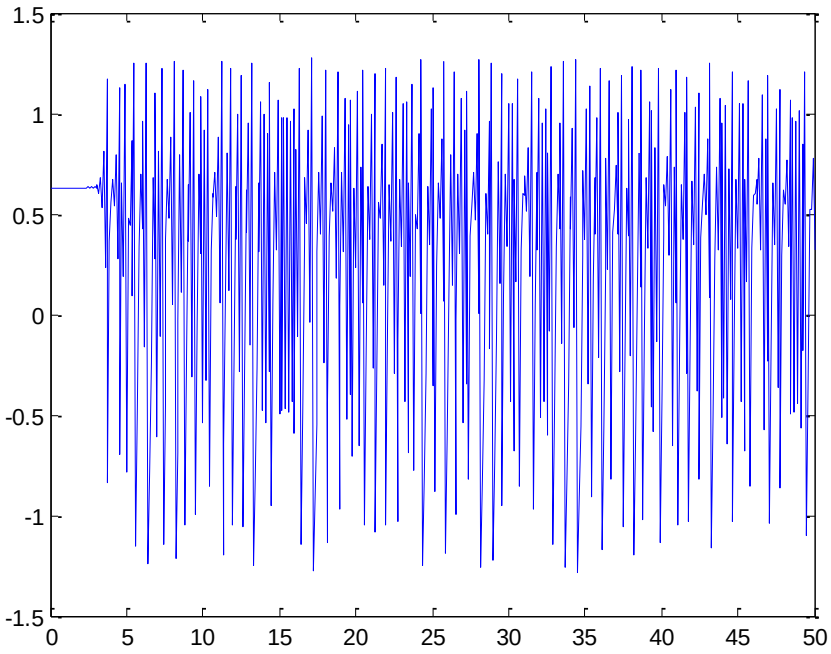
Şekil 7.4 - Parametrik Logistic Map Matlab kodu**.**

7.2. Henon Map Sistemi

Tez içerisinde Lorenz sistemine yakın olan Henon Map sistemi yazılımsal olarak modellenmiştir. Çözüm kümesinin grafiği Şekil 7.5'de görülebilmektedir. Sistemin zamana karşı değişimi de Şekil 7.6'da incelenebilir.



Şekil 7.5 - Henon Map y değerlerinin x değerlerine göre değişim grafiği.



Şekil 7.6 - Henon Map zamana göre x değeri değişim grafiği.

KAYNAKLAR

A. Castillo Atoche, G. S., "Synchronization of Chaotic Systems: Field Programable Gate Array And Nonlinear Controlfeedback Approach". *IWS 2006*, , s. 47, (2006).

Aseeri, M., Sobhy, M., & Lee, P., "Lorenz chaotic model using Filed Programmable Gate Array (FPGA)" . *Circuits and Systems, 2002. MWSCAS-2002.* , s. 30 vol.1, (2002).

Azzaz, M., Tanougast, C., Sadoudi, S., & Dandache, A. "Real-time FPGA Implementation of Lorenz's Chaotic Generator for Ciphering Telecommunications". *Circuits and Systems and TAISA Conference, 2009. NEWCAS-TAISA '09* , s. 1-4, (2009).

Birkhoff, G. D. "Dynamical Systems". *vol. 9 of the American Mathematical Society Colloquium Publications (Providence, Rhode Island: American Mathematical Society)*, (1927) .

Butcher, J. C., "Numerical methods for ordinary differential equations", John Wiley & Sons. ISBN 0471967580, (2003).

Chao, C. ,"Basic Logic Design", Lecture Notes on Verilog Course #90132300. Nanyang Technological University, Singapore, (2005).

CORE, T., "*FPGA Logic Cells Comparison",* http://www.1-core.com/library/digital/fpga-logic-cells/fpga-logic-cells.pdf, (2011).

Ding, Q., & Pan, J., "The Research of Optimization Parameter Based on Lorenz Chaotic Masking Secure Communication". *IEEE - ISBN: 978- 1-4244-8043-2* , s. 1136 – 1139, (2010).

Fiedler, G. "*Integration Basics",* http://gafferongames.com/game-physics/integration-basics/ , (2006).

Hall, C. D. "Manifesto on Numerical Integration of Equations of Motion Using Matlab", (2002).

Hénon, M. "A two-dimensional mapping with a strange attractor".*Communications in Mathematical Physics* , s. 50 (1): 69–77. doi:10.1007/BF01608556, (1976).

Huang, L., Wang, X., & Sun, G. "Design and circuit simulation of the new Lorenz chaotic system". *Systems and Control in Aeronautics and Astronautics (ISSCAA), 2010* , s. 1443 - 1447, (2010).

IEEE. "*754-2008 IEEE Standard for Floating-PointArithmetic",* http://ieeexplore.ieee.org/servlet/opac?punumber=4610933, (2009).

KAYNAKLAR(Devam Ediyor)

Jie-Hong (Roland) Jiang, S. D., "Logic synthesis in a nutshell". Y.-W. C.-T. Laung-Terng Wang içinde, *Electronic design automation: synthesis, verification, and test , ISBN 9780123743640* (Chapter 6). Bobs Books (JRM) , (2008).

Kolmogorov, A. N. (1941). On degeneration of isotropic turbulence in an incompressible viscous liquid. *Proceedings of the Royal Society of London: Mathematical and Physical Sciences (Series A)* , vol. 434, sayfa 15–17(1991).

Lee, W. F., "*Verilog Coding for Logic Synthesis",* John Wiley & Sons, (2003).

Li, Y., Yuan, C., & Huang, Y. , "A Novel Watermarking Technology Based on Lorenz Chaotic Attractor" , *Computational Intelligence and Natural Computing, CINC '09* , vol 2 s.330 - 332 , (2009).

Lian, K.-Y., & Liu, P. Synchronization with message embedded for generalized Lorenz chaotic circuits and its error analysis . *Circuits and Systems I:Fundamental Theory and Applications, IEEE Transactions* , s. vol 47, Issue 9,1418 - 1424 , (2000).

Littlewood, M. L. "On non-linear differential equations of the second order, I: The equation y" + k(1−y2)y' + y = bλkcos(λt + a), k large. *Journal of the LondonMathematical Society* , s. vol. 20, pages 180–189, (1945).

Lorenz, E. N. "Deterministic Nonperiodic Flo,". *Journal of the Atmospheric Sciences* , vol. s. 20 130-141, (1963).

Lorenz, E. N. *"The Essence of Chaos".* University of Washington Press, (1995).

Maxfield, C. The Origin of FPGAs. C. Maxfield içinde, *"The Design Warrior's Guide to FPGAs, ISBN 0750676043, 9780750676045* (s. 51-52). Elsevier, (2004).

NI., "*What is the Definition of Logic Cells, Logic Slices, Configurable Logic Blocks and Gates in Regards to FPGA Devices?",* http://digital.ni.com/public.nsf/allkb/33D4F29F1483548586256D740058B428, (2009).

Poincaré, J. H. "Sur le problème des trois corps et les équations de la dynamique. Divergence des séries de M. Lindstedt,". *Acta Mathematica vol. 13* , s. pages 1–270, (1890).

Strogatz, S. H., *"Sync: How Order Emerges from Chaos in the Universe, Nature,and Daily Life".* New York: Hyperion, (2000).

Turcotte, D.; Schubert, G., "*Geodynamics"* (2nd ed.). New York: Cambridge University Press. ISBN 0521661862, (2002).

Tzartzanis, N., "Verilog for Behavioral Modelling", University of Southern California, (1998).

KAYNAKLAR(Devam Ediyor)

Vahid, F. , "*Digital Design with RTL Design, Verilog and VHDL (2nd ed.)".* (s. 247.). John Wiley and Sons, (2010).

Verhulst, P.-F.,"Recherches mathématiques sur la loi d'accroissement de la population", *Nouv. mém. de l'Academie Royale des Sci. et Belles-Lettres de Bruxelles* , s. 18, 1-41, (1845).

Wang Zhong-Lin; Li Hong-Wei; Chen Zeng-Qiang*;* "*Design and implementation of a switch Chen chaotic system", Control Conference, CCC 2011, s. 550 - 554,* 2011.

Weisstein, E. W., *"Logistic Equation.".* MathWorld—A Wolfram http://mathworld.wolfram.com/LogisticEquation.html, (2011).

WH, Teukolsky, S., Vetterling, W., & Flannery., Section 17.1 Runge-Kutta Method. W. Press, S. Teukolsky, W. Vetterling, & Flannery içinde, "*Numerical Recipes: The Art of Scientific Computing (3rd ed.)"* (s. Section 17.1 , Section 17.2). New York: Cambridge University Press. ISBN 978-0-521-88068-8, (2007).

White, F. M., "*Viscous Fluid Flow"* (3rd. ed.). New York: McGraw-Hill. ISBN 0072402318, (2006).

Wikipedia., "*Binary numeral system"* içinde "Fractions in Binary" bölümü, http://en.wikipedia.org/wiki/Binary_numeral_system ,(2011).

Xilinx, "*Virtex-4 FPGA Manual UG070 (v2.6).",* "Configurable Logic Blocks" bölmü http://www.Xilinx .com/support/documentation/user_guides/ug070.pdf, (2008).

Yu Simin; Lu Jinhu*; "High Order Chua's Circuit and Its FPGA Realization" Control Conference, CCC 2007.*

Yu, W., & Bai, G. "A novel random number generator based on continuous-time chaos" . *Network Infrastructure and Digital Content, 2010 2nd IEEE International Conference* , s. 1052 - 1055, (2010)

ÖZGEÇMİŞ

Kişisel Bilgiler

ADI SOYADI : Emre GÜNGÖR

DOĞUM YERİ VE TARİHİ : Eskişehir/ 02.10.1984

Eğitim Durumu

Lisans Öğrenimi : Eskişehir Osmangazi Üniversitesi

Elektrik Elektronik Mühendisliği

Bildiği Yabancı Diller :İngilizce

İş Deneyimi

Stajlar : Savronik Elektronik Sanayi ve Ticaret A.Ş. /ESKİŞEHİR

EYUGGEM- Eskişehir Yazılım Üssü Genç Girişimci Eğitim Merkezi

Çalıştığı Kurumlar :Bilecik Üniversitesi Mühendislik Fakültesi

İletişim

Adres: Bilecik Üniversitesi Gülümbe Kampüsü Mühendislik Fakültesi

Tel: 0228 216 01 01 – 1387

E-Posta Adresi: emre.gungor@bilecik.edu.tr