İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

HATAYA BAĞIŞIKLI MİKROİŞLEMCİ TASARIMI

YÜKSEK LİSANS TEZİ

Buse USTAOĞLU

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

ARALIK 2015

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

HATAYA BAĞIŞIKLI MİKROİŞLEMCİ TASARIMI

YÜKSEK LİSANS TEZİ

Buse USTAOĞLU
(504131205)

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Müştak Erhan YALÇIN

ARALIK 2015

İTÜ, Fen Bilimleri Enstitüsü'nün 504131205 numaralı Yüksek Lisans Öğrencisi **Buse USTAOĞLU**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"HATAYA BAĞIŞIKLI MİKROİŞLEMCİ TASARIMI"** başlıklı tezini aşağıdaki imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Prof. Dr. Müştak Erhan YALÇIN** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Müştak Erhan YALÇIN** İstanbul Teknik Üniversitesi	.....
	Prof. Dr. Ece Olcay GÜNEŞ İstanbul Teknik Üniversitesi	.....
	Doç. Dr. Selçuk BAKTIR Bahçeşehir Üniversitesi	.....

Teslim Tarihi : **27 Kasım 2015**
Savunma Tarihi : **25 Aralık 2015**

Aileme,

ÖNSÖZ

Tez çalışmam boyunca bilgi ve tecrübeleriyle çalışmalarımda bana yardımcı olan ve yönlendiren degerli hocam Doç. Dr. Berna Örs YALÇIN'a, bu süreçte bilgilerini ve deneyimlerini paylaşıp teknik ve akademik olarak kendimi geliştirmemi sağlayan Anka Mikroelektronik Sistemler'in kurucuları İnan ERDEM'e ve Gökhan IŞIK'a, son olarak tüm hayatım boyunca maddi, manevi destekleriyle her zaman yanımda olan aileme sonsuz teşekkürlerimi sunarım.

Tezin savunulduğu ay yıl

Buse USTAOĞLU
Araştırma Görevlisi

İÇİNDEKİLER

Sayfa

KISALTMALAR

TDB : Temel Denetim Birimi
ALB : Aritmetik Lojik Birim
BHO : Bit Hata Oranı
SoC : System On Chip
FIT : Failure In Time
MTTF : Mean Time to Failure
MTTR : Mean Time to Repair
MTBF : Mean Time Between Failures
NBTI : Negative Bias Temperature Instability
SRAM : Static Random Access Memory
RTN : Random Telgraph Noise
SER : Soft Error Rate
LET : Linear Energy Transfer
MeV : Mega Electron Volt
SEU : Single Event Upset
SEE : Single Event Effect
SET : Single Event Transient
MBU : Multiple Bit Upset
HDL : Hardware Description Language
bps : bit per second
UART : Universal Asynchron Receiver Transmitter
LFSR : Linear Feedback Shift Register
FPGA : Field Programmable Gate Array
ASIC : Application Specific Integration Circuit
TMR : Triple Modular Redundancy
DWC : Duplication With Comparison
MIPS : Microprocessor without Interlocked Pipeline Stages
AES : Advanced Encryption Standart

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

HATAYA BAĞIŞIKLI MİKROİŞLEMCİ TASARIMI

ÖZET

Bu tez kapsamında gömülü mikroişlemcilerin içerisinde meydana gelebilecek hatalara rağmen işlevine devam edebilmesi için devreyi güçlendirmeye yönelik tasarım yapılmıştır. Mikroişlemciler birçok güvenliği kritik gömülü sistem uygulamalarında kullanılmaktadır ve sistemlerde bir beyin görevi gördüğü için düzgün bir şekilde çalışması kritik önem taşımaktadır. Transistör boyutlarının küçültülmesi devrelerin hatalara daha yakın olmasına yol açmakta ve bu sorun sistemlerde güvenilirlik problemini beraberinde getirmektedir. Üretim hataları, yıpranma gibi etkenlerin yanısıra elemanların faydalı çalışma ömrü içerisinde radyasyon gibi çevresel etkilerin yol açtığı tek durum etkileri kombinezonsal devrelerdeki lojik hatalar veya bellek hücrelerindeki verilerin değişmesine sebep olan bit bozumaları gibi geçici hatalara ve artan oranla oluşan geçici etkiler sonucu kalıcı hata oluşumlarına sebep olmaktadır. Bu sebeplerden dolayı hataya bağışıklı tasarım göz ardı edilemeyecek derecede gerekli ve önemli olmaktadır.

Bu çalışmada ilk olarak bir devrenin hatalar varken davranışının gözlemlenebilmesi için bir sistem gerçeklenmiştir. Öncelikle hata üretim devresi tasarlanmıştır. Ardından test edilecek tasarım olarak 8-bitlik İndirgenmiş Komut Takımı Bilgisayarı (Reduced Instruction Set Computing-RISC) mimarisi ile tasarlanmış Natalius mikroişlemcisi seçilmiştir. Gerekli ara birimler oluşturulup bağlantılar yapılmış ve sonuçlar elde edilerek grafikler yorumlanmıştır.

Kaydedici dosyası, mikroişlemci içerisinde sıklıkla erişilen kısımdır. Aritmetik ve mantıksal işlemler bu birim üzerinden yapılır ve içerisine gelebilecek bir hatanın tüm sistemi etkilemesi olasıdır. Bu nedenle mikroişlemcilerdeki kaydedici dosyalarını korumak için literatürde yer alan hataya karşı bağışıklık kazandırma yöntemleri incelenmiştir. Bu yöntemlerden donanım yedeklemesi sınıfına giren üçlü modül çoğullama yönteminin çoğunluk oylama devresinden ve bilgi yedeklemesi sınıfına giren matris kodlarının koruma bitleri hesaplarından faydalanılarak çoklu bit bozulmalarına karşı veriyi koruyan kodseti adında yeni bir yöntem önerilmiştir. TMR yöntemi 3 adet eş modül bulundurur ve çıkışa çoğunluk oylamasına göre karar verilir. Herhangi bir yere gelen hata maskelenir. Matris kodları ise hata tespit etme ve düzeltme kapasitesine sahiptir ve bu amaçla eşlik ve kontrol bitleri hesaplanır. Tasarımdaki temel fikir, kaydedici bitlerine ardışık olarak gelen hatalara karşı bir yöntem geliştirmektir. Bu nedenle veri kodsetlerine ayrılır. Ayrıca rastgele bitlere gelen hatalara karşı da veri korunabilmektedir. Eşlik ve kontrol bitlerine gelen hataların orjinal veriyi yanlış düzeltmesini engellemek için oylama mekanizması geliştirilmiştir. 2, 4 veya 8 biti koruyabilecek şekilde ayarlanabilir olması nedeniyle kullanışlı bir yöntem olmaktadır.

Bu yöntemi mikroişlemci içerisinde uygulayabilmek için MIPS mimarisi ile 32-bitlik hem tek çevrim hem de iş hatlı mikroişlemci tasarımı yapılmıştır. Mikroişlemci

üzerinde uygulama kodu çalıştırmak için Gelişmiş Şifreleme Standartı algoritması seçilmiş ve şifreleme modülü eklenerek komut genişletmesi yapılmıştır. Kaydedici dosyasının hataya karşı bağışıklık kazandırma yöntemleriyle tasarımı yapılmıştır. Hata üretim devresi kullanılarak farklı durumlarda simülasyon çıktıları alınmıştır.

Tasarımlar TSMC 90 nm teknolojisinin standart kütüphanesi ile Cadence firmasının RTL Compiler aracı kullanılarak sentezlenip yerleşimleri yapılmış alan, maksimum çalışma frekansı bilgileri verilmiştir. Ayrıca güvenilirlik analizleri yapılarak sonuçlar karşılaştırılmıştır.

FAULT TOLERANT MICROPROCESSOR DESIGN

SUMMARY

In this thesis, a fault tolerant microprocessor has been designed in order to operating correctly against faults. Microprocessors are used in safety-critical embedded system applications and they are main part of the system and their correct functionality is essential. With transistor scaling, circuits are more susceptible to errors and it leads to reliability concern in the systems. Besides the manufacture faults, wear-out factors, single event effects, that occur because of the environmental factors such as radiation effects, cause from logical errors in combinational circuits, to upsets changing data, to destructive hard errors with increasing soft error rate. Due to the these reasons, fault tolerant design is so necessary and crucial.

First of all in order to observe the behaviour of a circuit when faults exist, a fault injection circuit has been designed. Natalius microprocessor, which is 8-bit Reduced Instruction Set Computer (RISC) processor, has been chosen as design under test. Necessary interfaces have been formed and all the units have been connected to each other. Then, results have been obtained.

Register file is accessed frequently by the microprocessor. Arithmetic and logic operations are performed on register values. Therefore a fault, which occurs inside it, can effect whole system. That's why, fault tolerant techniques to protect register files have been examined in the literature. A new method, called as codeset, has been proposed by making use of the majority voter circuit of Triple Modular Redundancy (TMR), which is one of hardware redundancy method, and calculations of protection bits of matrix code, which is one of information redundancy method. There are 3 identical modules in TMR and output is decided based on the majority voting. TMR masks faults inside a module. Matrix codes method has detection and correction capability, parity and check bits are calculated for this purpose. The key idea of the design is developing a method in order to protect register file against burst errors. So, codesets have been created. Moreover, random faults in a register can be corrected. A voting mechanism has been developed and wrong correction of the original data has been prevented because of the faults that occurs parity or check bits. This technique is cost effective and configurable because codesets can be created to protect data against up to 2, 4 or 8-bit adjacent or random errors in any register.

In order to use the method in microprocessor, both 32-bit single cycle and pipeline microprocessors have been designed with MIPS architecture. Advanced Encryption Standard (AES) algorithm has been chosen and crypto module has been integrated into MIPS microprocessor so as to create an application platform. Register files have been designed with fault tolerant techniques. By using fault injection circuit, simulations have been performed in different situations.

Designs have been synthesized by using Cadence RTL Compiler on TSMC 90nm process. Area, maximum frequency results have been given. Furthermore, reliability analysis have been made and results have been compared.

1. GİRİŞ

Mikroelektronik teknolojileri askeri, endüstriyel, havacılık, uzay ve iletişim alanlarında geniş bir kullanım alanına sahiptir ve gün geçtikçe gelişmektedir. Performansın artırılması ve maliyetin düşürülmesi için yarıiletken teknolojilerinde eleman boyutları gittikçe küçültülmektedir. Tümleşik devrelerin ölçeklenmesi transistörlerin daha yüksek yoğunlukla dolgulanmasına ve bunun sonucu olarak radyasyon gibi çevresel koşullardan daha fazla etkilenmelerine yol açmaktadır [1]. Bu da sistemlerin güvenilir olarak çalışmasını etkilemektedir. Bu etkiyi en aza indirmek ve çevresel etkilerle oluşabilecek sorunlara rağmen sistemlerin işlevselliğini düzgün olarak devam ettirebilmek için elektronik devreleri güçlendirmeye yönelik tasarımların yapılmasının gerekliliği kaçınılmaz olmaktadır.

Metal oksit yarıiletken transistörler yüksek radyasyona maruz kaldıklarında elektron delik çiftleri oluştururlar [2]. Transistör kaynak ve difüzyon düğümlerinin topladığı lojik durum değerini değiştiren minimum yük değeri teknolojideki boyutların küçülmesiyle azalır. Böylece oluşabilecek hata olasılığı artar [3] [4].

Gömülü mikroişlemcilerin düşük geometrili teknolojilerde gerçeklenmesiyle mikroişlemcili sistemlerin çevresel koşullardan etkilenme olasılıklarını arttırmaktadır [5] [6]. Otomotiv sektörü, uzay ve tıp alanları gibi güvenliği kritik uygulamalar sıklıkla kullanıldığı için bu hataların oluşturabileceği problemler önem kazanmaktadır [7]. Kırmık üstü sistemlerdeki (System on Chip-SoC) mikroişlemciler kritik algoritmaları işletirler ve sistemdeki diğer elemanların uyumlu çalışması için gerekli iletişimi sağlarlar. Mikroişlemcilerde oluşacak herhangi bir hatanın beklenen çıktıyı vermemesi halinde tüm sistemi etkilemesi olasılığı çok yüksektir [8].

Kaydedici dosyaları mikroişlemci içerisinde çok sık erişilen ve güç tüketen bir birimdir. Ayrıca kaydedicilere gelebilecek hata sistemin diğer kısımlarına yayılabilir [9] [10] [11]. Bu sebeplerden dolayı tez kapsamında hataların oluşması durumunda mikroişlemcinin güvenilir şekilde çalışmasına devam etmesi için kaydedici

dosyalarındaki bellek hücrelerine yönelik literatürdeki tekniklerin yanısıra yeni bir devre bağışıklığını arttırma yöntemi geliştirilmiştir.

1.1 Genel Kavramlar

Hataların oranlarının hesaplanması, etkilerinin incelenmesinde çeşitli parametreler mevcuttur. Bu parametreler:

Hata Yoğunluğu (Fault Density): Cihazda bulunana her birim veri için hata sayısı ölçüsüdür. Bellekler için her megabayt ya da gigabayt verideki hata olarak ifade edilir. Bu parametre kalıcı hatalar için kullanılmaktadır [12].

Bozulma oranı (Failure Rate): Birim zamanda beklenen bozulmaların sayısı olarak tanımlanır. λ ile ifade edilir. Örneğin her 1000 saatte, bir işlemcinin işlevini yerine getiremezse, bozulma oranı $\lambda = 1/1000$ başarısızlık/saat olur.

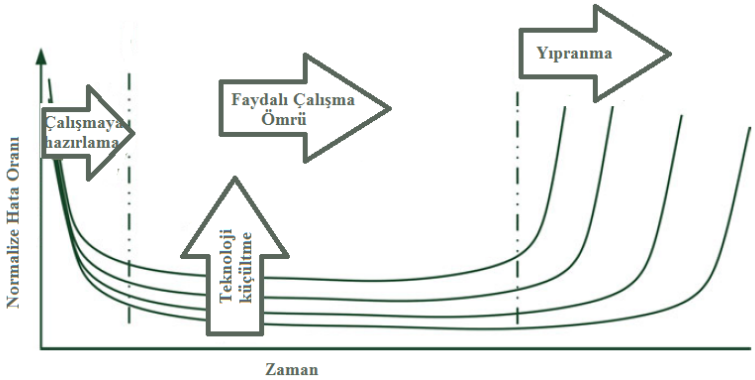
Bozulma oranı genel olarak tüm sistem için değil, eleman seviyesinde mevcuttur. Profesyonel kuruluşlar, sıklıkla kullanılan elemanların (diyot, anahtar, kapı, iki kararlı devre vb.) bozulma oranı kestirimlerini toplayıp yayınlamaktadırlar. Aynı zamanda yeni bir sistemin tasarımı standart elemanların yeni düzenlenişini içerir. Eleman bozulma oranı mevcut olduğunda yedeksiz bir sistemin tahmini bozulma oranı hesabı elemanlarınkinin toplanmasıyla elde edilebilir $\lambda = \sum_{i=1}^{n} \lambda_i$.

Bozulma oranı zamanın bir fonksiyonu olarak değişmektedir. Bir sistemin tipik bozulma oranı ölçümü çalışmaya hazırlama (burn-in), kullanılabilir (useful life) , yıpranma (wear-out) olarak 3 fazda sınıflandırılır. Bozulma oranı üretim hatası olan elemanların testlerle tespit edilmesiyle azalır, belirli bir zaman aralığında sabit kalır ve elektronik ve mekanik elemanların yıpranmasıyla tekrar yükselir [13].

Bozulma Ortalama Zamanı (Mean Time to Failure - MTTF): Elektronik bir sistemde hatanın ilk oluşumuna kadar olan ortalama zamandır. İstatistik bir değerdir ve genellikle milyon saat birimiyle ifade edilir. Hata ortalama zamanı, hata oranı sabit sistemler için hata enjekte zamanının tersidir ($MTTF = 1/\lambda$) [14].

Zamanla Bozulma Oranı(Failure In Time - FIT): 10^9 saatteki bozulma oranı ölçüsüdür. Yarıiletken teknolojisinde duyarlılık FIT/Cihaz olarak verilir. Bazı kaynaklarda bu parametre $\lambda_{FIT} = \lambda x 10^9$ olarak tanımlanır.

Onarım Ortalama Zamanı (Mean Time to Repair - MTTR): Hata oluşumu ve onarımı arasında geçen süredir. Sistemin çalışma moduna geri dönmesi örnek



Şekil 1.1: Zamana bağlı bozulma oranı

verilebilir.

Bozulmalar Arası Ortalama Zaman (Mean Time Between Failures - MTBF): Elektronik sistemdeki bir elemana başka bir hata gelmesine kadar geçen süredir. Hatalar arası ortalama zaman, hata ortalama zamanından farklı olarak onarım ortalama zamanını da içermektedir ($MTBF = MTTF + MTTR$).

Geçici Hata Oranı (Soft Error Rate - SER): Birim zamanda birim veride geçici hatanın oluşma olasılığıdır. Elektronik eleman ya da sisteme kullanılabilir fazda çevresel etkilerle rastgele oluşturulan hataların ölçüsüdür. Bazı kaynaklarda parametresi λ'dır ve birimi FIT'tir.

Güvenilirlik (Reliability): Bir sistemin verilen zaman aralığında düzgün olarak çalışmasına devam etme olasılığı olarak tanımlanır. Bir sistemin kesintiye uğramamak şartıyla doğru performansı göstermesinin bir ölçüsüdür. Verilen bir bozulma oranında güvenilirlik $R(t) = e^{-\lambda t}$ olarak ifade edilir.

Sabit bir zamanda geçici hata oranı değeri değiştirilerek de güvenilirlik hesaplanabilir.

Şekil 1.1 teknoloji küçültmesini vurgulayarak cihaz hata oranlarını göstermektedir. Müşteriye dağıtılan ürünlerde oluşan bozulmalar ağırlıklı olarak geçici hatalardan kaynaklanır. Çalışma ömrü içerisindeki hata oranı bu periyot içerisinde ortamda enjekte edilen hatalar tarafından etkilenir.

1.2 Tez Planı

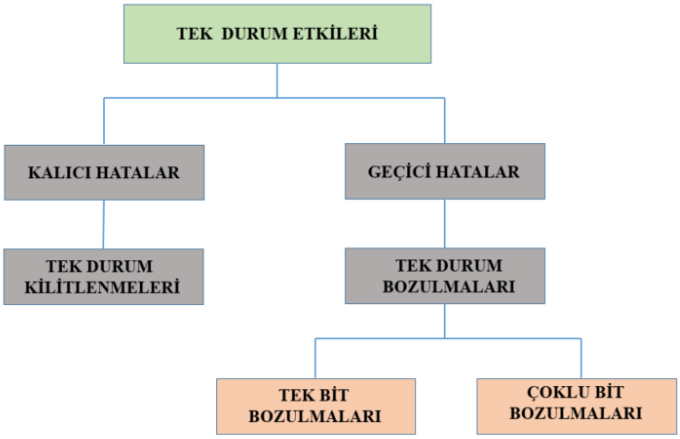
Bu tez kapsamında Bölüm 2'de kalıcı ve geçici hata türleri, bu hatalara neden olan faktörler ve güvenilirlik kavramı açıklanmıştır. Bölüm 3'de hatanın devre içerisine verilmesi için tasarlanan hata üretim devresi, test edilen tasarım içerisinde yapılan

değişiklikler ve bunların bir sistem haline getirilmesinin ardından yapılan testlerden çıkarılan sonuçlar anlatılmıştır. Bölüm 4'de hataya karşı devre bağışıklığı yöntemleri incelenmiş ve yeni bir yöntem önerilip güvenilirlik ölçümleri için formüller verilmiştir. Bölüm 5'de mikroişlemcinin hatalar varken de çalışmaya devam etmesi için açıklanan hata bağışıklığı kazandırma yöntemleri, tek döngülü ve iş hatlı mikroişlemci mikroişlemcinin kaydedici dosyalarına uygulanmıştır. Bölüm 6'da yapılan simülasyon sonuçları, maliyet ve güvenilirlik analizi karşılaştırmaları verilmiştir. Son olarak Bölüm 7'de tez çalışması özetlenmiştir.

2. HATA TÜRLERİ VE GÜVENİLİRLİK

Elektronik sistemlerde hatalar devrelerde geri dönüştürülemez fiziksel kusurların yol açtığı ve devredeki ilgili eleman değiştirilmediği sürece hep aktif olan kalıcı hatalar ile radyasyon etkileri ile oluşan kısa bir zaman sürecinde aktif olan geçici hatalar olarak iki sınıfa ayrılmaktadır.

Tek durum etkileri (Single Event Effects - SEE) yüksek enerjili partiküllerin tümleşik devrelerdeki devre elemanlarıyla etkileşmesi sonucu oluşur [15] [16]. Geçici ve kalıcı hatalar tek durum etkilerinin alt kümesidir. Şekil 2.1'de bu sınıflandırma gösterilmektedir.



Şekil 2.1: Tek Durum Etkileri

2.1 Kalıcı Hata

Cihazdaki bir veya daha fazla elemandaki kalıcı hasardır. Bu hatalara genellikle fiziksel kusurlar sebep olur. Elektronik bir sistemdeki kısa devreler, bağlantılardaki kopmalar veya bellekteki bir bitin aynı değere takılı kalması örnek olarak verilebilir. Kalıcı hatalara üretim hataları, yaşlanma, radyasyon gibi etkiler sebep olabilmektedir. Kalıcı hatalar devrede kalıcı bozulmalara yol açabileceği için hatalı çipler üretim sonrası testler sonucunda atılırlar ya da onarılırlar.

Devre yıpranması, ters kutuplu sıcaklık kararsızlığı (negative bias temperature instability - NBTI) ve elektriksel parametre kaymasına sebep olan rastgele telgraf gürültüsü (random telgraph noise - RTN) sonucu oluşur ve önemli ölçüde devrenin işlevsel sürecindeki performansını azaltır [17].

2.1.1 Bozulma Oranı Ölçümü

Elektronik elemanların bozulma oranlarıyla ilgili bilgi ticari ya da askeri kaynaklardan bulunabilmektedir. MIL-HDBK-217 gibi askeri standartlar, NTT gibi ticari standartlar mevcuttur [18].

Bozulma oranı ölçümleri çevre, basınç gibi faktörlerin kullanıldığı karmaşık modellere dayanır [19]. Elektronik bir elemana test ortamında sıcaklık, akım, gerilim yüklemeleri yapılarak hızlandırma testleri uygulanır. Bozulma oranı Chi^2 dağılımı kullanılarak hesaplanır. Aktivasyon enerjisi, sıcaklık, Boltzman sabiti parametreleri kullanılarak hesaplanan hızlandırma faktörü (accaleration rate) ile ters orantılıdır [20].

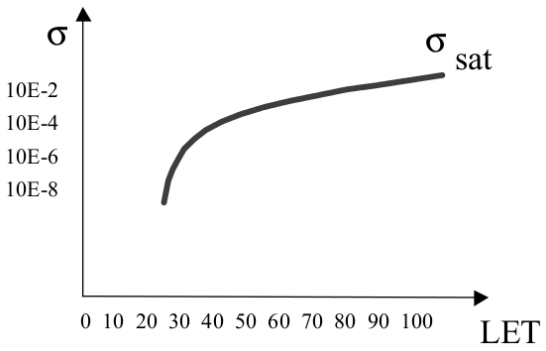
2.2 Geçici Hata

Geçici hatalar veri bozulması ile oluşan durumlardır ancak cihazın kendisi kalıcı olarak hasar görmez. Geçici hatalar uygulamaya göre farklı etkilere sahiptir. Belleklerdeki baskın hata türüdür. Bir taraftan sistem seviyesinde tespit edilebilen veya tespit edilemeyen veri bozulmalarına yol açabilir. Diğer taraftan bir devrenin arızalı çalışmasına hatta sistemin çökmesine sebep olabilir.

2.2.1 Radyasyon Etkileri

Radyasyon ortamı güneş aktivitesi tarafından üretilen çeşitli partiküllerden meydana gelmektedir. Bu partiküller iki ana türde sınıflandırılır:(1) elektron, proton, ağır iyonlar gibi yüklü partiküller ve (2) χ, gama ve ultraviyole ışınları gibi elektromanyetik radyasyon (foton). Yüklü partiküller silikon atomları ile etkileşime geçtiklerinde atomik elektronların uyarımı ve iyonlaşmasına neden olurlar.

Bir ağır iyon silikona çarpığında, serbest elektron-delik çiftleri oluşturarak enerjisini kaybeder. Protonlar ve nötronlar, materyal üzerinden geçtiğinde nükleer reaksiyona



Şekil 2.2: LET-arakesit değişim eğrisi

neden olabilir. Bu iyonizasyon devrede bozulmaya sebep olan geçici akım vurusu ile modellenen yük birikimi oluşturur.

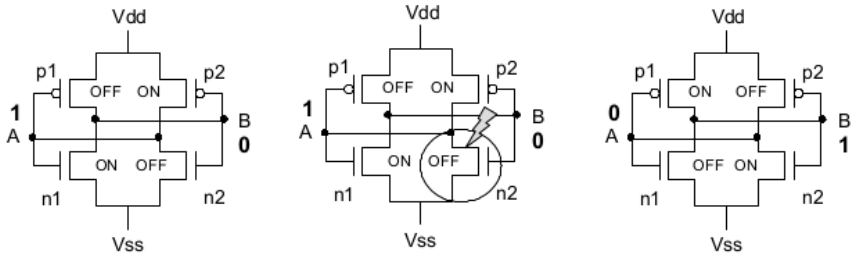
Farklı radyasyon kaynakları, farklı yük birikim dalgaformları gösterirler. Ayrıca, bu dalgaformları katkılama profili gibi teknolojik parametreler kadar bölge ve geliş açısına da bağlıdır. Yük birikim mekanizması genellikle partikül çarpma bölgesinde çift üstel akım vurusu ile modellenir.

Bir materyaldeki radyasyon etkisi enerji ve partikül akısı ile ölçülür. Bir cihaza aktarılan enerji Lineer Enerji Transferi(LET) olarak adlandırılır ve birim başına artan enerji ile ölçülür $(MeV/(mg/cm^2))$. Tek durum bozulmalarına (single event upset - SEU) neden olan minimum LET'e LET eşiği denmektedir.

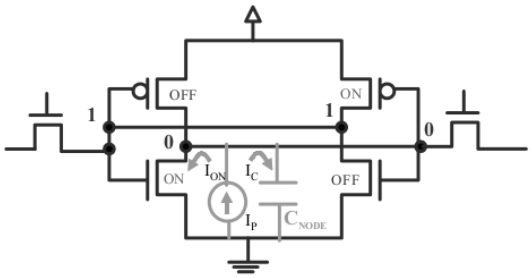
Bozulma miktarı ve partikül geçiş miktarının bilinmesiyle bir partikülün sebep olacağı bozulma olasılığı hesaplanabilir. Bozulma miktarının , her cm^2 başına düşen partikül sayısına bölünmesiyle elde edilen değer parçanın arakesiti olarak tanımlanır ve birimi $cm^2/cihaz$'dır. Sonuç olarak, bir cihazın duyarlılığı LET'e bağlı olarak arakesit fonksiyonu (σ) ile ölçülür.

Şekil 2.2'deki eğri incelendiğinde 25 MeV'nin altındaki LET için hata oluşmaz. 25 MeV'de 100.000.000'dan fazla partikül bir bozulmayı tetiklemek için devrenin duyarlı bölgesine geçmelidir. 50 MeV için her sn'de 10.000 partikül gerekirken 100 MeV için 100 partikül akısı bir bozulmayı tetiklemek için yeterlidir.

Kozmik ışınlardan gelen alfa partikülleri, yüksek enerji ve termal nötronlar SEU'ların oluşmasına etki ederler. Uzayda bulunan yüksek enerjili kozmik ışınlar uydu elektroniğinin ve uzay görevlerinin güvenilir olarak çalışmasına tehdit oluşturur. Çünkü atmosfere nüfuz eden kozmik ışınlar ikincil nötron, proton, elektron ve diğer



Şekil 2.3: SRAM bellek hücresine partikül çarpması



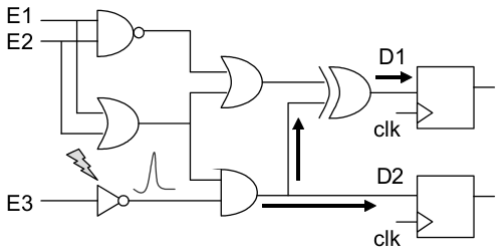
Şekil 2.4: SRAM bellek hücresindeki SEU etkisi

alt-atomik parçacıkların art arda sıralanmasına yol açar. Bunlardan bazıları yeryüzü seviyesine ulaşarak yarıiletken maddeden geçer ve bu nükleer etkileşim ile lojik durumu değiştirecek yük birikimine sebebiyet verir [21]. Yüksek enerjili nötronlar zemin seviyesinde SEU'ların baskın kaynağıdır. Boron yarı iletken elemanlarda sıklıkla kullanılan katkılama maddesidir. Termal nötronar bor atomları ile etkileşime geçtiklerinde ve özellikle dielektrik katmanlarında kullanılan borfosforsilikon camı (BPSG) bozulma oranını arttırır.

2.2.2 Radyasyonun Tümleşik Devreler Üzerindeki Etkileri

Bir partikül silikon içerisindeki kombinezonsal ya da ardışıl lojiğe çarpabilir. Şekil 2.3'de tipik devre yapısı gösterilmiştir. SEU etkisi ise Şekil 2.4'de gösterilmiştir. Bellek hücreleri 0 veya 1 değeri tutabilen iki kararlı duruma sahiptir. Her durumda iki transistör açık diğer ikisi kapalıdır. Yüklü bir partikül kapalı durumdaki transistörün savağı gibi bir bellek hücresinin duyarlı noktalarına çarptığında, geçici akım vurusu üretir ve tümler transistorün kapısını etkinleştirir. Bu etki tutulan değerin evrilmesine yani bellek hücresinde bit değişimine yol açar.

Yüklü partikül kombinezonsal lojik bloğa çarptığında, geçici akım vurusu oluşturur. Bu durum tek geçici etki (single transient effect-SET) olarak adlandırılır. Lojiğin hızına bağlı olarak SET Şekil 2.5'de ikinci tutucuda gerçek bir veri gibi saklanabilir ve



Şekil 2.5: Kombinezonsal devredeki SET etkisi

SEU'ya yol açar. SET, lojik çıkış yelpazesine bağlı olarak çoklu geçici akım vuruları oluşturup çoklu bit bozulmalarına (multiple bit upsets-MBU) sebep olabilir.

2.2.3 Tek Durum Bozulmaları Sınıflandırılması

Aynı zamanda gelen bozulma sayısına göre geçici hatalar sınıflandırılabilir. SEU birinci derece etki olarak sınıflandırılabilirken, MBU ikinci yada üçüncü derece etki sınıflarına girer.

Üç çeşit MBU vardır. İlki bir partikülün iki ayrı bellek hücresindeki ardışık iki duyarlı düğüme çarptığında meydana gelir. Bu tip bozulmalar ikinci derece etki olarak sınıflandırılır ve bellek hücrelerinin özel olarak yerleşimi ya da bellekteki verilerin birbirinden yeterince uzak yerleştirilmesi gibi yöntemlerle önlenebilir.

İkinci tip MBU bir partikülün aynı bellek hücresindeki iki ardışık düğüme çarpmasıyla oluşur ve üçüncü derece etki sınıfına girer. Bu durum fiziksel serimde kritik düğüm jonksiyonlarını geniş uzaklıklar ile ayırarak ve her bir jonksiyon alanının birbirini görecek şekilde dizerek azaltılabilir.

Üçüncü tip MBU, birden çok partikülün çoklu bellek hücrelerinin duyarlı noktalarına çarpmasıyla meydana gelir. Ardışık bellek hücrelerindeki çoklu bozulmaların çoğunluğuna tek bir partikül sebep olur. Birden fazla yüklü partikülün bir saniyeden az bir periyotta ardışık hücrelerle etkileşime geçmesi olasılığı düşüktür [22] [23].

2.2.4 Geçici Hata Oranı Ölçümü

Elektronik bir cihazın geçici hata hassasiyetini ölçmek için temel parametre cihazın partikül çarpması sonucu bozulmasına sebep olan minimum yükü ifade eden kritik yüktür (Q_{kritik}). Kritik yük genellikle SPICE gibi tümleşik devre simülatörleri kullanılarak hesaplanır. Anlık darbeler cihazın duyarlı noktalarına enjekte edilir. Bu

darbeler alfa partikülü ya da nötron çarpmasıyla oluşmuş elektron-delik çiftlerinden üretilen akımı ifade eder. Yük değeri **2.1** ile hesaplanır.

$$Q_{yuk} = CxV_{dd}^2 \tag{2.1}$$

Eşitlikte C kapasite V_{dd} besleme gerilimidir. Yük ile besleme gerilimi arasında karesel bir ilişki olduğundan, bir cihazın geçici hata hassasiyeti gerilim ile karesel olarak artar. Örneğin 0.6 μm teknolojisindeki SRAM bellek hücresinde besleme gerilimini $3.3V$'tan $2.2V$'ta düşürmek Q_{kritik}'i $91.4fC$'den $51.5fC$'ye düşürür. Besleme gerilimini azaltarak yapılan güç tasarrufu yöntemlerinde yüksek geçici hata oranı kaçınılmaz olmaktadır [24].

2.2.5 Güvenilirlik Ölçümü

Matematiksel olarak güvenilirlik $R(t)$ $[0-t]$ zaman aralığında sistemin düzgün olarak çalışmaya devam etme olasılığıdır. $P(t)$ ise verilen zaman aralığındaki bozulma olasılığıdır ve aralarındaki ilişki **2.2** şeklindedir.

$$R(t) + P(t) = 1 \tag{2.2}$$

Hataların Poisson dağılımı ile oluştuğu ve bit bozulmalarının istatistiksel olarak birbirinden bağımsız olduğu varsayıldığında bir sistemin güvenilirliği **2.3** eşitliği ile modellenir [25].

$$R_{sistem}(t) = \sum_{i=0}^{N} \binom{n}{i} .(1-R(t))^i .R(t)^{n-i} \Rightarrow R_i(t) = R(t) \tag{2.3}$$

Eğer sistem sabit bozulma oranına sahipse yani faydalı çalışma ömrü periyodu içerisindeyse $R(t) = e^{-\lambda t}$ olarak ifade edilir ve eşitlik **2.4** şeklinde olmaktadır.

$$R_{sistem}(t) = \sum_{i=0}^{N} \binom{n}{i} .(1-e^{-\lambda t})^i .e^{(-\lambda t)*(n-i)} \Rightarrow e^{-\lambda t} = e^{-\lambda t} \tag{2.4}$$

Ayrıca sabit bir zamanda geçici hata oranı (GHO) değeri değiştirilerek de güvenilirlik hesabı yapılabilir.

3. HATANIN DEVRE İÇİNE VERİLMESİ

Hatanın devre içerisine verilmesi bir sistemin güvenilirliğini test etme tekniğidir. Bu teknik hatalar mevcut iken sistem davranışını gözlemeye dayanır.

Yapay hatalar sisteme verilerek, sistemin davranışı gözlemlenir. Böylece hatanın oluşması ve yayılmasının sistem performansı üzerindeki etkileri gözlemlenebilir. Bu işlemler simülasyonlarla ya da çalışan prototipler test edilerek yapılabilir. Ancak bir sistemin hataya karşı bağışıklığının kalitesini ölçmek için kullanılması daha faydalıdır.

Bir sisteme hata verme teknikleri üç ana sınıfa ayrılır [26]:

- **Donanım tabanlı hata verme tekniği:** Ağır iyon radyasyonu veya elektromagnetik bozucuların olduğu bir ortam oluşturarak ya da tümleşik devrelerin pin değerleri değiştirilerek donanıma fiziksel seviyede müdahale edilir
- **Yazılım tabanlı hata verme tekniği:** Bu yöntem uygulamalar ve işletim sistemlerine yöneliktir harici bir donanım gerektirmez. Kaynak kodunun değiştirilmesini gerektirir.
- **Simülasyon tabanlı hata verme tekniği:** Donanım tanımlama dilleri ile tasarlanan yöntemdir. Sistemin simülasyon modeli için testler yazılır. Gerçek sisteme harici girişler eklenmemektedir.

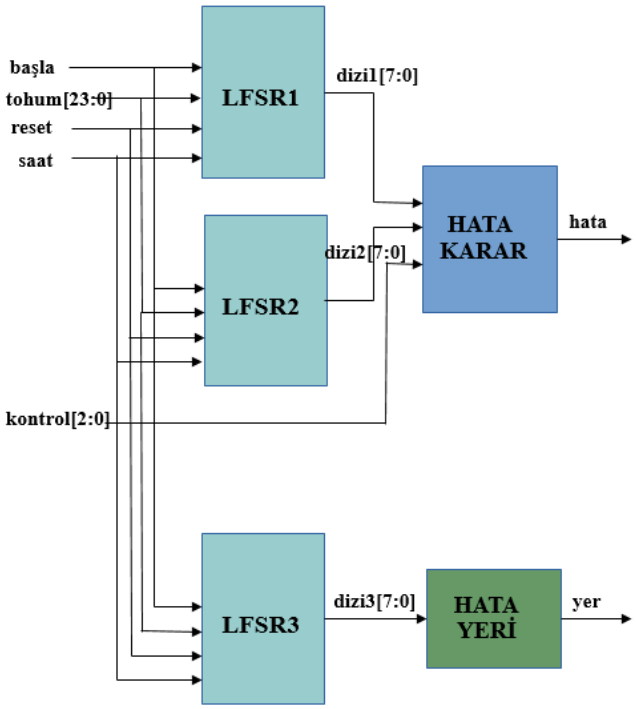
Hata sisteme verildiğinde devre davranışını ya da program çıktısını etkilemesi hatanın geldiği yere, türüne ve süresine bağlıdır.

3.1 Hata Üretim Devresi Tasarımı

Tasarlanan hata üretim devresi hem donanım hem de simülasyon tabanlı yöntemlerin özelliklerini içermektedir. Kontrolü, kullanım kolaylığı ve değişik tasarımlara uygulanabilirliği sebebiyle [27]'deki çalışmadan temel alınarak devre tasarlanmıştır. Bu devre doğrusal geribeslemeli ötelemeli kaydedici (linear feedback shift register-LFSR), hata karar ve hata yeri bloklarından oluşan devre Şekil 3.1'de gösterilmiştir.

Algorithm 1 Hata Üretim Algoritması

if $reset = 1$ **then**
  $dizi_{1,2,3} = 0$
else if $basla = 1$ **then**
  $dizi_{1,2,3} = tohum$
else
  $dizi_{1,2,3}[0] = \sim dizi_{1,2,3}[1] \oplus dizi_{1,2,3}[2] \oplus dizi_{1,2,3}[3] \oplus dizi_{1,2,3}[7]$
  $dizi_{1,2,3}[7:1] = dizi_{1,2,3}[6:0]$
end if
if $kontrol = 0$ **then**
  $hata = 0$
else if $kontrol = 1$ **then**
  if $dizi_1[0] = dizi_2[0]$ **then**
    $hata = 1$
  else
    $hata = 0$
  end if
else if $kontrol = 2$ **then**
  if $dizi_1[1:0] = dizi_2[1:0]$ **then**
    $hata = 1$
  else
    $hata = 0$
  end if
else if $kontrol = 3$ **then**
  if $dizi_1[2:0] = dizi_2[2:0]$ **then**
    $hata = 1$
  else
    $hata = 0$
  end if
  —
  —
  —
else if $kontrol = 6$ **then**
  if $dizi_1[5:0] = dizi_2[5:0]$ **then**
    $hata = 1$
  else
    $hata = 0$
  end if
else if $kontrol = 7$ **then**
  $hata = 1$
end if
$yer = dizi_3$



Şekil 3.1: Hata Üretim Devresi Blok Diyagramı

Hata Algoritma 1'de verilen şekilde üretilmektedir.

3.1.1 Doğrusal Geribeslemeli Ötelemeli Kaydedici Blokları

Hata üretim sisteminin ana özelliği bir devreye istenilen sıklıkta hata verebilmesidir. Bu amaçla hata enjekte sistemi sözde rastgele sayı dizilerinden faydalanmaktadır. Sözde rastgele sayı dizileri doğrusal geribeslemeli ötelemeli kaydediciler ile üretilir. İki adet 8-bitlik LFSR paralel çalışarak ürettikleri sayılar karşılaştırılır ve böylece hata sinyalinin oluşturulup oluşturulmayacağıa karar verilir. Üçüncü LFSR tercihe göre eklenir ve hata yeri belirlemede kullanılır.

3.1.2 Hata Karar Bloğu

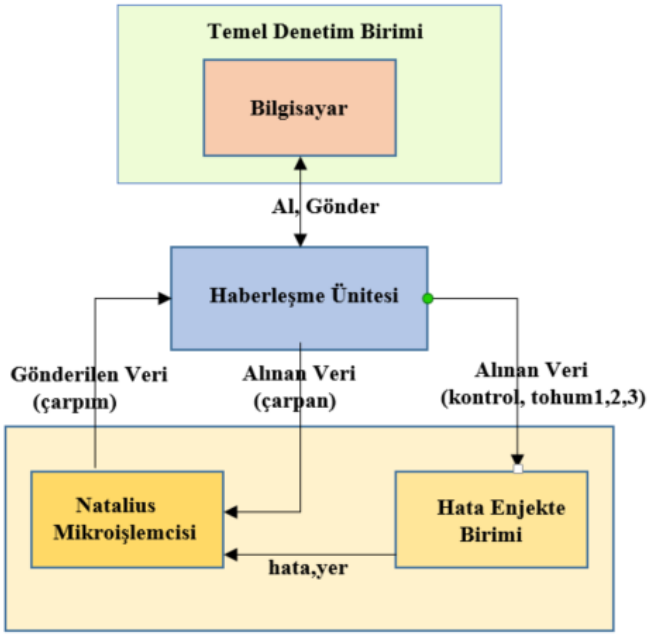
Karar devresi dışarıdan kullanıcının girdiği 3-bitlik konrol girişine göre belirlenmektedir. Kontrol girişi "000" seçilirse devre hatasız, "111" seçilirse kalıcı hatalı olarak çalışmaktadır. Bu iki değer arasında kalan değerler seçildiğinde geçici hata oluşmaktadır. Kontrol girişinin değerine göre rastgele sayılardan gelen verilerin karşılaştırılacak bit sayısı belirlenir. Karşılaştırılan veriler eşit olduğunda hata oluşur. Bit sayılarının artması eşit olma olasılıklarını düşüreceğinden hata oluşması $^1/_2$ oranında azalır.

3.1.3 Hata Yeri Bloğu

Bu blok tercihen devrede bulundurulabilir. Girişi LFSR3 tarafından üretilen 8-bitlik sayıdır. Eğer hata bellekte bir hücreye verilecekse bu sayı satır ve sütun olarak bölünüp koordinat belirtilebilir.

3.2 Hatanın Devre İçerisine Verilmesi İçin Deneysel Bir Sistem Oluşturulması

Tasarlanan hata üretim devresinin kullanılabilirliğini göstermek ve hatalar mevcut olduğunda devre davranışını gözlemlemek amacıyla bir tasarım seçilerek içerisine hata verilerek ve sonuçların toplanacağı deneysel br sistem oluşturulmuştur.



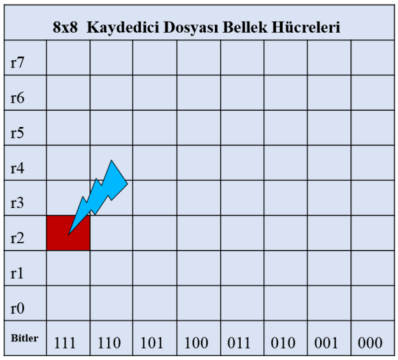
Şekil 3.2: Hata Enjekte Sistemi

3.2.1 Test Edilen Tasarım

Bu sistemdeki test edilen tasarım gömülü bir mikroişlemci olan, Verilog donanım tanımlama dili ile tasarlanıp Sahada Programlanabilir Kapı Dizileri (Field Programmable Gate Array - FPGA) üzerinde gerçeklenmiş Natalius adında küçük boyutlu bir mikroişlemcidir. Natalius 8-bit aritmetik ve lojik birim (arithmetic and logic unit - ALU), 8x8 genel amaçlı kaydedici, 8-bit adres portu, sıfır ve elde bayrağı, 16x11 yığın bellek içermektedir. 2048 adet 16-bitlik komut depolar ve her bir komut 3 saat periyodu ile çalışır.

3.2.1.1 Devre İçerisine Hata Verme Noktaları

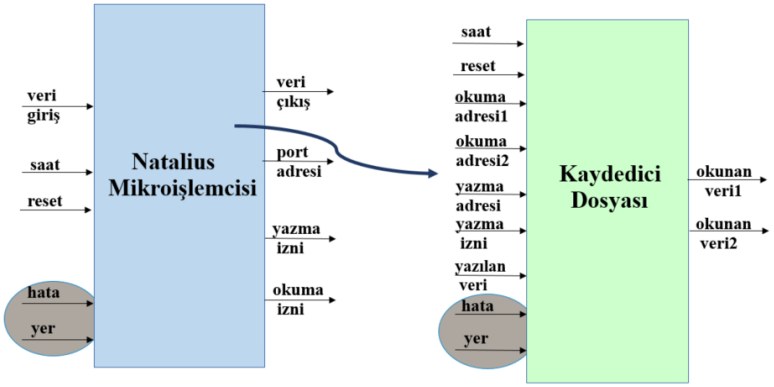
Hata, aritmetik ve mantıksal işlemlerin yapıldığı ve içerisinde oluşabilecek hatanın tüm sistemi yani program çıktılarını etkilemesi ve hataya karşı bağışıklık kazandırma yöntemi tasarımı yapılırken bu birime odaklanılması sebebiyle mikroişlemcinin kaydedici dosyasındaki bellek hücrelerine verilmektedir. Şekil 3.3'de tek durum etkilerinin veya yıpranmanın sebep olduğu bellek hücresine gelen hata gösterilmiştir.



Şekil 3.3: Kaydedici dosyasındaki bir bite hata gelmesi

Mikroişlemci ve içerisindeki kaydedici dosyasına hata enjekte etmek için gerekli değişiklikler yapılmalıdır. Hata girişlerinin eklenmiş olduğu mikroişlemci ve içerisindeki kaydedici dosyası Şekil 3.4'de gösterilmiştir.

Hata girişleri eklendikten sonra kaydedici dosyası tasarımının içerisinde de uygun değişiklikler yapılmalıdır. Algoritma 2'de bu yapılan tasarım değişiklikleri açıklanmaktadır.



Şekil 3.4: Hata Girişleri Eklenmiş Mikroişlemci ve Kaydedici Dosyası

3.2.1.2 Uygulama kodu

Mikroişlemci için uygulama kodu olarak En Anlamlı Bit İlk Çarpıcı Algoritması seçilmiştir. Algoritma 3'de gösterilen çarpım işlemi topla ve kaydır işlemlerine

Algorithm 2 Kaydedici dosyasının bellek hücrelerine hata enjekte etmek için tasarımda yapılan değişiklikler

```
if hata = 1 then
    if yazmaizni = 1 then
        if yazmaadresi = hatayeri then
            bellek[yazmaadresi] = hataliyazilanveri
        else
            bellek[yazmaadresi] = yazilanveri
            bellek[hatayeri] = hataliveri
        end if
    else
        bellek[yazmaadresi] = hataliveri
    end if
else
    if yazmaizni = 1 then
        bellek[yazmaadresi] = yazilanveri
    end if
end if
okunanveri1 = bellek[okumaadresi1]
okunanveri2 = bellek[okumaadresi2]
```

dayanmaktadır. C döngü aracılığıyla üzerinde işlem yapılan ve çarpım sonucunu veren değişkendir. A ve B çarpılacak sayılardır. Döngü içerisinde her seferinde C iki katına çıkarılır ve B'nin ilgili biti 1 değerinde ise C sayısına A çarpanı eklenir. Mikroişlemcide her işlem yazmaçlar aracılığı ile yapıldığından ve yazmaçların 8-bitlik olması sebebiyle 8 adet döngü gerçekleşmektedir.

Algorithm 3 En Anlamlı Bit İlk Çarpıcı Algoritması

Require: $A, B = (b_{n-1}, b_{n-2}, \ldots, b_1, b_0)_2$ pozitif tam sayılar
Ensure: $C = A \times B$

```
C = 0
for i = n − 1 : 0 do
    C = 2 × C
    if b_i = 1 then
        C = C + A
    end if
end for
```

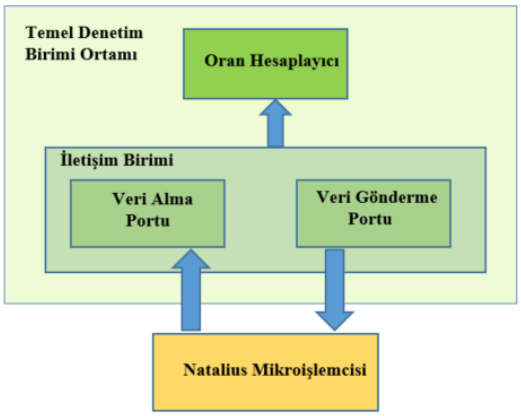
Komut kümesi kullanılarak yazılmış kod hatanın etkilerini gösterecek grafiklerde anlaşılır olması açısından önemlidir. Buna göre $r1$ ve $r2$ yazmaçları algoritmadaki A ve B çarpanlarına karşılık gelmekte, port adresine göre değerleri dışarıdan işlemciye giriş olarak verilmektedir. $r3$ ise T'ye karşılık gelir, sonuç üretildiğinde karşılık gelen port adresine göre işlemci çıkışına verilir. Değeri 1 olan $r4$ ve başlangıçta 9 olan

*r*6 her seferinde *r*6'dan *r*4'ün çıkarılıp karşılaştırılmasıyla 8 adet döngüyü sağlarlar. Değeri 128 olan *r*5 ise *r*2'nin en anlamlı bitinin kontrol edilmesini sağlayan sabittir.

```
1  forever  csr square
2           jmp forever
3  square   ldm r1,1
4           stm r1,1
5           ldm r2,2
6           csr multsoft
7           stm r3,7
8           ret
9  multsoft ldi r3,0
10          ldi r4,1
11          ldi r6,9
12          ldi r5,128
13 multloop add r3,r3
14          ldi r7,0
15          oor r7,r2
16          oor r7,r5
17          cmp r2,r7
18          jpz addbit
19 continue add r2,r2
20          sub r6,r4
21          cmp r6,r4
22          jnz multloop
23          ret
24 addbit   add r3,r1
25          jmp continue
```

3.2.2 Temel Denetim Birimi

Temel Denetim Birimi (TDB) tasarım ile veri alışverişini gerçekleştirmek için haberleşme ünitesini yönetir ve gelen verinin analizinin yapılmasını sağlar. Sistemi 0'dan 255'e kadar olan 8-bitlik *N* sayısına bağlı olarak otomatik olarak çalıştırır. Çizelge 3.1'de görülebileceği üzere en anlamsız ilk 4-bit çarpan, sonraki 3-bit kontrol numarası ve en anlamlı bit ise hata türünü belirler.



Şekil 3.5: Temel Denetim Birimi

Çizelge 3.1: *N* bit grupları

7	6 5 4	3 2 1 0
tür	kontrol	çarpan
0	0..7	0...15
1	0..7	0...15

3.2.3 Haberleşme Ünitesi

Bu birim TDB ile tasarım arasındaki haberleşmeden sorumludur. Veri aktarımı Evrensel Asenkron Alıcı ve Verici (Universal Asynchronous Receiver and Transmitter - UART) protokolü ile yapılmaktadır ve hızı 9600 bps'dir. Bu birim için sonlu durum makinası tasarlanmıştır ve Şekil 3.6'de mekanizmanın akış şeması verilmiştir. 0'dan 7. duruma kadar TDB tarafından gönderilen LFSR bloklarının başlangıç tohumu, kontrol çarpan ve hata türü bilgileri alınır. 8-11 durumları arasında çarpım sonuçları beklenir ve ve TDB'ye gönderilir. Bu süreç belirlenen sayıda sonuç toplanana kadar devam eder ve durum makinesi yeni veri beklemek için başlangıç durumuna geri döner.

3.2.4 Analizler

Analizler bu çarpıcı algoritmasıyla bir sayının karesi alınarak yapılmıştır. Kaydediciler 8-bit genişliğinde olduğu ve sonuç değerinde taşma olmaması için karesi alınacak sayı 4-bitlik seçilmiştir. Tasarım Xilinx Spartan3s-500e kartında gerçeklenmiştir. Çarpıcı algoritması ilk sonucu 5940ns sonra üretmiştir. Çünkü bir komut 3 saat periyodunda işletilmektedir ve 1 saat periyodu 20ns'dir. Her bir N değeri için 3 farklı zamanda 256 adet ölçüm alınmıştır ve ölçüm sonuçları toplam 768 sonucun yüzdesi olmaktadır. Her bir ölçümün davranışsal simülasyon süresi $256x5.94\mu s = 1.52064ms$ olmaktadır. Hata türü olarak kaydedicideki bellek hücresindeki bit değeri 0 yapılmıştır.

3.2.4.1 Belli Bir Kaydedici Bitine Gelen Hata

Bu analizde hata yeri bloğu kullanılmamıştır. Hata $r2$ kaydedicisinin en anlamlı bitine verilmiştir. Çünkü bu bit her adımda kontrol edildiğinden bu uygulama açısından yeri kritiktir.

Çizelge 3.2'de $r2$ kaydedicisinin içerisinde bulunan verilerin 1 ve 0 sayılarına göre yapılan gruplama ve kontrol girişinin değerine göre kaydedicilerdeki verilerin hata

Şekil 3.6: Haberleşme Ünitesi Akış Şeması

oranları gösterilmiştir. Örneğin 15 sayısının ikili gösterimi "1111"dir ve 4 adet 1 içerir. Kaydedici 8-bitlik olduğundan verinin 50%'si 1 içermektedir. Hata oranı ise kontrol girişinin 1 olmasından dolayı 50%'dir. Yani bu veri için yazmaç ikisinin çarpımı sonucu çıkan 25%'lik oran ile hata yapar.

Şekil 3.7'de çarpım sonuçlarının kontrol değerlerine göre doğruluk yüzdeleri verilmiştir ve iki aşamada değerlendirilebilir. İlk olarak aynı veri grubunda bulunan çarpanlar hatadan farklı zamanlarda etkilenseler bile program çıktısı olan çarpım sonuçları benzer davranış gösterdiği için tek bir grafikte toplanmışlardır. Verigrubu 0'da yanlış sonuç alınmamaktadır çünkü zaten gelen hata türü 0'dır ve hata maskelenmektedir. Veri grubu 1'den 4'e kadar çarpandaki 1 sayısı arttığından çarpım sonucundaki doğruluk oranı azalmaktadır.

İkinci olarak, kontrol girişlerine bağlı olarak doğruluk oranı belirlenmektedir. 0 iken hata verilmemektedir ve tüm sonuçlar doğru çıkmaktadır, 7 iken kaydedicinin ilgili

Çizelge 3.2: 4-bitlik çarpanların içerisindeki 1-0 sayıları ve kontrol girişine bağlı olarak hatalı veri oranları

Veri Setleri	0	1,2,4,8	3,5,6,9,10,12	7,11,13,14	15
1-0 sayıları	0-8	1-7	2-6	3-5	4-4
Kontrol Girişleri	Kontrol değerine bağlı hatalı veri oranları				
1	50/0	43.75/6.25	37.5/12.5	31.25/18.75	25/25
2	25/0	21.88/3.13	18.75/6.25	15.62/9.38	12.5/12.5
3	12.5/0	10.94/1.56	9.38/3.12	7.81/4.69	6.25/6.25
4	6.25/0	5.46/0.76	4.69/1.56	3.90/2.34	3.12/3.12
5	3.125/0	2.73/0.38	2.34/0.78	1.95/1.17	1.56/1.56
6	1.565/0	1.37/0.19	1.17/0.39	0.97/0.58	0.78/0.78
7	100/0	87.5/12.5	75/25	62.5/37.5	50/50

Çizelge 3.3: Hata oranına bağlı olarak maksimum hatalı bit sayısı

Kontrol değeri	0	1	2	3	4	5	6	7
Maksimum hatalı bit sayısı	0	49	25	13	6	3	2	64

biti sürekli olarak 0'a takılı kalmaktadır ve hata bitinin değeri 0 olduğu için veri grubu 0 haricindekilerde doğru çarpım sonucu alınmamaktadır. Kontrol girişi 1-6 arasında geçici hatalar enjekte edilmektedir. Yani hata olmadığında gelen veri doğru bir şekilde kaydediciye yazılabilmektedir. Ancak kaydediciden hatalı veri okunduğunda bu tüm programa yayılmakta ve sonucun yanlış olmasına sebep olmaktadır. Özellikle hata oranı yüksek olan kontrol girişi 1 olduğunda doğru sonuç alınamamaktadır ve 2 için doğruluk yüzdesi çok az olmaktadır. 3'ten 6'ya kadar oran iyice azalacağından doğru sonuç sayısı artmaktadır.

3.2.4.2 Rastgele Kaydedici Bitlerine Gelen Hata

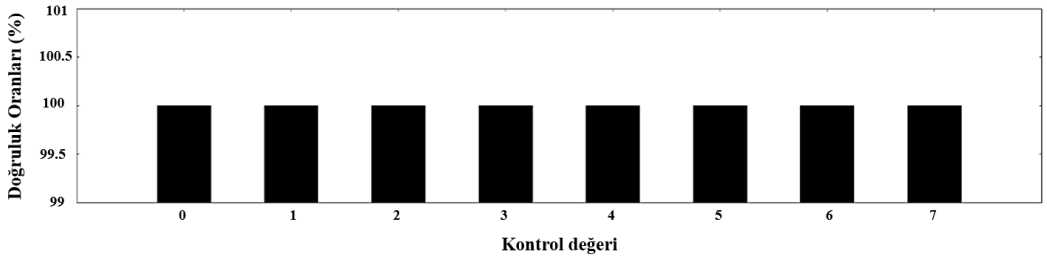
Bu analizde hata yeri devresi kullanılmıştır ve her komut döngüsünde hata sinyali aktifse rastgele bir kaydedicinin rastgele bitine hata verilmektedir yani program sonuna kadar çoklu hata oluşmaktadır. Hata değeri, enjekte yapılan kaydedicideki bit ile aynı ise ya da hata gelen kaydedicideki değer o anda okunmuyorsa ilgili komut döngüsünde hatadan etkilenilmez. Çizelge 3.3'de hata oranına bağlı olarak program sonunda oluşabilecek maksimum hatalı kaydedici bit sayısı verilmiştir. Örneğin; hata oranı 50% iken programda 99 komut işletildiğinden program sonuna kadar en fazla 49 tane bellek hücresine hata gelebilir.

Şekil 3.8'de aynı veri grubunda bulunan çarpan değerlerinin bu analizde farklı davranışlar gösterdiği görülmektedir.

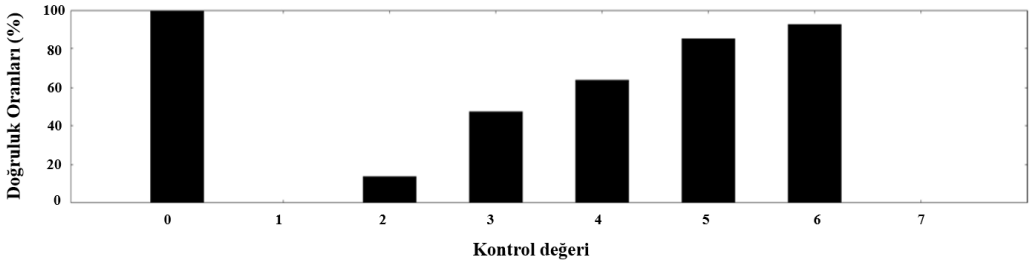
3.2.5 Sonuç ve Yorumlar

Tasarlanan hata enjekte sistemi ile ilk olarak kaydedici dosyasına yeri belli tek bitlik hata verilmiştir. Algoritmayı gerçekleyen uygulama kodunda en kritik kaydedici bitine hata verilmiş ve program çıktıları gözlemlenmiştir. İçerisinde aynı sayıda 1 ve 0 bulunduran değerlerin doğruluk oranları birbirine çok benzemektedir. Eğer hata daha az kritik olan rastgele bir bite verilirse geçici hatalar için sonuçların doğruluk oranları daha yüksek olacaktır. Ancak kalıcı hata için sonuç değişmeyecektir çünkü bu programda kaydedicilerin tamamı kullanılmıştır.

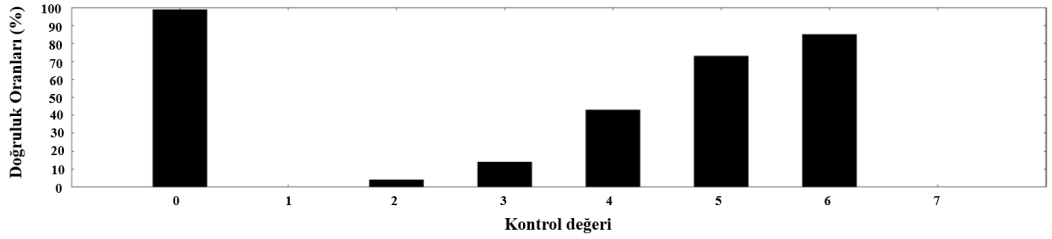
Hata yeri rastgele olarak verilen durumda aynı verisetinde bulunan değerler farklı oranlarda doğru sonuçlar vermektedir. Çünkü her komut döngüsünde eğer hata sinyali aktifse farklı kaydedicilerin farklı bitlerine hatalar gelmektedir ve bu da sonucun birden fazla bitteki hatadan etkilenmesine yol açmaktadır.



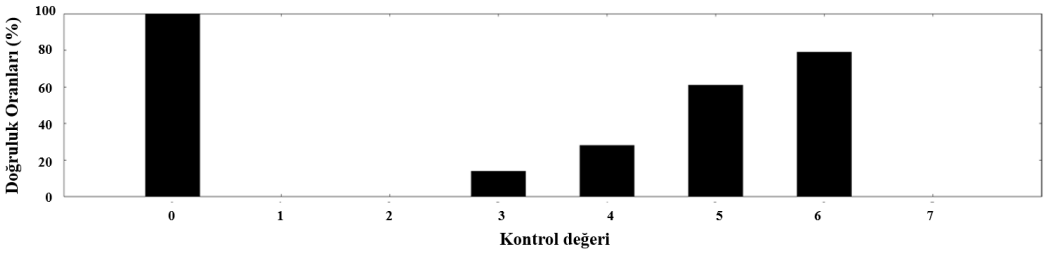
Veri Grubu 0



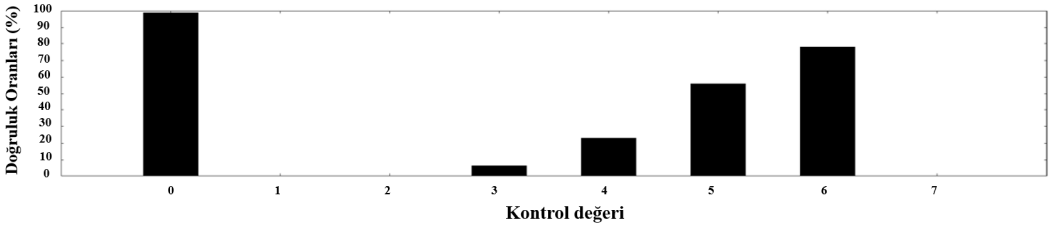
Veri Grubu 1



Veri Grubu 2

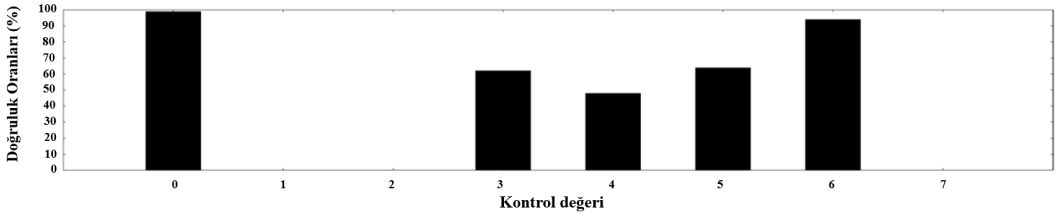


Veri Grubu 3

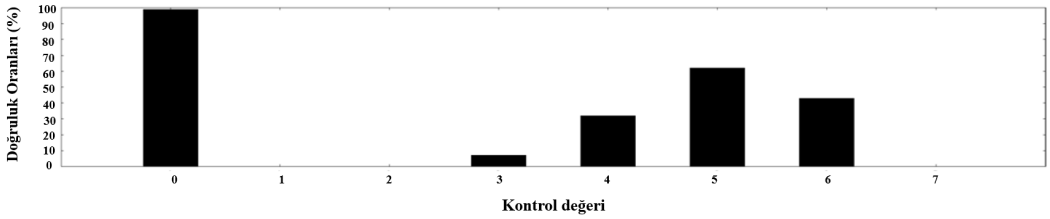


Veri Grubu 4

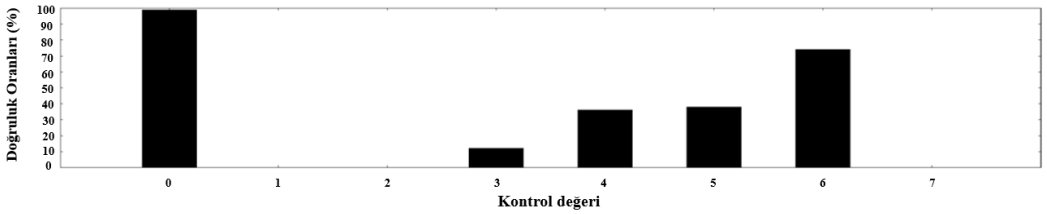
Şekil 3.7: Veri Grupları



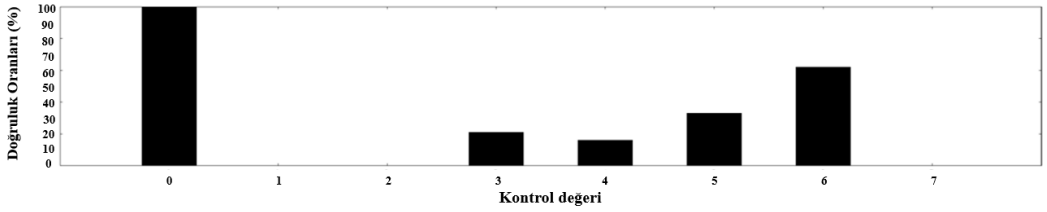
Çarpan değeri 7



Çarpan değeri 11



Çarpan değeri 13



Çarpan değeri 14

Şekil 3.8: Rastgele gelen hatalar için aynı verigruplarının farklı davranışı

4. HATAYA KARŞI DEVRE BAĞIŞIKLIĞINI ARTTIRMA YÖNTEMLERİ

Devrelerin hataya bağışıklı hale getirilmesi yedekleme ile yapılmaktadır. Yedekleme işi yapmak için gereken minimum kaynaktan daha fazlasının kullanılmasıdır.

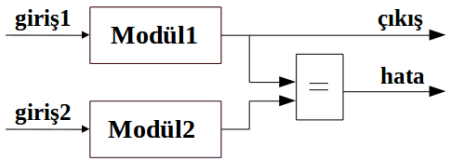
Gömülü mikroişlemcilerin doğru bir şekilde çalışmasına devam etmeleri için çeşitli hataya karşı bağışıklık kazandırma yöntemleri mevcuttur. Üçlü modül çoğullama, çiftleme ve karşılaştırma gibi donanım yedeklemesi ve Hamming kodları gibi bilgi yedeklemesi en sık kullanılan yöntemlerdir.

4.1 Donanım Yedeklemesi

Donanım yedeklemesi tasarımda hatalı olan elemanın etkisini kamufle etmek veya hatanın varlığını tespit etmek için fazladan donanımların kullanılmasıdır. Bir sistemde tek işlemci bulundurmak yerine iki ya da üç tane aynı işi yapan işlemci bulundurmak örnek olarak verilebilir. Devre boyutlarının büyümesi, güç tüketiminin artması gibi sisteme dezavantajlar getireceğinden gömülü sistemler için basit yapılar kullanılmaktadır.

4.1.1 Çiftleme ve Karşılaştırma

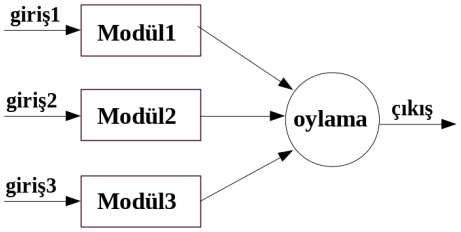
İki eş modül paralel olarak çalıştırılır ve çıkışları karşılaştırılır, eğer sonuçlar birbirinden farklı iseler Şekil 4.1'de görülebileceği gibi hata sinyali üretilir. Bu yöntem bir modülün hata tespitinde kullanılır. Hata tespit edildiğinde, sistemi tekrar çalışır duruma getirecek bir mekanizma bulunmamaktadır.



Şekil 4.1: Çiftleme ve Karşılaştıma Sistemi

4.1.2 Üçlü Modül Çoğullama Yöntemi

Donanım yedeklemesi için en yaygın yöntemdir ve temel yapısı Şekil 4.2'de gösterilmiştir. Devreler üçlenerek paralel olarak çalıştırılırlar. Doğru çıkışa çoğunluk oylaması karar verir. Eğer devrelerden birinde hata oluşursa çoğunluk oylaması diğer iki devreden gelen sonucun doğru olduğuna karar vererek hatayı maskeler.

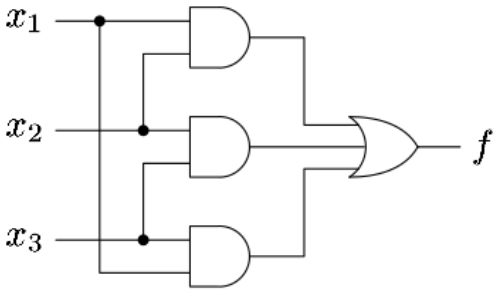


Şekil 4.2: Üçlü Modül Çoğullama Sistemi

Şekil 4.3'de oylama devresi, Çizelge 4.1'de ise doğruluk tablosu verilmiştir.

Çizelge 4.1: Çoğunluk Oylaması

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Şekil 4.3: Oylama devresi

4.2 Bilgi Yedeklemesi

Bilgi yedeklemesi rastgele erişilebilir bellek teknolojileri ve saklama cihazları için geçici hatalara karşı kullanılan bağışıklık kazandırma yöntemidir. Ayrıca sayısal haberleşmede gürültülü kanallardan geçen mesajların bozulmasını önler. Hamming

kodları tek bit düzeltme ve çift bit tespit etme özelliği ile bellekte herhangi bir kelimeye gelen hatalara karşı koruma sağlar.

4.2.1 Tek Bit Düzeltme Çift Bit Tespit Etme Kodları

Tek Bit Düzeltme Çift Bit Tespit Etme Kodları Single-Error Correcting Double-Error Detection Codes - SECDED) en yaygın kullanılan bilgi yedeklemesi yöntemleridir. Eşlik ve kontrol bitleri Hamming kodlama yöntemleri kullanılarak hesaplanır [28]. Eğer bir eşlik biti ile genişletilirse bir hata düzeltilip ve iki hata tespit edilebilir. Veri bitlerinin kodlanması için k tane veri bitine ek olarak $r \geq [log_2(k+r+1)]$ eşitsizliğini sağlayan r adet kontrol biti kullanılır ve kod $n = r+k$ bit olur. Örneğin; 8 bitlik veri için 4 kontrol biti, 16 bitlik veri için 5 kontrol biti gereklidir. Kontrol bitleri 2'nin kuvvetine, veri bitleri de diğer yerlere gelir ve kodlanmış kelime $(B_{12},B_{11},B_{10},B_9,B_8,B_7,B_6,B_5,B_4,B_3,B_2,B_1) = (D_7,D_6,D_5,D_4,C_3,D_3,D_2,D_1,C_2,D_0,C_1,C_0)$ biçimini alır.

$$B_{2^i} = C_i = \sum_{j=1, j\neq 2^q}^{k+r} a_{i,j}B'_j, \forall i \in 0,1,...,r-1 \qquad \textbf{(4.1)}$$

Diğer kontrol bitleri de bu formülden yola çıkarak hesaplanabilir:

$$B_1 = C_0 = B_3 \oplus B_5 \oplus B_7 \oplus B_9 \oplus B_{11} = D_0 \oplus D_1 \oplus D_3 \oplus D_4 \oplus D_6$$
$$B_2 = C_1 = B_3 \oplus B_6 \oplus B_7 \oplus B_{10} \oplus B_{11} = D_0 \oplus D_2 \oplus D_3 \oplus D_5 \oplus D_6$$
$$B_4 = C_2 = B_5 \oplus B_6 \oplus B_7 \oplus B_{12} = D_1 \oplus D_2 \oplus D_3 \oplus D_7$$
$$B_8 = C_3 = B_9 \oplus B_{10} \oplus B_{11} \oplus B_{12} = D_4 \oplus D_5 \oplus D_6 \oplus D_7$$

$(B'_{r+k},...,B'_j,...,B'_2,B'_1)$ bitleri hata içerebilen, bellekten okunan bitler olmak üzere hata düzeltimi r adet sendrom bitleri hesaplanarak yapılır. Sendrom bitlerini hesaplamak için, öncelikle B'_j $(j \neq 2^i)$ veri bitlerinden kontrol bitleri üretilir ve hesaplanan her B''_{2^i} kontrol biti denk gelen B'_{2^i} biti ile XOR işlemine tabi tutulur.

$$S_i = B'_{2^i} \oplus \sum_{j=1, j\neq 2^q}^{k+r} a_{i,j}B'_j, \forall i \in 0,1,...,r-1 \qquad \textbf{(4.2)}$$

Eğer tek bitlik hata varsa B'_m haricindeki tüm bitler için $B'_j = B_j$dir. Bu bit için $B'_m = B_m \oplus 1$dir. $S_i = a_{i,m}$ hatalı bitin yerini gösterir. B'_m biti ters çevrilerek B_m doğru değeri elde edilir. Eğer hiç hata yoksa sendrom bitlerinin hepsi 0'dır yani kelime hatasızdır.

Çift bit hata tespit etmek için genişletilmiş Hamming kodu kullanılır. Eklenen 5. bit ise bütün veri veya kontrol bitlerinin XOR işlemi sonucu olan eşlik biti hesabına dayanır.

$$C_4 = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7$$

4.1 ve **4.2**'deki ilişkiler eşlik matrisleriyle daha düzenlenmiş şekilde ifade edilebilir. k adet veri bitinden r adet Hamming kontrol bitlerini hesaplamak için $C^T = P_C D^T$ ilişkisi $C = [C_{r-1}, ..., C_1, C_0]$ ve $D = [D_{k-1}, ..., D_1, D_0]$ kontrol ve veri bitlerinin satır matrisi C^T ve D^T bunların transpozesi ve P_C kxr boyutunda eşlik matrisidir ve 8 bit veri **4.1**'den hesaplanır:

$$\begin{array}{c} \begin{array}{cccccccc} B_{12} & B_{11} & B_{10} & B_9 & B_7 & B_6 & B_5 & B_3 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

r adet sendrom biti k adet veri biti ve r adet kontrol bitlerinden $S^T = P_S B^T$ eşitliği kullanılarak hesaplanır. $S = [S_{r-1}, ..., S_1, S_0]$ ve $B = [B_{k-1}, ..., B_1, B_0]$ olmak üzere C^T ve D^T bunların transpozesi $(kxr)xr$ boyutunda eşlik matrisidir ve literatürde H matrisi olarak geçer.

$k = 8$ için P_S, P_C tarafından $r = 4$ sütun eklenerek elde edilir.

$$\begin{array}{c} \begin{array}{cccccccccccc} B_{12} & B_{11} & B_{10} & B_9 & B_8 & B_7 & B_6 & B_5 & B_4 & B_3 & B_2 & B_1 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Diğer bir SEC-DED kodu H matrisinin değişik düzenlenmesiyle oluşturulan Hsiao kodudur. Hamming kodlarına göre kodlayıcı ve kod çözücüdeki XOR kapılarının sayılarını azaltır [29].

Çoklu hataların düzeltilmesi için de birçok hata düzeltme kodları (Error Correction Codes - ECC) mevcuttur. Ancak bunların gerçeklenmesi alan, hız ve güç maliyetlerini yükseltmektedir.

Çizelge 4.2: 32 bit kod kelimesi için matris kodlarının gösterilimi

X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	C_0	C_1	C_2	C_3	C_4
X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	C_5	C_6	C_7	C_8	C_9
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}
X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}	X_{31}	C_{15}	C_{16}	C_{17}	C_{18}	C_{19}
P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7					

4.2.2 Matris Kodları

Bu yöntemde koruma bitleri matris formatında gösterilmiştir. n bitlik kod kelimeleri $k1$ satır ve $k2$ sütuna bölünmüştür. Çizelge **??**'de gösterildiği gibi her bir $k1$ satırına tek bit düzeltme çift bit tespit yapan kontrol bitleri, her bir sütuna da dikey eşlik bitleri eklenmiştir [30] [31].

Eşlik bitleri şu şekilde hesaplanır:

$$P_l = X_l \oplus X_{l+8} \oplus X_{l+16} \oplus X_{l+24}$$

l 8 eşlik biti için 0'dan 7'ye kadar olan sütun numarasıdır.

$$C_0 = X_0 \oplus X_1 \oplus X_3 \oplus X_4 \oplus X_6$$
$$C_1 = X_0 \oplus X_2 \oplus X_3 \oplus X_5 \oplus X_6$$
$$C_2 = X_1 \oplus X_2 \oplus X_3 \oplus X_7$$
$$C_3 = X_4 \oplus X_5 \oplus X_6 \oplus X_7$$
$$C_4 = X_0 \oplus X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus X_5 \oplus X_6 \oplus X_7$$

Bütün satırlar için kontrol bitleri cb her satır için kontrol bit numarası, o satır numarası j kontrol bitinin ve i ilk satırdaki veri bitinin pozisyonu olmak üzere $C_{yeni} = C_{j+(cb+o)}$ ve $X_{yeni} = X_{i+(k_2*o)}$ eşitlikleriyle hesaplanır.

Her bir satır için Hamming kod çözücüsü kullanılır ve kod çözme iki adımda gerçekleşir. Birincisi, yatay kontrol bitleri kaydedilen veri bitleri tarafından üretilir, kaydedilmiş kontrol bitleri ile karşılaştırılır ve sendrom bitleri üretilir. İkincisi sendrom bitleri kullanılarak her satır için tek hata tespit (single error detection-SED)/çift hata tespit (double eror detection-DED) ya da hata yok (no error-NE) sinyalleri üretilir. Eğer herhangi bir satırda çift hata tespit edilirse dikey sendrom bitleri ve veri bitleri kullanılarak **4.3**'deki eşitlikle herhangi bir satırdaki tek veya çift hata düzeltilir.

$$X_{i_{dzeltilmis}} = (X_{i_{hatali}} \oplus O_i) \oplus (DED_j * SP_k) \quad \textbf{(4.3)}$$

Bu eşitlikte hatalı bit, O_i i'ye karşılık gelen kod çözücü çıkışı, DED_j j numaralı satırdaki çift bit tespit etme sinyali ve SP_k karşılık gelen eşlik bitinin sendrom bitidir. Örneğin; X_{10} için SP_2 kullanılmaktadır.

Matris kodu yönteminde hata tespit ve düzeltme prosedürü için

- Kaydedilen bitler okunur:
  01111001011001111100110011011100
- Kaydedilen verinin eşlik ve kontrol bitlerinin hesaplanır:
  $C_0 - C_{19} = 10001011111010001001$
- Sendrom kontrol bitleri üretilir:
  $SC_0 - SC_{19} = 01100000000000000000$
- Eşlik bitleri hesaplanır:
  $P_0 - P_7 = 00001110$
- Sendrom eşlik bitleri üretilir:
  $SP_0 - SP_7 = 11000000$
- Hatalı bitler **4.3** kullanılarak düzeltilir:
  $X_{0_{dzeltilmis}} = (X_{0_{hatali}} \oplus O_0) \oplus (DED_0 * SP_0)$
  $X_{0_{dzeltilmis}} = (0 \oplus 0) \oplus (1 * 1)$
  $X_{0_{dzeltilmis}} = (0) \oplus (1) = 1$ Başlangıç değeri
  $X_{1_{dzeltilmis}} = (X_{1_{hatali}} \oplus O_1) \oplus (DED_0 * SP_1)$
  $X_{1_{dzeltilmis}} = (1 \oplus 0) \oplus (1 * 1)$
  $X_{1_{dzeltilmis}} = (1) \oplus (1) = 0$ Başlangıç değeri
- Düzeltilmiş veri çıkışa verilir

Matris kodları en fazla 4 bit ardışık hata düzeltebilir ancak bunun olasılığı düşüktür. Eğer kelime içerisinde ikiden fazla hata varsa matris kodları herhangi bir satırdaki iki hatayı diğer satırlarda sadece tek hata olmak şartıyla düzeltebilir. Örneğin hatalar X_6, X_7 ve X_8, X_9 gibi 2 bitin aynı satırda diğer ikisinin farklı satırda olduğu yere denk

gelirse düzeltme işlemi gerçekleşir ancak X_0, X_1 ve X_2, X_3'e gelen ardışık hatalar için bu durum geçerli değildir.

4.3 Önerilen Hata Tolerans Yöntemi

Bu tez çalışması için üçlü modül çoğullama yönteminin çoğunluk oylama mekanizması ile matris kodlarının eşlik ve kontrol biti hesap yöntemleri değiştirilip bu iki yapı birleştirilerek çoklu hataya karşı veriyi koruyabilen KodSeti olarak adlandırılan yeni bir yöntem önerilmiştir.

4.3.1 Kod Setleri

32-bitlik kelime 2, 4 veya 8 adet kod setine bölünebilir. Çizelge 4.3'te kod seti 4 düzenlemesi gösterilmektedir. Dikey eşlik bitleri verinin dikey satırlarının, yatay eşlik bitleri de yatay satırların XOR işlemiyle hesaplanır.

Kod setinde, verinin, yatay eşlik ve dikey eşlik bitlerinin kendi aralarında XORlanması aynı sonucu verir ve bu da kesişim biti olarak adlandırılmıştır.

Çizelge 4.4'de 32-bit veride düzeltilebilen ardışık hatalar için kodseti konfigürasyonları verilmiştir. KodSeti yönteminde gereksinime göre en uygun olan kullanılabilir. Fakat daha çok bitin hatalı olması durumunda düzeltme için gereken yatay ve dikey eşlik bitleri artmaktadır.

Dikey eşlik bitlerinin hesaplanması:

$$P_i = X_i \oplus X_{i+16}$$

Yatay eşlik bitlerinin hesaplanması:

$$C_i = X_i \oplus X_{i+4} \oplus X_{i+8} \oplus X_{i+12}$$

Kesişim bitlerinin hesaplanması:

$$DX_i = X_i \oplus X_{i+4} \oplus X_{i+8} \oplus X_{i+12} \oplus X_{i+16} \oplus X_{i+20} X_{i+24} \oplus X_{i+28}$$

$$DP_i = P_i \oplus P_{i+4} \oplus P_{i+8} \oplus P_{i+12}$$

$$DC_i = C_i \oplus C_{i+4}$$

$$DX_i = DP_i = DC_i$$

Çizelge 4.3: KodSeti grupları

X_0	X_4	X_8	X_{12}	C_0
X_{16}	X_{20}	X_{24}	X_{28}	C_4
P_0	P_4	P_8	P_{12}	D_0

X_1	X_5	X_9	X_{13}	C_1
X_{17}	X_{21}	X_{25}	X_{29}	C_5
P_1	P_5	P_9	P_{13}	D_1

X_2	X_6	X_{10}	X_{14}	C_2
X_{18}	X_{22}	X_{26}	X_{30}	C_6
P_2	P_6	P_{10}	P_{14}	D_2

X_3	X_7	X_{11}	X_{15}	C_3
X_{19}	X_{23}	X_{27}	X_{31}	C_7
P_3	P_7	P_{11}	P_{15}	D_3

Çizelge 4.4: 32-bit veri için kodseti düzenlemeleri

KodSeti	Ardışık Hatalar	Yatay Eşlik Bitleri	Dikey Eşlik Bitleri
8	8	16	16
4	4	8	16
2	2	4	16

4.3.2 Oylama Mekanizması

Oylama mekanizması düzeltme işlevinin gerekli olup olmadığına karar verir. Eğer hata eşlik ya da kontrol bitlerinde oluşursa orjinal verinin bozulmaması için düzeltme işlevi yapılmamalıdır. Çizelge 4.5'e göre düzelt sinyali veri bitleri hatalı olduğunda aktif olmaktadır.

4.3.3 Düzeltme İşlevi

Düzeltme işlevi düzeltme sinyali aktif olduğunda gerçekleştirilir ve **4.4**'deki eşitlikle yapılır:

$$Xduzeltilmis_{j+4*(i+4)} = Xhatali_{j+4*(i+4)} \oplus (SP_{j+4*i} \& SC_{j+4}) \qquad \textbf{(4.4)}$$

Düzeltme işlevi örneği:

- Orjinal veri kaydediciye yazılır.

  $X = 00110011100000011111010011101000$

- Orjinal verinin yatay ve dikey eşlik bitleri hesaplanır:

  for $i = 0:3$ **do**

  $C_i = X_i \oplus X_{i+4} \oplus X_{i+8} \oplus X_{i+12}$

  $C_{i+4} = X_{i+16} \oplus X_{i+20} \oplus X_{i+24} \oplus X_{i+28}$

Çizelge 4.5: Oylama Mekanizması

DX	DC_1	DP_1	düzelt
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

end for

$C = 11010000$

for $i = 0 : 15$ **do**

$P_i = X_i \oplus X_{i+16}$

end for

$P = 1100011101110101$

- Kaydedici içerisindeki veri okunur:

  $X_{hatali} = 00110011100\mathbf{111}01111010011101000$

  İlgili kaydedicinin 20:18 bitlerindeki verilere hata gelmiş olduğu varsayılsın.

- Hatalı verinin yatay ve dikey eşlik bitleri hesaplanır:

  $Ce = 10011101$

  $Pe = 1100011101101001$

- Kesişim bitleri hesaplanır ve karşılaştırılır:

  for $i = 0 : 3$ **do**

  $DX_i = X_i \oplus X_{i+4} \oplus X_{i+8} \oplus X_{i+12} \oplus X_{i+16} \oplus X_{i+20} \oplus X_{i+24} \oplus X_{i+28}$

  $DP_i = P_i \oplus P_{i+4} \oplus P_{i+8} \oplus P_{i+12}$

  $DC_i = C_i \oplus C_{i+4}$

  end for

  $DX = 0100\ DP = 1001\ DC = 1001$

  DX'in eş DP ve DC bitlerinden farklı olduğu görülebilmektedir. Yani orjinal veri kaydedilmiş veriden farklıdır.

- Düzeltme bitleri hesaplanır:

$duzelt = (DC \,\&\, DP \,\&\, \overline{DX}) \| (\overline{DC} \,\&\, \overline{DP} \,\&\, DX)$

$duzelt = 1101$

Düzeltme bitlerine göre veride 3 hata bulunmaktadır.

- Sendrom bitleri hesaplanır

  $SC = C \oplus Ce = 01001101$

  $SP = P \oplus Pe = 0000000000011100$

  Sendrom bitleri orjinal ve hatalı verinin yatay ve dikey eşlik bitleri farkını gösterir.

- Verideki hatalı bitler düzeltilir:

  for $j = 0:3$ **do**

  if $duzelt_i = 1$ **then**

  for $j = 0:3$ **do**

  $X_{duzeltilmis_{j+4*i}} = X_{hatali_{j+4*i}} \oplus (SP_{j+4*i} \,\&\, SC_j)$

  $X_{duzeltilmis_{j+4*i}} = X_{hatali_{j+4*i}} \oplus (SP_{j+4*i} \,\&\, SC_{j+4})$

  end for

  else

  $X_{duzeltilmis_{j+4*i}} = X_{hatali_{j+4*i}}$

  $X_{duzeltilmis_{j+4*i}} = X_{hatali_{j+4*i}}$

  end if

  end for

  $X_{duzeltilmis} = 00110011100000011111010011101000$

- Düzeltilmiş veri çıkışa verilir.

Çizelge 4.6: KodSeti yöntemi için rastgele gelen hataların düzeltme olasılıkları

Hata sayısı	Kodseti-2	Kodseti-4	Kodseti-8
1	100	100	100
2	50	75	87.5
3	0	37.5	65.625
4	0	9.375	41.015
5	0	0	20.507
6	0	0	7.69
7	0	0	1.92
8	0	0	0.24

Çiftleme ve karşılaştırma yöntemi sadece hata tespit edebilmektedir. Üçlü çoğullama yöntemi bir kelimeye gelen tüm hatalara karşı toleranslıdır ancak güç ve alan tüketimi fazladır. Matris kodlarında en fazla 4 tane ardışık hata düzeltilebilmektedir. Bu tasarım ile kodseti düzenlemesine göre 2, 4 veya 8 adet ardışık veya rastgele hata düzeltilebilmektedir. Yatay veya dikey eşlik bitlerinden birine hata geldiğinde verinin kaydediciden doğru olarak okunması sağlanabilmektedir. Eğer kaydedilmiş yatay veya dikey eşlik bitlerine hata gelirse verinin yanlış olarak düzeltilme işlevi önlenebilmekte ve orjinal veri bozulmamaktadır.

4.4 Hataya Karşı Bağışıklık Kazandırma Yöntemlerinin Güvenilirlik Ölçümü

Eğer tek bir bit üçlü modül çoğullama yöntemi ile korunuyorsa çıkışın hatalı olması olasılığı herhangi iki bit hatalı veya üç bitin hepsi hatalı ise oluşur [32]. Yani herhangi bir bite hata geldiğinde TMR sistemi için güvenilirlik **4.5** eşitliği ile hesaplanır.

$$R_{TMR-bit}(t) = \sum_{i=0}^{1} \binom{3}{i} .(1-R(t))^i R^{3-i}(t) = 3R^2(t) - 2R^3(t) \qquad \textbf{(4.5)}$$

W-bit genişliğindeki bir kelime TMR yöntemi ile korunuyorsa eşitlik **4.6** şeklinde olur.

$$R_{TMR-kelime}(t) = R^W_{TMR-bit}(t) \qquad \textbf{(4.6)}$$

Bilgi yedeklemesi ile koruma yönteminde hata tespit etme ve düzeltme kodları kullanıldığı için eklenen koruma bitleri de hesaba katılır. Genel ifade **4.7** eşitliğindeki gibidir.

$$R_{EDAC}(t) = \sum_{i=0}^{N} \binom{r+k}{i} .(1-R(t))^i .R^{r+k-i}(t) \qquad \textbf{(4.7)}$$

Bu eşitlikte N korunan bit, i hatalı bit sayısına karşılık gelmektedir. $r+k$ ise daha önce açıklandığı gibi koddaki bit sayısıdır.

Tek bit hata düzeltme kodu için eşitlik **4.8** şeklinde yazılabilir. Yani kelimeye hiç hata gelmeme olasılığına, düzeltilebilir tek bit hata olasılığı eklendiğinde güvenilirlik artar.

$$R_{SEC}(t) = R^{r+k}(t) + (r+k).(1-R(t)).R^{r+k-1}(t) \qquad \textbf{(4.8)}$$

İki, üç veya daha fazla sayıda hata düzelten Hamming kodları için eşitlik genişletilebilir.

Matris ve KodSeti yöntemleri için rastgele gelen hatalarda W-bit uzunluğundaki bir kelimenin güvenilirlik hesabı için bitlerin bozulma oranları haricinde geldiği yerler de hesaba katılmalıdır. Matris kodları için güvenilirlik hesabı hataların geldiği satırlara bağlıdır. Yani matris kodundaki veri herhangi bir satırda en fazla 2 hataya kadar korunabilir. 3. bir hata aynı satıra gelirse veri korunamaz. KodSeti yönteminde ise bir blok içerisinde tek bit korunduğundan 2. bir hatanın aynı bloğa gelmemesi gerekmektedir. Bu nedenle eşitlik **4.7** genel ifadesi yeniden düzenlenerek **4.9** haline getirilir.

$$R_{EDAC-kod}(t) = \sum_{i=0}^{N} \binom{r+k}{i} .(1-R(t))^{i}.R(t)^{r+k-i}.CC(i) \qquad \textbf{(4.9)}$$

$CC(i)$, veride i adet hatanın düzeltme kapasitesini ifade eder.

32-bit genişliğindeki kelime için hataların rastgele ve herhangi bir bite eşit olasılıkla gelmesi durumunda düzeltme kapasiteleri Çizelge 4.6'da verilmiştir. Kodseti-4 için 32-bit kelimeye karşı 16-bit dikey eşlik ve 8-bit yatay eşlik biti olduğundan toplam 24 tane koruma biti vardır. $r+k$ ifadesi 56'ya eşit olmaktador. Ayrıca Çizelge 4.6'daki düzeltme kapasitesi yerine konulduğunda **4.9** eşitliği açılırsa **4.10** ifadesi elde edilir. KodSeti-2, KodSeti-8 ve Matris Kodları için de ilgili değerler yerine konularak güvenilirlik hesaplanabilir.

$$\begin{aligned} R_{kodseti-4}(t) = {} & R^{56}(t) + 56.(1-R(t)).R^{55}(t) + \binom{56}{2}.(1-R(t))^{2}.R^{54}(t).\frac{3}{4} \\ & + \binom{56}{3}.(1-R(t))^{3}.R^{53}(t).\frac{3}{8} + \binom{56}{4}.(1-R(t))^{4}.R^{52}(t).\frac{3}{32} \end{aligned} \qquad \textbf{(4.10)}$$

Faydalı çalışma periyodunda oluşturulan geçici hata oranları değiştirilerek güvenilirlik aynı zamanda hata olasılık hesapları yapılabilir. Tek bir bitin hata oranı BHO ile ifade edildiğinde Çizelge 4.7'de verilen BHO ve bir bitteki güvenilirlik arasındaki ilişkiden görülebileceği üzere sabit bir zamanda BHO değiştirilerek kelimedeki hata olasılığındaki değişim benzer eşitliklerden hesaplar yapılarak incelenebilir.

$$R_{kelime} = (1-BHO)^{k} \qquad \textbf{(4.11)}$$

$$BHO_{TMR} = 3BHO^2 - 2BHO^3 \tag{4.12}$$

$$R_{TMR-kelime} = (1 - [3BHO^2 - 2BHO^3])^k \tag{4.13}$$

$$R_{EDAC-kod} = \sum_{i=0}^{N} \binom{r+k}{i} .BHO^i .(1-BHO)^{r+k-i} .CC(i) \tag{4.14}$$

Çizelge 4.7: BHO ve güvenilirlik arasındaki ilişki

BHO	$1 - e^{-\lambda t} = 1 - R(t)$	t
0	0	0
1.10^{-6}	$0,999..95x10^{-6}$	0,1
$1,1.10^{-6}$	$1,099..91x10^{-6}$	0,11
$1,2.10^{-6}$	$1,199..28x10^{-6}$	0,12
—	—	—
—	—	—
1.10^{-5}	$0,999..95x10^{-5}$	1
—	—	—
—	—	—
$0,999..99$	$0,999..55$	$0,99..99.10^6$
1.10^0	1	1.10^6

5. HATAYA BAĞIŞIKLI MİKROİŞLEMCİ TASARIMI

Bu kısımda ilk aşamada tek çevrimli ve iş hatlı MIPS-32 gömülü mikroişlemcisi tasarımı yapılmıştır. Asıl amaç olarak mikroişlemci devresini güçlendirmek için kaydedici dosyasının önceki bölümde anlatılan hata bağışıklığı kazandırma yöntemleriyle tasarımı yapılmıştır. Tasarımları mikroişlemci üzerinde test etmek için güvenliği kritik bir uygulama olan Gelişmiş Şifreleme Standardı (Advanced Encryption Standard -AES) algoritması seçilmiştir ve bu algoritmaya özgü işlemlerin yapılabilmesi için mikroişlemcide kod genişletmesi yapılmıştır.

5.1 MIPS-32 Mikroişlemci Tasarımı

Mikroişlemci tasarımı yaygın olarak kullanılan ve temel bir yapıya sahip MIPS mimarisi ile [33]'den faydalanarak tasarlanmıştır.

Tasarlanan mikroişlemci Çizelge 5.1'de verilen komutları desteklemektedir.

Çizelge 5.1: MIPS-32 Komutları

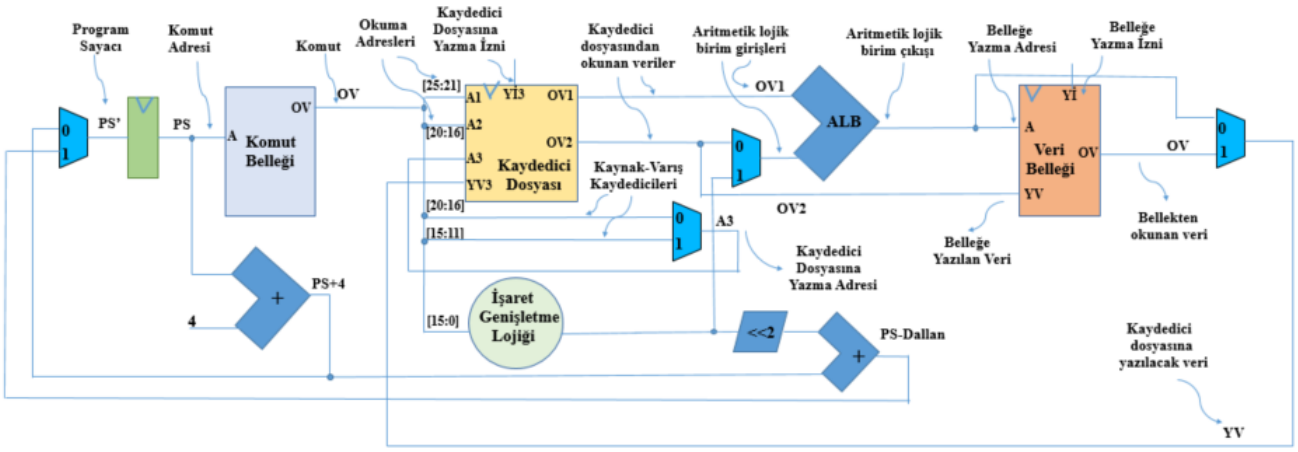
İşlem kodu	**Ad**	**Açıklama**	**İşlev**
000000	R-type	bütün R-tipi komutlar	[rs] = [rd] fonk [rt]
100011	LW	kelimeyi başlat	[rt]=[Adres]
101011	SW	kelimeyi depola	[Adres]=[rt]
000100	BEQ	eşit ise dallan	if ([rs] == [rt]) PC = BTA
000101	BNE	eşit değil ise dallan	if ([rs] != [rt]) PC = BTA
000010	J	atla	PC = JTA
001000	ADDI	doğrudan topla	[rt] = [rs] + SignImm
001100	ANDI	doğrudan ve	[rt] = [rs] & ZeroImm
001101	ORI	doğrudan veya	[rt] = [rs] | ZeroImm
001110	XORI	doğrudan harici veya	[rt] = [rs] ^ ZeroImm
001111	LUI	doğrudan üstten başla	[rt] = Imm, 16´b0
100100	LBU	işaretsiz baytı başlat	[rt] = ZeroExt ($[Adres]_{7:0}$)

Çizelge 5.2'de ise R-tipi komutlar verilmiştir.

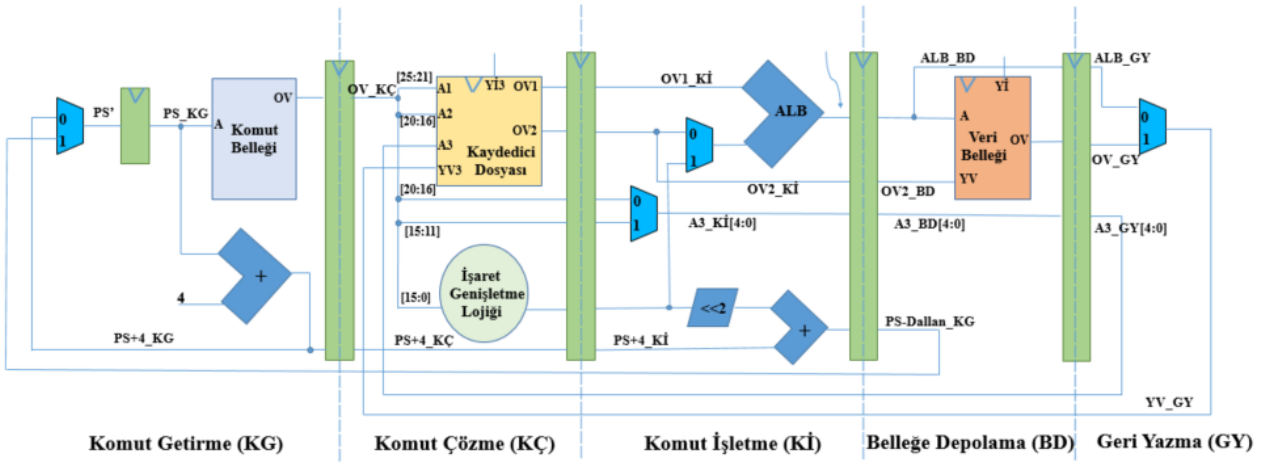
Şekil 5.1'de tek çevrim mikroişlemci, şekil 5.2'de ise iş hatlı mikroişlemci genel yapısı gösterilmiştir.

Çizelge 5.2: R-tipi Komutlar

Fonksiyon	Ad	Açıklama	İşlev
100100	AND	ve	[rd] = [rs] & [rt]
100110	XOR	harici veya	[rd] = [rs] ^ [rt]
100111	NOR	veya değil	[rd] = !([rs] | [rt])
101010	SLT	daha küçüğe ayarla	[rt] = ZeroExt ($[Adres]_{7:0}$)
000000	SLL	lojik sola kaydır	[rs] = [rd] crypt [rt]
000010	SRL	lojik sağa kaydır	[rs] = [rd] crypt [rt]
000110	SRLV	lojik değeri sağa kaydır	[rd] = [rt]<<[rs] 4:0
000100	SLLV	lojik değeri sola kaydır	[rd] = [rt]>>[rs] 4:0
001000	JR	kaydediciye atla	[rs] = [rd] crypt [rt]



Şekil 5.1: Tek çevrim mikroişlemci genel yapısı



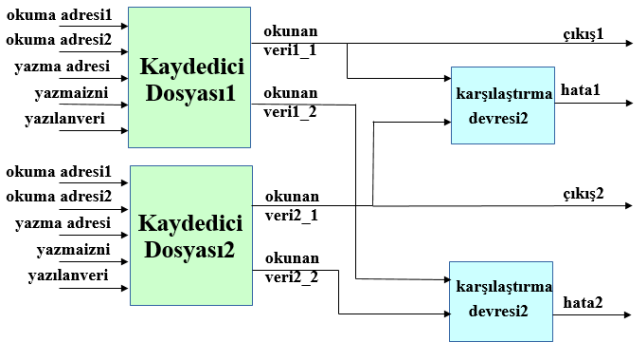
Şekil 5.2: İş hatlı mikroişlemci genel yapısı

5.2 Hataya Bağışıklı Kaydedici Dosyası Tasarımı

Mikroişlemci içerisinde hataya en yatkın birimlerden biri kaydedici dosyasıdır. Çünkü aritmetik ve mantıksal işlemler bu kaydediciler üzerinden yapılır [34] ve işlemci çekirdeği tarafından sıklıkla erişilen birimdir. Veriler uzun periyotlar boyunca bu birimde tutulur. Kaydedici dosyasında oluşabilecek bir hata yanlış verilerin okunmasıyla işlemcinin diğer kısımlarına yayılıp bozulmalara ve yanlış program çıktılarının alınmasına sebep olabilir. Bu sebeple kaydedici dosyalarının korunması için literatürde birçok yöntem önerilmiştir [35] [36] [37].

5.2.1 Çiftleme ve Karşılaştırma Yöntemi ile Tasarım

Şekil 5.3'de kaydedici dosyasının çiftleme ve karşılaştırma yöntemi ile tasarımı gösterilmiştir. İki adet eş modülün çıkışları karşılaştırılmaktadır ve herhangi bir hata oluştuğunda hata sinyali aktif olmaktadır.



Şekil 5.3: Çiftleme ve Karşılaştırma ile Korunan Kaydedici Dosyası

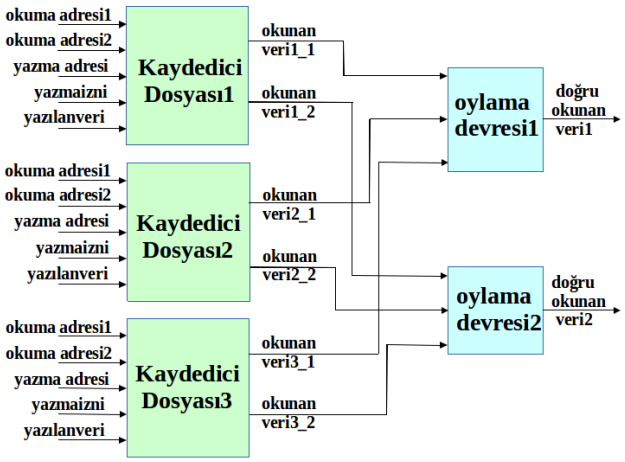
5.2.2 Üçlü Modül Çoğullama Yöntemi ile Tasarım

Şekil 5.4'de kaydedici dosyası üçlü modül çoğullama ile korunmaktadır. Üç adet eş kaydedici dosyasında okunan veriler çoğunluk oylama bloğuna girmektedir.

5.2.3 Matris Kodları Yöntemi ile Tasarım

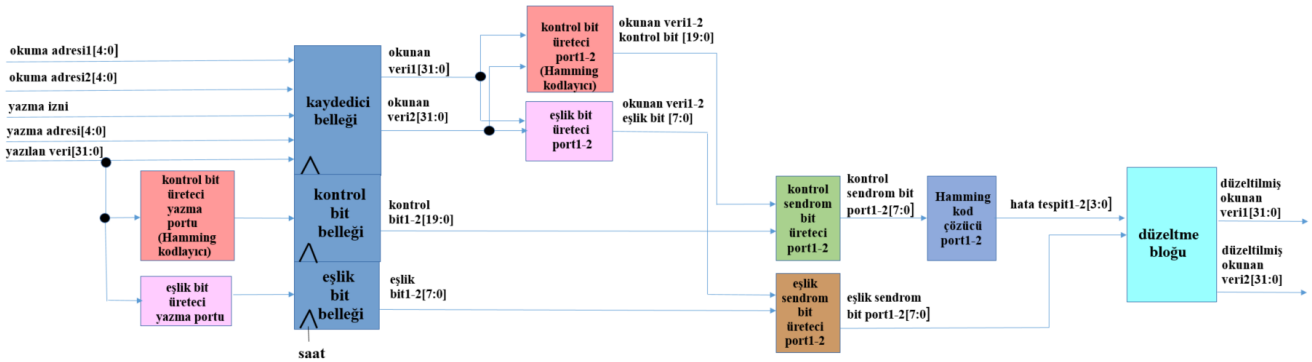
Şekil 5.5'de kaydedici dosyası matris kodu yöntemi ile korunmaktadır.

- **orjinal kaydedici dosyası:** Bu blok tipik 1-yazma 2-okuma portu olan kaydedici dosyasıdır.
- **kontrol bit dosyası:** Bu blok kontrol bit dosyasını içerir.
- **eşlik biti dosyası:** Bu blok eşlik bit dosyasını içerir.



Şekil 5.4: Üçlü Modül Çoğullama ile Korunan Kaydedici Dosyası

- **kontrol bit üreteci-Hamming kodlayıcı:** Eşlik bitlerinin okuma ve yazma işleminde hesaplanması için 3 eş modül bulunur.
- **eşlik-sendrom bit üreteci:** Sendrom eşlik bitlerinin üretimi için iki adet eş modüldür.
- **kontrol-sendrom bitüreteci:** Sendrom kontrol bitlerinin üretimi için iki adet eş modüldür.
- **Hamming kod çözücü:** Hata tespit bitlerinin üretimi için iki adet eş modüldür.
- **düzeltme bloğu:** Hamming kod çözücülerden üretilen hata tespit sinyallerine bağlı olarak doğrulama işlemi gerçekleştirilir.

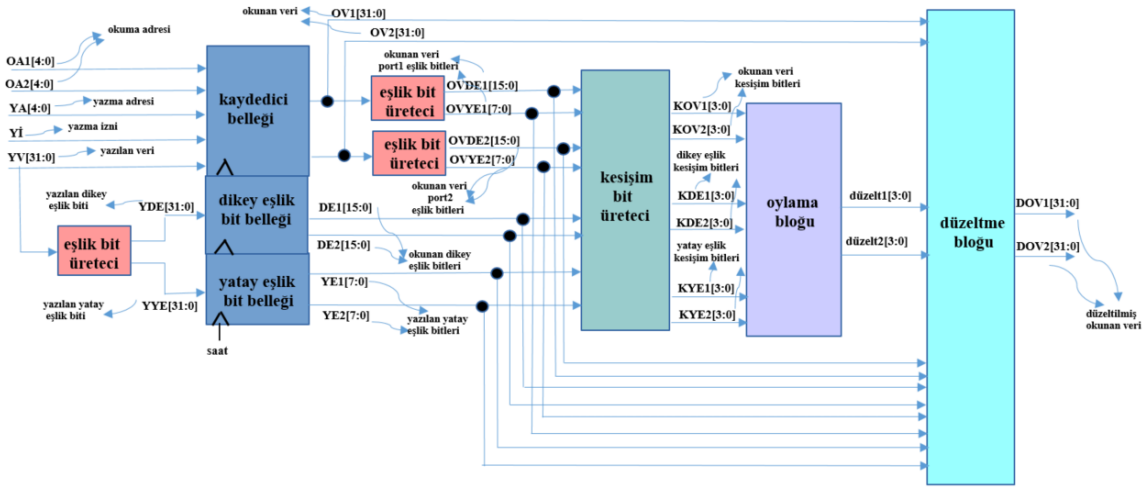


Şekil 5.5: Matris Kodu Yöntemi ile Korunan Kaydedici Dosyası

5.2.4 Kod Seti Yöntemi ile Tasarım

Şekil 5.6 devrenin blok ve bağlantılarını göstermektedir. Matris kodu yöntemi ile ortak bloklar vardır ancak Hamming kodlayıcı ve kod çözücü bulundurmaması ve oylama mekanizması içermesi gibi temel farklılıklar bulunmaktadır.

- **eşlik bit üreteci:** Yatay ve dikey eşlik bitlerinin okuma ve yazma işleminde hesaplanması için 3 eş modül bulunur.
- **kesişim bit üreteci:** Karşılaştırılacak kesişim bitlerinin hesaplandığı modüldür.
- **oylama bloğu:** Oylama mekanizması gerçekleştirilir. Verinin düzeltilip düzeltilmemesine karar veren düzeltme sinyali üretilir.
- **düzeltme bloğu:** Eğer veride hata varsa düzeltme sinyaline bağlı olarak eşlik bitlerinden sendrom bitleri hesaplanır ve doğrulama işlemi gerçekleştirilir.

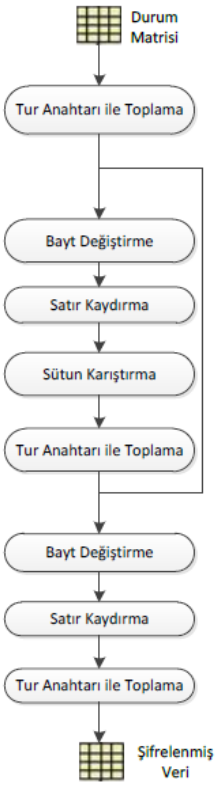


Şekil 5.6: Kod seti yöntemi ile kaydedici dosyası tasarımı

5.2.5 Kaydedici Dosyasındaki Bellek Birimi İçin Güvenilirlik Ölçümü

Bir bellekteki güvenilirlik içerisindeki tüm kelimelerin güvenilirliğinin çarpımına eşittir [38] [39].

$$R_{TMR-bellek}(t) = R^{M}_{TMR-kelime}(t) \tag{5.1}$$



Şekil 5.7: AES Algoritmasının Blok Yapısı

$$R_{EDAC-bellek}(t) = R^{M}_{EDAC-kod}(t) \tag{5.2}$$

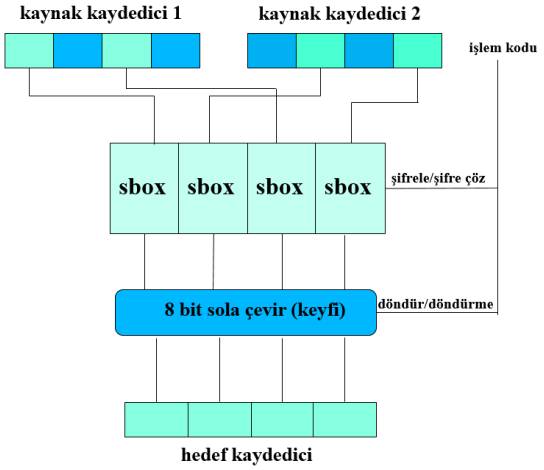
Eşitlik **5.1** ve **5.2**'de M bellekteki kaydedici sayılarını ifade etmektedir.

5.3 Mikroişlemciye şifreleme modülü eklenmesi

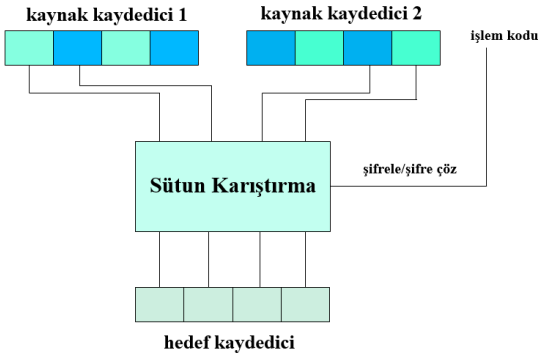
Şifreleme modülü Gelişmiş Şifreleme Standardı (AES-128) algoritması baz alınarak tasarlanmıştır [40]. Bu algoritmanın blok yapısı Şekil 5.7'de gösterilmiştir. Mikroişlemci ile etkili olarak bütünleştirilebilmesi için özel fonksiyonlar mevcuttur. Şifreleme komutlarını desteklemek için MIPS mikroişlemcisine komut genişletilmesi yapılmıştır. Şifreleme komutları R-tipi formatında olsa bile daha belirgin bir gösterim için dördüncü kategoriye alınmıştır.

5.3.0.1 sbox4s-isbox4s-sbox4r komutları

Bu komutlar AES algoritmasının her turunda gerçekleştirilen bayt değiştirme ve ters bayt değiştirme işlemini gerçekleştirmektedir. Bu komutların yapısı Şekil 5.8'de gösterilmiştir. Buna göre girişte iki adet kaynak saklayıcı bulunmakta ve bu saklayıcıların şekilde gösterilen baytları alınarak S-kutusundan geçirilmektedir.



Şekil 5.8: sbox4s, isbox4s ve sbox4r komutlarının işlevi



Şekil 5.9: mixcol4s, imixcol4s komutlarının işlevi

S-kutusu çıkışlarının bağlı olduğu bayt kaydırma birimi bu donanım bloğunun anahtar üretimi için de kullanılmasını sağlamaktadır ve sbox4r komutu ile bu işlem gerçekleştirilmektedir. sbox4s komutu şifreleme işlemi için kullanılırken isbox4s komutu şifre çözme için kullanılmaktadır ve S-kutuları yerine ters S-kutuları kullanılması ile elde edilmiştir [41].

5.3.0.2 mixcol4s – imixcol4s komutları

Sütun karıştırma ve ters sütun karıştırma işlemlerinin gerçekleştirilmesini sağlayan bu komutların yapısı Şekil 5.9'de gösterilmiştir. Bu komut yapısı gereğince, matrisin iki farklı sütununu tutan iki kaynak saklayıcısının şekilde gösterilen baytları alınarak tasarlanan sütun çarpıcı modülünden geçirilmektedir. mixcol4s ve imixcol4s komutları arasındaki tek fark, imixcol4s için ters sütun çarpıcı modülünün kullanılmasıdır [41].

Çizelge 5.3: Şifreleme İşlem Kodu

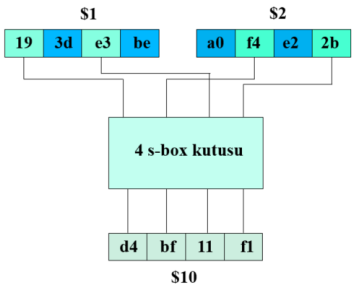
010011	CRYPT	şifrele-şifre çöz	[rs] = [rd] crypt [rt]

Çizelge 5.4: Şifreleme Komutları

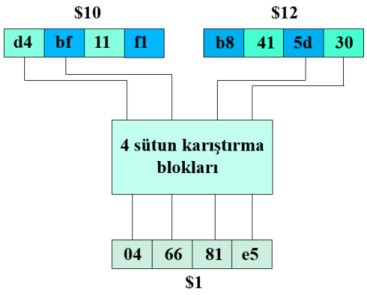
Fonksiyon	İsim
100101	SBOX4R
100000	SBOX4S
100001	MIXCOL4S
100010	ISBOX4S
100011	IMIXCOL4S

5.3.0.3 AES-128 program kodu ile simülasyon sonuçları

Mikroişlemci tasarlandıktan sonra verilen komutlarla AES-128 program kodu yazılıp MARS Simulatörü ile derlenerek makine koduna çevrilmiştir. Program şifreleme yapmaktadır, mesaj ve anahtar bilgisi [40]'deki AES-128 örneğinden alınmıştır ve Çizelge 5.5'de gösterilmiştir.



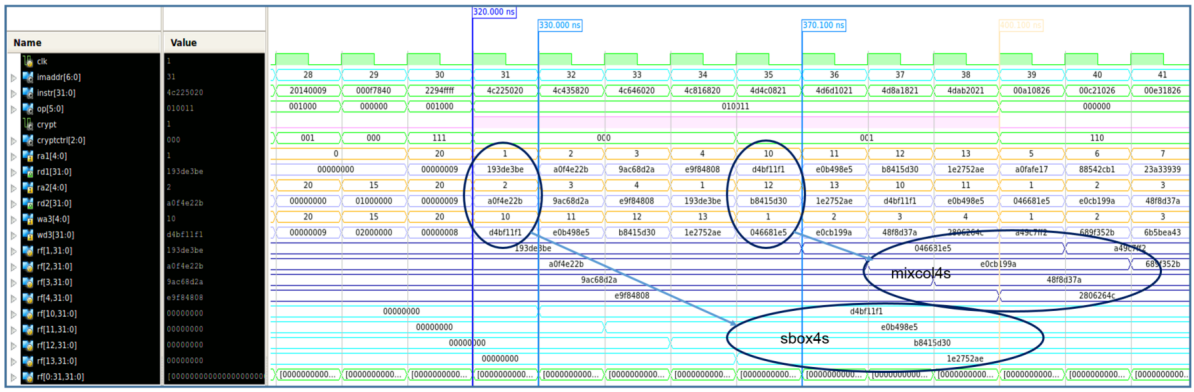
Şekil 5.10: sbox4s komut işlev örneği



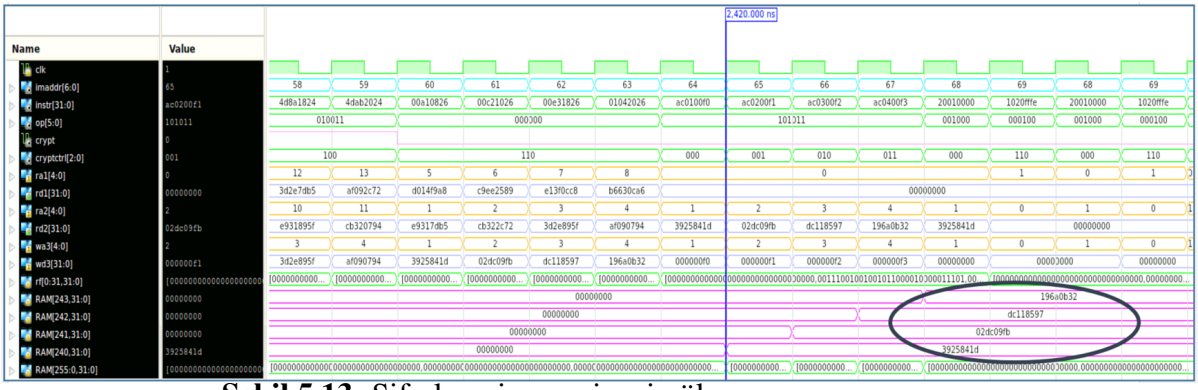
Şekil 5.11: mixcol4s komut işlev örneği

Çizelge 5.5: Mesaj-Anahtar-Şifrelenmiş veri

Mesaj	3243f6a8-885a308d-313198a2-e0370734
Anahtar	2b7e1516-28aed2a6-abf71588-09cf4f3c
Şifrelenmiş veri	3925841d-02dc09fb-dc118597-196a0b32



Şekil 5.12: sbox4s-mixcol4s simülasyon sonucu



Şekil 5.13: Şifrelenmiş mesajın simülasyon sonucu

6. TESTLER VE ANALİZLER

6.1 Simülasyon Sonuçları

Anlatılan yöntemler ile kaydedici dosyasının tasarımları yapılıp mikroişlemci ile bütünleştirilmesinden mikroişlemcide şifreleme algoritması işletilirken ve KodSeti-4 yönteminin uygulanabilirliğini göstermek için testler yapılmış ve simülasyon çıktıları alınmıştır. Hatalar anlatılan hata üretim devresi ile verilmiştir.

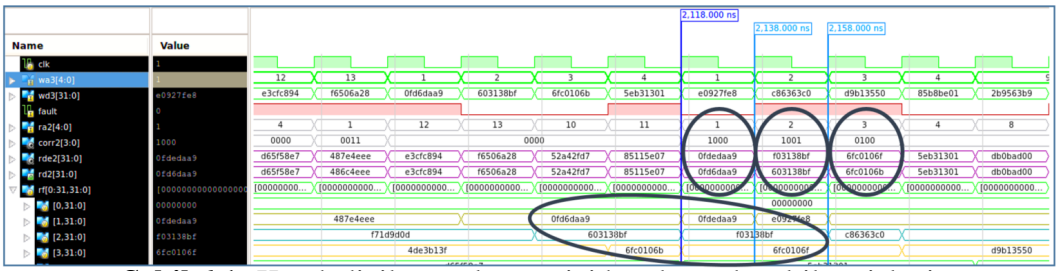
Hata gelebilecek yerler ve hata oranları çok çeşitlilik göstermektedir. Bu nedenle yapılan testler için çeşitli özellikler tanımlanmıştır 4-bit hatalar gelecek şekilde varsayımlar sınırlandırılmıştır.

İlk simülasyon için;

- Hata oranı 50% olarak belirlenmiştir.
- Sadece kaydedici dosyasına hatalar gelmektedir.
- Hatalar ilk 3 kaydedicinin bitlerine verilmiştir.
- $r1$ kaydedicisinin $[20:17]$ bitleri hatalı olarak seçilmiştir.
- $r2$ kaydedicisinin $[31:28]$ bitleri hatalı olarak seçilmiştir.
- $r3$ kaydedicisinin $[3:0]$ bitleri hatalı olarak seçilmiştir.
- Hatanın değeri 1 seçilmiştir.

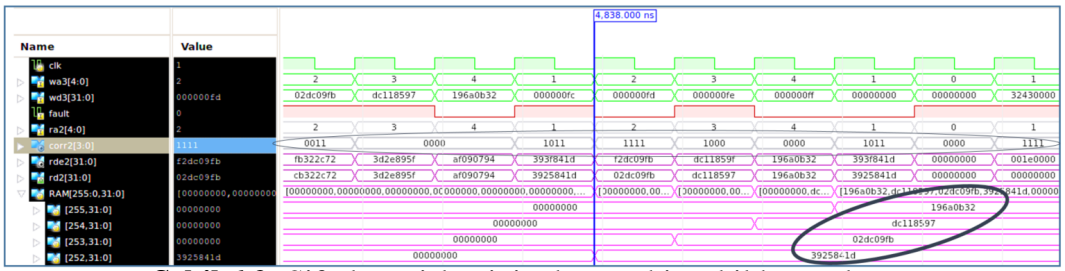
Şekil 6.1'de gelen veri hata sinyali pasif iken ilgili kaydedicilere doğru bir şekilde yazılmaktadır. $2118ns$'de ilk üç kaydedici hatadan etkilenmiştir. İlgili adreslerden okuma işlemi gerçekleşirken düzeltme bitleri kaç adet bitin düzeltilmesi gerektiğine dair bilgi verir. Çünkü kaydedicinin ilgili bitine gelen hatanın değeri o bitte bulunan değerle aynı olabilir. Örneğin; $r1$ kaydedicisinin değeri $0fd6daa9$ iken hata ile birlikte $0fdedaa9$ yani [20:17] bitlerine bakıldığında "1011" bitleri "1111" değerini almıştır.

Hata değerinden ötürü sadece 1 bit etkilenmiştir ve düzeltme sinyali de "1000" değerini alarak kodsetindeki 3. gruptaki bitin düzeltileceğini bildirmektedir. Veri okunuken yani okuma adresi 1 olduğunda düzeltilerek kaydedici dosyasının veri okuma portuna verilmektedir. Aynı şekilde diğer hatalı olan kaydediciler de okunacakları zaman düzeltilerek çıkışa verilirler.

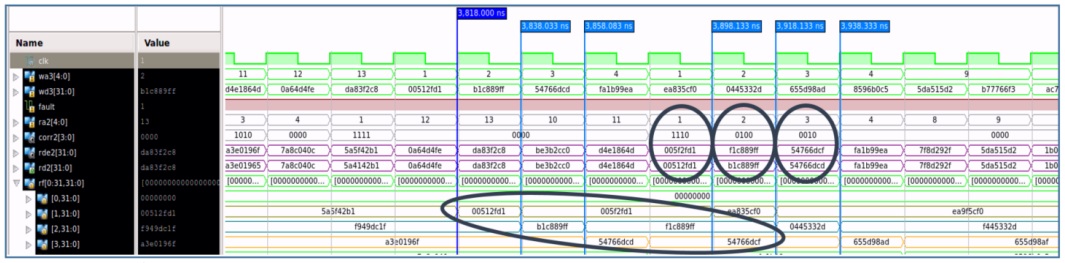


Şekil 6.1: Kaydedicilere gelen geçici hataların düzeltilme işlevi

Devamında ise şifreleme işlemi gerçekleştikten sonra şifrelenmiş mesaj 32-bitlik veriler halinde veri belleğine yazılmıştır ve Şekil 6.2'de gösterilmiştir. Şifreleme işlemi doğru bir şekilde gerçekleşmiştir.

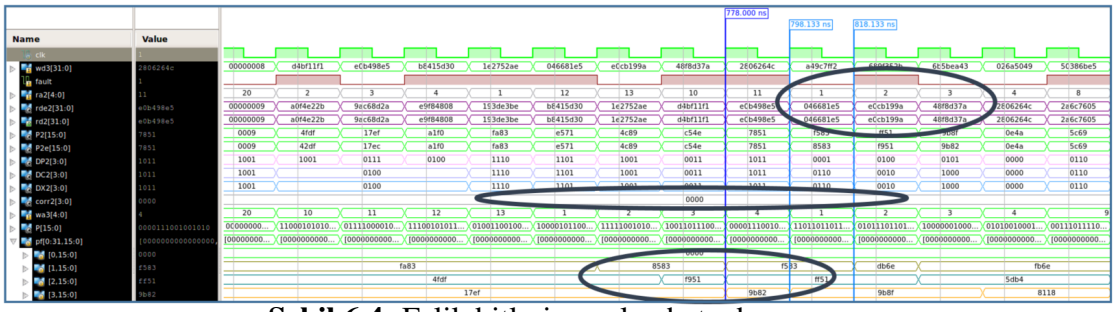


Şekil 6.2: Şifreleme işleminin düzgün bir şekilde yapılması



Şekil 6.3: Kaydedicilere gelen kalıcı hataların düzeltilme işlevi

Şekil 6.3'de ise hata oranı 100% olacak şekilde ayarlanarak ilgili kaydedicilerdeki bitler sürekli hatalı yapılmıştır. İki şekil de düzeltme sinyalindeki değişimi göstermektedir. Düzeltme işlevi hata oranı ile etkilenmemektedir ve veri içerisindeki maksimum düzeltebilecek bit sayısını geçmemek şartıyla yapılabilmektedir. Ancak hata hiçbir zaman ortadan kalkmadığından zamanla başka bitlere gelebilecek geçici hatalara karşı veri korunamaz.



Şekil 6.4: Eşlik bitlerine gelen hata durumu

Diğer bir simülasyon için eşlik bitlerine hata gelmesi ele alınmıştır. Şekil 6.4 eşlik bit dosyasındaki hatalı kaydedicileri göstermektedir. 778.190*ns*'de ilk üç kaydedicideki değerler değişmiştir ancak orjinal veride bozulma olmadığı için oylama mekanizması ile düzeltme bitleri pasif kalarak yanlış düzeltme işlevi engellenmiştir.

Şekil 6.5: Kaydedici bitlerine rastgele gelen hata durumu

Hataların rastgele olarak kaydedici bitlerine geldiği simülasyon ise Şekil 6.5'de gösterilmiştir. Burada hata aynı kodseti grubuna geldiği için veri doğru bir şekilde düzeltilememektedir. 3858*ns*'de 2. kaydedicinin değeri *db*677*c*2*a* iken *db*677*d*3*a* olarak değişmiştir. İkili düzende düşünüldüğünde aynı kodseti grubunda olan 4., 8. ve 24. bitlere hata gelmiştir. 3898*ns*'de okunduğunda verinin düzeltilememiş olduğu görülmektedir.

6.2 Zaman ve Alan Karşılaştırmaları

Açıklanan hata tolerans yöntemleri ile tasarlanan kaydedici dosyası TSMC 90 nm standart kütüphanelerinde sentezlenip alan ve zaman karşılaştırılması yapılmıştır.

Çizelge 6.1'de alan sonuçları verilmiştir. Bu sonuçlara göre KodSeti yöntemi 2 ve 4 konfigürasyonları için diğer yöntemlere göre daha az alan kaplamaktadır. Üçlü modül çoğullama ile yapılan tasarım ise en fazla alanı kaplamaktadır.

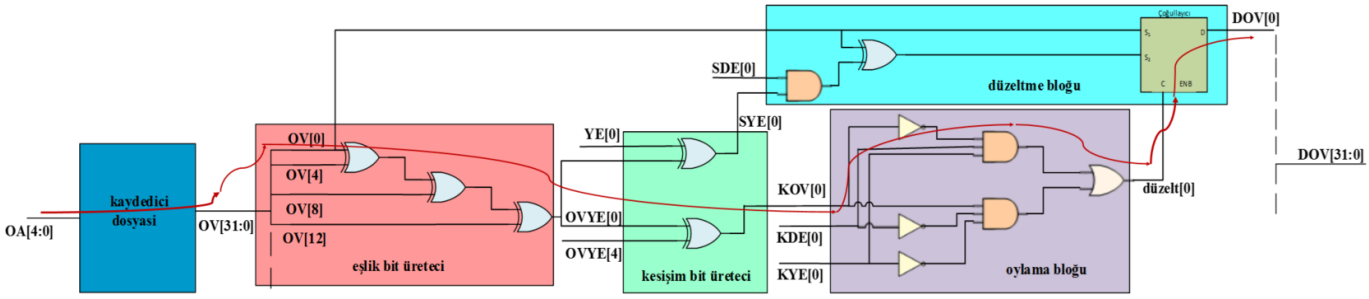
Çizelge 6.1: Kaplanan alan karşılaştırılması

Yöntemler	Toplam Alan (μm^2)	Yüzde Aşım %
ORJİNAL	42567	100
KODSETİ-2	71615	168
KODSETİ-4	76881	180
KODSETİ-8	88597	208
MATRİS	85910	201
ÇİFTLEME	85449	200
ÜÇLEME	127313	299

Çizelge 6.2: Maksimum çalışma frekanslarının karşılaştırılması

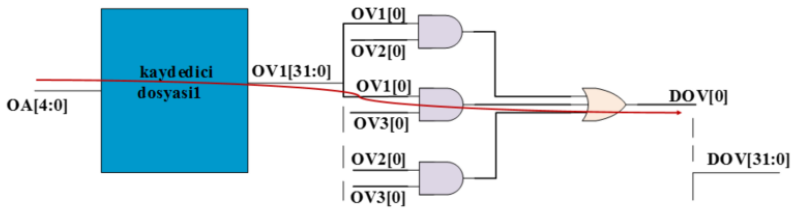
Yöntemler	Maksimum Çalışma Frekansı (MHz)
ORJİNAL	576
KODSETİ-2	412
KODSETİ-4	449
KODSETİ-8	451
MATRİS	315
ÇİFTLEME	373
ÜÇLEME	550

Çizelge 6.2'de ise tasarımların maksimum çalışma frekansları verilmiştir. Orjinal kaydedici dosyasından sonra en hızlı çalışan tasarım TMR'dır. Çünkü kombinezonsal kısım paralel olarak çalışmaktadır. Şekil 6.6 ve Şekil 6.7'de TMR ve kodseti yöntemi için kritik yollar gösterilmiştir.

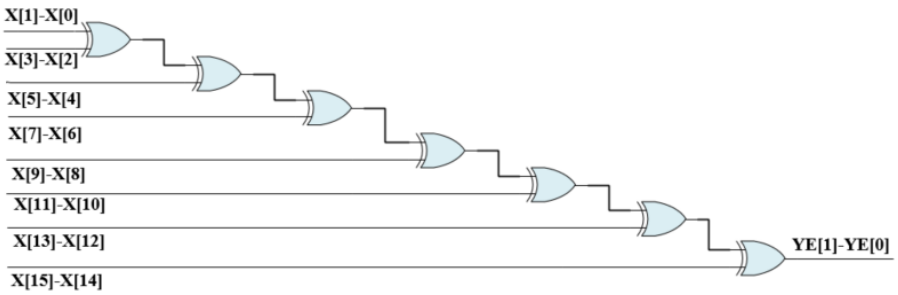


Şekil 6.6: KodSeti yöntemi için kritik yol

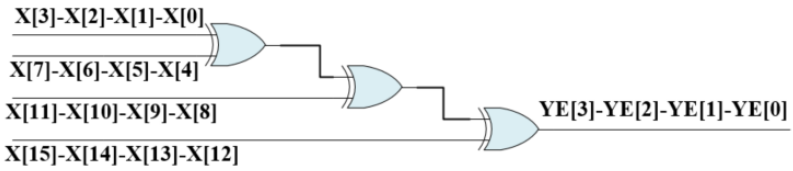
KodSeti-8 konfigürasyonu için maksimum çalışma frekansı daha fazladır. Bunun sebebi Şekil 6.8'de gösterilen KodSeti-2'de bir bit yatay eşlik biti hesaplamak için 7 XOR zinciri, Şekil 6.9'de gösterilen KodSeti-4 için 3 XOR zinciri ve Şekil 6.10'de gösterilen KodSeti-8 için tek bir XOR gerekmektedir.



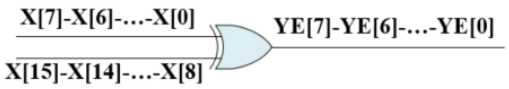
Şekil 6.7: TMR yöntemi için kritik yol



Şekil 6.8: KodSeti-2 için gereken XOR zinciri



Şekil 6.9: KodSeti-4 için gereken XOR zinciri



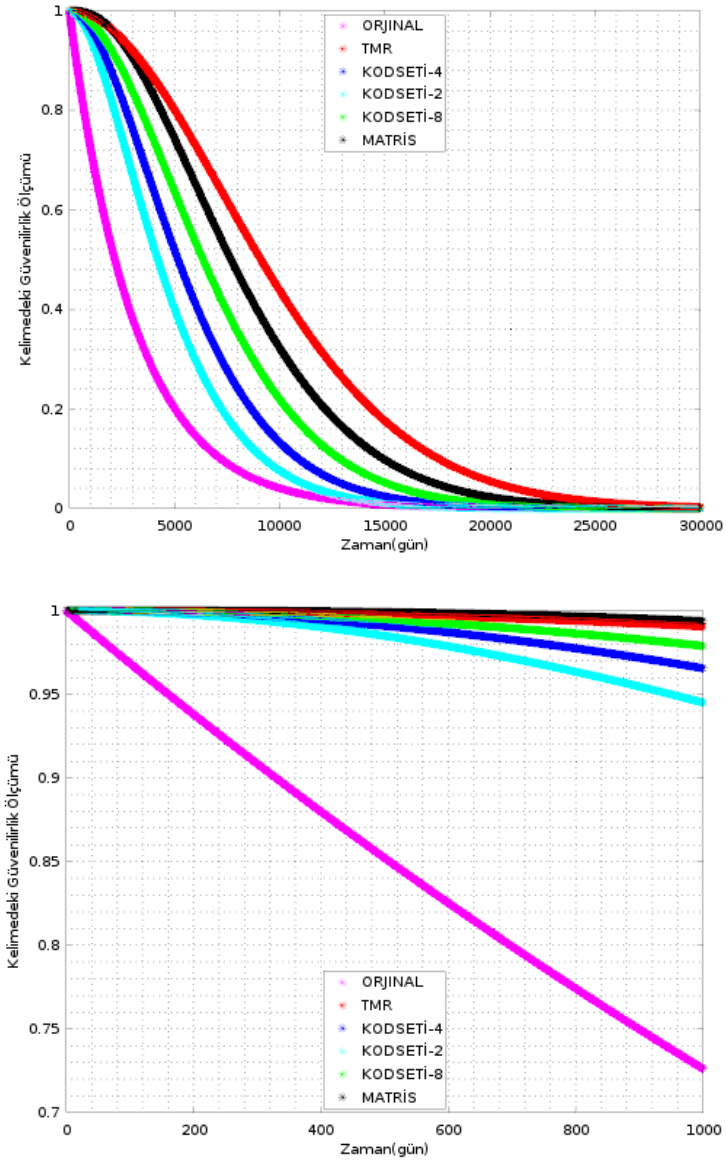
Şekil 6.10: KodSeti-8 için gereken XOR zinciri

6.3 Güvenilirlik Analizi

Tasarımların güvenilirlik analizleri yapılmıştır. Önceki bölümde verilen formüllere dayanarak hatalar kaydedici bitlerine rastgele olarak gelmektedir. DWC yöntemi sadece hata tespiti yapabildiğinden bu analizler içerisinde incelenmemiştir.

Şekil 6.11 32-bit uzunluğundaki bir kaydedicinin $\lambda = 10^{-5}$ bozulma oranında zamana bağlı güvenilirlik analizini göstermektedir. İlk 1000 güne bakıldığında koruma olmayan kaydedicinin güvenilirliği hızla azalmakta iken anlatılan yöntemlerle korunan kaydedicinin güvenilirlikleri başabaş gitmektedir. 25000 güne kadar TMR yönteminin güvenilirliği diğer yöntemlere göre daha fazladır.

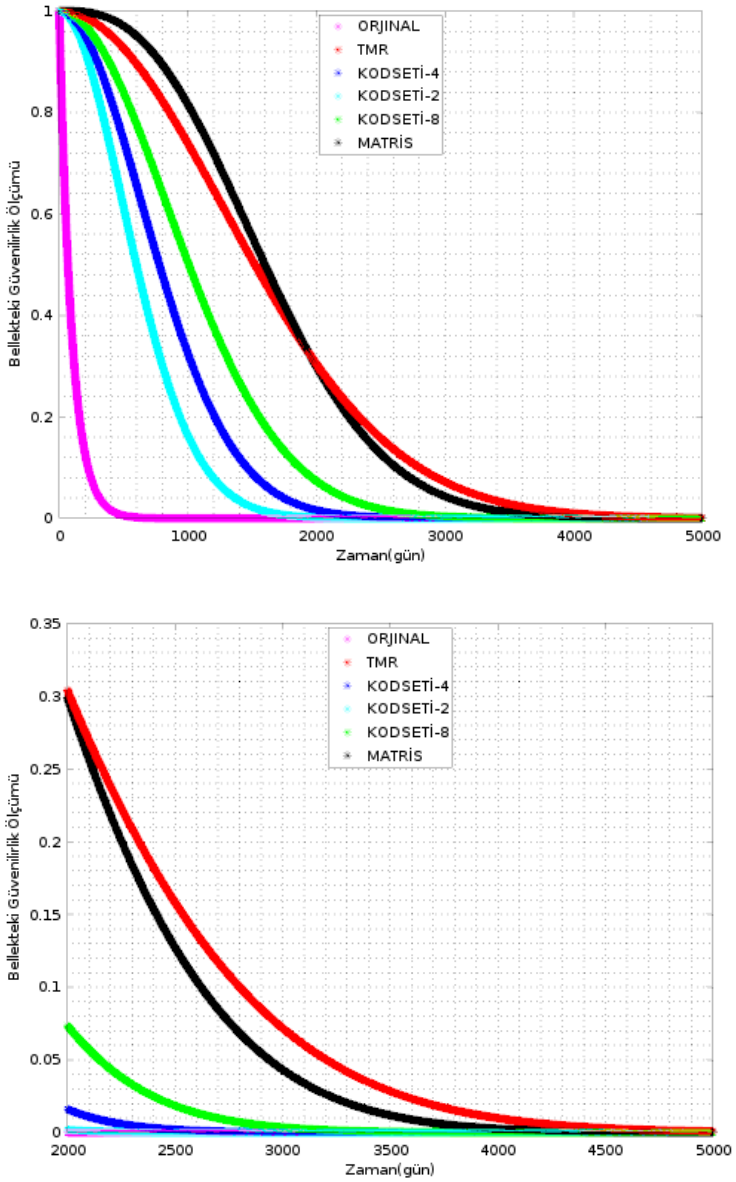
Şekil 6.12 32x32 kaydedici belleğinin için güvenilirlik analizini göstermektedir.



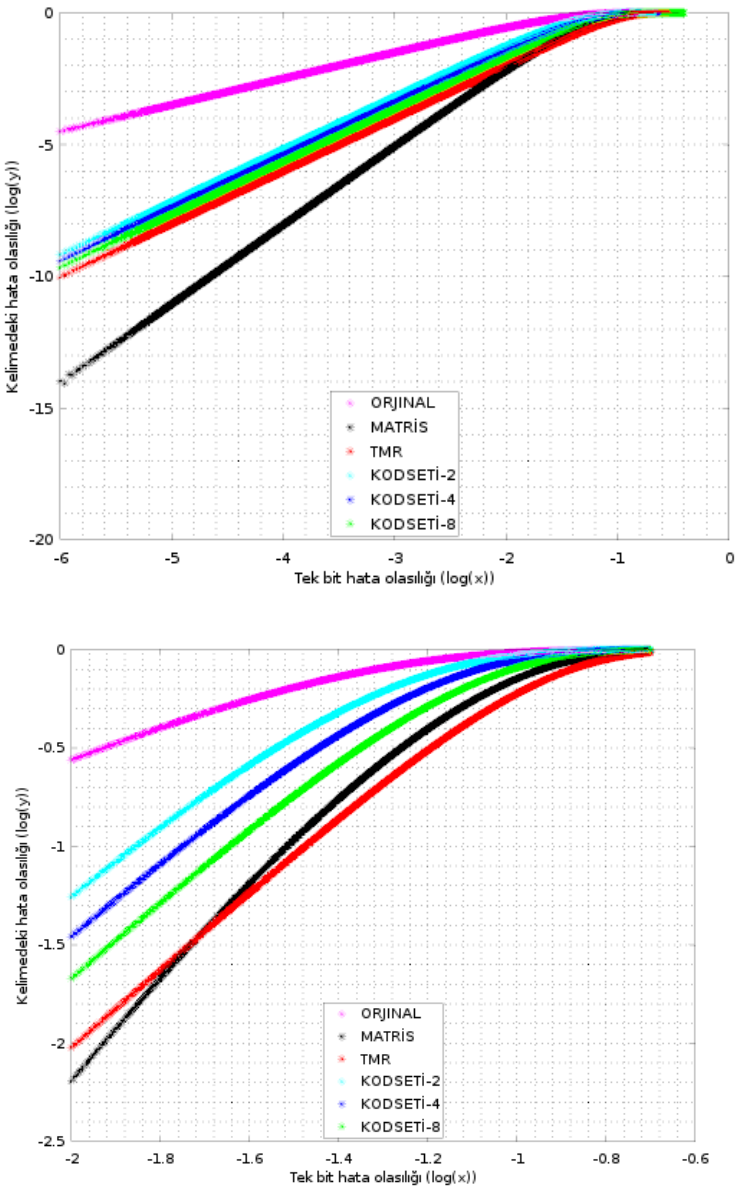
Şekil 6.11: Kod kelimesi güvenilirlik analizi ($\lambda = 10^{-5}$ bozulma/gün)

Şekil 6.13 ise sabit zamanda değişen bit hata oranınının değişimine bağlı olarak kod kelimesi için hata olasılığını göstermektedir. Grafiklerde eksenlerdeki değerler logaritma sonuçlarıdır. Başlangıçta matris kodları için kod kelimesindeki hata olasılığı diğer yöntemlere göre daha düşüktür. BHO arttırıldığında tüm yöntemler için kod kelimesindeki hata olasılıkları 1'e yaklaşmaktadır.

Rastgele gelen bitler için KodSeti yönteminin güvenilirliği diğer yöntemlere göre düşüktür. Çünkü 2. hatadan itibaren hatanın geldiği yer önem kazanmaktadır. Ancak KodSeti-4 ve 8 için ardışık hata koruma kapasitesi Matris Kodları yöntemine göre daha fazladır.



Şekil 6.12: Kaydedici belleği güvenilirlik analizi ($\lambda = 10^{-5}$ bozulma/gün)



Şekil 6.13: Bit hata oranına bağlı olarak kelimedeki hata olasılığı

7. SONUÇ

Bu tez kapsamında mikroişlemcilerin içerisinde oluşabilecek hatalara karşı düzgün bir şekilde çalışmasına devam etmesi için, hataya karşı bağışıklık yöntemleri incelenmiş ve tasarımları yapılmıştır. Mikroişlemcide en sık erişilen, verinin uzun periyotlar boyunca tutulduğu birim olan kaydedici dosyasına odaklanılmıştır.

Hata oluştuğunda devre davranışını gözlemleyebilmek için hata üretim devresi tasarlanmıştır. Test edilen tasarım olarak 8-bitlik bir mikroişlemci seçilmiş üzerine çarpıcı kodu uygulaması yazılmıştır. Kaydedici dosyasında gerekli tasarım değişiklikleri ve hata üretim devresi ile tasarımı haberleştirmek için arabirimler ve bağlantılar yapılarak bir sistem oluşturulmuştur. Kaydedici dosyasının bellek hücresine bir bitlik hatalar verilmiştir. Hata ilk olarak algoritmadaki en kritik kaydedicinin bir bitine yani belirli bir yere daha sonra da rastgele bir kaydedicinin rastgele bir bitine verilmiştir. Çıktılar incelendiğinde belirli bir yere gelen hatanın girişe verilen ve içerisindeki 1-0 sayısına göre gruplandırılan veriler için aynı grupta bulunanların benzer davranış gösterdiği tespit edilmiştir. Her komut döngüsünde farklı bir bite gelebilecek hataların doğru sonuç oranını düşürdüğü ve aynı gruptaki giriş verilerinin farklı davranışta olduğu gözlemlenmiştir.

Hataya bağışıklık kazandırma yöntemi olarak literatürde bulunan yöntemlerden donanım yedeklemesi sınıfına giren TMR yönteminin çoğunluk oylama devresi, bilgi yedeklemesi sınıfına giren Matris Kodları yönteminin eşlik ve kontrol biti hesaplamalarından faydalanılarak KodSeti adında yeni bir yöntem önerilmiştir. Bu yöntem çoklu bit hatalarına karşı koruma sağlar ve ihtiyaca göre 2, 4 veya 8 biti koruyabilecek şekilde ayarlanabilir. 32-bitlik bir veri gruplara ayrılarak her bir gruptaki eşlik, kontrol ve kesişim bitleri hesaplanır. Eşlik veya kontrol bitlerine gelebilecek bir hatanın orjinal veriyi bozmaması için oylama mekanizması düzeltme işlevine gerek olup olmadığına karar verir.

Mikroişlemciyi hataya bağışıklı hale getirebilmek için literatürde incelenen ve önerilen yöntem ile kaydedici dosyalarının tasarımları yapılmıştır. Bu tasarımlar

orjinal kaydedici dosyasının yerine konulabilmektedir. Ayrıca mikroişlemci üzerinde güvenliği kritik bir algoritma olan AES seçilerek, algoritmaya özgü komutların işletilebilmesi için kod genişletmesi yapılmış ve ardından uygulama kodu yazılmıştır. Bu algoritma çalışırken ve KodSeti yöntemi için simülasyon sonuçları alınmış, verilerin hatalar geldiğinde doğru bir şekilde düzeltildiği görülmüştür.

Anlatılan yöntemlerle tasarlanan kaydedici dosyaları Cadence RTL Compiler'da TSMC 90nm'de sentezlenip alan ve zaman karşılaştırmaları yapılmıştır. KodSeti yönteminin 2 ve 4 konfigürasyonları ile tasarlanan kaydedici dosyası diğerlerine göre daha az alan kaplamaktadır. Ancak maksimum çalışma frekansına, kritik yolun daha kısa olduğu TMR yöntemi ile yapılan tasarım sahiptir.

Güvenilirlik analizleri hataların Poisson dağılımı ile oluştuğu varsayılarak yapılmıştır. İlk olarak sabit bir bozulma oranı verildiğinde zamana bağlı olarak kaydedici ve kaydedici belleği için güvenilirlik ölçümleri yapılmıştır. Ardından sabit zamanda bit hata oranı değiştirilerek kaydedicideki hata olasılığının değişimi gözlemlenmiştir. Matris Kodları ve KodSeti yöntemleri için rastgele gelen hatalarda bozulma ya da geçici hata oranları haricinde hataların geldiği yerler de önemli olmaktadır.

Kodseti yöntemi değişik konfigürasyonlarda gerçeklenebilir, çoklu bit ardışık ya da rastgele hatalara karşı koruma sağlar. Rastgele bitlere gelen hatalar için güvenilirlik diğer yöntemlere göre düşüktür ancak TMR yöntemine göre daha az alan kaplar. KodSeti 4 ve 8 konfigürasyonlarının ardışık hataları koruma kapasitesi ve maksimum çalışma frekansı Matris Kodlarına göre daha fazladır. Literatürdeki yöntemlere alternatif bir yöntem olarak kullanılabilir.

KAYNAKLAR

[1] **Roy, K.**, **Mak, T. ve Cheng, K.T.** (2003). Test consideration for nanometer scale CMOS circuits, *VLSI Test Symposium, 2003. Proceedings. 21st*, s.313–315.

[2] **Schwank, J.**, **Shaneyfelt, M.**, **Fleetwood, D.**, **Felix, J.**, **Dodd, P.**, **Paillet, P. ve Ferlet-Cavrois, V.** (2008). Radiation Effects in MOS Oxides, *Nuclear Science, IEEE Transactions on*, ***55***(4), 1833–1853.

[3] **Mukherjee, S.** (2008). *Architecture Design for Soft Errors*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[4] **Liden, P.**, **Dahlgren, P.**, **Johansson, R. ve Karlsson, J.** (1994). On latching probability of particle induced transients in combinational networks, *Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on*, s.340–349.

[5] **Bower, F.**, **Shealy, P.**, **Ozev, S. ve Sorin, D.** (2004). Tolerating hard faults in microprocessor array structures, *Dependable Systems and Networks, 2004 International Conference on*, s.51–60.

[6] **Mehdizadeh, N.**, **Shokrolah-Shirazi, M. ve Miremadi, S.** (2008). Analyzing fault effects in the 32-bit OpenRISC 1200 microprocessor, *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, s.648–652.

[7] **Blome, J.A.**, **Gupta, S.**, **Feng, S.**, **Mahlke, S. ve Bradley, D.**, (2006), Cost-Efficient Soft Error Protection for Embedded Microprocessors.

[8] **Gizopoulos, D., P.A. ve Zorian, Y.** (2004). *Embedded Processor-Based Self-Test*, cilt 28, Springer US.

[9] **Montesinos, P.**, **Liu, W. ve Torrellas, J.** (2007). Using Register Lifetime Predictions to Protect Register Files against Soft Errors, *Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on*, s.286–296.

[10] **Vera, X.**, **Abella, J.**, **Carretero, J.**, **Chaparro, P. ve Gonzalez, A.** (2009). Online error detection and correction of erratic bits in register files, *On-Line Testing Symposium, 2009. IOLTS 2009. 15th IEEE International*, s.81–86.

[11] **Memik, G.**, **Kandemir, M. ve Ozturk, O.** (2005). Increasing register file immunity to transient errors, *Design, Automation and Test in Europe, 2005. Proceedings*, s.586–591 Vol. 1.

[12] **Breuer, M.** (2005). Multi-media applications and imprecise computation, *Digital System Design, 2005. Proceedings. 8th Euromicro Conference on*, s.2–7.

[13] **Dubrova, E.** (2013). *Fault-tolerant design.*, Berlin: Springer.

[14] **Tezzaron Semiconductor**, (2004), Soft Error in Electronic Memory, `http://www.tezzaron.com/about/papers/`.

[15] **Xilinx**, (2012), Considerations Surrounding Single Event Effects in FPGAs, ASICs, and Processors, `http://www.xilinx.com/support/documentation/white_papers/wp402_SEE_Considerations.pdf`.

[16] **Mastipuram R. and Wee E.C.**, (2004), Soft errors' impact on system reliability, `http://www.pld.ttu.ee/IAF0030/454636.pdf`.

[17] **SquareTrade**, (2008), Report on Xbox 360 failure rates, `http://blog.squaretrade.com/2008/02/xbox-fail-rates.html`.

[18] **White M.,Bernstein, J.B.**, (2008), Microelectronics Reliability: Physics-of-Failure Based Modeling and Lifetime Evaluation, `http://https://nepp.nasa.gov/files/16365/08_102_4_%20JPL_White.pdf`.

[19] **ITEM Software**, (2007), Reliability Prediction Basics, `http://www.reliabilityeducation.com/ReliabilityPredictionBasics.pdf`.

[20] **Ellerman P.**, (2012), Calculating Reliability using FIT - MTTF: Arrhenius HTOL Model, `http://www.microsemi.com/document-portal`.

[21] **Baumann, R.** (2001). Soft errors in advanced semiconductor devices-part I: the three radiation sources, *Device and Materials Reliability, IEEE Transactions on*, ***1***(1), 17–22.

[22] **Reed, R.**, **Carts, M.**, **Marshall, P.**, **Marshall, C.**, **Musseau, O.**, **McNulty, P.**, **Roth, D.**, **Buchner, S.**, **Melinger, J. ve Corbiere, T.** (1997). Heavy ion and proton-induced single event multiple upset, *Nuclear Science, IEEE Transactions on*, ***44***(6), 2224–2229.

[23] **Giot, D.**, **Roche, P.**, **Gasiot, G. ve Harboe-Sorensen, R.** (2007). Multiple-Bit Upset Analysis in 90 nm SRAMs: Heavy Ions Testing and 3D Simulations, *IEEE Transactions on Nuclear Science*, ***54***, 904–911.

[24] **Hazucha, P.**, **Svensson, C. ve Wender, S.** (2000). Cosmic-ray soft error rate characterization of a standard 0.6-/spl mu/m CMOS process, *Solid-State Circuits, IEEE Journal of*, ***35***(10), 1422–1429.

[25] **Koren, I. ve Krishna, C.M.** (2007). *Fault-Tolerant Systems*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[26] **Benso, A. ve Prinetto, P.** (2010). *Fault Injection Techniques and Tools for Embedded Systems Reliability Evaluation*, Springer Publishing Company, Incorporated, 1st sürüm.

[27] **Seward, S. ve Lala, P.** (2003). Fault injection in digital logic circuits at the VHDL level, *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE*, s.161–.

[28] **M., N.** (2011). *Soft Errors in Modern Electronic Systems*, Springer, US, 1th sürüm.

[29] **Hsiao, M.** (1970). A Class of Optimal Minimum Odd-weight-column SEC-DED Codes, *IBM Journal of Research and Development*, ***14***(4), 395–401.

[30] **Argyrides, C.**, **Pradhan, D. ve Kocak, T.** (2011). Matrix Codes for Reliable and Cost Efficient Memory Chips, *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, ***19***(3), 420–428.

[31] **Argyrides, C.**, **Zarandi, H. ve Pradhan, D.** (2007). Matrix Codes: Multiple Bit Upsets Tolerant Method for SRAM Memories, *Defect and Fault-Tolerance in VLSI Systems, 2007. DFT '07. 22nd IEEE International Symposium on*, s.340–348.

[32] **Naseer, R.** (2008). A Framework For Soft Error Tolerant SRAM Design, *Doktora Tezi*, University of Southern California.

[33] **Patterson, D.A. ve Hennessy, J.L.** (2008). *Computer Organization and Design, Fourth Edition, Fourth Edition: The Hardware/Software Interface (The Morgan Kaufmann Series in Computer Architecture and Design)*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 4th sürüm.

[34] **Nurmi, J.** (2007). *Processor Design: System-On-Chip Computing for ASICs and FPGAs*, Springer Publishing Company, Incorporated, 1st sürüm.

[35] **Abazari, M.**, **Fazeli, M.**, **Patooghy, A. ve Miremadi, S.** (2012). An efficient technique to tolerate MBU faults in register file of embedded processors, *Computer Architecture and Digital Systems (CADS), 2012 16th CSI International Symposium on*, s.115–120.

[36] **Esmaeeli, S.**, **Hosseini, M.**, **Vosoughi Vahdat, B. ve Rashidian, B.** (2011). A multi-bit error tolerant register file for a high reliable embedded processor, *Electronics, Circuits and Systems (ICECS), 2011 18th IEEE International Conference on*, s.532–537.

[37] **Amiri-Kamalabad, M.**, **Miremadi, S. ve Fazeli, M.** (2008). A Power Efficient Approach to Fault-Tolerant Register File Design, *VLSI Design, 2008. VLSID 2008. 21st International Conference on*, s.21–26.

[38] **Miremadi, S. ve Zarandi, H.** (2005). Reliability of protecting techniques used in fault-tolerant cache memories, *Electrical and Computer Engineering, 2005. Canadian Conference on*, s.820–823.

[39] **Saleh, A.**, **Serrano, J. ve Patel, J.** (1990). Reliability of scrubbing recovery-techniques for memory systems, *Reliability, IEEE Transactions on*, ***39***(1), 114–122.

[40] **of Standards, N.I.** (2001). *Announcing the Advanced Encryption Standard (AES)*, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, MD.

[41] **Tillich, S. ve Großschädl, J.** (2006). Instruction Set Extensions for Efficient AES Implementation on 32-bit Processors, ***L. Goubin ve M. Matsui,*** *(düzenleyenler), Cryptographic Hardware and Embedded Systems – CHES 2006*, cilt4249 of *Lecture Notes in Computer Science*, Springer Verlag, s.270 – 284.

EKLER

EK A.1 : AES-128 Assembly Kodu
EK A.2 : Mikroişlemci Tasarım Dosyaları Hiyerarşisi
EK A.3 : Ardışık hataları incelemek için oluşturulan verilog tanım dosyası

EK A.1

```
1        lui  $1, 0x3243
2        ori $1, $1, 0xF6A8
3        lui  $2, 0x885A
4        ori  $2, $2, 0x308D
5        lui  $3, 0x3131
6        ori $3, $3, 0x98A2
7        lui  $4, 0xE037
8        ori $4, $4, 0x0734
9        lui  $5, 0x2B7E
10       ori $5, $5, 0x1516
11       lui  $6, 0x28AE
12       ori $6, $6, 0xD2A6
13       lui  $7, 0xABF7
14       ori $7, $7, 0x1588
15       lui  $8, 0x09CF
16       ori $8, $8, 0x4F3C
17        lui $15, 0x0100
18        xor $1, $5, $1
19        xor $2, $6, $2
20        xor $3, $7, $3
21        xor $4, $8, $4
22        sbox4r $9, $8, $8
23       xor $9, $5, $9
24       xor $5, $9, $15
25       xor $6, $5, $6
26       xor $7, $6, $7
27       xor $8, $7, $8
28       addi $19, $0, 2
29       addi $20, $0, 9
30       sll $15, $15, 1
31 Loop: addi $20, $20,-1
32       sbox4s $10, $1, $2
33       sbox4s $11, $2, $3
34       sbox4s $12, $3, $4
35       sbox4s $13, $4, $1
36       mixcol4s $1, $10, $12
37       mixcol4s $2, $11, $13
38       mixcol4s $3, $12, $10
39       mixcol4s $4, $13, $11
40       xor $1, $5, $1
41       xor $2, $6, $2
42       xor $3, $7, $3
43       xor $4, $8, $4
44       sbox4r $9, $8, $8
45       xor $9, $5, $9
46       xor $5, $9, $15
47       xor $6, $5, $6
48       xor $7, $6, $7
49       xor $8, $7, $8
50       beq $20, $19 Round
51       sll $15, $15, 1
52       bne $20, $0  Loop
```

```
53        sbox4s $10, $1, $2
54        sbox4s $11, $2, $3
55        sbox4s $12, $3, $4
56        sbox4s $13, $4, $1
57        mixcol4s $1, $10, $12
58        mixcol4s $2, $11, $13
59        mixcol4s $3, $12, $10
60        mixcol4s $4, $13, $11
61        xor $1, $5, $1
62        xor $2, $6, $2
63        xor $3, $7, $3
64        xor $4, $8, $4
65        sw $1, 240($0)
66        sw $2, 241($0)
67        sw $3, 242($0)
68        sw $4, 243($0)
69  End:  addi $1, $0, 0
70        beq $1, $0 End
71  Round:lui $15, 0x1B00
72        j   Loop
```

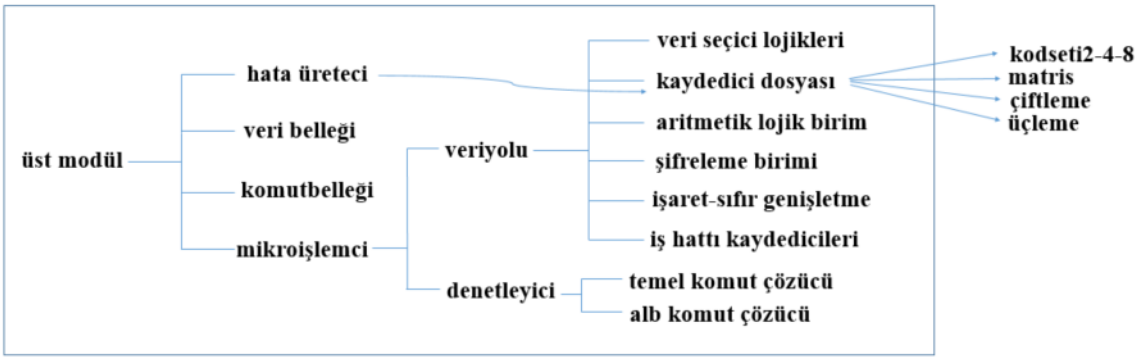
EK A.2

Şekil A.1: Mikroişlemci tasarım klasörleri

Şekil A.2'de tasarım ağacı gösterilmiştir. Yapılacak seçimlere istenilen hata bağışıklığına sahip tasarımlar mikroişlemci ile bütünleştirilebilir.

EK A.3

Ardışık hataları incelerken tasarım dosyalarının anlaşılabilir olması için bir tanım dosyası oluşturulmuştur. Verilen örnek tanım dosyası incelendiğinde 15 adet kaydedicinin $[29:27]$ bitlerinin değerlerini lojik 1'de tutacak şekilde ayarlanmıştır. Hata bağışıklık yöntemi olarak kodseti-4 yöntemi seçilmiştir. Kaydedici dosyası hatalı, kontrol ve eşlik bit dosyaları hatasız olarak belirlenmiştir.



Şekil A.2: Tasarım Ağacı

```
////KAYDEDICI DOSYASINA HATA VERME ILE ILGILI TANIMLAR

////Hata verilecek kaydedicideki bit sayilari
`define FT_WIDTH   3 //hatali bitler
`define FT_VAL     7//hata turu
`define FT_PLACE  29//hata baslangic yeri

////Hata verilecek kaydedici sayisi tanimlari
`define FT_LOC_NO 15 //hatali kaydedici sayisi
`define FT_LOC     0 //hata baslangic yeri


////HATA BAGISIKLIK YONTEMI SECIMI
//`define DWC
//`define TMR
//`define MATRIX
//`define PARITY
`define BURST


////UCLU MODUL COGULLAMA YONTEMI ICIN TANIMLAR
//`define RF_FT1 //kaydedici dosyasi1 hatali
//`define RF_NONFT1 //kaydedici dosyasi1 hatasiz
//`define RF_FT2 //kaydedici dosyasi2 hatali
//`define RF_NONFT2 //kaydedici dosyasi2 hatasiz
//`define RF_FT3 //kaydedici dosyasi3 hatali
//`define RF_NONFT3 //kaydedici dosyasi3 hatasiz


////KODSETI YONTEMI ICIN TANIMLAR
`define DATASET   4
`define DATASET4

`define RF_FT //kaydedici dosyasi hatali
//`define RF_NONFT //kaydedici dosyasi hatasiz
//`define CBF_FT //kontrol bit dosyasi hatali
`define CBF_NONFT//kontrol bit dosyasi hatasiz
//`define PBF_FT//eslik biti hatali
`define PBF_NONFT//eslik biti hatasiz

//eslik bitleri iCin tanimlar
```

```
42 `define PB_WIDTH  16
43 `define PB_DEPTH  32
44 //kontrol bitleri iCin tanimlar
45 `define CB_WIDTH  8 //Kod seti 8-icin-16 4-icin-8 2-icin-4
46 `define CB_DEPTH  32
```

ÖZGEÇMİŞ

Ad Soyad: Buse USTAOĞLU

Doğum Yeri ve Tarihi: İstanbul-30.08.1991

Adres: Göztepe/İSTANBUL

E-Posta: ustaoglubu@itu.edu.tr

Lisans: İstanbul Teknik Üniversitesi(2013)

Y. Lisans: İstanbul Teknik Üniversitesi(2015)

Mesleki Deneyim ve Ödüller:
Araştırma Görevlisi- İTÜ (12/2013-devam)
ASIC Tasarım ve Dogrulama Mühendisi- Anka Mikroelektronik Sistemler (7/2013-12/2013)

Yayın ve Patent Listesi:
Yeniçeri R., **Ustaoğlu B.**, Yalçın M.E., "Throughput Enhancement for a New Timedelay Sampleddata System Based True Random Bit Generator", European Conference on Circuit Theory and Design (ECCTD), 2013 21st, 8-12 September 2013
B. Ustaoglu, B. Ors, "Mikroislemci Tabanli Bir Sisteme Hata Enjekte Etme Yontemi Gelistirilmesi ve Hata Tespit Mekanizmasinin Gerceklenmesi", Elektrik - Elektronik Ve Bilgisayar Muhendisligi Sempozyumu, 27-30 Kasim 2014, Bursa.
Ahmet Çagrı Bağbaba, **Buse Ustaoğlu**, İnan Erdem, Gökhan Işık, Berna Örs, "Leon3 Tabanlı SoPC Tasarımı ve Uygulama Gerçeklenmesi", GOMSIS2014.
B. Ustaoğlu, Ç. Bağbaba, B. Örs, İ. Erdem, "Seri Çevresel Arayüzü için Evrensel Doğrulama Metodu ile Test Ortamının Oluşturulması", Sinyal Isleme ve Iletisim Uygulamalari (SIU) Kurultayi, 16-19 Mayis 2015, İnönü Üniversitesi, Malatya, Türkiye.
Ustaoğlu B.,Ors B. "Design and implementation of a custom verification environment for fault injection and analysis on an embedded microprocessor", Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), 2015 3rd International Conference on,29 April-1 May 2015.
Bağbaba Ç, **Ustaoğlu B.**, Ors B., Erdem İ., "A Layered UVM Based Testbench Design for SpaceWire", Electrical and Electronics Engineering (ELECO), 2015,8th International Conference on,27-29 Nov. 2015
Ustaoğlu B., Ors B. "Fault Tolerant Register File Design for MIPS AES-Crypto Microprocessor", International Conference on Electronics, Circuits, and Systems (ICECS), 2015 22nd, 6-9 December 2015.

TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

B. Ustaoglu, B. Ors, “Mikroislemci Tabanli Bir Sisteme Hata Enjekte Etme Yontemi Gelistirilmesi ve Hata Tespit Mekanizmasinin Gerceklenmesi”, Elektrik - Elektronik Ve Bilgisayar Muhendisligi Sempozyumu, 27-30 Kasim 2014, Bursa.

Ustaoğlu B., Ors B. "Design and implementation of a custom verification environment for fault injection and analysis on an embedded microprocessor", Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), 2015 3rd International Conference on,29 April-1 May 2015.

Ustaoğlu B., Ors B. "Fault Tolerant Register File Design for MIPS AES-Crypto Microprocessor", International Conference on Electronics, Circuits, and Systems (ICECS), 2015 22nd, 6-9 December 2015.